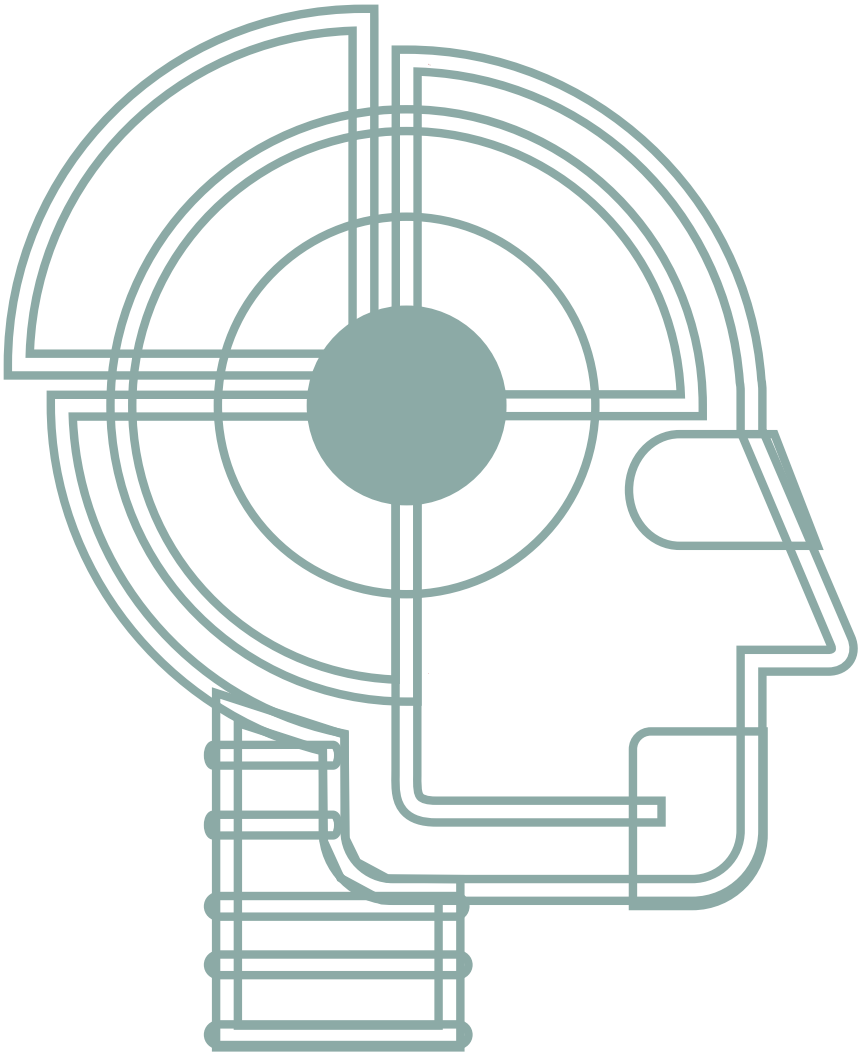
#MİLLİ TEKNOLOJİ HAMLESİ

DENEYAP TÜRKİYE

YAPAY ZEKÂ

LİSE



T.C. SANAYİ VE TEKNOLOJİ BAKANLIĞI | T.C. GENÇLİK VE SPOR BAKANLIĞI | TÜBİTAK | TÜRKİYE TEKNOLOJİ TAKIMI

TÜBİTAK Deneyap Kitapları 14

Yapay Zekâ
LİSE

Doç. Dr. Utku KÖSE
Doç. Dr. Koray ÖZSOY
Dr. Öğr. Üyesi Bekir AKSOY

TÜBİTAK Deneyap Kitapları *DENEYAP TÜRKİYE* Projesi kapsamında hazırlanmıştır.

ISBN: 978-605-312-518-1
Yayıncı Sertifika No: 47703

Yayım Tarihi: Eylül 2023

TÜBİTAK Başkanı: Prof. Dr. Hasan MANDAL
Bilim ve Toplum Başkanı: Ömer KÖKÇAM
Genel Yayın Yönetmeni: Fatma BAŞAR
Editörler: Dr. İpek PİRPİROĞLU GENCER - Havva Hilal KAÇAR
Düzelti: Dr. Mustafa ORHAN
Telif İşleri Sorumlusu: Havva Hilal KAÇAR

TÜBİTAK Bilim ve Toplum Başkanlığı
Tunus Caddesi No: 80 Kavaklıdere 06680 Ankara
Tel: (312) 298 96 50
e-posta: deneyap@tubitak.gov.tr
https://yayinlar.tubitak.gov.tr/deneyap-atolyesi

DENEYAP

Teknoloji Atölyeleri

YAPAY ZEKÂ

LİSE

Doç. Dr. Utku KÖSE
Doç. Dr. Koray ÖZSOY
Dr. Öğr. Üyesi Bekir AKSOY

TÜBİTAK

İçindekiler

Sunuş

Tüm dünyada olduğu gibi ülkemizde de teknolojik gelişmeler diğer bir ifade ile dijital endüstri hızlı biçimde değişmektedir. Endüstri başta olmak üzere ekonominin sağlık, havacılık-savunma sanayi, otomotiv, imalat, tarım ve eğitim başta olmak üzere tüm alanlarında dijital dönüşümü gerçekleştirmek için; Büyük Veri, Sensörler/Algılama Sistemleri, Nesnelerin İnterneti, Artırılmış Gerçeklik, Bulut Teknolojileri, Siber Güvenlik, Akıllı Fabrikalar, Eklemeli imalat, Yapay Zekâ gibi teknolojiler kullanılmaktadır. İnternet teknolojilerin hızla gelişmesi ile üretilen büyük verilerin hacimleri, geliştirilen algoritmalar ve veri depolama sistemlerindeki gelişmeler yapay zekâ teknolojilerini bugün çok daha popüler hale getirmiştir. Bu nedenle ülkemizdeki öğrencilerin teknolojik gelişmelere entegrasyonunda ve milli teknolojik hamlelerin geliştirilmesinde en önemli uygulamalı eğitim kurumlarından birisi de TÜBİTAK Deneyap Teknoloji Atölyeleri'dir.

Yapay zekâ teknolojileri, öğrencilerin bilişsel ve analitik düşünme becerisini geliştirmek için uygun bir teknolojik yöntemdir. Bu yöntem uygun bir eğitmen entegrasyonuyla gerçekleştirildiğinde öğrencilerin analitik ve zihinsel düşünme ile mesleki ve toplumsal gelişimlerine üst düzey problem çözme, yaratıcılık becerilerini geliştirmesine katkı sağlayacağı düşünülmektedir. Hazırlanan kitapta öğrenci ve eğitmenlere yapay zekâ teknolojileri program geliştirme ve uygulama yapma sürecinde rehberlik etmesi hedeflenmiştir. Öğretmenlerin öğrencilerin gelişmesine yardımcı olabilmesi için adım adım kod yazma öğretim modeli kullanmıştır. Bu amaç için bu kitabın her aşamasında yazarlar tarafından geliştirilen "Algıla, Tasarla, Harekete Geç, Yürüt ve Karar Ver" öğrenme döngüsü kullanılmıştır.

Öğrenme döngüsünde yapay zekâ teknolojileri algoritmalarının temel kavramların oluşturulmasında eğitmen temelli algıla yaklaşımı sunulmuştur. Öğrencilerin verilen temel yapay zekâ bileşenleri ile ileri seviye dikkat ve motivasyonlarını sağlama yaklaşımları ön plana çıkarılmaya çalışılmıştır. Tasarla bölümünde eğitmen yapay zekâ veri setini öğretme ve hazırlama işlemlerini gerçekleştirmektedir. Hareket geç bölümünde eğitmen öğrencilerden Tasarla bölümünde öğretilen veri setlerine uygun yapay zekâ yöntemlerine ait kodları birlikte yazar. Öğrenciler Yürüt bölümünde Tasarla ve Harekete Geç aşamalarında hazırlamış oldukları uygulamalar için yapay zekâ modellerine ait eğitim ve tahminlenmeden elde edilecek sonuçları doğru biçimde kodlaması sağlar. Karar ver bölümünde yapay zekâ modelinden elde edilen sonuçların başarısını test eder. Bu öğretim döngüsünden sonra eğitmen öğrencilere her haftaya ait birer etkinlik uygulaması vererek öğrencilerin yapay zekâ algoritmalarına yönelik Python programlama dili kullanarak kod yazma, hata ayıklama, modeli eğitme, tahminleme ve modelden elde edilecek doğruluk başarım sonuçlarını test eder.

Kitapta yapay zekâ algoritmalarının modellenmesi için kolay öğrenebilen, nesne tabanlı uygulamaları ile oldukça sıklıkla tercih edilen açık kaynak kodlu Python programlama dilindeki Spyder editörü kullanılmıştır. Yapay zekâ modellerini uygulamak için Python programlama dilinde sıklıkla kullanılan Numpy, Matplotlib, Scipy, Scikit-Learn, TensorFlow, Keras, Pytorch kütüphaneleri kullanılmıştır. Ayrıca her hafta öğrencilerin yapay zekâ modellerini daha iyi anlayabilmeleri için açık erişimli internet sitelerinden farklı konularda farklı veri setleri kullanılarak

uygulamaları yapması sağlanmıştır. Eğitmen etkinlik öncesinde öğrencilere yapay zekâ modelleri kullanılarak gerçekleştirilecek uygulamaları için açık erişimli internet sitesinden veri setini yükleyerek öğrenciler tarafından kullanılmasını sağlar. Bu kitap lise seviyesinde yapay zekâ eğitimi vermek isteyen eğitmenler için uygulamalı biçimde hazırlanmıştır. Kitapta, öğrenciler ilk aşamada yapay zekanın kuramsal temellerini, ikinci aşamada problemin belirleyerek açık erişimli internet sitelerinden veri setlerini yüklenmesi ve verilerin analizi adımlarını ve son aşamada ise yapay zekâ modelleri kullanılarak problemi çözme, tahminleme ve raporlama adımlarını kazanması amaçlanmıştır. Konuların tamamı öğrenme döngüsü kavramı içerisinde oluşturulmuştur. Kitap 8 haftalık bir ders planı olarak tasarlanmıştır. Her haftaya bir bölüm düşecek şekilde sekiz haftalık içerik hazırlanmıştır. Öğrencilerin öğrenmiş oldukları yapay zekâ modelleri ile ilgili 4. haftadan itibaren projeler geliştirilmesi hedeflenmiştir.

Sevgili eğitmenlerimiz ve öğrencilerimiz başta olmak üzere kitabın bütün eğitim camiamıza katkı sağlaması dileğiyle…

Öğretim Kılavuzu

YAPAY ZEKÂ DERSİ ÖĞRETİM KILAVUZU

Öğrenme, bireyin yaşantılar sonucu davranışlarında meydana gelen oldukça uzun süreli değişimlerdir. Eğitim üçgenine göre eğitmen ve bilgi etkileşimi "öğretim", eğitmen ve öğrenci ilişkileri "pedagoji", öğrenci ve bilgi etkileşimi "öğrenme" sürecini oluşturmaktadır (Amerikan Ulusal Araştırma Konseyi, 2012). İnternet devrimi ve bilgisayar kullanımının yaygınlaşması eğitim alanında da bir "dijital devrim" olarak ortaya çıkmıştır. İçinde bulunduğumuz dönem "bilgi çağı" olarak adlandırılmaktadır. 21. Yüzyıl öğrenenleri için bilişsel, kişisel, kişiler arası alan olmak üzere üç temel yeterlik başlığı belirlemiştir Amerikan Ulusal Araştırma Konseyi (2012). Söz konusu yeterliklerden de hareketle yapay zekanın ele alındığı bu öğretim kitabında da bağlam temelli öğrenme-öğretme ve problem çözmenin bilişsel yaklaşım içerisinde harmanlandığı bir strateji benimsenmiş durumdadır.

Bireyin gerçek hayatta karşılaştığı olgu, olay veya kullanmış olduğu teknolojinin ders içerikleriyle ilişkilendirilmesi bağlam temelli öğrenme-öğretme olarak tanımlanır (MEB, 2012). Öğrenciye aktarılacak yeni bilgiler ile önceki öğrenmeleri arasında kurulan köprü veya bağlantı olarak düşünülebilir. Bu kitapta yapay zekâ çözümleriyle bağlantılı içerik de bireyin gerçek hayatta karşılaştığı, hatta günümüzde güncelliğini koruyan olgular, olay ve teknolojiler çerçevesinde ele alınmıştır. Bu olaylar özellikle yapay zekâ yönünde problem çözme odaklı eylemlerin kullanılması yönünde irdelenmiş ve böylelikle problem çözme yoluyla yapay zekanın (ve ilgili tekniklerin) öğretilmesi hedeflenmiştir.

Bağlam temelli öğrenme adımından sonra konumlandırılan problem çözme, sorunun nedenini belirleyerek çözüm için alternatifleri deneyerek güçlükleri yenme ve istenilen sonuca ulaşma çabasıdır. Problemler, tecrübelerimiz ve öğrenme süreçleriyle elde ettiğimiz bilgi-becerilerinin yapılandırılmasını gerekli kılmaktadır (Jonassen, 2007). Bu kitaptaki yapay zekâ öğretimi de bireylere her yeni yapay zeka tekniğinin problem çözme odaklı uyarlanmasını önermekte, problem çözümleri yapay zekanın ve ilgili tekniklerin kavranmasını kolaylaştırmaktadır. Bu yolla da esasında bilişsel yaklaşım içerisinde bir süreç de tecrübe edilmektedir.

Bilişsel yaklaşıma göre eğitim, bireyin çeşitli yaşantılar yoluyla zihnindeki şemaları geliştirme sürecidir. Zihinsel şemalar bilgileri düzenleme, yerleştirme ve kullanma biçimlerini içermektedir. Öğrencinin zihnindeki şemaların zengin ve gelişmiş olması alınan bilgilerin daha kolay özümsenmesini sağlamaktadır. Etkili ve verimli bir eğitim için bireylerin zihinsel değişimine katkı sağlayan öğrenme türlerine ağırlık verilmelidir. Öğrencilerin girişimcilik, yaratıcı düşünme, eleştirel düşünme, karmaşık problemleri çözme, etkili iletişim ve takım çalışması gibi becerileri kazanmalarına yönelik olarak tasarlanmalıdır. Benzer şekilde kitap içerisinde sunulan içerik özellikle kodlama çözümleri aracılığıyla bireylerin zihin şemalarını bir arada etkili bir şekilde

kullanarak etkili ve verimli bir bilgi-beceri kazanımı oluşturmalarını sağlamakta, bu yolla yapay zekâ öğretimi kapsamlı bir şekilde tamamlanmaktadır.

Günümüzde bilgi teknolojilerindeki gelişmeler ile bilim arasında doğrudan bir ilişki vardır. Bilgi teknolojileri bilimsel çalışmalar içerisindeki veri düzenlemesi, veri yönetimi-analizi ve elde edilen bulguların yorumlanması gibi aşamalarda aktif bir biçimde kullanılmaktadır (Hacer, 2004). Söz konusu aşamalardan hareketle kitaptaki içerik de algıla, tasarla, harekete geç, yürüt ve karar ver öğrenme döngüleri içerisinde organize edilmiştir. Bu döngüler bağlam temelli öğrenme-öğretme ve problem çözme ile bilişsel yaklaşımı desteklerken, bireylere bilimsel çalışmalarda izleyebilecekleri adımlara aşina olmalarını mümkün kılmaktadır. Söz konusu döngüler ve kitaptaki yapay zekâ uygulamalarıyla bağlantılı Python programlama diline ilişkin genel bilgiler ilerleyen paragraflar altında açıklanmıştır:

ALGILA- TASARLA – HAREKETE GEÇ- YÜRÜT-KARAR VER ÖĞRENME DÖNGÜSÜ

Algıla-Tasarla-Hareket Geç-Yürüt-Karar Ver öğrenme döngüsü adımları, yapay zekanın uygulamalarla etkili öğretilebilmesi için Python programlama dilini de dikkate almak üzere şu şekilde kullanılmalıdır:

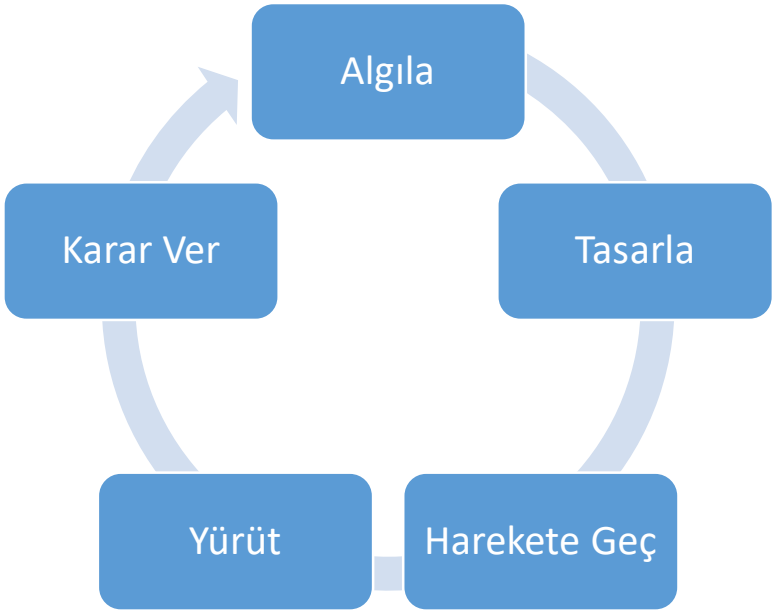
Algıla: Bu bölümde, ilgili hafta ile ilgili eğitmen öğrencilere teorik bilgileri görsel destekler ile aktarır. Anlatılacak konu ile ilgili olasılı avantaj ve dezavantajları detaylı biçimde anlatarak öğrencilerin dersin konusu ile ilgili algısal düşüncelerini arttırır. Eğitmen, Python programlama geçmiş bilgilerini aktive etmek ve Python yapay zekâ kütüphaneleri hakkında öğrencilerin dikkat ve motivasyonlarını sağlamak ile görevlidir. Eğitmenin bir önceki derste yapılan yapay zekâ uygulamalarını mutlaka özetlemesi gerekir.

Tasarla: Bu bölümde eğitmen öğrenciler ilk olarak ilgili hafta ile ilgili belirlenen problemleri anlaması sağlayarak ilgili haftada belirlenen yapay zeka modeli kullanarak problem çözmesi sağlanır. Bunun için Python programlama editöründe yapay zekâ veri setini öğretme ve hazırlama işlemlerini yapar. Öğrenciler, eğitmenin göstermiş olduğu kodları Python programlama editöründe uygulamaya gerçekleştirir. Bu bölüm dahil, ilerleyen uygulama odaklı bölümlerde dersteki kodlar ve veri setleri için Github üzerinde https://github.com/deneyapyz/lise/ platformu tanımlanmıştır.

Haraket Geç: Bu bölümde öğrenciler Python editöründe kod yazmak için aktif rol alırlar. Eğitmen uygulayıcı pozisyonundadır. Ayrıca öğrencilerin kod yazarken yaptıkları hatalarda destek olacaktır. Fakat eğitmenin sağladığı destek gereğinden fazla da olmamalıdır. Eğitmen öğrencilerden eğitim ve test verilerini ayırmasını ve model kurmada başarılı olmasını bekler.

Yürüt: Bu bölümde eğitmen Python kodlama ekranında program kod satırlarını adım adım yazar. Öğrenci ise eğitmenin kod yazdığı her bir adımdan sonra kodu uygulayarak ilgili haftaya ait uygulamayı gerçekleştirir. Burada, öğrenciler Python editöründe kod yazmak için aktif rol alırlar. Eğitmen uygulayıcı pozisyonundadır. Eğitmen öğrencilerden modeli eğitme ve tahminlenmesini doğru biçimde kodlamasını bekler. Öğrencilerin takıldıkları noktalarda destek olacaktır.

Karar Ver: Öğrenci Python editöründe gerçekleştirmiş olduğu yapay zekâ uygulamasındaki başarısını test eder. Elde edilen test sonuçlarını hata matrisi üzerinde gözlemleyerek yapay zekâ modeli başarısını test eder. Github platformundaki ders klasörü ile etkileşim devam eder.



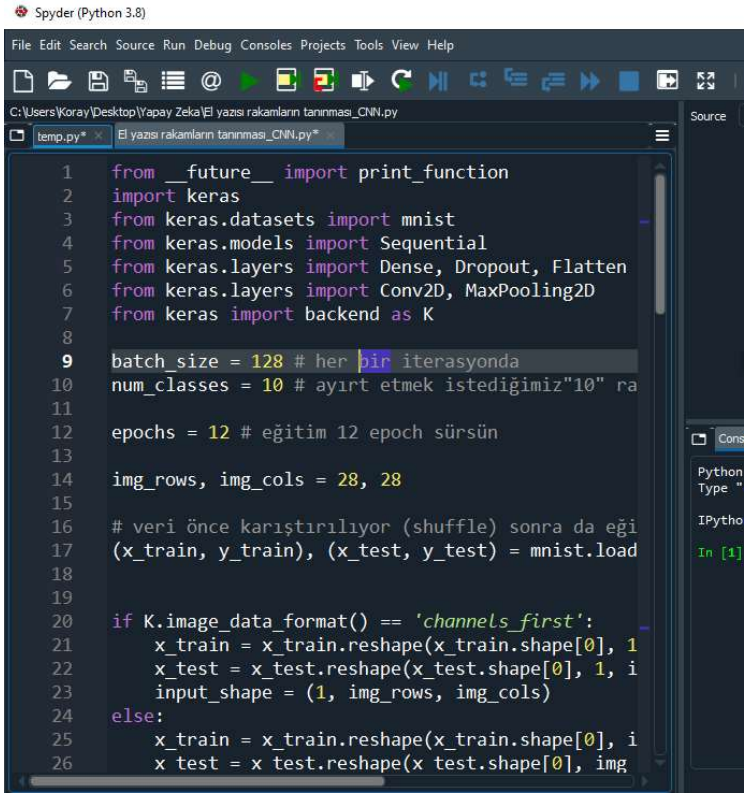
Şekil 1. Öğrenme Döngüsü

PYTHON PROGRAMLAMA ORTAMI

Python

Günümüzde teknolojinin hızla gelişmesi ile bilgisayar progralama dillerine olan ilgide gün geçtikçe artmaktadır. C++, Visual Studio.Net, Python gibi birçok yaygın biçimde programlama dili kullanılmaktadır. Özellikle açık kaynak kodlu olan ve sürekli olarak kütüphaneleri geliştirilen Python programlama dili bu diller içerisnde öne çıkan programlama dilidir. Python programlama dili basit bir kod yapısına sahip olduğu için öğrenilmesi oldukça kolaydır. Python kolay öğrenilebilirliği, ücretsiz ve açık kaynak kodlu olmasından dolayı sürekli geliştirilebilmesi sayesinde kodlamaya yeni başlayanların ilk tercihi olmaktadır. Python, 2020 IEEE araştırmasına göre dünya çapında en çok kullanılan ve tercih edilen programlama dillerinden birisidir (Cass, 2020). Yapay zekâ denince akla ilk olarak Python dili gelmektedir. Bu durum da Python'ın dünya çapında büyük bir kitlesinin olmasına neden olmaktadır. Ancak yapay zekâ uygulamalarında Matlab, C++, Lush gibi standart dillerde kullanılır. Python programlama dilinde yapay zekâ işlemede genellikle Keras, Numpy, Pytorch, Matplotlib, Scipy, Scikit-Learn kütüphaneleri kullanılmaktadır. Python programlama dilinin diğer programlama dillerinden farkı, çok yüksek seviyeli olması dolayısıyla az sayıda komutla çok sayıda işlemin pratik şekilde gerçekleştirilebilmesi, nesne yönelimli programlama yaklaşımının hızlı bir şekilde hayata geçirilebilmesi ve her geçen zaman genişleyen, aktif bir kütüphane havuzuna sahip olmasıdır. Python'da IDLE, Spyder ve Pycharm gibi farklı kod yazma editörleri kullanabilmektedir. Sahip

olduğu yetenekler dolayısıyla Python programlama dili üzerinden internet / web platformları geliştirme, bilimsel ve sayısal hesaplama süreçleri işletme, yapay zekâ sistemleri oluşturma, robotik çözümlerin kod altyapısını destekleme gibi birçok farklı alanda uygulamalar gerçekleştirilmektedir.



Şekil 2. Python Spyder kodlama ortamı.

Kitap içeriği hem yüz yüze hem de uzaktan eğitime uyumlu bir yapıda inşa edilmiştir. Özellikle uzaktan eğitim süreçlerinde, içerikte sunulan açıklamaların ve yönlendirici kutucukların dikkate alınması öğretim-öğrenim süreçlerinin kalitesini artırabilecektir. Kitapta içerik akışı Python ile yapay zeka, yapay zeka matematiği ve bulanık mantık, olasılık çözümleri için bayes öğrenme, karar ağaçları makine öğrenme algoritması, yapay sinir ağları, etmen tabanlı modelleme, zeki optimizasyon ve derin öğrenme olmak üzere 8 haftalık program altında kullanıma sunulmuştur. Öğrenciler 8 haftalık program sonucunda şu kazanımları elde etmektedir:

- Algoritmik yapıları öğrenir ve basit bir yapay zekâ oluşturur.
- Python ile bulanık mantık tabanlı matematiksel işlemleri gerçekleştiren yapay zeka tabanlı çözüm üretir.
- Makine ve Bayes öğrenme ile regresyon, sınıflandırma ve kümeleme yöntemlerini uygun olanı modeller.
- Karar ağaçları algoritmasının kullanarak örnek uygulama yapar.
- Yapay sinir ağları modeli ile yüksek bir başarım oranı ile model oluşturur.
- Etmen tabanlı modeller davranışlarını kullanarak mimari tasarlar ve problemi çözer.
- Zeki optimizasyon teknikleri kullanarak koloni optimizasyon davranışını uygular.

- Yapay Zekâ alanı açısından derin öğrenme algoritmalarını kullanarak örnek uygulamalar yapar.

Kitap içeriği Python programlama dili içerdiği için yapay zekâ odaklı konuların anlaşılması konusunda Python kodlama bilgi ve becerisi de gerektirmektedir. Kitap üzerinden öğretim ve öğrenim süreçlerine geçilmeden hemen önce Python ile ilgili temel eğitimler alınabileceği gibi, https://owncloud.tubitak.gov.tr/index.php/s/6wbNcqcaJh8h1OZ adresi altında sunulan 17 videoluk anlatım serisi ile Python kodlama temelleri öğrenilebilmektedir. Öğrencilerin özellikle ilgili video serisinden destek alarak, 1 haftalık bir Python ön-öğrenim süreci geçirmeleri kitapta anlatılanların daha iyi anlaşılması adına önemlidir. Yine videolar altında anlatılan örnek kodlar https://github.com/deneyapyz/pythonvideolari Web adresinde sunulmuş durumdadır.

Kitaptaki öğretim sürecinin etkin olması adına çeşitli Web unsurlarından ve özel içerik kutucuklarından faydalanılmıştır. Söz konusu bileşenlerin genel işlevleri kısaca şöyledir:

- **GitHub:** Güncel yazılım projelerinin ve geliştirici profillerinin paylaşıldığı GitHub platformunun öğrenciler tarafından önerilmesi önemli olduğu için kitap içeriğinde anlatılan yapay zekâ konularıyla bağlantılı kodlar ve veri setleri içerikte aktarılan GitHub linkleri altında kullanıma açılmıştır.
- **QR (Kare) Kodlar:** Öğrencilerin ders ve öğretim süreçlerine ilgilerini aktif tutmak ve yapay zekâ çerçevesinde daha etkin ve güncel fikirler oluşturabilmek adına çeşitli video linklerine bağlantılar sunulmuştur. Öğrenciler (ve hatta eğitmenler) ders süreçlerinde ilgili QR (kare) kodları okutup videolara erişebilecek, ders sırasında videoları tartışabilecek ya da ders harici zamanlarda videoları izleyerek güncel konulara dair kazanımlar edinebilecektir.
- **Eğitmene Not:** Eğitmene Not kutucukları, kitap içeriğinin öğrencilere aktarımı aşamasında eğitmenlere tavsiyeler ya da bazı kritik işlem adımlarını aktarabilmek adına kullanılmıştır.
- **Düşün, tartış…:** Düşün, tartış kutucukları ders sırasında yapay zekâ ile ilgili güncel konulardan hareketle sınıf içi ya da bireysel fikirlerin oluşturulup tartışılmasını sağlamak için kullanıma sunulmuştur. Bu kutucuklar altındaki sorular (tartışma konuları) özellikle öğrencilerin entelektüel bilgi düzeylerini güncel konularla harmanlamayı hedeflemektedir.
- **Biliyor musunuz?:** Biliyor musunuz kutucukları yapay zekâ ile ilgili güncel, önemli ve dikkate çekici konuları / gelişmeleri öğrenciler ile kısa notlar üzerinden paylaşmayı hedeflemiştir.
- **Dünyadan Haberler:** Dünyadan Haberler kutucukları, ders haftaları içerisindeki yoğun kodlama süreçlerini esnekleştirmek ve güncel yapay zekâ gelişmelerini öğrencilere kaynaklar üzerinden aktararak bilgi birikimi desteklemek adına sunulmuştur.

Proje Yarışması: Kitap ile sağlanan yapay zekâ eğitimini etkin bir proje içerisinde harmanlamak ve öğrencilerin elde ettikleri bilgi-becerilerini nihai bir çalışma altında, rekabetçi bir yönde işe koşmasını sağlamak için proje yarışması oluşturulmuştur. Detayları kitabın sonunda sunulmuş olan proje yarışması, iklim değişikliği (krizi) özelinde hedef bir veri seti (problem) için öğrencilerin özgür bir şekilde yapay zekâ çözümü üretmelerini ve ürettikleri çözümün taşıdığı niteliklere göre

puanlanmasını içeren prosedürleri kapsamaktadır. Yarışma süreci 7. hafta sonundan 8. hafta dersi 2 gün öncesine kadar sürmektedir.

Buraya kadarki açıklamalardan da hareketle, haftalık öğretim sürecinin şöyle bir akış içerisinde gerçekleştirilmesi önerilmektedir: Eğitmen öğrencilere ilgili haftaya ait konuyu teorik olarak detaylı biçimde aktarır. Sonrasında, öğrencilere ilgili haftanın veri seti, flash disk ortamında ya da çevrim içi indirme linkleri yoluyla verilir. Eğitmen tarafından öğrenciler ile birlikte Python programlama dilinde ilgili haftaya ait örnek uygulamayı kod olarak yazar ve yürütür. Eğitmen ve öğrenci birlikte uygulama kodlarından elde edilen sonucu değerlendirir. Bu noktada içerikte sunulan eğitmene not kutucukları ve diğer destekleyici kutucuklar da dikkate alınarak, öğretim-öğrenim süreçlerinin etkinliği artırılır.

Kaynakça

National Research Council. (2012). Education for life and work: Developing transferable knowledge and skills in the 21st century. National Academies Press.

Meb 2011, Bağlam Temelli Öğrenme-Öğretme Yaklaşımı. Available from: https://www.researchgate.net/publication/334959478_BAGLAM_TEMELLI_OGRENME-OGRETME_YAKLASIMI [accessed Jun 14 2021].

Jonassen, D. (2011). Designing for problem solving. In R. Reiser & J. Dempsey (eds.), *Trends and Issues in Instructional Design and Technology*, (pp. 64–74). Boston, MA: Pearson Education.

Hacer, T. O. R., & Erden, O. (2004). Ilkögretim ögrencilerinin bilgi teknolojilerinden yararlanma düzeyleri üzerine bir arastirma. TOJET: The Turkish Online Journal of Educational Technology, 3(1).

Cass, S. (2020). The top programming languages: Our latest rankings put Python on top-again-[Careers]. IEEE Spectrum, 57(8), 22-22.

1. Hafta: Python ile Yapay Zekâ

Ön Bilgi:

- Temel bilgisayar programlama ve algoritma bilgisi verilecektir.
- Python anlatım videoları 1. Python Dili Özellikleri, 2. Python Yukleme Kodlama Ort., 3. Python Kodlama İçin Spyder, 4. Veri Tipleri, 5. Veri Tipleri Dönüşümleri, 6. Atama Operatörleri ve Aritmetiksel Operatörler ve ayrıca 7. Karşılaştırma Op., Mantıksal Operatörler. 8. If Yapısı, 9. Giriş-Çıkış Komutları, 10.-11. Döngüler ve 12. Veri Yapıları-Veri Modelleri öğrenciler tarafından hafta öncesinde izlenmiş olmalıdır.

Haftanın Kazanımları:

- Öğrenciler "yapay zekâ" kavramını temel düzeyde açıklar.
- Öğrenciler algoritma kavramını açıklar.
- Öğrenciler basit bir yapay zekâ örneği oluşturabilir.
- Öğrenciler Python ile yapay zekâ temel kütüphanesi ile giriş seviyesinde uygulamalar yapar.
- Öğrenciler Github platformu ile yapay zekâ örnekleri üzerinden karar vermeyi öğrenir.

Haftanın Amacı:

Bu haftanın amacı, "yapay zekâ" kavramının tüm öğrenciler tarafından doğru bir şekilde anlaşılmasını sağlamaktır. Haftanın bir diğer amacı, Python ile örnek kodlar yazarak yapay zekâ uygulamaları ile öğrencilerin ilgisini çekerek etkinliklere yönelik ilgilerini artırmaktır. Ayrıca, öğrencilerin problem algılaması ve bu probleme ait yapay zekâ modelleme kavramının ne olduğunu tanımlaması ve basit bir veri organizasyonu oluşturabilmesi de hedeflenmektedir. Ders süresince kullanılacak Python editör ve temel yapay zekâ kütüphanelerini tanıtmak ve öğrencilerin bu kütüphaneleri kullanarak temel düzeyde yapay zekâ uygulamaları yapmaları hedeflenmektedir.

Kullanılacak Malzemeler:

Bilgisayar, kâğıt, kalem, Python kod editörü

Haftanın İşlenişi:

Algıla: Öğrenciler yapay zekâ kavramı üzerine tartışabilir ve problem çözüm şekillerini algılayabilir.

Tasarla: Öğrenciler yapay zekâda veri organize etme işlemlerini temel düzeyde gerçekleştirebilir.

Harekete Geç: Öğrenciler Python programlama dili ile eğitim ve test verilerini ayırma, model kurma gibi süreçleri gerçekleştirebilir.

Yürüt: Eğitmen, öğrencilerin Python ile model eğitme ve tahminleme işlemlerini gerçekleştirebilmeleri için uygulamalara birebir ve etkili rehberlik eder. Böylelikle öğrenciler de yapay zekâ model eğitme ve tahminleme çözümlerini uygulayabilir.

Karar Ver: Öğrenciler alternatif yapay zekâ modelini düzenleyerek / kodlayarak çözüm üretebilir.

1. ALGILA

1.1. Yapay Zekâ Kavramı Üzerine Tartışma

Eğitmen, öğrencilerin "yapay zekâ" kavramı üzerinde tartışmalarını sağlar. Eğitmen, öğrencilere aşağıdaki soruları yönelterek tartışmayı yönetir.

- Yapay zekâ nedir?
- Yapay zekâ bileşenleri nelerdir?
- Yapay zekâda problem çözüm şekilleri nelerdir?
- Yapay zekâ ile nerelerde karşılaşabiliriz?
- Robotlar yapay zekâ ile dünyayı ele geçirebilirler mi?
- Yapay zekâ insan beynini nasıl taklit ediyor?
- Yapay zekâ hayatımızı nasıl şekillendirecek?
- Python nedir?
- Python programlama dile ile yapay zekâ nasıl öğrenilir?
- Python için hangi temel yapay zekâ kütüphaneleri kullanılır?
- Program/modelleme nedir?

Soruların cevaplarının öğrencilerin bulması sağlanır. Eğitmen, yapay zekâ ve bileşenleri program ve modelleme kavramları için devam eden paragraflarda verilen tanımları dikkate alarak öğrencileri yönlendirebilir veya öğrenciler kavramı yanlış tanımladıklarında tanımlamaları doğrudan kendisi yapabilir.

Yapay zekâ, insan zekâsını modelleyebilmek adına insan gibi akıl yürütme, anlam çıkartma, genelleme yapabilme, geçmiş deneyimleri ile öğrenebilme gibi yetkinlikleri bir bilgisayara ya da makineye kazandırabilmektedir. *Oxford İngilizce Sözlük*'te ise yapay zekâ görsel algılama, konuşma tanıma, karar verme ve diller arasında çeviri gibi normalde insan zekâsını gerektiren görevleri yerine getirebilen bilgisayar sistemleri olarak tanımlanmaktadır. Her iki tanımda da vurgulanan nokta, yapay zekânın bir görevi gerçekleştirirken insan beyni gibi öğrenme fonksiyonunu gerçekleştiren bilgisayar sistemleri olmasıdır. Yapay zekâ ile insanlara göre daha akıllı çalışma yeteneğine sahip olup daha hızlı akıl yürütmekte ve daha doğru karar vermektedir (Yılmaz, 2021).

Yapay zekâ Şekil 1.1'de gösterildiği gibi veri seti, öğrenme algoritmaları ve karar verme süreci olmak üzere üç ana bileşenden oluşur. Yapay zekâda metin, görüntü verileri, zaman, uzunluk ölçümleri, video kayıtlarının düzenlenmesi ile veri seti oluşmaktadır. Öğrenme algoritmaları yapay sinir ağları, makine öğrenmesi, derin öğrenme gibi birçok alt daldan oluşmaktadır. Yapay zekâ öğrenme algoritmaları karmaşık bir yapıya sahip gibi gözükse de temelde algoritma ve programlamadan oluşmaktadır.

Şekil 1.1. Yapay zekâ bileşenleri

Algoritma: Bilgisayarlarda bir problemin çözümünde izlenecek yol genel tanımıyla algoritma olarak adlandırılır. Çocuklar günlük hayatta kullandıkları tabletlerden, oynadıkları oyunlara, yaşamlarının her alanında algoritmaları kullanmaktadır. Algoritma mantığı veya bir problemi adım adım çözebilme yeteneği özellikle fen ve matematik gibi sayısal derslerde oldukça önemlidir. Örneğin matematik dersinde bölme, çarpma ve çıkarma gibi işlemleri yaparken algoritmalardan faydalanılır. Çocuklar, problemleri bölümlere ayırma ve çözüme ulaşabilmek için yapmış oldukları işlem basamakları algoritmik düşünceye örnek olarak verilebilir.

Program: Yapay zekâ uygulamaları için algoritma adımları bilgisayar tarafından gerçekleştirilen program kodlarının bütünüdür. Programlar yapay zekâ uygulamalarının bilgisayar tarafından yapılması için gereken adımların bir programlama dili aracılığıyla aktarılmasıdır. Öğrenciler de bu derste yapay zekâ uygulamaları geliştirmek için Python programlama dili kullanacaktır.

Düşün, tartış…

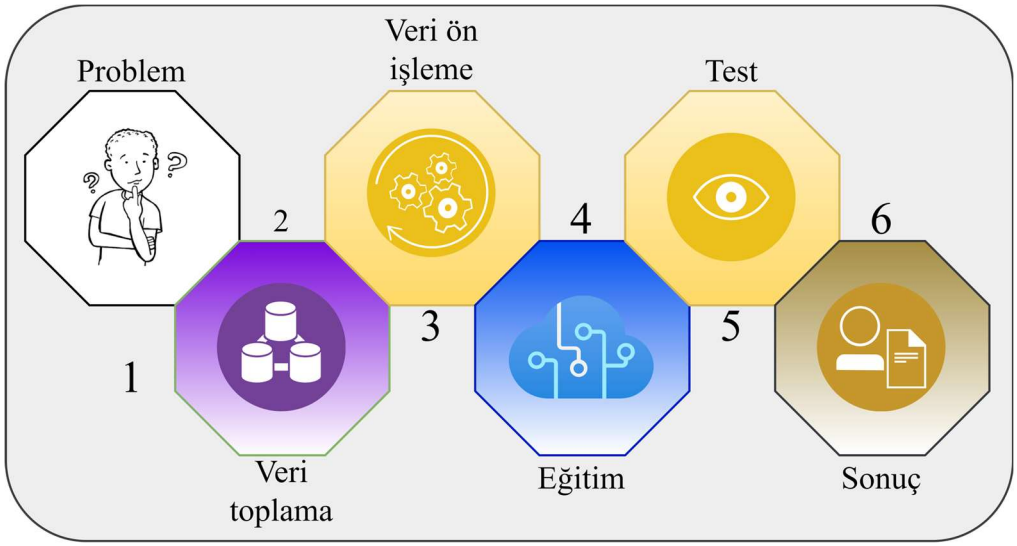
İnsanlar teknolojiye neden ihtiyaç duymaktadır? Yapay Zekâ'nın buradaki rolü nedir; neden bu kadar önemli ve popüler olmuştur? Öğrenciler olarak bu konudaki bireysel düşüncelerinizi notlandırabilir; ardından sınıf ortamında tartışabilir ve / veya Web ortamındaki güncel tartışmalar ile düşüncelerinizi karşılaştırabilirsiniz.

1.2. Yapay Zekâda Problem Modelleme

Yapay zekâda problemin modellemesinde kullanılan birçok otomatik karar verme mekanizması vardır. Yapay zekâ modelleri ile veri girişi sağlandığında uzman bir kişinin vereceği kararı "eğitilmiş" matematiksel algoritmalar yapar. Ayrıca yapay zekâ modelleri karar sürecinin yorumlanmasına yardımcı olurlar. Yapay zekâ modelleri, sistemin en doğru kararı vermesi veya maliyeti en aza indirmesi için büyük miktarda veriyi işlemektedir. Farklı ortamlarda gelen verileri

analiz eden yapay zekâ modeli tüm verileri gözden geçirerek uzmanlardan oluşan bir ekibin yapacağı belirli bir karar sürecini tek başına gerçekleştirir. Yapay zekâ modellemesi veri akışı Şekil 1.2'de gösterildiği gibi şu şekildedir:

- **Problem:** Öğrenciler yapay zekâ çözümleri için problemin sınıflandırma, regresyon veya kümeleme türünü belirler. Örneğin kedi ve köpek verilerinden oluşan veri setinin sınıflandırma problemi olduğunu, yaya sayısına göre trafik ışıklarının süresinin belirlenmesinin bir regresyon problemi olduğunu analiz eder.
- **Veri Toplama:** Öğrenciler problemin çözümü ile ilgili açık erişimli internet sitelerinde yer alan (Kaggle, Github vs.) veya kendi toplamış oldukları verileri bir araya getirir.
- **Veri Ön işleme:** Bu aşamada, toplanan veriler üzerinde eksik verileri tamamlama, anlamsız verileri çıkarma gibi veri önişleme aşamaları gerçekleştirilerek yapay zekâda oluşabilecek problemlerin önlenmesi sağlanır. Veri ön işlemede ilk olarak veriler küçükten büyüğe veya anlamlı olacak şekilde sıralanmalıdır. Ardından aykırı veri tespitinde veya gürültülü verilerin tespitinde kümeleme algoritmalarından faydalanabilir. Yine eksik verilerin tamamlanması aşamasında ise regresyon gibi yöntemler kullanılabilir (Web Kaynağı 1.2)
- **Eğitim:** Veri önişleme sonrasında anlamlandırılan verilerden eğitim için ayrılan veriler yapay zekâ modelleri ile eğitir.
- **Test:** Eğitilen modellerin doğruluğu test verileri ile değerlendirilerek anlamlı sonuçlar elde edilmeye çalışılır.
- **Sonuç:** Test edilen veriler üzerinde en doğru sonuç veren modeli seçer.



Şekil 1.2. Yapay zekâ problem modelleme

Eğitmene Not

Eğitmen, öğrencilerin Yapay Zekâ yönündeki mesleki eğilimlerini sorar; farklı meslek eğilimleri gösteren öğrencilere Yapay Zekâ'nın mesleklerindeki olası etkilerini sorarak sorgulamasını ister.

Biliyor musunuz?

Yapay Zekâ'nın en etkili kullanıldığı alanlardan biri de sağlıktır. Sağlık alanında hastalıkların erken teşhisinde, tedavi planlamasında ve hatta ilaç keşfinde Yapay Zekâ yaygın bir şekilde kullanılmaktadır. Yapay Zekâ bu alanda doktorlarla yarışmaktadır!

1.3. Python ile Yapay Zekâ İşlemleri

Python, kolay öğrenebilen, nesne tabanlı uygulamaları ile oldukça sık tercih edilmektedir. Python programlama dilinde yapay zekâ işlemede genellikle Numpy, Matplotlib, Scipy, Scikit-Learn, TensorFlow, Keras, Pytorch kütüphaneleri kullanılmaktadır. Yapay zekâ modellemesi için yapay zekâ kütüphanelerin kısa açıklaması şöyledir:

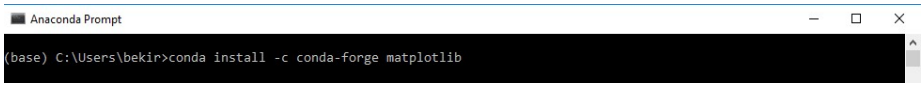
Numpy: Hesaplama işlemleri, diziler ve diziler üzerinde hızlı işlemler yapılabilmesi için kullanılan önemli kütüphanelerden birisidir (Şekil 1.3).

Şekil 1.3. Anaconda Spyder için Numpy kütüphanesinin kurulumu

Matplotlib: Görüntüleri yüksek kalitede gösterebilmek için kullanılan bir çizim kütüphanesidir (Şekil 1.4).

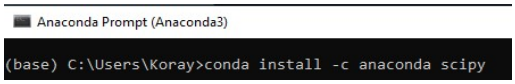
Şekil 1.4. Matplotlib kütüphanesi ile ilgili örnek grafikler

Anaconda Spyder'da Matplotlib kütüphanesini yüklemek için anaconda prompt ekranında Şekil 1.5'te görüldüğü gibi "conda install -c conda-forge matplotlib" komut satırının yazılması gerekmektedir (Şekil 1.5).



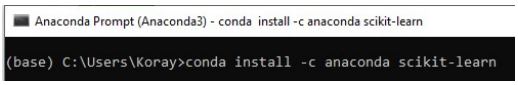
Şekil 1.5. Anaconda Spyder için Matplotlib kütüphanesinin kurulumu

Scipy: Veri analizinde kümeleme, regresyon (tahmin), veri işleme gibi işlemleri gerçekleştirebilen çok yönlü bir kütüphanedir (Şekil 1.6).



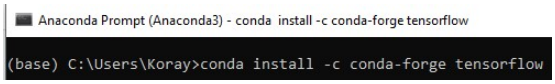
Şekil 1.6. Anaconda Spyder için Scipy kütüphanesinin kurulumu

Scikit-Learn: Makine öğrenme modelleri oluşturmak için kullanılan bir kütüphanedir. Regresyon (tahminleme), kümeleme ve sınıflandırma için kullanılan pek çok öğrenme algoritmasına sahiptir (Şekil 1.7).



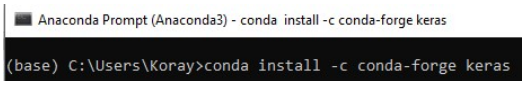
Şekil 1.7. Anaconda Spyder için Scikit-Learn kütüphanesinin kurulumu

TensorFlow: Google tarafından geliştirilen açık kaynaklı kodlu bir derin öğrenme kütüphanesidir (Şekil 1.8).



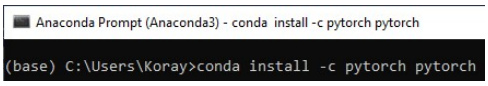
Şekil 1.8. Anaconda Spyder için TensorFlow kütüphanesinin kurulumu

Keras: Derin sinir ağları ile hızlı eğitim yapabilmek için tasarlanan açık kaynak kodlu bir sinir ağı kütüphanesidir (Şekil 1.9).



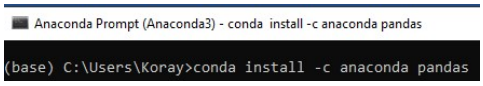
Şekil 1.9. Anaconda Spyder için Keras kütüphanesinin kurulumu

Pytorch: Derin öğrenme modellerinde esneklik ve hız ile geliştiricilerin işlerini oldukça kolaylaştıran bir kütüphanedir (Şekil 1.10).



Şekil 1.10. Anaconda Spyder için Pytorch kütüphanesinin kurulumu

Pandas: Tablosal veri işlemesi ve analizi için kullanılan Python temel kütüphanesidir (Şekil 1.11).



Şekil 1.11. Anaconda Spyder için Pandas kütüphanesinin kurulumu

1.4. Python ile Yapay Zekâ Veri İşleme

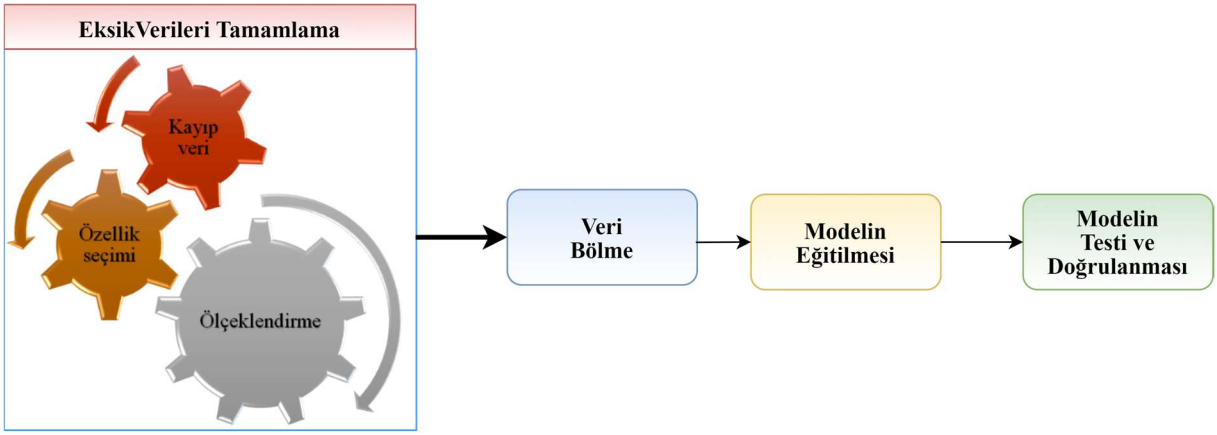
Yapay zekâda veri işleme hem makine öğrenmesi hem de derin öğrenme modellerinden doğru bir biçimde yararlanmak için kullanılan bir tekniktir. Makine öğrenimi yapay zekânın bir alt kümesidir, bilgisayarların verilerden anlamlı sonuçlar elde edilmesini sağlayan önemli tekniklerden birisidir. Derin öğrenme ise, beyindeki sinir ağlarını örnek alarak çalışan karmaşık sorunların çözülmesini sağlayan makine öğreniminin alt kümesidir. Yapay zekâda toplanan veriler üzerinde; eksiklik, gürültü (yanlış veri) ve tutarsızlık gibi farklı nedenlerden dolayı veri işleme problemleri yaşanabilir. Yapay zekâ veri işleme aşamaları Şekil 1.12'de gösterildiği gibi şu şekilde sıralanmaktadır:

Eksik Verileri Tamamlama: Yapay zekâ veri işleme sürecinde ilk aşamada toplanan verilerde eksik değerler bulunur. Eksik verileri tamamlamak için öznitelik oluşturma, sınıflandırma, ölçeklendirme-normalize etme ve uç verileri tespit etme yöntemleri kullanılır.

Veri Bölme: Yapay zekâda veriler eğitim ve test olmak üzere ikiye ayrılır. Eğitim verisi, modelin eğitildiği verileri, test verisi ise modelin eğitilmeyen veriler üzerindeki sonuçlarını görmek için kullanılır.

Modelin Eğitilmesi: Yapay zekânın doğru bir tahmin yapabilmesi için temizlenmiş verilerin eğitilmesi gerekir.

Modelin Testi ve Doğrulanması: Test verileri ile eğitilen model doğrulanarak yapay zekâ modeli değerlendirilir.



Şekil 1.12. Yapay Zekâ ile Veri Analizi

Dünyadan Haberler

Dubai'de şirketler pandemi sonrası yapay zekâyı daha insancıl kılma arayışında.

Dijital teknolojiye küresel bağımlılık hızla büyüdükçe, markalar daha insansı teknolojiyle tüketicilere nasıl daha iyi ulaşabileceklerini bulma arayışında. Dubai'deki girişimler, kullanıcılarının güvenini kazanmak için yapay zekâyı "etik veriler" olarak adlandırdıkları yöntemle eğitiyor.

Pandemi ile gelen sokağa çıkma yasakları teknolojiye ne kadar ihtiyacımız olduğunu bir kez daha gözler önüne serdi. Bu da dijital ortamda alışveriş yapan ve sayıları hızla artan kişilerin nasıl daha iyi ağırlanacağı konusunda markaları çözümler üretmeye itti.

Orta Doğu'daki tüketicilerin ihtiyaçlarını internetten karşılama oranı gün geçtikçe artıyor. Oysa 2020 başından beri tekstil sektörünün satışları sert bir düşüş yaşadı. Bazı şirketlere göre bunun sebebi internet üzerindeki işlemlerin insani yanının olmaması. Dubai merkezli Getbee adlı şirket ise "*kişiselleştirilmiş alışveriş platformu*" adını verdiği oluşumugeliştirerek, markaların müşterileri ile daha iyi bağlantıya girmelerini sağlıyor. Müşteriler seçtikleri temsilciler sayesinde ürünler hakkında bilgi alabiliyor ve yönlendirilebiliyor.

Şirketin tepe yöneticisi Thea Myhrvold platformlarını yaklaşık 20 markanın kullandığını ve küresel moda gelirlerinin düşüşe geçtiği 2020'de kendilerinin online satışlarının yüzde 28 arttığını söylüyor (Web Kaynağı 1.3)

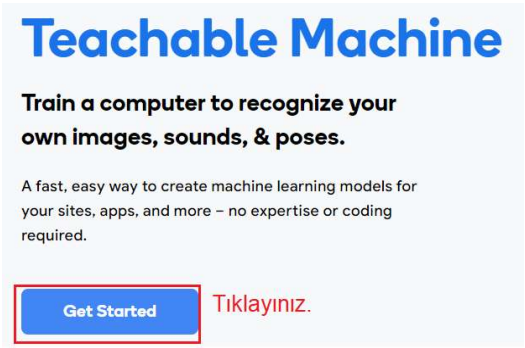
Eğitmene Not

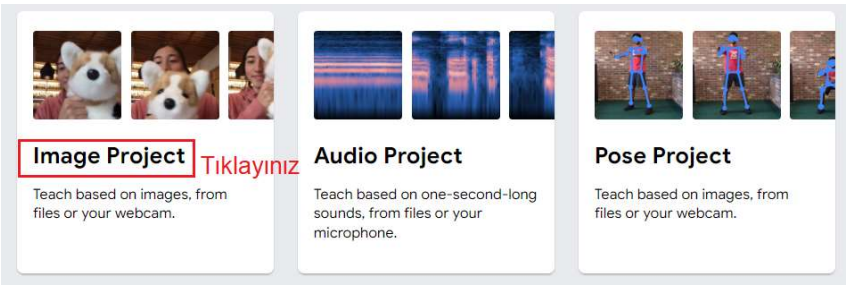
Eğitmen, kitap başlangıcında ifade edilen; iklim değişikliği odaklı proje yarışması konusundaöğrencileri bilgilendirir. Bu noktada, kitap sonunda yer alan açıklamaları dikkate almak suretiyle genel bir bilgilendirme yaparak öğrencileri proje yarışması konusunda ilk haftadan itibaren motive eder. Genel motivasyon iklim değişikliği (krizi) konusunun ciddiyetini ve yapay zekanın bu yöndeki bilimsel araştırmalar konusundaki çözüm potansiyelini irdelemek üzerine olmalıdır (Proje yarışmasının amacı ve genel yapısı hakkında bilgiler için kitap başlangıcındaki öğretim kılavuzu tekrar incelenmelidir).

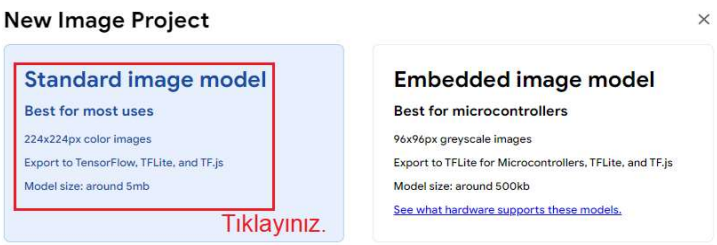
1.5. Eğitilebilir Yapay Zekâ Platform ile Taş Kâğıt Makas Oyununun Modellenmesi

Öğrenciler aşağıda verilen linki tıklayabilir.

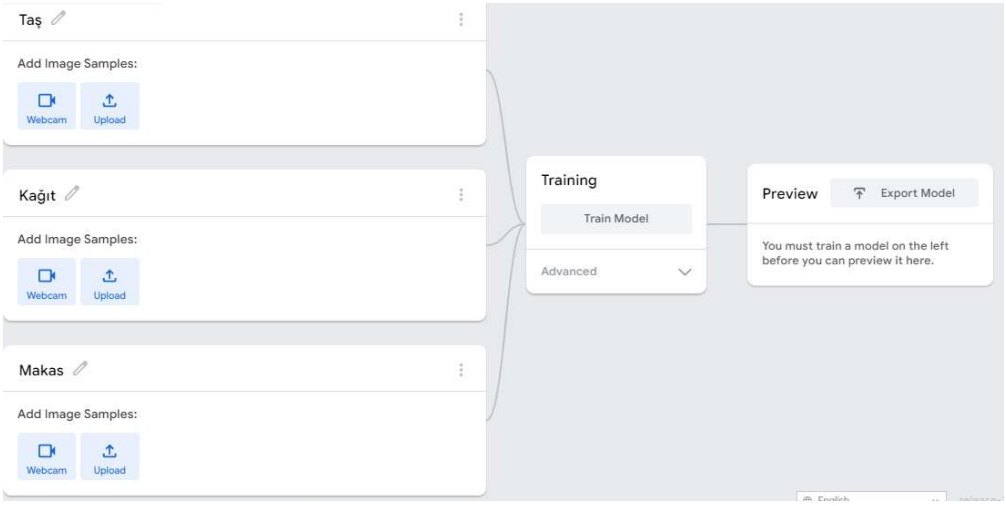
https://teachablemachine.withgoogle.com/



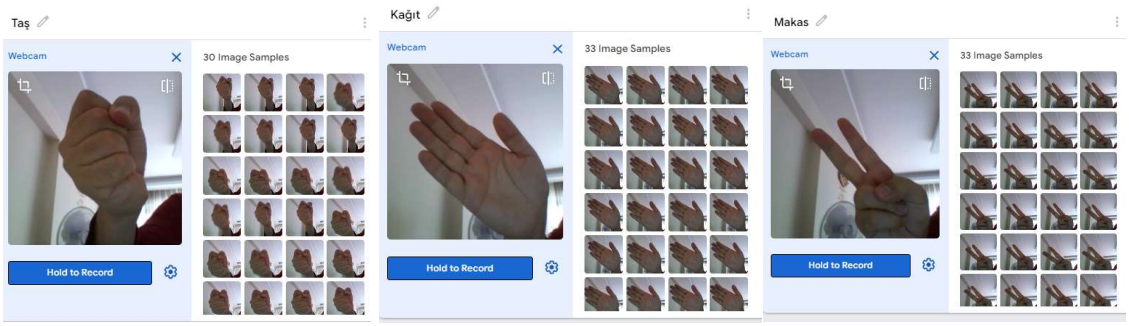




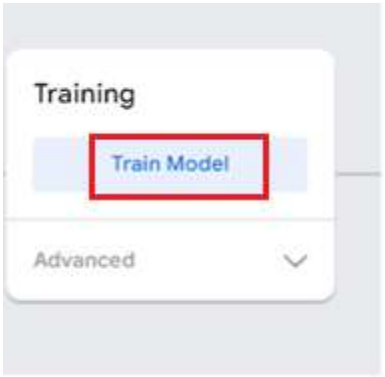
Öğrenciler class isimlerini “Taş”, “Kâğıt” ve “Makas” olarak isimlendirir.



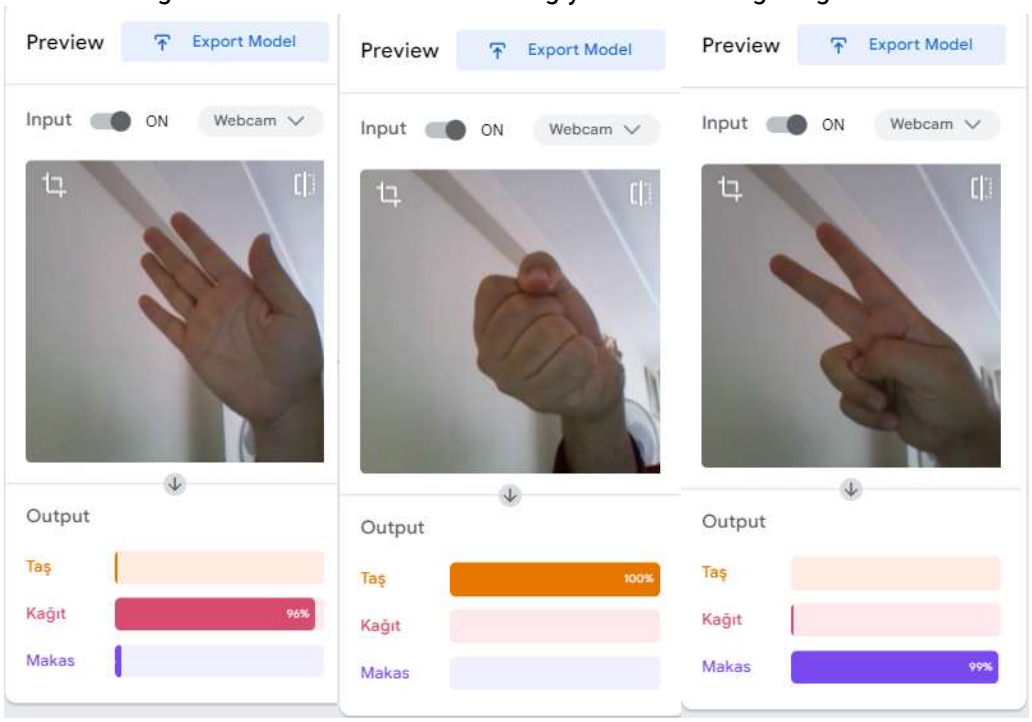
Öğrenciler “hold to record” butonuna basarak taş, kâğıt ve makas oyunu için görüntü veri seti oluşturur.



Öğrenciler “train model” butona basarak görüntü veri setinin eğitim işlemini gerçekleştirir.



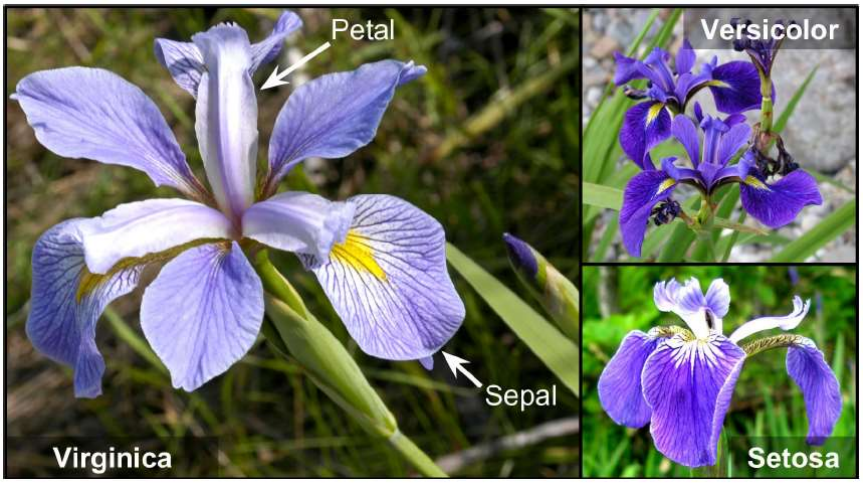
Öğrenciler modeli eğittikten sonra web cam aracılığıyla modelin doğruluğunu test eder.



2. TASARLA

2.1. Veri Setini Öğreniyorum

Sınıftaki öğrencilere Şekil 1.13'te görüldüğü gibi iris bir çiçeğin üç türüne *(setosa, versicolor, virginica)* ait fotoğraf gösterilir. Daha sonra üç türe ait 50'şer tane, toplamda 150 tane olmak üzere üst ve alt çiçek yaprakları ölçülmüş veri seti verilir. Bu ölçümden dört nitelikli *"alt yaprak uzunluğu" cm, "alt yaprak genişliği" cm, üst yaprak genişliği" cm, "üst yaprak uzunluğu" cm* ve 150 elemanlı bir veri seti gösterilir.



Şekil 1.13. İris çiçeği ve türleri (O'Reilly, 2021)

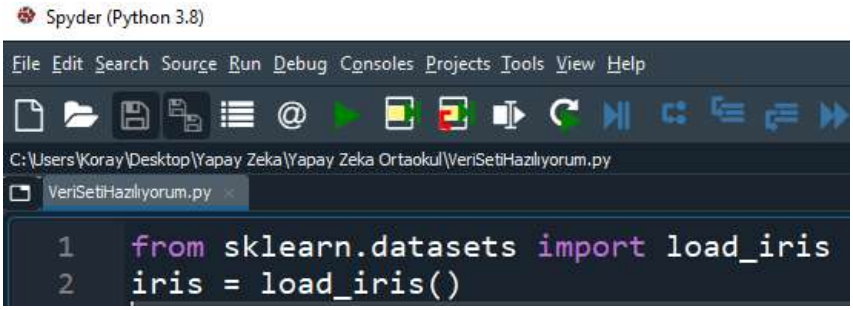
Her bir öğrenci Çizelge 1.1'de verilen iris çiçeğine ait yaprak ölçülerini ve türünü inceler ve karşılaştırır. Öğrenci veri seti tablosundaki alt ve üst yaprak uzunluğunu-genişliğini giriş parametresi olarak isimlendirir. Ölçümlerin değerlerine göre iris çiçeği yaprak türlerini ise çıkış parametresi olarak isimlendirir. Öğrenci veri setindeki ondalıklı sayılarda virgül yerine noktanın kullanıldığını öğrenir.

Çizelge 1.1. İris çiçeği türlerinin yaprak ölçüm değerleri

Örnek Numara	Alt yaprak uzunluğu (cm)	Alt yaprak genişliği (cm)	Üst yaprak uzunluğu (cm)	Üst yaprak genişliği (cm)	Tür
1	5.1	3.5	1.4	0.2	Setosa
2	4.9	3.0	1.4	0.2	Setosa
…	…	…	…	…	…
51	7.0	3.2	4.7	1.4	Versicolor
52	6.4	3.2	4.5	1.5	Versicolor
…	…	…	…	…	…
101	6.3	3.3	6.0	2.5	Virginica
102	5.8	2.7	5.1	1.9	Virginica
…	…	…	…	…	…
150	5.9	3.0	5.1	1.8	Virginica

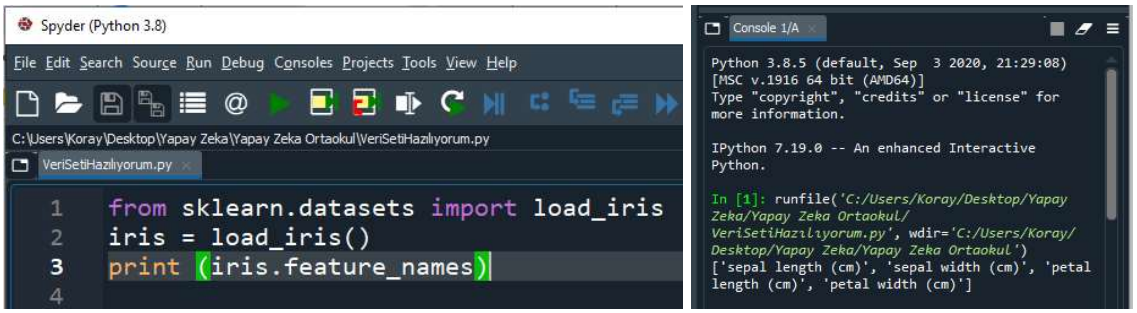
2.2. Python ile Veri Seti Hazırlıyorum

Öğrenci Python scikit-learn kütüphanesinin içnde hazır olarak yer alan iris veri setini kullanarak basit bir sınıflandırma yapar. Tasarım aşamasında iris çiçeğinin alt ve üst yaprak uzunluk ve genişlik verilerini kullanarak çiçeğin türünü sınıflandırmaya çalışır. Şekil 1.14'te gösterilen gerekli kütüphane ve veri setini indirir.



Şekil 1.14. Gerekli kütüphaneleri ve veriyi alma

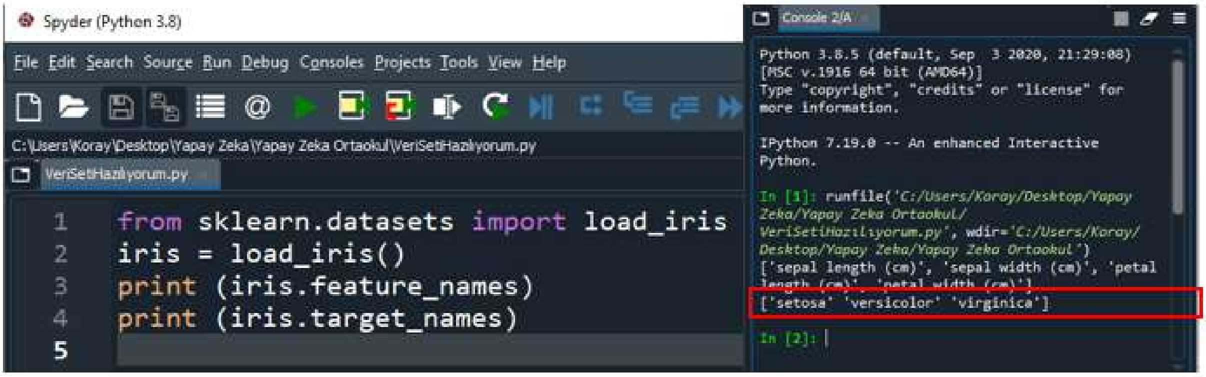
Dikkate alınan veri ön işlemi yapılmış durumdadır. Burada veriler sklearn kütüphanesinden hazır olarak alınmıştır. Birden çok değer alan ve değişim gösteren her şeye değişken adı verilir. Yapay zekâda sonuca etki eden değişkene bağımsız değişken, başka bir değişkene bağlı olan yani etkilenen değişkene bağımlı değişken denir. Daha sonra bağımlı/bağımsız hedef değişkenin değeri kategorik sınıftan sayısala çevrilmiştir. Sklearn kütüphanesinden alınan veriyi daha iyi anlayabilmek için verimizdeki bağımsız değişkenlerin (nitelikler) isimleri Şekil 1.15a'da verilen kod satırı kullanılarak elde edilir. Şekil 1.15b'de dört bağımsız değişkenin isim listesi görülmektedir.



(a) (b)

Şekil 1.15. Veri analizi a) bağımsız değişkenleri görüntüleme kodu b) bağımsız değişkenler görüntüsü

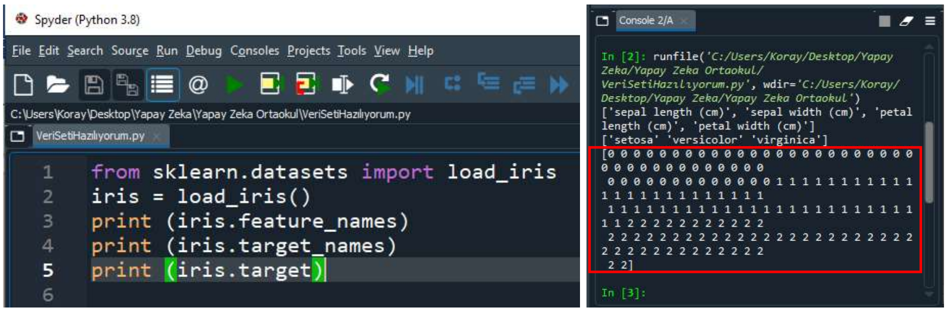
Sklearn kütüphanesinden alınan veriyi daha iyi anlayabilmek için verimizdeki bağımlı değişkenlerin isimleri Şekil 1.16a'da verilen kod satırı kullanılarak elde edilir. Şekil 1.16b'de bağımlı değişkenlerin isimleri listesi görülmektedir.



(a) (b)

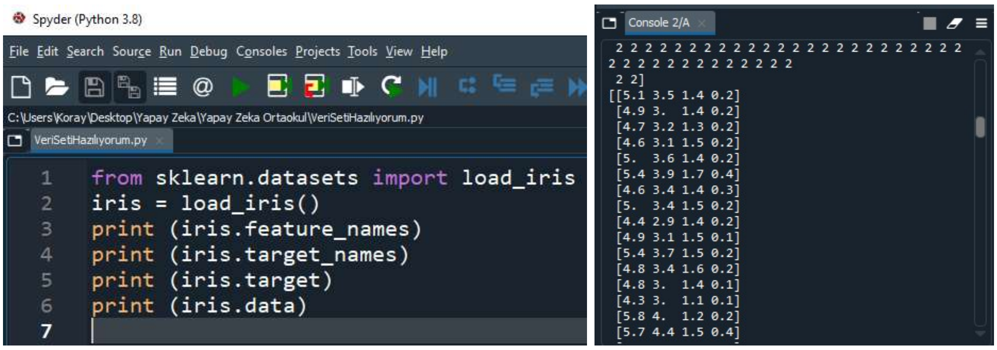
Şekil 1.16. Veri analizi a) bağımlı değişkenleri görüntüleme kodu b) bağımlı değişkenler görüntüsü

Öğrenciler iris çiçek türüne ait üç farklı sınıfı listeler. Şekil 1.17'de bağımlı değişkenlerimiz olan setosa için "0", versicolor için "1" ve virginica için "2" ile kodlanarak sayısallaştırılmıştır. Şekil 1.17a ve 1.17b'de iris veri kodu, Şekil 1.18a ve 1.18b'de bağımsız değişkenlerin sayısal görüntüsü verilmiştir.



(a) (b)

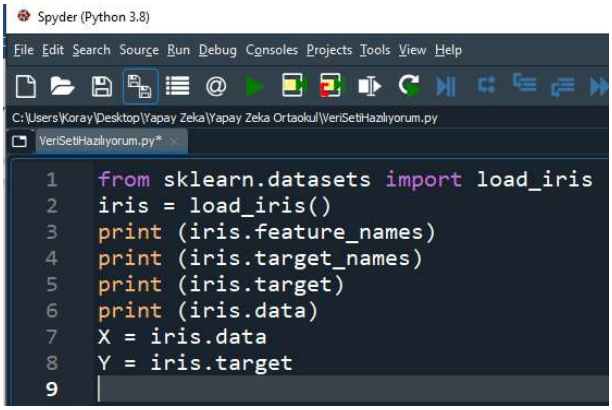
Şekil 1.17. Veri analizi a) bağımlı değişkenleri sayısallaştırma kodu b) görüntüsü



(a) (b)

Şekil 1.18. Veri analizi a) bağımsız değişkenleri sayısallaştırma kodu b) görüntüsü

Veri seti hazırlıyorum son aşamasında öğrenciler bağımlı ve bağımsız değişkenleri oluşturmak için bağımsız nitelikleri X, bağımlı niteliği Y değişkenine atar.



Şekil 1.19. Bağımlı ve bağımsız değişkenleri X ve Y değişkenine atama kod satırı

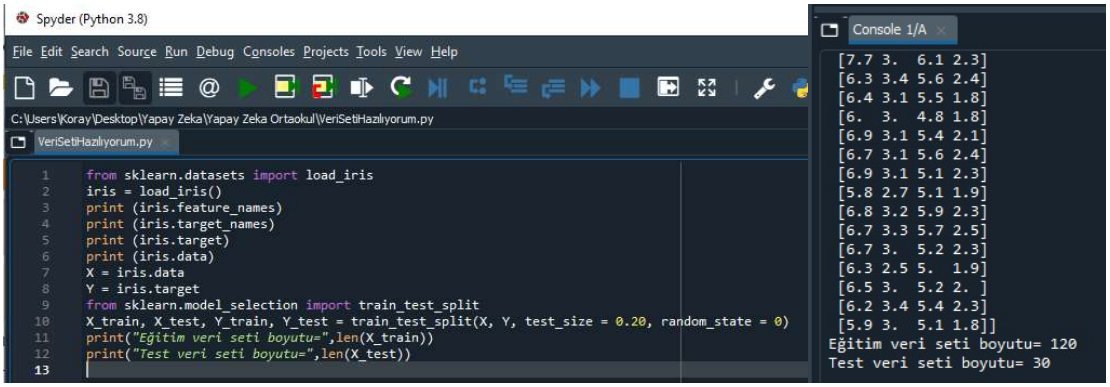
3. HAREKETE GEÇ

3.1. Eğitim ve Test Verilerini Ayırma

Öğrenciler iris çiçeği türüne ait verilerin %80'ini eğitim için, %20'sini ise test için ayırır. Şekil 1.20a'da gösterildiği gibi veri setlerini ayırmak için Python kodalarını yazar. Şekil 1.20b'de ekran görüntüsü verilmiştir.

Eğitmene Not

Eğitmen, öğrencilere veri setini eğitim ve test verileri olarak ayırmayı gösterirken eğitim veri setinin test veri setinden daha büyük olması gerektiğini belirtir. Eğer test ve eğitim veri setiyüzdesel olarak birbirine yakın olursa modelden elde edilen sonuçların yanlış olabileceğini belirtir. Bu nedenle eğitim veri seti genellikle %75'den başlayarak %80, %85'e kadar ayrılırken test veri seti ise %15 ile %25 arasında değişir. Modelde eğitim ve test verileri yüzdeleri kullanılan veri setine bağlı olarak değişebilmektedir.

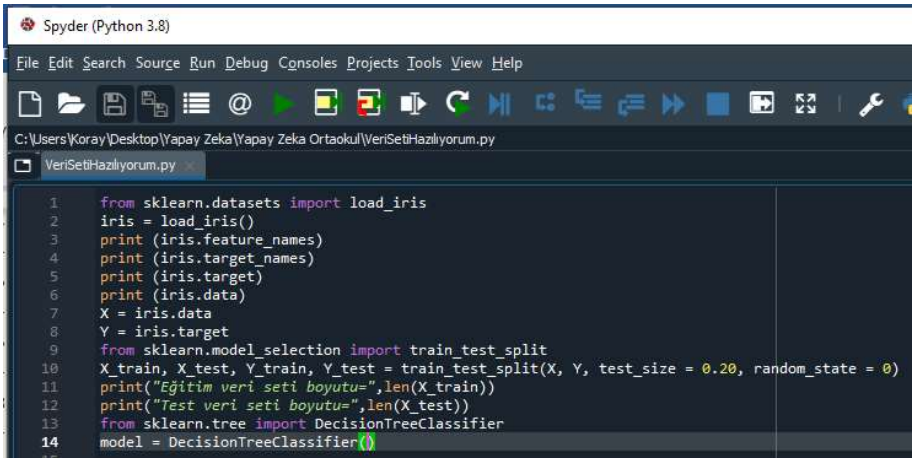


(a) (b)

Şekil 1.20. Eğitim ve test verileri ayırma a) kod b) görüntüsü

3.2. Model Kurma

Öğrenciler model için scikit-learn kütüphanesinden karar ağaçları sınıflandırıcısını çağırır ve model adında bir değişkene aktarır. Şekil 1.21'de verilen karar ağaçlarına ait algoritmanın Python kodunu editöre yazar.



Şekil 1.21. Karar ağaçları ait algoritmanın kodu

Ağaç yapısında dallardaki birleşme noktaları (düğümler) bir özelliğe (özniteliğe) tekabül etmekte, her bir dal ise bir kararı işaret etmektedir. Karar ağacında düğümler, sınıflandırılacak gruptaki özellikleri temsil eder ve dallar ise düğümlerin alabileceği değerleri temsil eder.

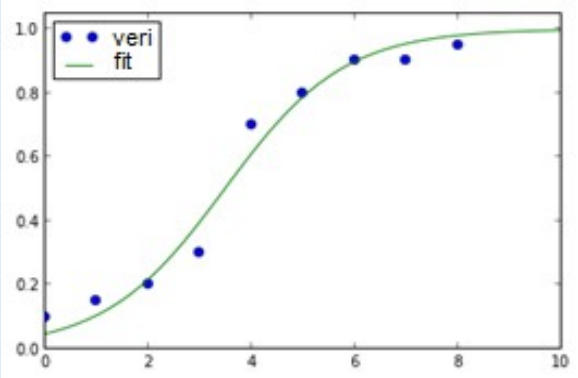
4. YÜRÜT

4.1. Modeli Eğitme

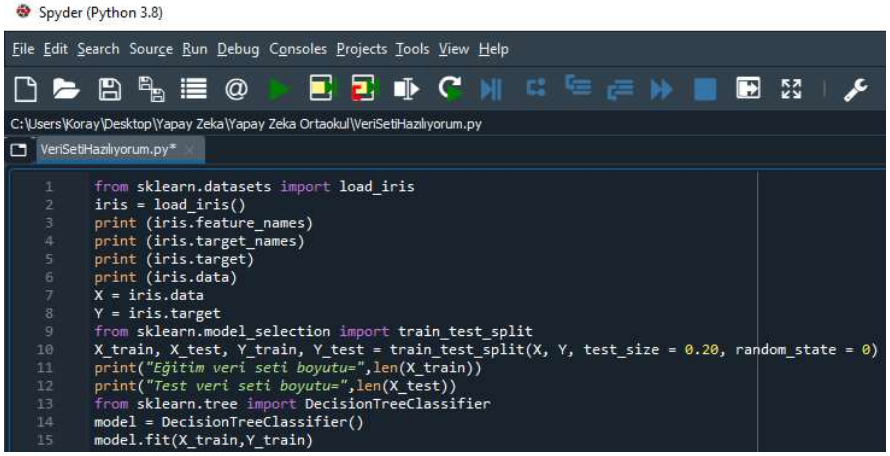
Öğrenci, Şekil 1.22'de gösterildiği gibi Python programlama editöründe iris çiçeği türlerine ait eğitim verilerinin **model.fit** komutu ile eğitimini gerçekleştirir.

Eğitmene Not

Eğitmen, öğrencilere Şekil 1.23'te gösterildiği gibi, eğri üzerinden .fit komutunu anlatır. Fit eğrisi üzerine verileri mümkün olan en yakın noktaya yerleştirmek için model eğitilir.



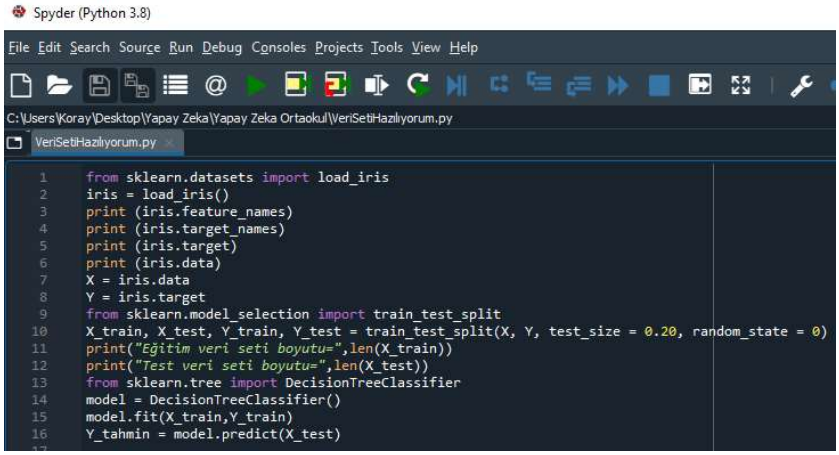
Şekil 1.22. Fit eğrisi



Şekil 1.23. Modeli eğitme

4.2. Model Tahmini

Öğrenci, Şekil 1.24'te gösterildiği gibi Python programlama editöründe iris çiçeği türlerine ait test verilerinin **model.predict** komutu ile giriş verilerine göre çiçeğin hangi tür çiçek sınıfına ait olduğunu tahmin eder.



Şekil 1.24. Modeli tahmin etme

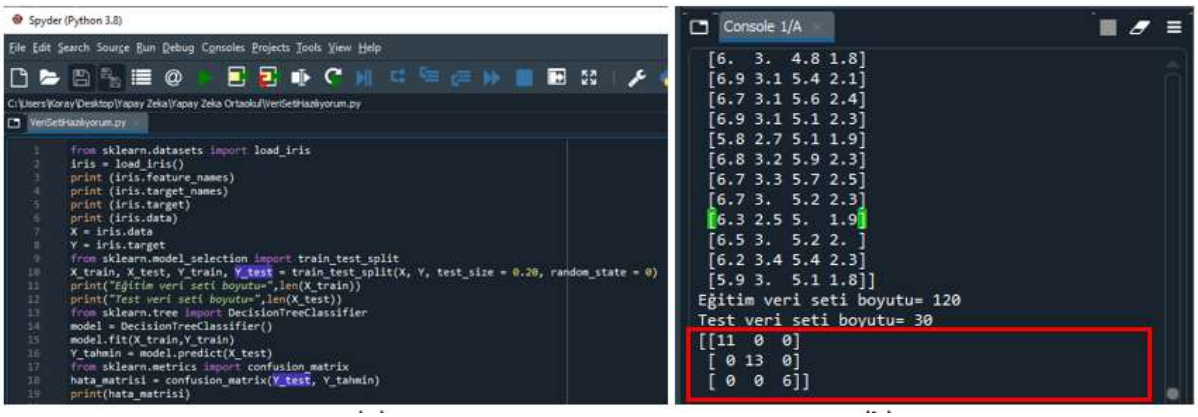
Eğitmene Not

Eğitmen, Makine Öğrenmesi algoritmalarında eğitim ve test süreçlerinin her zaman için uygulandığını ve bu süreçlerin gerekliliğini yeri geldikçe vurgular; öğrencilerin bu yönde farkındalığını geliştirir.

5. KARAR VER

5.1. Tahmin-Test Sonuçlarını Karşılaştırma

Öğrenciler, https://github.com/deneyapyz/lise/ adresi altındaki Hafta1 klasöründe yer alan iris.xlsx dosyasını indirip inceler. Burada amaç, karar ağaçları modeli ile eğitilen iris çiçeği sınıfını hata matrisi (confusion matrix) kullanarak modelin başarısını ölçmektir. Hata matrisi yapay zekâ sınıflandırmaları için performans ölçer. Şekil 1.25a'daki Python programlama editörüne hata matrisine ait kodları yazar. Şekil 1.25b'de ise hata matrisine ait sonucu görür.

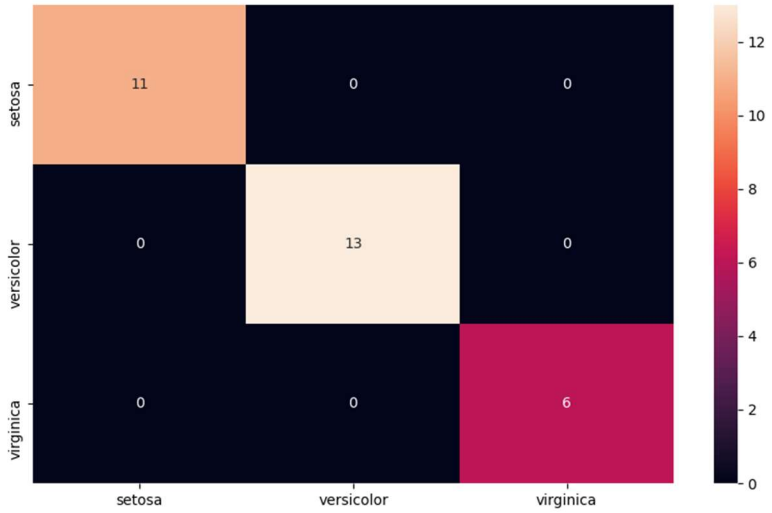


(a) (b)

Şekil 1.25. Hata matrisi a) kod b) sonuç görüntüsü

5.2. Hata Matrisini Değerlendirme

Öğrenciler, iris çiçeğine ait toplam 150 kayıt verinin %80'ini (120) eğitim, %20'i (30) test eğitim seti olarak ayırmıştı. Karar ağaçları modelini 120 kayıt ile eğitmişti. Geriye kalan 30 kayıt (X_test) ise modeli tahmin etmek için kullanıldı (Y_tahmin). Öğrenciler, Şekil 1.26'de gösterildiği gibi 30 test kaydına ait gerçek sonuçlar (Y_test) ile tahmin sonuçlarını karşılaştırdı ve hata matrisi elde etti. Öğrenci matristeki sayıların toplamının test verisi olan 30'a eşit olduğunu gördü.



Şekil 1.26. Hata matrisi

Öğrenciler, hata matrisini incelediğinde 30 adet test kaydında Setosa sınıfına ait 11 tane kayıt var olduğu ve hepsinin doğru tahmin edildiğini görmüştür. Ayrıca 13 tane versicolorve 9 tane virginica var olduğunu ve bunların hepsinin de başarıyla virginica olarak doğru tahmin edildiğini görmüştür.

Biliyor musunuz?

İris çiçeği için kullanılan çözüm adımlarını, atmosfer olaylarının tahmininde, hastalık teşhisinde veya bir müşterinin satın alabileceği ürünün tahmininde kullanabilirsiniz. Tek yapmanız gereken problemi modelleyip veri setini hazırlamak!

Biliyor musunuz?

Türkiye'nin en önemli bilim insanlarından birisi de dünyaca ünlü matematikçi, birçok matematiksel kavramın mucidi **Ord. Prof. Dr. Cahit Arf**'tır.

Cahit Arf, dünyanın daha tanıştığı Yapay Zekâ teknolojisini düşünerek, 'Makine Düşünebilir mi ve Nasıl Düşünebilir?' adlı çalışmasıyla Yapay Zekâ'nın gücünü 20. Yüzyıl ortalarında tartışmaya açmıştır. Söz konusu çalışma Atatürk Üniversitesi Üniversite Çalışmalarını Muhite Yayma ve Halk Eğitimi Yayınları Konferanslar Serisi altında yayınlanmıştır. Bu bakımdan Prof. Arf, Yapay Zekâ ve düşünen makine konusuna yönelik gösterdiği öngörüler bakımından dünyadaki sayılı bilim insanları arasındaki yerini almıştır.

Ülkemizin önemli değerlerinden olan Prof. Arf, günlük hayatın koşturması arasında dikkatinizden kaçmış olabilir. Hemen 10 TL'lik kâğıt banknotun arkasına bakabilirsiniz!

Prof. Arf'ın bilim dünyasına kattığı önemli kavramlar hakkında bilgi sahibi olmak için şu anahtar kelimeleri Web'te aratabilirsiniz: Arf Değişmezleri, Arf Halkaları, Arf Kapanışı.

6. UYGULAMANIN PYTHON KODLARI

```
from sklearn.datasets import load_iris
iris = load_iris()
print (iris.feature_names)
print (iris.target_names)
print (iris.target)
print (iris.data)
X = iris.data
Y = iris.target
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.20, random_state = 0)
print("Eğitim veri seti boyutu=",len(X_train))
print("Test veri seti boyutu=",len(X_test))
from sklearn.tree import DecisionTreeClassifier
model = DecisionTreeClassifier()
```

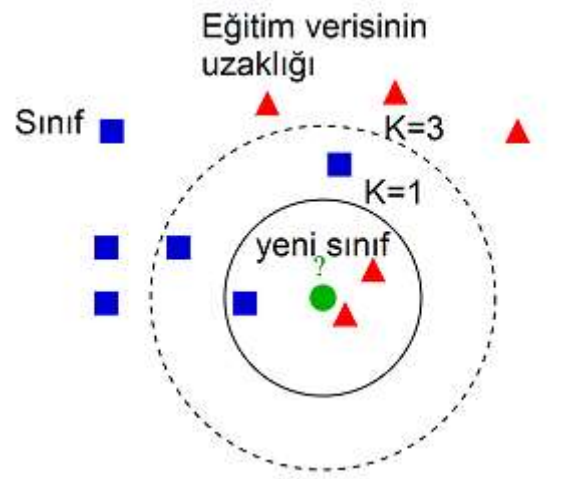
```
model.fit(X_train,Y_train)
Y_tahmin = model.predict(X_test)
from sklearn.metrics import confusion_matrix
hata_matrisi = confusion_matrix(Y_test, Y_tahmin)
print(hata_matrisi)
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt
index = ['setosa','versicolor','virginica']
columns = ['setosa','versicolor','virginica']
hata_goster = pd.DataFrame(hata_matrisi,columns,index)
plt.figure(figsize=(10,6))
sns.heatmap(hata_goster, annot=True)
```

7. İLAVE ETKİNLİK

Öğrenciler, iris veri çiçeği türlerine ait veri setini kullanarak k-en yakın komşu algoritmasının (k-NN) eğitimini gerçekleştirir. Bu ilave etkinlik ile öğrencilerin farklı bir yapay zekâ algoritması kullanarak uygulama gerçekleştirmek için gereken adımları sıralamaları gerektiğini kavrarlar (Bu etkinliğe ait kodlar Github Hafta1 klasörü altında, H1_kNN_algoritmasi.py dosyasında yer almaktadır).

Eğitmene Not

Eğitmen, öğrencilere k-NN algoritmasını şu görsel vasıtasıyla ifade eder:



Şekil 1.27. Fit eğrisi

PYTHON KODLARI

```
from sklearn.datasets import load_iris
iris = load_iris()
print (iris.feature_names)
print (iris.target_names)
print (iris.target)
print (iris.data)
X = iris.data
Y = iris.target
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.20, random_state = 0)
print("Eğitim veri seti boyutu=",len(X_train))
print("Test veri seti boyutu=",len(X_test))
from sklearn.neighbors import KNeighborsClassifier
model =  KNeighborsClassifier ()
model.fit(X_train,Y_train)
Y_tahmin = model.predict(X_test)
from sklearn.metrics import confusion_matrix
hata_matrisi = confusion_matrix(Y_test, Y_tahmin)
print(hata_matrisi)
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt
index = ['setosa','versicolor','virginica']
columns = ['setosa','versicolor','virginica']
hata_goster = pd.DataFrame(hata_matrisi,columns,index)
plt.figure(figsize=(10,6))
sns.heatmap(hata_goster, annot=True)
```

Düşün, tartış…

Yapay Zekâ ile problem çözmek için gerekenler, modelleme ve veri seti gibi görünüyor… Peki, veri içeriği ve muhtemel siber saldırılar güvenli bir Yapay Zekâ sistemi elde edebilmemiz için önemli midir? İlgili sorular sınıf ortamında öğrencilerle tartışılabilir ve öğrenciler fikirlerini Web ortamındaki görüşlerle karşılaştırabilir.

Kaynakça

O'Reilly. (2021). Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow Concepts, Tools, and Techniques to Build Intelligent Systems.

Yılmaz, D. Ö. Ü. A., (2021). Yapay Zekâ. Kodlab yayın dağıtım yazılım ltd. Şti.

Web Kaynağı 1.1: https://erdincuzun.com/makine_ogrenmesi/decision-tree-karar-agaci-id3-algoritmasi-classification-siniflama/

Web Kaynağı 1.2: https://www.mshowto.org/veri-on-isleme-nedir.html

Web Kaynağı 1.3: https://tr.euronews.com/next/2021/03/31/sci-tech-dubai-de-sirketler-pandemi-sonras-yapay-zekay-daha-insanc-l-k-lma-aray-s-nda

2. Hafta: Yapay Zekâ Matematiği ve Bulanık Mantık

Ön Bilgi:

- Python ile yapay zekâ mantığı ve veri organizasyonu işlemleri gerçekleştirilecektir.
- Python kodlamada karşılaştırma, fonksiyonlar ve kütüphane kullanımı (Bu konuda Python anlatım videoları: 7. Karşılaştırma Op., Mantıksal Operatörler. 8. If Yapısı, 10.-11. Döngüler, 13. Fonksiyonlar, 16. Python Yapay Zekâ kütüphaneleri yardımcı araç olarak kullanılabilecektir).

Haftanın Kazanımları:

- Öğrenciler "yapay zekâ" ekseninde bulanık mantık tabanlı matematiksel işlemleri gerçekleştirir.
- Öğrenciler, bulaşık makinesi modellemesine ait parametrelere göre üyelik fonksiyon sayıları, isimleri alt ve üst limitleri belirler.
- Öğrenciler Python programlama dilinde scikit-fuzzy kütüphanesindeki bulanık mantık modellemesi için gerekli olan **"antecedent", "arrange", consequent", "ControlSystem", "ControlSystemSimulation" ve "compute"** komutlarını uygulayarak öğrenir.
- Öğrenciler, bulaşık makinesi modellemesinde bulaşık miktarı değerini bulanık mantık ile ("az", "normal" ve "çok" şeklinde) organize eder ve ilgili kurallar çerçevesinde tasarlanan sistemi Python programlama dilinde oluşturur.
- Öğrenciler bulaşık makinesi modellemesi örneğinde öğrendiği bulanık mantık komutlarını kullanarak "otomatik fren sistemi" örneğini uygulayarak sentezler.
- Öğrenciler Python ile bulanık mantık modelleme ve çözüm üretmeyi uygular.

Haftanın Amacı:

Bu haftanın amacı, "yapay zekâ matematiği" ve "bulanık mantık" kavramlarının tüm öğrenciler tarafından anlaşılmasını sağlamaktır. Haftanın bir diğer amacı, Bulanık mantık kullanılarak farklı örnekler ile öğrencilerin etkinliklere yönelik ilgilerini artırmaktır. Ayrıca, öğrencilere Python ile bulanık mantık çözüm üretme yetenekleri kazandırılacaktır.

Kullanılacak Malzemeler:

Bilgisayar, kâğıt, kalem, Python kod editörü, örnekler için çıktı görselleri

Haftanın İşlenişi:

Algıla: Öğrenciler yapay zekâ matematiği ve bulanık mantık sistemi kavramlarını tanımlayabilir.

Tasarla: Öğrenciler yapay zekâ tabanlı problem çözümleri için verileri organize edebilir.

Harekete Geç: Öğrenciler Python programlama dili ile yapay zekâ eğitim ve test verilerini ayırma, Bulanık Mantık modeli kurma gibi süreçleri gerçekleştirebilir.

Yürüt: Eğitmenin, öğrencilere Python ile Bulanık Mantık tabanlı model tasarlama işlemlerini gerçekleştirebilmeleri için uygulamalarına birebir ve etkili rehberlik eder. Öğrenciler de ilgili doğrultuda Bulanık Mantık tabanlı uygulama kodlayarak hedef problem için çözüm üretir.

Karar Ver: Öğrenciler alternatif yapay zekâ modelini düzenleyerek / kodlayarak çözüm üretebilir.

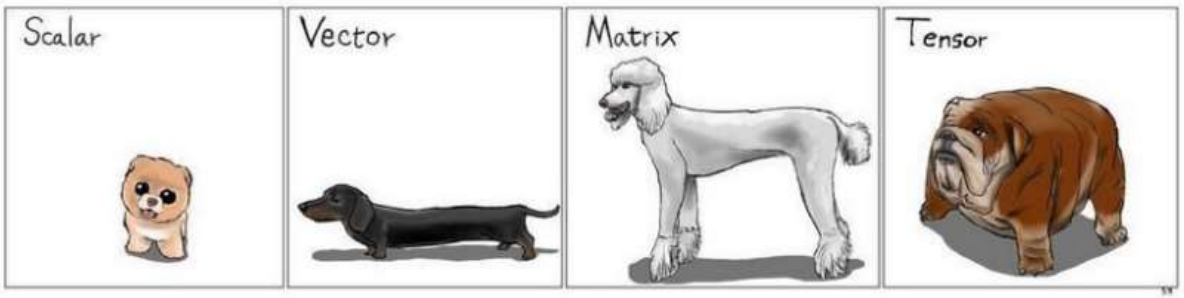
1. ALGILA

1.1. Mantık

Matematiksel becerileri kazanabilmek için matematiksel düşünme tarzını geliştirerek uygulamaya koymak mantığın temelini oluşturur (Korkmaz, 2019). Mantık, insan zihni gibi doğruyu ve yanlışı ayırt etmek için kullanılan bir düşünme disiplinidir. Mantık kavramı doğru (1) veya yanlıştan (0) oluşan bir yargının sonucudur. Örneğin; "İstanbul, Türkiye'nin doğusundadır." cümlesi bir önermedir ve bu önerme yanlıştır. "23 Nisan, Ulusal Egemenlik ve Çocuk Bayramı'dır." önermesi ise doğru bir önermedir. Mantık kavramı insan beyninin problem çözme teknikleri ile oldukça benzerdir. İnsan beyni ilk olarak problemin kaynağı olan faktörleri tespit ettikten sonra, problemin çözümü için çözüm aşamalarını mantıksal olarak gerçekleştirir.

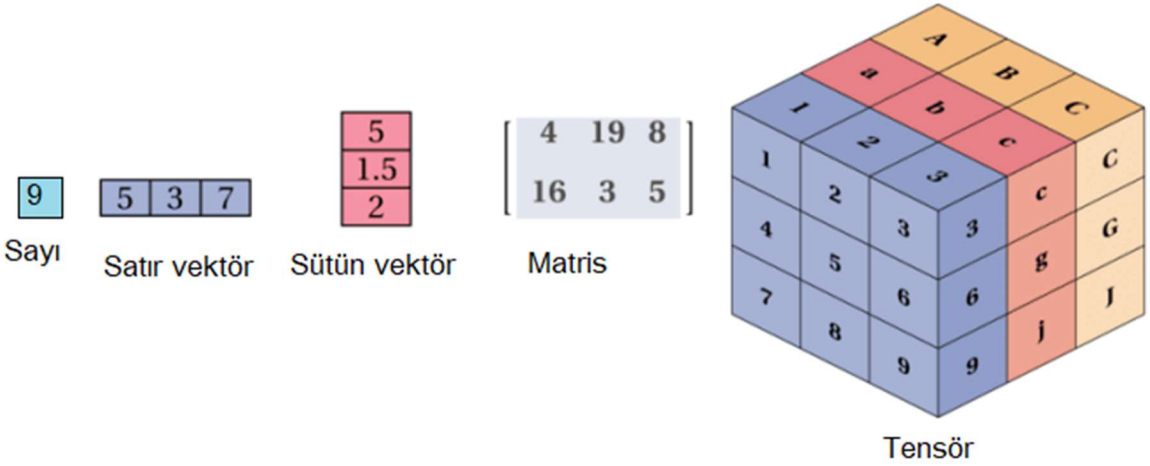
1.2. Python ile Yapay Zekâ Matematiği

Öğrenciler ilk haftada öğrendikleri Python ile numpy, pandas ve scipy yapay zekâ kütüphaneleri ile basit bir şekilde matematiksel işlemleri gerçekleştirir. Python ile yapay zekâ matematiği sayılar, matrisler, vektörler, tensörler gibi kavramları kapsar (Şekil 2.1).



Şekil 2.1. Skalar, vektör, matris ve tensör görselleri (Web Kaynağı 2.1)

İnsanın yaşını tanımlarken kullanılan sayılar ifadesi boyutsuz olarak kullanılır. İnsanın yaşı yerine fotoğrafı ise 2 boyutlu sayılardan oluşan matris ile ifade edilir (Şekil 2.2). Matrislerin satır ve/veya sütunlarını oluşturan her bir boyutuna vektör denir. Şekil 2.2'de gösterildiği gibi tensör kavramı ise, insanın damarındaki kanın akışı ile damarın hacimsel ve yüzeysel alan değişimi ifade edilirken kullanılır. Örneğin zekâ küpü tensöre bir örnek olarak verilebilir.



Şekil 2.2. Yapay zekâ matematiğindeki veri türleri

1.3. Bulanık Mantık Tekniği

Bulanık mantık, insan düşüncesini fonksiyonlara dönüştüren hesaplamalı matematiksel işlemlerdir. Bulanık mantık kavramı klasik yanlış (0) ve doğru (1) mantığı arasında derecelendirme yapılarak kümelendirme işlemi gerçekleştirir. Problemin çözümü esnasında bazı önermelerin doğru ya da yanlış olarak değerlendirilmesi mümkün olmayabilir. Bu durumun sebebi, önermelerin doğru bir şekilde ölçülememesinden kaynaklanmaktadır. İnsan beyninde gerçek durumlar daha çok ara değerler ile temsil edilebileceği için klasik (keskin) mantığa karşı **"bulanık mantık"** kavramı öne sürülmektedir. Öğrenciler bulanık mantık ile modelleme işleminde, giriş parametreleri ile çıkış parametrelerini ilişkilendirilerek küme oluştur ve bulanık mantık kurallarını tanımlar. Örneğin bir kişinin yaşını tahmin etmeye çalışalım. Kişinin yaşını görsel olarak anlamak kesin bir sonuç değildir. Kişinin kesin yaşını belirlemek için ya kimliğine bakılır ya da tıbbi testler uygulanır. Bu sonuçlara bakarak, kişinin yaşı hakkında net bir bilgi elde edilir. Örneğin "Bu kişi 18 yaşından büyüktür." gibi bir önermeye yüzde yüz yanlış ya da yüzde yüz doğru gibi bir yorum getirmek mümkündür. Ancak önerme "Bu kişi gençtir." şeklinde ortaya atılırsa bulanık mantık kavramı ortaya çıkacaktır. Şekil 2.3'te bulanık mantığı işlem sırası gösterilmiştir. Yapay zekâda bulanık mantık bileşenleri şöyledir:

Giriş: Yapay zekâ bulanık mantık sistemlerinde ilk olarak veri girişi yapılır. Böylece bulanık mantık sistemi çalışmaya başlar.

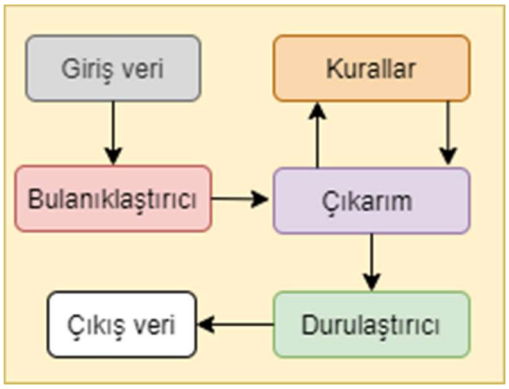
Bulanıklaştırıcı: Bulanık mantık sistemine girilen verileri 0 ile 1 arasındaki bulanık değerlere dönüştürür.

Kurallar: Bulanıklaştırma işlemi sırasındaki çıkarım kuralları belirlenir.

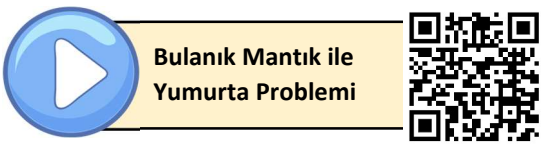
Çıkarım: Bulanık kuralları kullanarak bulanık giriş değerleri için bulanık sonuçlar çıkarır.

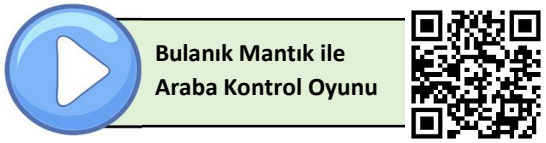
Durulaştırıcı: Çıkarımdan elde edilen sonuçları istenilen gerçek değerlere dönüştürür.

Çıkış: Bulanık mantık sisteminden elde edilen çıkış verisidir.



Şekil 2.3. Bulanık mantık sistemin genel görüntüsü





2. TASARLA

Öğrenciler, ilk olarak bulanık mantık modeli ile bulaşık makinesi problemini anlar. Şekil 2.4'te gösterilen bulaşık makinesi bulanık mantık modeline uygun olarak giriş ve çıkış parametrelerini belirler. Bulaşık makinası bulanık mantık modeli ile belirlenen bulaşık miktarı ve kirlilik seviyesine göre bulaşıkları yıkamak için gerekli olan süreyi belirler.

Şekil 2.4. Bulaşık Makinası için Bulanık Mantık Sisteminin Giriş – Çıkış Parametreleri

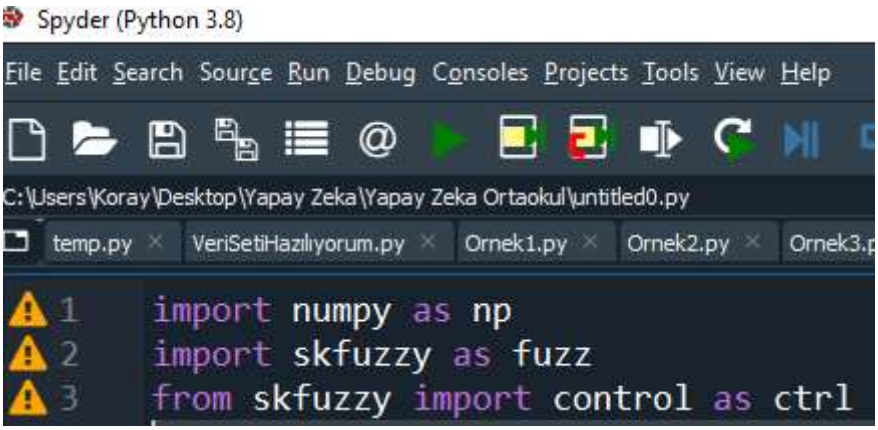
3. HAREKETE GEÇ

Öğrenciler bulaşık makinesi bulanık mantık modeline ait giriş-çıkış parametrelerine göre tüm parametrelerin üyelik fonksiyon sayılarını, isimlerini alt ve üst limitlerini belirleyerek Python kodlarını hazırlar. Öğrenciler, ilk olarak, Şekil 2.5'te gösterilen Anaconda Prompt (Anaconda 3) uygulama ekranından scikit-fuzzy kütüphanesini yükler.

```
C:\Users\Koray>pip install scikit-fuzzy
```

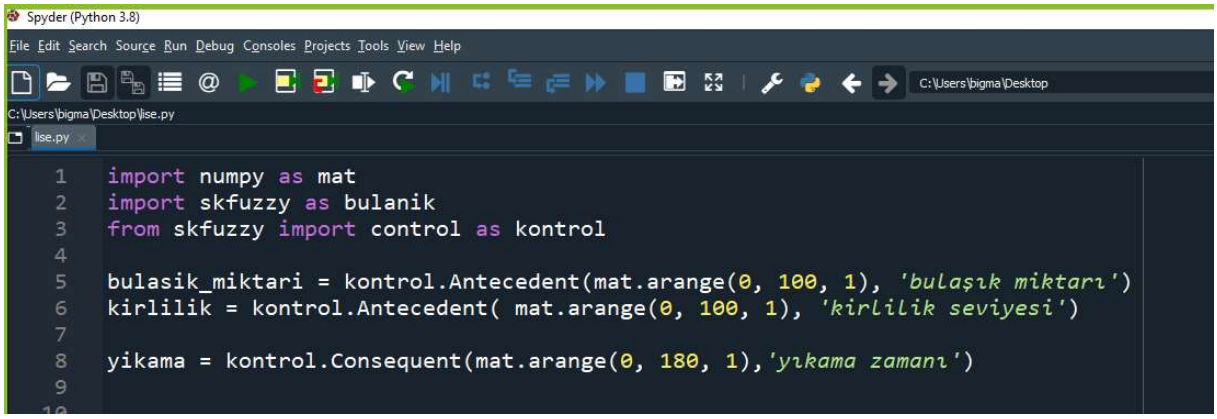
Şekil 2.5. Scikit-fuzzy kütüphanesi

Şekil 2.6'daki bulanık mantık (fuzzy) kütüphaneleri kodlarını yazar. 2. ve 3. satırda yer alan bulanık mantık kütüphanesini editöre yükler.



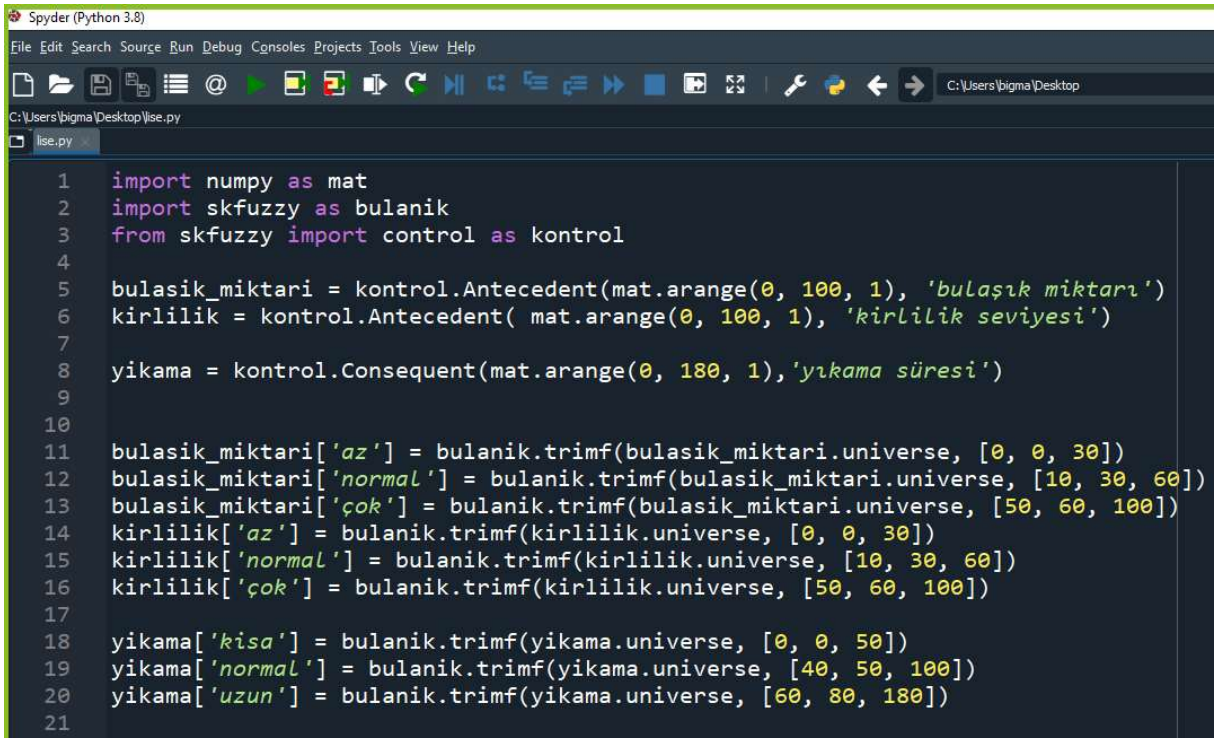
Şekil 2.6. Bulaşık makinası bulanık modeli için Python kütüphanelerin kod satırı

Öğrenciler, kütüphane kodlarını yazdıktan sonra giriş parametreleri olan bulaşık miktarı ve kirlilik seviyesi yüzde olarak aldığında çıkış parametreleri olan yıkama süresini 0 ile 180 dakika arasında belirler. Bunun için giriş parametrelerini belirlemede bulanık mantık kontrol sistemlerinde giriş değişkeni olan "**kontrol.antecedent**" komutunu kullanarak "**mat.arrange"** ile giriş aralığını belirler. Çıkış parametresini belirlerken bulanık mantık kontrol sistemlerinde çıkış değişkeni olan **"kontrol.Consequent"** ile yine **"mat.arrange"** komutunu kullanarak çıkış aralığını belirler.



Şekil 2.7. Bulanık mantık kontrol sistemlerinde giriş ve çıkış parametrelerin belirlenmesi

Öğrenciler, üyelik fonksiyon kümeleri oluştururken, giriş-çıkış değerlerinden kaçının "sözel giriş/çıkış değeri" ile isimlendirileceğine karar verir. Öğrenciler, Şekil 2.8'de gösterildiği gibi **"trimf"** komutu kullanarak giriş parametreleri olan bulaşık miktarı ve kirlilik derecesini **"az"**, **"normal"**, **"çok"** ve çıkış parametresi olan yıkama süresini de **"kısa"**, **"normal"**, **"uzun"** şeklinde sisteme tanıtır.



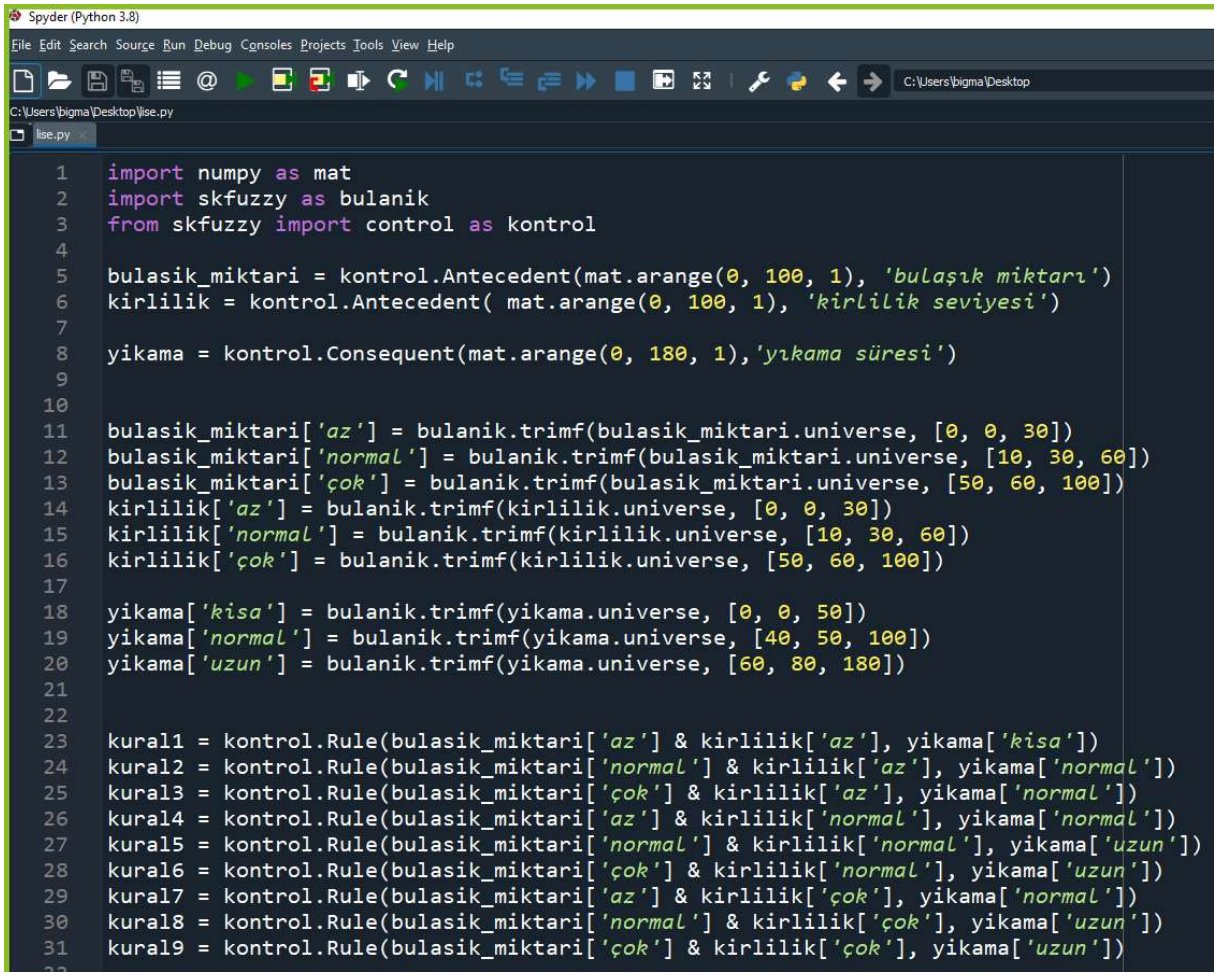
Şekil 2.8. Dilsel değişken isimlendirilmesi

Öğrenciler bulaşık makinesi bulanık mantık kontrol sisteminin giriş ve çıkış parametrelerine göre kurallarını Çizelge 2.1'de gösterildiği gibi oluşturur.

Çizelge 2.1. Bulaşık makinası bulanık kontrol sistemi için kural tablosu

	Giriş parametreleri		Çıkış parametresi
Kurallar	**Bulaşık Miktarı**	**Kirlilik**	**Yıkama zamanı**
Kural-1	Az	Az	Kısa
Kural-2	Normal	Az	Normal
Kural-3	Çok	Az	Normal
Kural-4	Az	Normal	Normal
Kural-5	Normal	Normal	Uzun
Kural-6	Çok	Normal	Uzun
Kural-7	Az	Çok	Normal
Kural-8	Normal	Çok	Uzun
Kural-9	Çok	Çok	Uzun

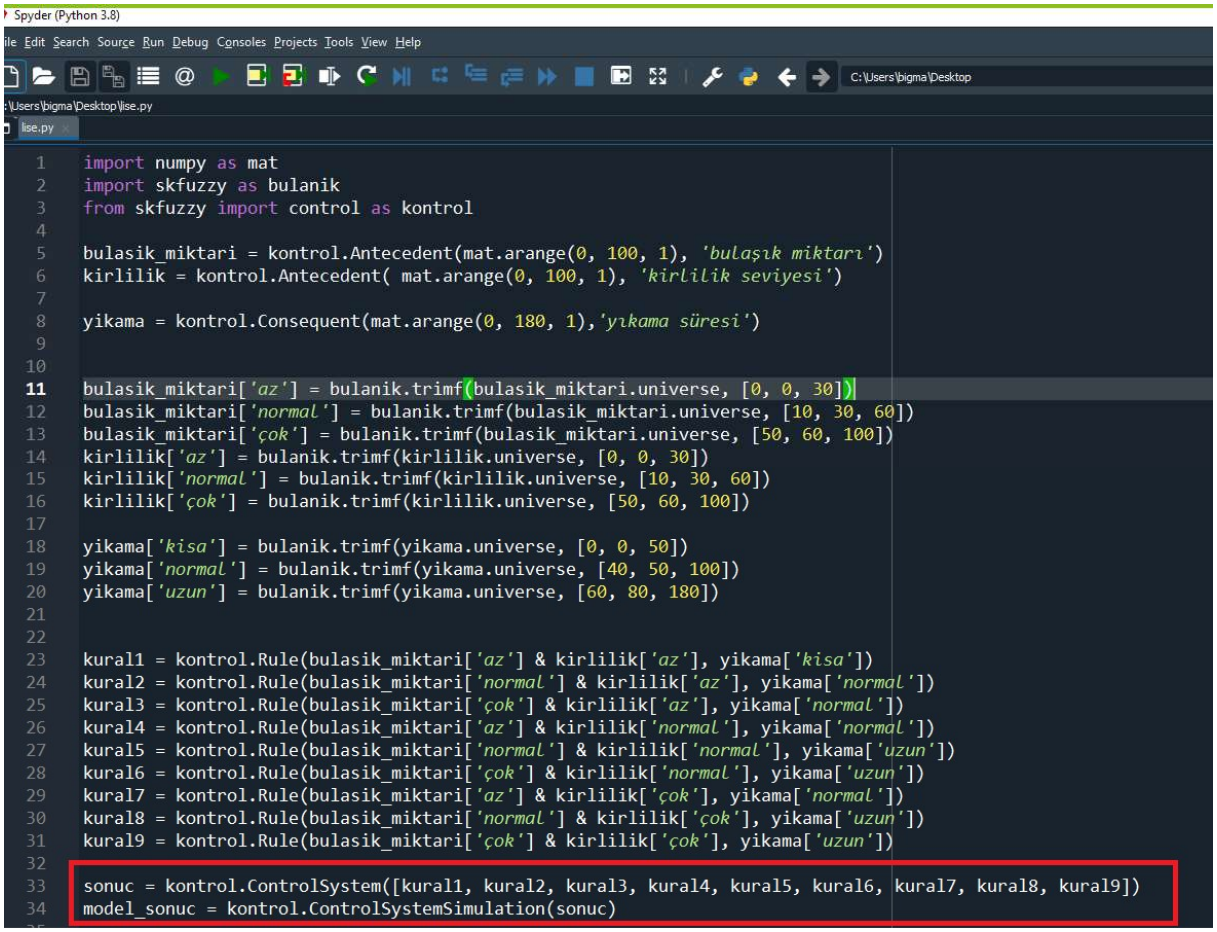
Öğrenciler daha önce hazırladıkları kurallara göre Şekil 2.9'da görüldüğü gibi Python kodlarını yazar.



Şekil 2.9. Bulaşık makinesi bulanık mantık modeli kuralları için Python kodları

4. YÜRÜT

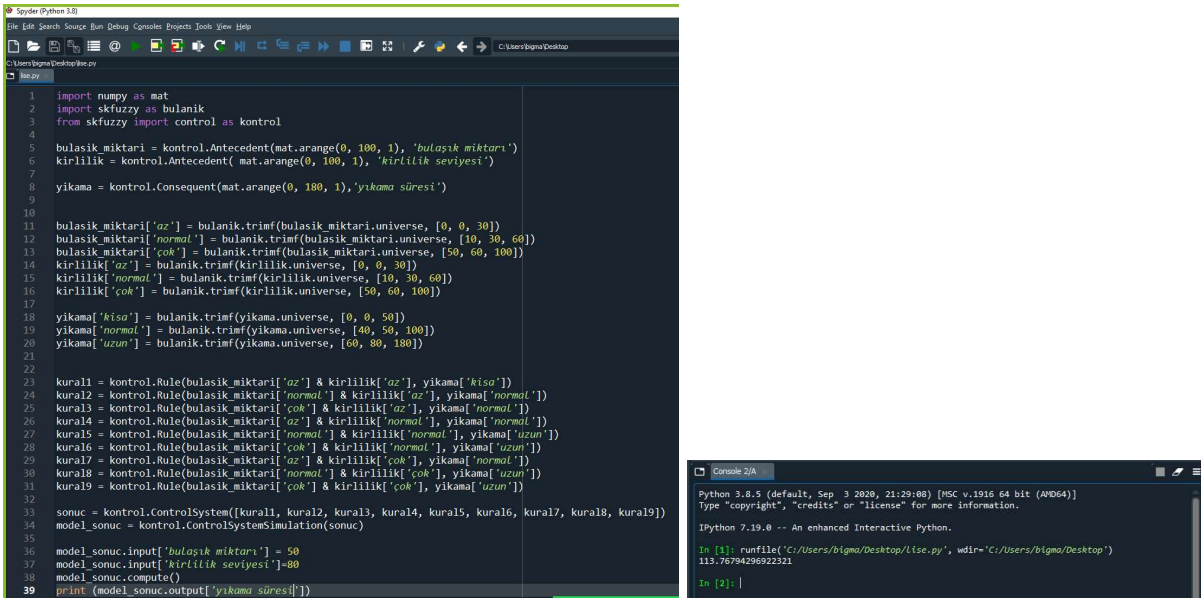
Öğrenciler bulaşık makinesi kontrol sistemi için giriş ve çıkış parametrelerini üyelik fonksiyonları ile modeller. Ardından yine kontrol modellemesi için belirlenen dokuz kurala göre, **"ControlSystem"** komutu kullanılmak suretiyle bulanık mantık sistemi oluşturulur. Oluşturulan bulanık kontrol sisteminin simülasyonu için **"ControlSystemSimulation"** komutunu kullanır.



Şekil 2.10. Bulanık kontrol sisteminin oluşturulması ve simülasyon kod satırı

5. KARAR VER

Öğrenciler, dokuz kurala göre oluşturmuş oldukları bulanık mantık modelinin sonuçlarına Şekil 2.10'da verilen kod satırlarını kullanarak karar verir (İlgili koda Github platformu: https://github.com/deneyapyz/lise/ kapsamında yer alan, Hafta2 klasörü altındaki H2_bm_bulasik-makinesi.py adlı dosyadan da erişilebilir). Şekil 2.11a'da **"model_sonuc.input"** komutu ile öğrenci bulanık mantık sistemine göndermek istediği bulaşık miktarı ve kirlilik seviyesini girer. Daha sonra **"model_sonuc.compute()"** komutu kullanılarak oluşturulan bulanık mantık sisteminde girilen bulaşık miktarı ve kirlilik seviyesine göre yıkama süresini bulanık mantık modelinde hesaplatır. Şekil 2.11b'de %50 bulaşık miktarı ve %80 kirlilik seviyesi için bulanık mantık modeli yıkama süresini 113,7 saniye olarak hesaplar.



Şekil 2.11: Bulaşık makinası örneği için bulanık mantık modelinin değerlendirilmesi

Eğitmene Not

Eğitmen, verilen bulaşık makinesi bulanık mantık modelini Python kodlarında

model_sonuc.input['bulaşık miktarı'] = 30
model_sonuc.input['kirlilik seviyesi']=60
model_sonuc.input['bulaşık miktarı'] = 10
model_sonuc.input['kirlilik seviyesi']=20
model_sonuc.input['bulaşık miktarı'] = 80
model_sonuc.input['kirlilik seviyesi']=100

gibi farklı örnekler üzerinden elde edilen yıkama sürelerini, uygulamalı olarak öğrencilere gösterir.

6. UYGULAMANIN PYTHON KODLARI

```
import numpy as mat
import skfuzzy as bulanik
from skfuzzy import control as kontrol


bulasik_miktari = kontrol.Antecedent(mat.arange(0, 100, 1), 'bulaşık miktarı')
kirlilik = kontrol.Antecedent( mat.arange(0, 100, 1), 'kirlilik seviyesi')
```

```
yikama = kontrol.Consequent(mat.arange(0, 180, 1),'yıkama süresi')


bulasik_miktari['az'] = bulanik.trimf(bulasik_miktari.universe, [0, 0, 30])
bulasik_miktari['normal'] = bulanik.trimf(bulasik_miktari.universe, [10, 30, 60])
bulasik_miktari['çok'] = bulanik.trimf(bulasik_miktari.universe, [50, 60, 100])
kirlilik['az'] = bulanik.trimf(kirlilik.universe, [0, 0, 30])
kirlilik['normal'] = bulanik.trimf(kirlilik.universe, [10, 30, 60])
kirlilik['çok'] = bulanik.trimf(kirlilik.universe, [50, 60, 100])

yikama['kisa'] = bulanik.trimf(yikama.universe, [0, 0, 50])
yikama['normal'] = bulanik.trimf(yikama.universe, [40, 50, 100])
yikama['uzun'] = bulanik.trimf(yikama.universe, [60, 80, 180])


kural1 = kontrol.Rule(bulasik_miktari['az'] & kirlilik['az'], yikama['kisa'])
kural2 = kontrol.Rule(bulasik_miktari['normal'] & kirlilik['az'], yikama['normal'])
kural3 = kontrol.Rule(bulasik_miktari['çok'] & kirlilik['az'], yikama['normal'])
kural4 = kontrol.Rule(bulasik_miktari['az'] & kirlilik['normal'], yikama['normal'])
kural5 = kontrol.Rule(bulasik_miktari['normal'] & kirlilik['normal'], yikama['uzun'])
kural6 = kontrol.Rule(bulasik_miktari['çok'] & kirlilik['normal'], yikama['uzun'])
kural7 = kontrol.Rule(bulasik_miktari['az'] & kirlilik['çok'], yikama['normal'])
kural8 = kontrol.Rule(bulasik_miktari['normal'] & kirlilik['çok'], yikama['uzun'])
kural9 = kontrol.Rule(bulasik_miktari['çok'] & kirlilik['çok'], yikama['uzun'])

sonuc = kontrol.ControlSystem([kural1, kural2, kural3, kural4, kural5, kural6, kural7, kural8,
kural9])
model_sonuc = kontrol.ControlSystemSimulation(sonuc)

model_sonuc.input['bulaşık miktarı'] = 50
```

```
model_sonuc.input['kirlilik seviyesi']=80
model_sonuc.compute()
print (model_sonuc.output['yıkama süresi'])
```

Düşün, tartış…

Benzeri bir çözümü buzdolabı için tasarlayabilir miyiz? Problemi tartışıp, modelleyelim… Bu konuda yapılan modellemeler sınıf ortamında tartışılabilecek ya da öğrenciler Web ortamında benzeri yöndeki uygulamaları araştırarak modellemeleri ile karşılaştırmalar yapabilecektir.

Eğitmene Not

Eğitmen, öğrencilerin 1. ve 2. hafta içerisinde elde ettikleri bilgi-becerileri değerlendirmek, aktif katılımlarını sağlamak ve onları gelecek haftalara motive etmek için 'Kahoot' uygulamasını ya da benzeri bir Web 2.0 uygulamasını kullanarak 'bilgi yarışması' düzenleyebilir.

1. ve 2. hafta bağlamında sorulabilecek sorular; Yapay Zekâ kavramının temelleri, veri analiz süreçleri, Makine Öğrenmesi temelleri, değinilen önemli Python kodları, Bulanık Mantık temelleri ve uygulamaları yönünde olabilir.

7. İLAVE ETKİNLİK

Öğrenciler, ilk olarak bulanık mantık ile otomatik fren sistemi problemini anlar. Şekil 2.12'de gösterildiği gibi otomatik fren sistemi için bulanık mantık kontrol sistemi giriş parametreleri mesafe ve hız, çıkış parametresi ise uygulanması gereken fren basınç değerlerini bulanık mantık modeli ile belirler. Bu etkinlik kapsamında problem çözümüne ilişkin bulanık mantık sistemi kodlanır (İlgili koda Github platformu Hafta2 klasörü altındaki H2_bm_fren-sistemi.py adlı dosyadan erişilebilir).

Şekil 2.12. Bulanık mantık ile otomatik fren sistemi

PYTHON KODLARI:

```
import numpy as mat
import skfuzzy as mantik
from skfuzzy import control as kontrol

mesafe = kontrol.Antecedent(mat.arange(0, 50, 1), 'mesafe')
hiz = kontrol.Antecedent( mat.arange(0, 100, 1), 'hiz')
fren_basinci = kontrol.Consequent(mat.arange(0, 100, 1),'fren_basinci')

mesafe['çok yakın'] = mantik.trimf(mesafe.universe, [0, 0, 10])
mesafe['yakın'] = mantik.trimf(mesafe.universe, [5, 15, 25])
mesafe['uzak'] = mantik.trimf(mesafe.universe, [20, 30, 40])
mesafe['çok uzak'] = mantik.trimf(mesafe.universe, [35, 50, 50])

hiz['çok yavaş'] = mantik.trapmf(hiz.universe, [0, 0, 20, 30])
hiz['yavaş'] = mantik.trapmf(hiz.universe, [20, 30, 45, 55])
hiz['hızlı'] = mantik.trapmf(hiz.universe, [45, 55, 70, 80])
hiz['çok hızlı'] = mantik.trapmf(hiz.universe, [70, 80, 100,100])

fren_basinci['çok düşük'] = mantik.trimf(fren_basinci.universe, [0, 20, 40])
fren_basinci['düşük'] = mantik.trimf(fren_basinci.universe, [20, 40, 60])
fren_basinci['yüksek'] = mantik.trimf(fren_basinci.universe, [40, 60, 80])
fren_basinci['çok yüksek'] = mantik.trimf(fren_basinci.universe, [60, 100, 100])

kural1 = kontrol.Rule(mesafe['çok yakın'] & hiz['çok yavaş'] , fren_basinci['çok yüksek'])
kural2 = kontrol.Rule(mesafe['yakın'] & hiz['çok yavaş'] , fren_basinci['çok düşük'])
kural3 = kontrol.Rule(mesafe['çok yakın'] & hiz['yavaş'] , fren_basinci['çok yüksek'])
kural4 = kontrol.Rule(mesafe['yakın'] & hiz['yavaş'] , fren_basinci['düşük'])
kural5 = kontrol.Rule(mesafe['uzak'] & hiz['yavaş'] , fren_basinci['çok düşük'])
kural6 = kontrol.Rule(mesafe['çok yakın'] & hiz['hızlı'] , fren_basinci['çok yüksek'])
```

```
kural7 = kontrol.Rule(mesafe['yakın'] & hiz['hızlı'] , fren_basinci['düşük'])
kural8 = kontrol.Rule(mesafe['uzak'] & hiz['hızlı'] , fren_basinci['çok düşük'])
kural9 = kontrol.Rule(mesafe['çok yakın'] & hiz['çok hızlı'] , fren_basinci['çok yüksek'])
kural10 = kontrol.Rule(mesafe['yakın'] & hiz['çok hızlı'] , fren_basinci['yüksek'])
kural11 = kontrol.Rule(mesafe['uzak'] & hiz['çok hızlı'] , fren_basinci['düşük'])
kural12 = kontrol.Rule(mesafe['çok uzak'] & hiz['çok hızlı'] , fren_basinci['çok düşük'])

fren_kontrol = kontrol.ControlSystem([kural1, kural2, kural3, kural4, kural5,kural6,
                                      kural7, kural8, kural9, kural10, kural11, kural12])
frenleme = kontrol.ControlSystemSimulation(fren_kontrol)

v = int(input("Hızı gir (0-100 km/h) : "))
s = int(input("mesafeyi gir (m) : "))

if (v/2 <= s):
      print ("fren basılması gerek yoktur")
else:
   frenleme.input['hiz'] = v
   frenleme.input['mesafe'] = s

   frenleme.compute()
   print ("fren basıncı (%): ", frenleme.output['fren_basinci'])
   print ("yeni hiz değeri: ", v-(v*frenleme.output['fren_basinci']/100))
```

Biliyor musunuz?

Bulanık Mantık özellikle kontrol sistemlerinde oldukça sık kullanılmıştır. Bu yolla otomatik sulama sistemleri, akıllı güvenlik sistemleri ve daha birçok çözüm tasarlanmıştır. Gerçekleştirilen çözümler için Web'i tarayın!

Kaynakça

Korkmaz, S. (2019). 9. Sınıf Öğrencilerinin Mantık Konusundaki Kavram Yanılgıları (Yüksek Lisans Tezi, Necmettin Erbakan Üniversitesi Eğitim Bilimleri Enstitüsü).

Web Kaynağı 2.1: https://medium.com/zaferdemirkol/herkes-i%C3%A7in-yapay-zekâ-matemati%C4%9Fi-2-vekt%C3%B6r-matris-i%CC%87%C5%9Flemleri-d7c63ad53f9

3. Hafta: Makine Öğrenmesi Kavramı ve Olasılıklı Çözümler için Bayes Öğrenmesi

Ön Bilgi:

- Python ile yapay zekâ mantığı, veri organizasyonu, yapay zekâ matematiği öğrenecektir.
- Python kodlamada döngüler, fonksiyonlar ve kütüphane yapılarının kullanımı (Bu konuda Python anlatım videoları: 10-11. Döngüler, 13. Fonksiyonlar, 14-16. Python kütüphaneleri serisi yardımcı araç olarak kullanılabilecektir).

Haftanın Kazanımları:

- Öğrenciler, makine öğrenmesi kavramını ve Bayes öğrenmesi algoritmasını kullanır.
- Öğrenciler, verileri makine öğrenme algoritmaları ile gerçek hayat problemlerine uygular.
- Öğrenciler makine öğrenmesindeki temel aşamaları bilir ve uygulamalarda kullanır.
- Öğrenciler regresyon, sınıflandırma ve kümeleme yöntemlerini uygular.
- Öğrenciler, sonucun sebebini bulurken sonucun hangi olasılıkla hangi sebepten kaynaklandığını veren Bayes Teoremini örneklerle kavrar.
- Öğrenciler Python programlama dilini kullanarak aracın özelliklerine göre satış durumunu tahminleme etkinliği için program kodunu yazabilir.

Haftanın Amacı:

Bu haftanın amacı, “yapay zekâ alt dalı olan makine öğrenmesi” ve “Bayes öğrenme” kavramlarının tüm öğrenciler tarafından doğru bir şekilde anlaşılmasını sağlamaktır. Haftanın bir diğer amacı, makine ve Bayes öğrenmelerini kullanılarak farklı örnekler ile öğrencilerin ilgisini çekerek etkinliklere yönelik ilgilerini artırmaktır. Ayrıca, öğrencilere Python ile makine öğrenme algoritmaları ile örnek modelleme ve çözüm üretme yeteneği kazandırılacaktır.

Kullanılacak Malzemeler:

Bilgisayar, kâğıt, kalem, Python kod editörü, örnekler için çıktı görselleri

Haftanın İşlenişi:

Algıla: Öğrenciler makine öğrenmesi kavramına ilişkin temelleri ve Bayes öğrenme tekniğini tanımlayarak yorumlayabilir.

Tasarla: Öğrenciler Bayes öğrenme tekniği ile araç satış probleminin giriş ve çıkış parametrelerini tasarlayabilir.

Harekete Geç: Öğrenciler Python programlama dili ile veri setindeki metinsel değerleri sayısallaştırıp eğitim ve test verileri ile model kurabilir. Bu hafta için Github platformu (https://github.com/deneyapyz/lise/) ile etkileşim Harekete Geç aşaması ile başlatılabilmektedir.

Yürüt: Eğitmenler öğrenciler ile etkileşimli bir biçimde Bayes öğrenmesi kullanarak modeli eğitir ve aracın satış tahmin durumunu belirler. Öğrenciler söz konusu uygulama üzerinden Bayes öğrenmesi tekniği ile etkin çözüm üretme süreçlerine ilişkin uygulama kodlama ve yürütme işlemlerini uygulayabilir.

Karar Ver: Öğrenciler alternatif yapay zekâ modelini düzenleyerek / kodlayarak çözüm üretebilir.

1. ALGILA

1.1. Makine Öğrenmesi Temelleri

Günümüzde teknolojinin hızla gelişmesiyle bilgisayar, cep telefonu, tablet gibi akıllı cihazların ve internet teknolojisinin hızla yayılması ile üretilen veriler artmış ve **"büyük veri"** kavramı ortaya çıkmıştır.

Bunun en önemli nedeni ise hayatımızın her alanında Instagram, Twitter ve Youtube paylaşımları, nesnelerin interneti (IoT) gibi dijital bir hareket olmasıdır. Büyük veri; verinin hacmi, veri hızı, veri çeşitliliği, verinin düzensiz olması, verinin değerli olması olmak üzere beş bileşenden oluşur. Veri hacmi, verinin yüksek hacimli olduğunu belirtir. Örneğin bir uçağın motorunda ve diğer bileşenlerinde milyonlarca sensör mevcuttur. Bu sensörler uçakların yaptığı her bir hareketi anlık olarak toplayarak terabaytlar seviyesinde veri üretmektedir (Şekil 3.1).

Şekil 3.1. Uçak sensörlerinden bir yıl boyunca elde edilen veri boyutu (Web Kaynağı 3.1)

Büyük verinin ikinci bileşeni ise verinin hızıdır. Örneğin Şekil 3.2'de gösterildiği gibi bir dakika içerisinde 200+ milyon e-posta, 4 milyon Facebook beğenmesi, 1+ milyon Instagram beğenmesi, milyonlarca atılan tweetler gibi veriler oldukça hızlı aktarılmaktadır. (Verinin kaynağı ve hangi yıla ait olduğu yazılırsa daha iyi olur.)

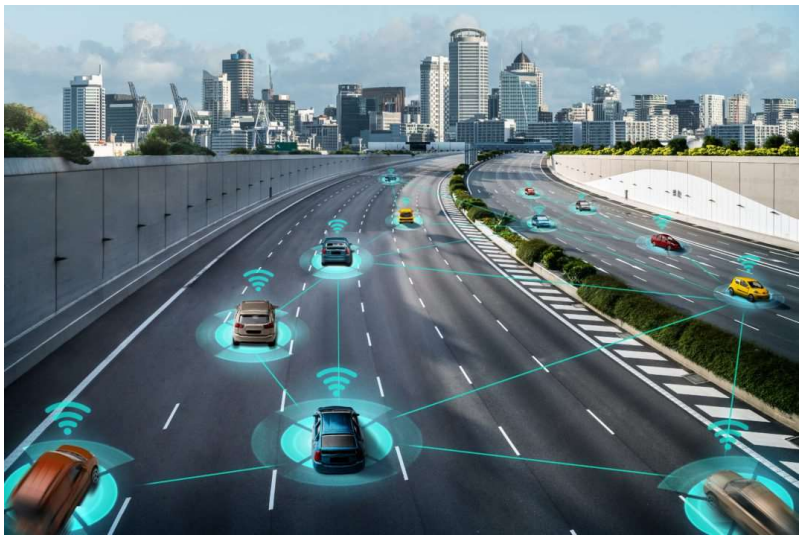
Şekil 3.2. Bir dakika içesinde sosyal medya araçlarında gelen veriler (Web Kaynağı 3.2)

Büyük verinin diğer önemli bileşeni ise veri çeşitliliğidir. Verilerin belirli bir yapısı yoktur. Şekil 3.3'te gösterildiği gibi Sensörden alınan verilerin toplandığı .log dosyaları, resim, ses metin .txt dosyaları, şirket veri tabanlarının .csv (excel) dosyaları, twitterdan alınan .json verileri gibi birçok farklı veri çeşitliliği vardır.

Şekil 3.3. Büyük veride kullanılan bazı veri çeşitleri (Web Kaynağı 3.3)

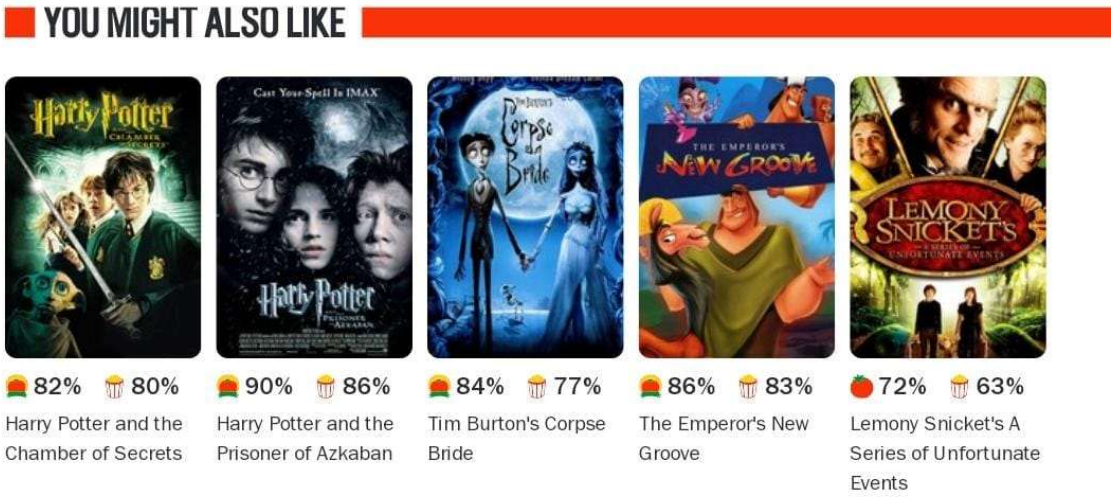
Büyük verinin dördüncü bileşeni ise verinin düzensiz, karmaşık ve kirli olmasıdır. Şekil 3.4'te gösterildiği gibi akan bir trafikte arabanın üzerinde bulunan sensörler vasıtasıyla araçların hızları alınmaktadır. Böylece araçların ortalama hızların göre, ileride karşılacakları, trafik lambalarının kırmızı veya yeşil yanma sürelerini arttırılması ayarlanmaktadır. Ancak veriler analiz edilken bir (1) tane aracın hızı için -10 km/saat olarak yanlış bir değer gelmektedir. Aracın hızının eksi (-)

değer gelmesi oluşturulacak modelde yanlış bir sonuca neden olabilir. Bu durum verinin düzensiz ve karmaşık olmasına bir örnektir. Bu nedenle bu verilerin öncelikle doğru bir formata getirilmesi ya da veri setinden çıkarılması (temizlenmesi) gerekir.



Şekil 3.4. Trafikte araçlardan sensörler vasıtasıyla hızların alınması görüntüsü (Web Kaynağı 3.4)

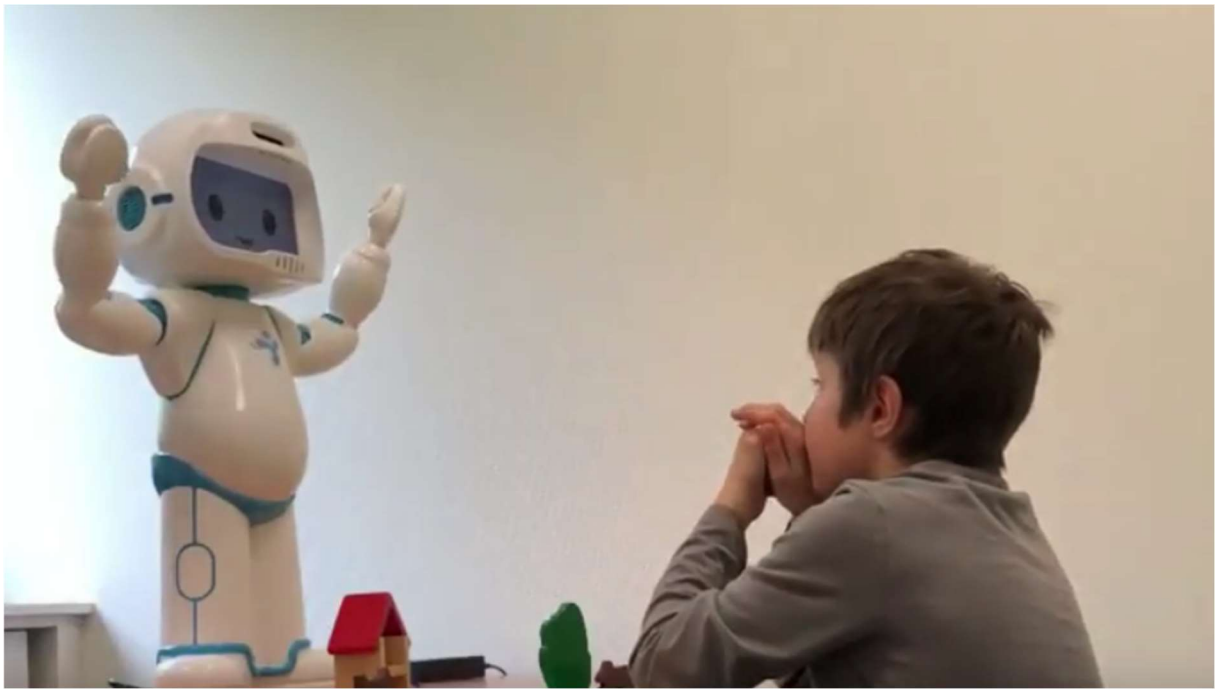
Büyük verinin son bileşeni verinin anlamlandırılmasıdır. Şekil 3.5'te gösterildiği gibi bir sinema uygulaması ile iki öğrenciden ilki farklı türde 10 tane film, diğeri ise yine ilk öğrenciye benzer 9 film almaktadır. Verinin anlamlandırılmasında 9 film alan öğrenciye benzer filmler alan öğrenciler incelenerek 10. (onuncu) film öğrenciye tavsiye olarak sunulmaktadır.



Şekil 3.5. Film tavsiye uygulaması (Web Kaynağı 3.5)

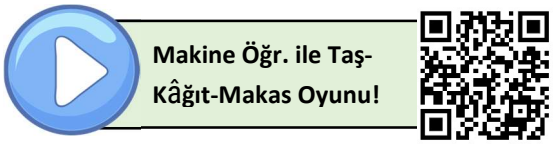
Büyük verileri anlamlandıran bireyler, kurumlar ve ülkeler kendilerini geliştirirken anlamlandıramayanlar ise zamanla teknolojinin gerisinde kalarak popülerliklerini ve sermayelerini kaybetmektedirler. Büyük veriyi değerlendirmek için makine öğrenme yöntemi sıklıkla

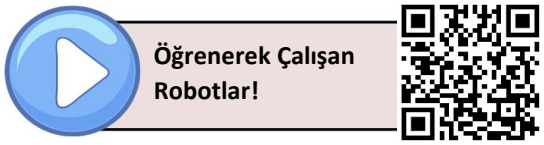
kullanılmaktadır. Makine öğrenmesi akıllı cihazların verileri işleyerek, kendi kendine karar vererek bir problemi çözme sürecidir.



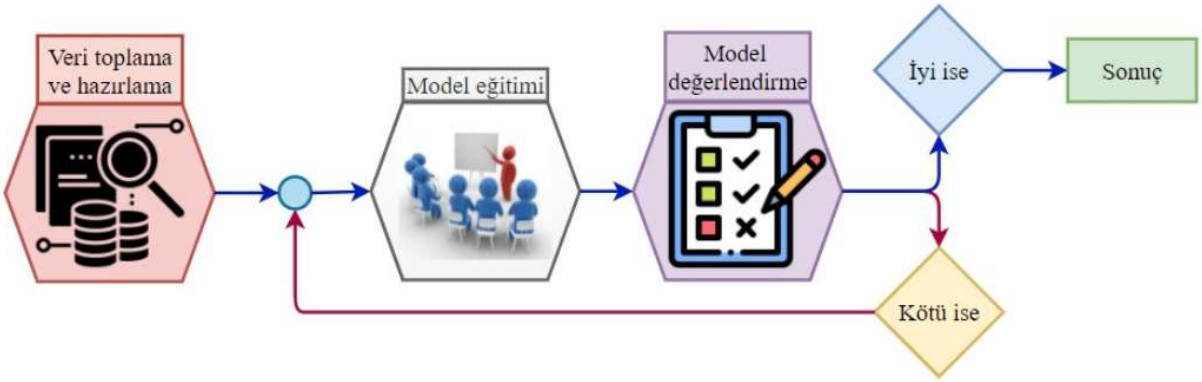
Şekil 3.6. İnsan robot etkileşiminde makine öğrenmesi (Web Kaynağı 3.6)

Makine öğrenmesi, bilgisayarın meydana gelen bir olay ile ilgili topladığı bilgi ve tecrübeleri öğrenebilmesi amacıyla matematiksel modellerin kullanılmasıdır. Makine öğrenmesinde temel amaç elde edilen veriler ile gelecekte oluşabilecek benzer olaylar hakkında kararlar verebilmek veya geçmişteki durumlar hakkında sonuç oluşturmaktır (Azure, 2021). Çok büyük miktarlardaki verilerin elle işlenip bir sonuca varılabilmesi oldukça zordur. Bu nedenle makine öğrenmesi algoritmaları kullanılarak bu işlem kolay ve kısa sürede gerçekleştirilebilir. Makine öğrenmesi, doğal dil işleme, nesne tanıma, arama motorları, robot hareket kontrolü, yüz tanıma gibi birçok uygulamada kullanılmaktadır (Bilgin, 2017).



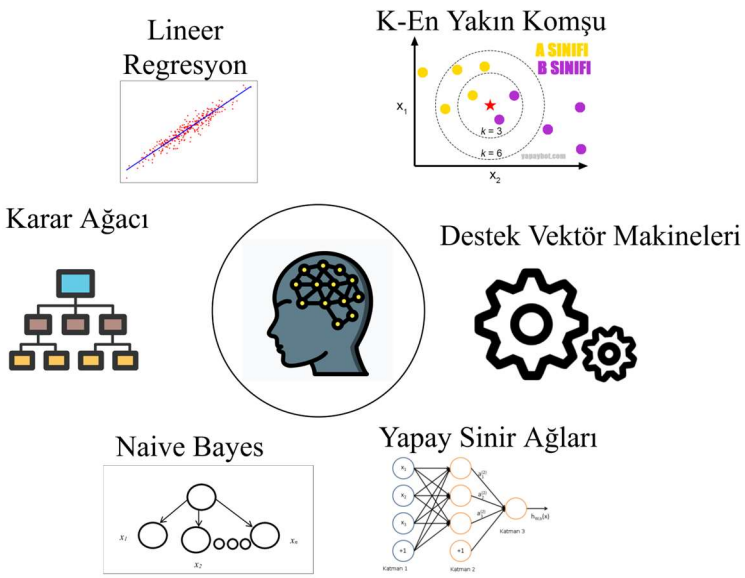


Şekil 3.7'de gösterildiği gibi makine öğrenmesi temel olarak dört aşamadan oluşmaktadır. İlk aşamada, akıllı cihazlardan veriler toplanarak işlenir. İşleme sürecinde uygun olmayan veriler veri setinden çıkarılarak veri bütünlüğü sağlanır. İkinci aşamada, veri setindeki veriler bir kısmı eğitim verisi diğer kısmı test verisi olarak ikiye ayrılır. Eğitim verileri makine öğrenmesindeki modelleri eğitmek için kullanılır. Üçüncü aşamada, modellerden elde edilen sonuçlar test verileri ile analiz edilerek makine öğrenmesi modelinin doğruluğu test edilir. Son aşamada ise, test verilerinden elde edilen sonuçlar değerlendirilir (Çevik, 2020).



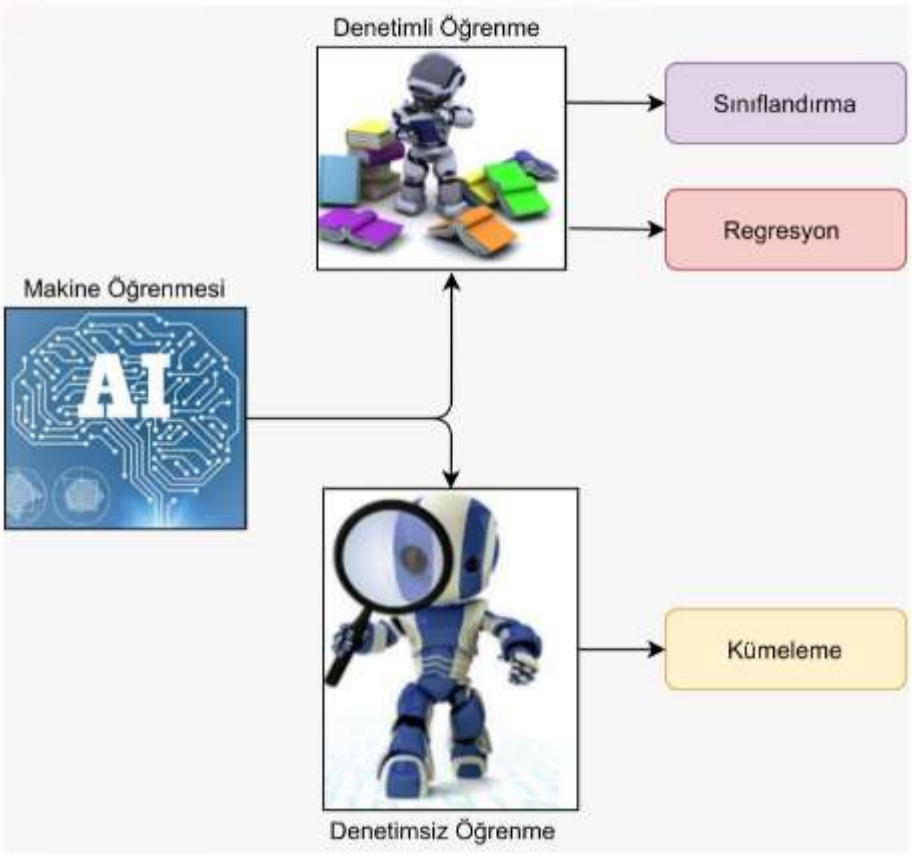
Şekil 3.7. Makine öğrenmesinde temel aşamalar

Makine öğrenmesi tekniklerinde sıklıkla naive-bayes algoritmaları, destek vektör makineleri, karar ağacı algoritmaları, k-en yakın komşu algoritmaları kullanılmaktadır (Şekil 3.8).



Şekil 3.8. Yaygın olarak kullanılan makine öğrenme algoritmaları

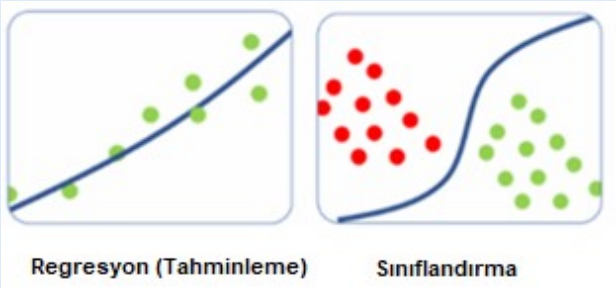
Bu algoritmalardan bir kısmı sınıflandırma uygulamalarında bir kısmı da tahmin etme işlemlerinde kullanılır. Şekil 3.9'da gösterildiği gibi makine öğrenmesi algoritmaları denetimli ve denetimsiz öğrenme olmak üzere ikiye ayrılmaktadır (Şahinarslan, 2019).



Şekil 3.9. Makine öğrenme yöntemleri

Eğitmene Not

Eğitmen, öğrencilere regresyon ve sınıflandırma mantığını Şekil 3.10'da verilen görsele göre anlatır.



Şekil 3.10. Regresyon ve sınıflandırma görseli

1.2. Bayes Öğrenme Tekniği

Naive Bayes öğrenme tekniği temeli Bayes teoremine dayanır. *Bayes Teoremi* bir sonucun sebebini bulurken sonucun hangi olasılıkla hangi sebepten kaynaklandığını bulur. Ülkemizde en popüler alışveriş internet sitesinde akıllı cep telefonu satışını düşünelim. Online mağazada A ve B iki ayrı marka cep telefonu satılmaktadır.

Edinilen tecrübe ve tutulan kayıtlardan cep telefonu ile ilgili bilgiler şu şekildedir:

1. Markalar; A ve B
2. Günlük satış miktarı; A marka 200, B marka 160 adet
3. İade edilen cep telefonu oranı; %5

Bir gün için online mağazada satılan cep telefonlarından A marka olanların iade edilme olasılığını hesaplayalım. Toplam 360 adet cep telefonu, 200 adedi A marka satılıyor. Bu durumda, günlük 18 adet cep telefon iade edilmektedir. İade edilen cep telefonların markalarına göre dağılımını eşit kabul edersek iade edilen cep telefonlarının 9 adeti A markasına aittir. O halde A marka cep telefonunun iade edilme olasılığı 9/200=%4,5'dur.

Bu durumu özetleyen Bayes öğrenme formülü şu şekildedir:

$$P(\text{İade edilen Cep Tlf} \mid A) = \frac{P(A \mid \text{iade edilen cep tlf}) * P(\text{iade cep tlf oranı})}{P(\text{Satış oranı} \mid A)}$$

Formüle sayısal veriler uyarlandığında şu sonuç elde edilir:

$$P(\text{İade edilen Cep Tlf} \mid A) = \frac{0{,}5 * 0{,}05}{0{,}555} = 0{,}045\,(\%4{,}5)$$

Dünyadan Haberler

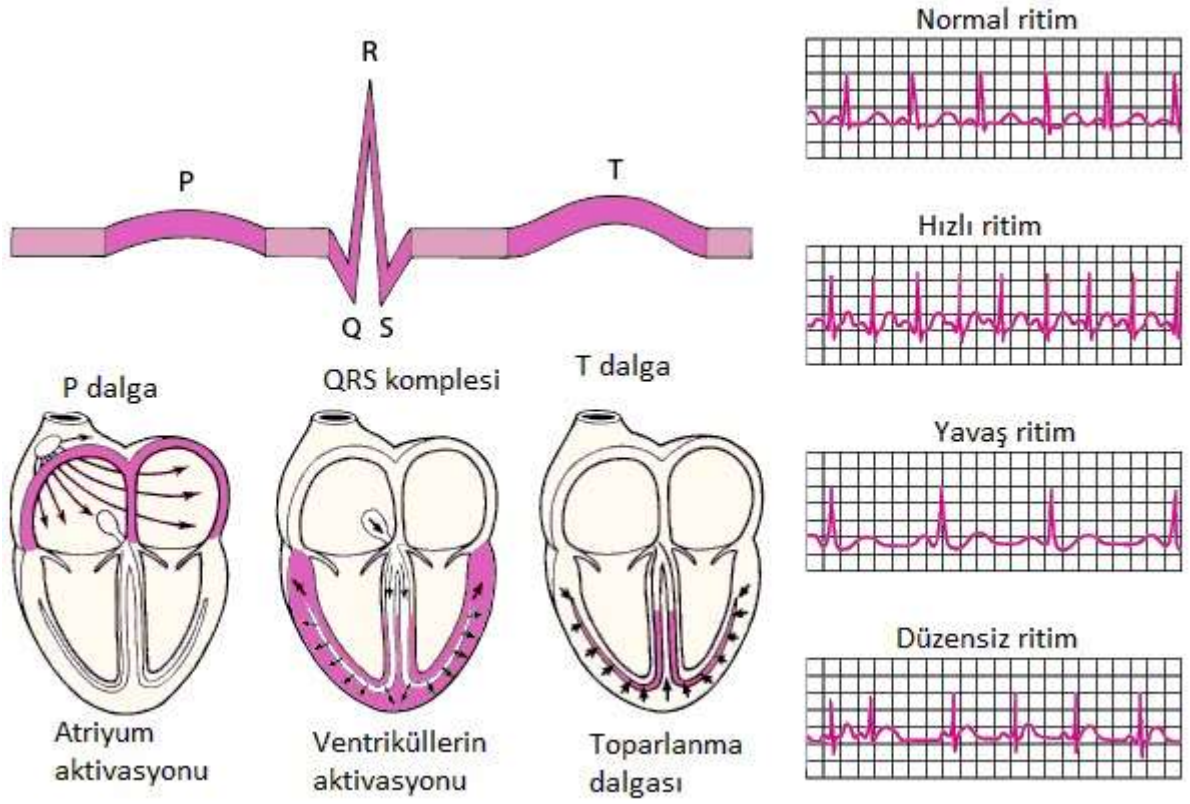
Bozulan iklimimizi yapay zekâ yardımıyla düzeltebilir miyiz?

Yapay zekâ, yaşamlarımızda giderek yaygınlaşan ve birçok alanda karşımıza çıkan bir teknoloji. Kullandığımız cihazlar ve aldığımız hizmetler de bu teknolojiden her geçen gün daha fazla faydalanıyor. Bilim insanları, girişimciler ve devletler, en büyük toplumsal sorunlara getirilebilecek yeni çözümleri keşfedebilmek amacıyla yapay zekâdan faydalanıyor. Dünya'nın iklim davranışlarını ve bunların gelecekte nasıl değişebileceğini anlamak, gündemimizin en başlarında yer alıyor. Dünya genelinde dolaşımda olan yüklü miktardaki dijital veriyi, teknoloji yardımıyla daha iyi anlayabiliyoruz. Peki, çevresel değişimleri azaltmamızda ve geleceğe uyum sağlamamızda yapay zekâ bize nasıl yardımcı olabilir?

Uydular vasıtasıyla toplanan iklimsel verilerin miktarı, tarih boyunca görülmemiş seviyelere ulaşmış durumda. Karşılaştığımız hava tahminleri ise hiç olmadığı kadar fazla ayrıntıya yer veriyor. Fakat iklim modelleri ve senaryoları, birçok belirsizlik barındırmaya devam ediyor. Eldeki yüklü verileri yapay zekâdan faydalanarak değerlendiren bilim insanları, bu sayede iklim bilimini geliştirerek toplumun ve doğanın geleceğe uyum sağlamasında yardımcı olacak daha tutarlı iklim tahminleri üretmeyi hedefliyor (Web Kaynağı 3.7).

2. TASARLA

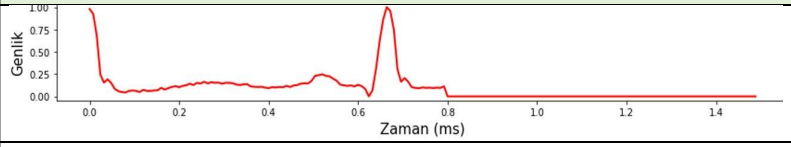
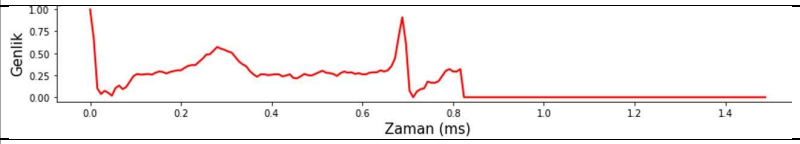
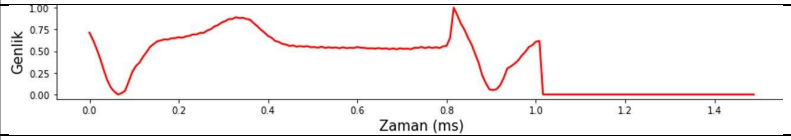
Öğrenciler, ilk olarak bayes öğrenme ile Elektro Kardiyografi (EKG) sinyallerinin değerlendirilme problemini anlar. Şekil 3.11'de gösterilen EKG sinyallerinin özelliklerine göre sinyalinin türlerini modeller.



Şekil 3.11. EKG sinyallerin türleri

Öğrenciler Çizelge 3.1'de verilen Bayes öğrenme ile EKG sinyalleri için giriş çıkış parametresini belirler.

Çizelge 3.1. Bayes öğrenme için giriş ve çıkış parametreleri

Veri Sayısı	GİRİŞ PARAMETRESİ	ÇIKIŞ PARAMETRESİ	
	EKG Sinyali	EKG Sinyal Türü	
1		Normal Sinüs Ritmi	TEST VERİ SETİ
…	…	…	
…	…	…	
18200		Supraventriküler erken atım	
…	…	…	
…	…	…	
21892		Sınıflandırılamayan atım	
…	…	…	EĞİTİM VERİ SETİ
…	…	…	
…	…	…	
109446		Sınıflandırılmayan ritim	

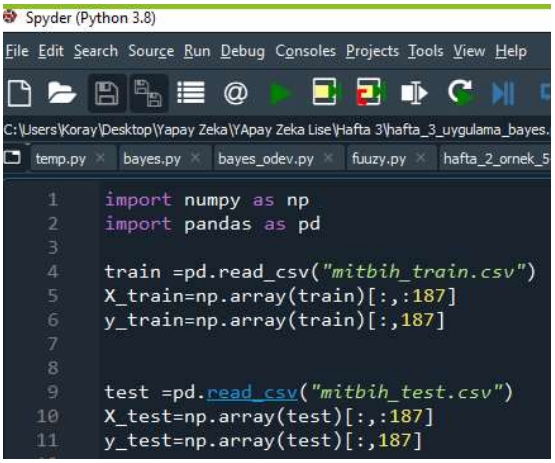
Bu bölümde EKG sinyallerine ait 21892 (Test verisi) 87554 (Eğitim verisi) olmak üzere toplam 109446 adet veri setinde (Web Kaynağı 3.8) EKG sinyali giriş parametresine göre EKG sinyal türünün makine öğrenmesi ile tahminlenmesi amaçlanmıştır.

VERİ SETİ İÇİN İNDİRME LİNKİ

https://www.kaggle.com/shayanfazeli/heartbeat
(Uygulama kapsamında kullanılan veri setleri dosyaları: mitbih_train.csv ve mitbih_test.csv şeklindedir.)

3. HAREKETE GEÇ

Öğrenciler, verilen mitbih_test.csv ve mitbih_train.csv isimli veri seti dosyalarını basit bir Bayes öğrenme tahminleme işlemi yapar. Tasarım aşamasında EKG sinyaline göre EKG sinyal türünü tahmin etmeye çalışır. Şekil 3.12'de gösterildiği gibi mitbih_test.csv ve mitbih_train.csv dosyaları yüklemek için gerekli kütüphane komutlarını yazar (İlgili kod Github platformunda Hafta3 klasörü altında H3_bayes_kalp.py dosyası ile sunulmuş durumdadır.).

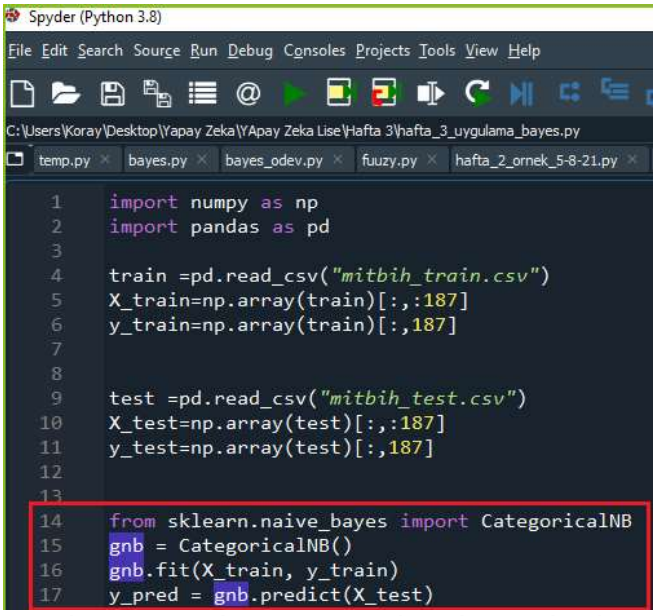


Şekil 3.12. Gerekli kütüphaneleri ve komutların gösterimi

Öğrenciler mitbih_train.csv ve mitbih_test.csv dosyaları içerisinde yer alan toplam 188 adet sütun verisi mevcuttur. Bu verilerden ilk 187 adeti EKG giriş sinyaline ait sayısal giriş değerleridir. 188. (son sütun) ise, EKG sinyalinin türünü ifade etmektedir. Söz konusu 5,6 (eğitim) ve 10,11 (test) numaralı satırlarda yazılan kod satırlarında 187 değeri giriş parametrelerine ait sayısal değerleri içerir.

4. YÜRÜT

Öğrenciler 14 numaralı kod satırında EKG sinyalinin özelliklerine göre EKG sinyal türü için oluşturmuş olduğu bayes öğrenme algoritmasını kullanabilmek için **"sklearn.naive_bayes"** kütüphanesinde yer alan **"CategoricalNB"** özellik fonksiyonunu çağırır. 15 numaralı satırda **CategoricalNB()** fonksiyonu "**model**" isimli bir değişkene aktarılır. 16 numaralı satırda, **"gnb.fit"** komutu yazarak giriş eğitim verilerine göre çıkış eğitim verileri için Bayes algoritması ile eğitim gerçekleştirir. 17 numaralı kod satırında ise, Bayes algoritması ile eğitim işlemi sonunda giriş test verilerine göre satış tahminini gerçekleştirir.

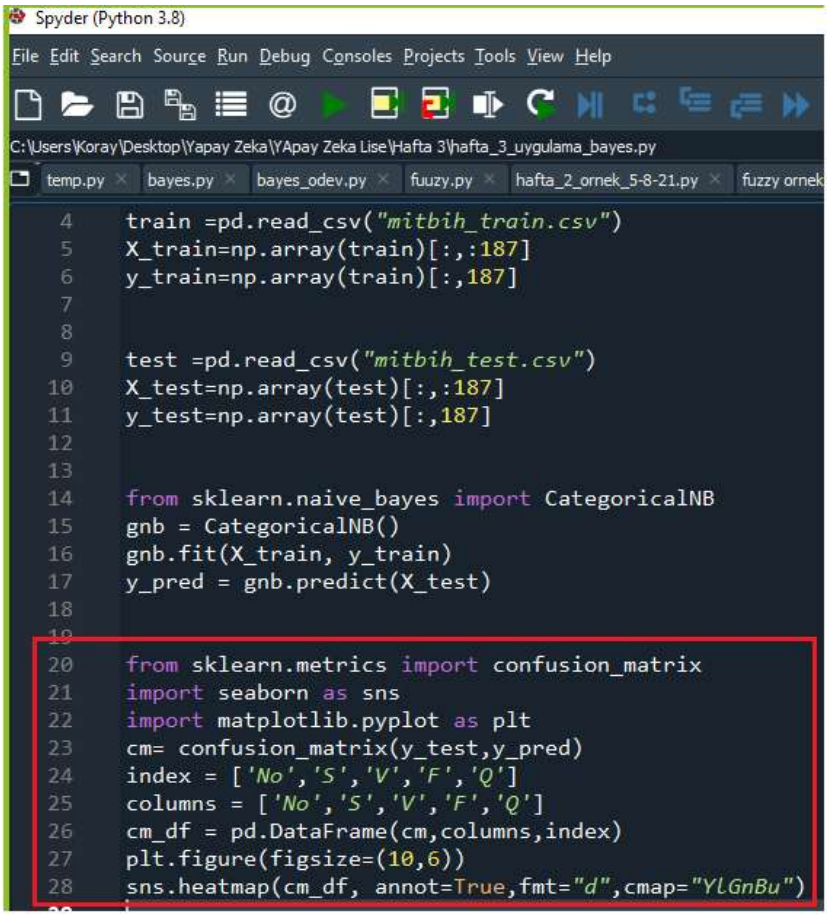


Şekil 3.13. Bayes algoritmasında eğitim ve tahminleme işlemleri için kod ekranı

5. KARAR VER

5.1. Tahmin-Test Sonuçlarını Karşılaştırma

Öğrenciler, Bayes öğrenme modeli ile eğitilen EKG sinyali özelliklerine göre EKG sinyal türünü tahminlemesine ait hata matrisini (confusion matrix) kullanarak modelin başarısını ölçer. Şekil 3.14'te gösterildiği gibi 20 numaralı komut satırında, sklearn.metrics" kütüphanesinden confusion_matrix özelliği yüklenir. 21 numaralı satırda ise, hata matrisini görüntüleyebilmek için gerekli olan seaborn kütüphanesini yükler. 22 numaralı satırda ise, hata matrisindeki grafikleri çizebilmek için **matplotlib** kütüphanesi içerisinde yer alan **pyplot** fonksiyonunu yükler. 23 numaralı kod satırında ise, **"cm değişkeni"** ile oluşturulan hata matrisinin satır (y_test) ve sütunları (y_pred) oluşturur. 24 ve 25 numaralı kod satırlarında ise index ve colums değişkenleri kullanarak hata matrisinde yer alacak olan metinleri belirler. 26 numaralı kod satırında **cm_df** değişkeni ile "**y_test"** ve "**y_pred"** değerlerinin veri çerçevesine (DataFrame) aktarılmasını sağlar. 27 numaralı kod satırında ise **"plt.figure"** komutu ile 10x6 cm çerçeve boyutunda boş bir çizim ekranı açar. 28 numaralı kod satırında ise **"sns.heatmap** komutu ile oluşturulan veri çerçevesini renkli olarak çizer. Burada annot=True ile sayısal değerler gösterilirken, fmt="d" sayısal değerler tam sayı olarak ve cmap="YlGnBu" ile yeşil gri ve mavi renkler ile gösterilmiştir.



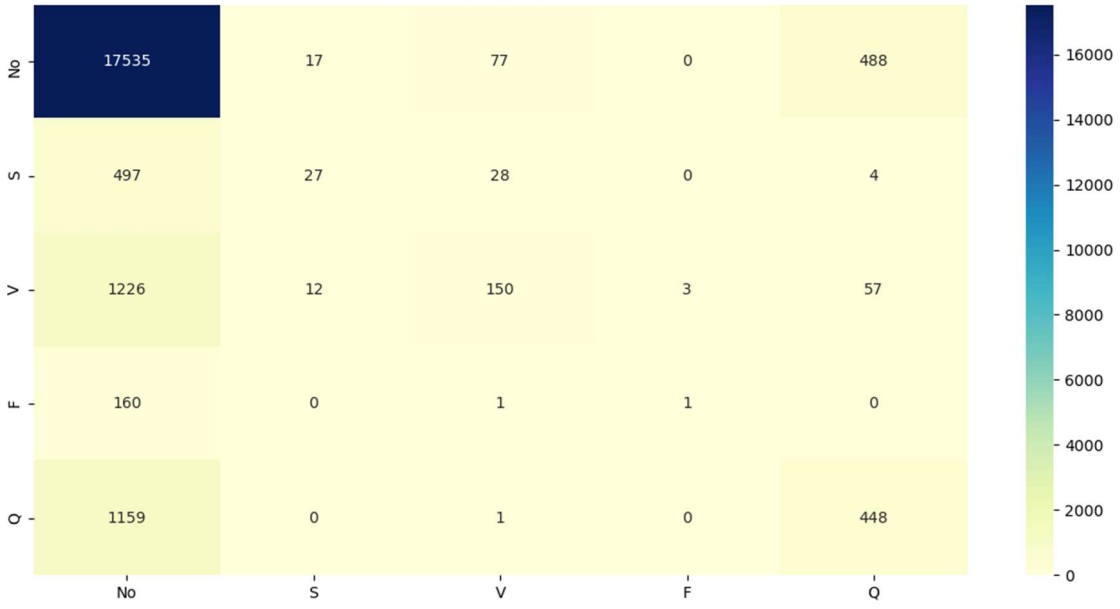
Şekil 3.14. Bayes algoritmasında eğitim ve tahminleme işlemleri için kod ekranı

5.2. Hata Matrisini Değerlendirme

Öğrenciler, EKG sinyal özelliklerine ait toplam 109.446 adet veri setinde (Kaggle, 2021) kayıt verinin %80'ini (87554) eğitim, %20'i (21892) test eğitim seti olarak ayırmıştı. Bayes modeli 87554 kayıt ile eğitilmişti. Geriye kalan 21892 kayıt (x_test) ise modeli tahmin etmek için kullandı (y_pred). Öğrenciler, Şekil 3.15'te gösterildiği gibi 21892 test kaydına ait gerçek sonuçlar (y_test) ile tahmin sonuçlarını karşılaştırdı ve hata matrisi elde etti. Böylece öğrenci, matristeki sayılar toplamının test verisi olan 21892'ye eşit olduğunu görmektedir.

Biliyor musunuz?

Yapay Zekâ, özellikle Makine Öğrenmesi algoritmaları açısından Veri Madenciliği alanıyla iş birliği içerisindedir (hatta sıklıkla karıştırılır). Aslında Veri Madenciliği daha çok verinin kullanımı ve olmayan bilgiye erişimi amaçlarken, Yapay Zekâ alanı ise zekice çözüm yapılabilmesi amacı taşımaktadır. Veri Madenciliği alanını Yapay Zekâ'dan ayıran konu *Birliktelik Kuralları*'dır.



Şekil 3.15. Hata matrisi

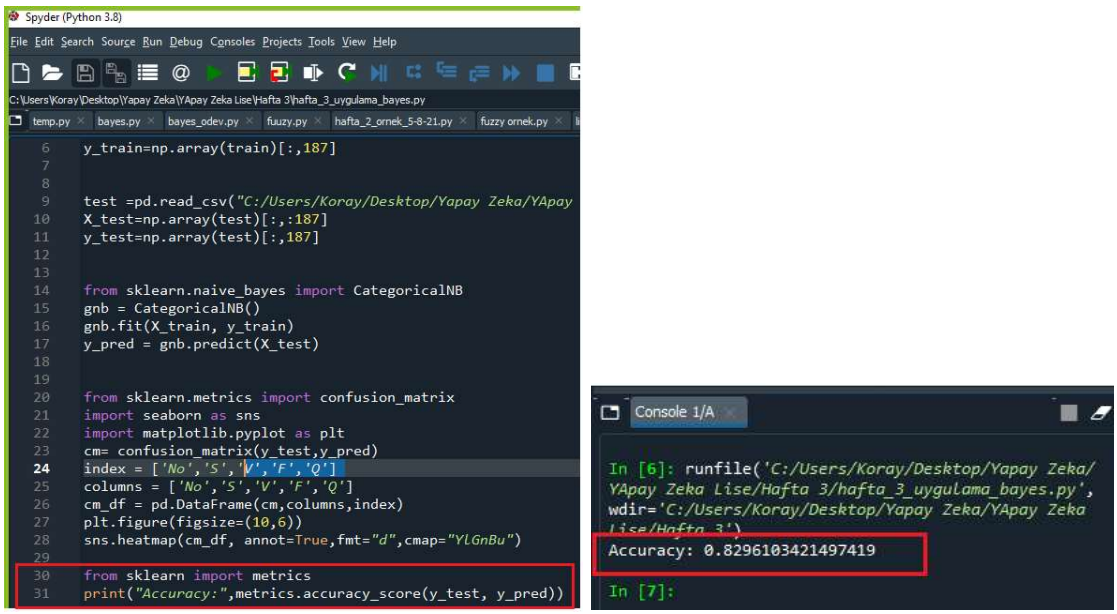
Eğitmene Not

Eğitmen, ilgili hata matrisinde satır ve sütunlarda verilen kısaltmalara ait ifadeleri öğrencilere açıklar.

No	S	V	F	Q
Normal Sinüs Ritmi	Supraventriküler erken atım	Erken ventriküler kasılma	Ventriküler ve normal atımın karışımı	Sınıflandırılamayan atım

Öğrenciler, hata matrisini incelediğinde 18117 adet test kaydında EKG sinyal türüne ait 17535 adetini "Normal Sinüs Ritmi (No)" doğru, 582 adetinin ise yanlış tahmin edildiğini görmüştür. 556 adet "Supraventriküler erken atım (S)" test verisinden 27 adetini doğru, 529 adeti yanlış tahminlendiğini görür. 1448 adet "Erken ventriküler kasılma (V)" test verisinden 150 adetini doğru, 1298 adeti yanlış tahminlendiğini görür. 162 adet "Ventriküler ve normal atımın karışımı (F)" test verisinden 1 adetini doğru, 149 adeti yanlış tahminlendiğini görür. Son olarak 1608 adet test verisinden 448 doğru, 1160 adetini yanlış olarak tahminlediğini görür.

Öğrenciler Şekil 3.16'da gösterildiği gibi 30 numaralı kod satırında bayes öğrenme modeli ile eğitilen modelin doğruluğunu değerlendirmek için **"sklearn"** kütüphanesinden **"metrics"** fonksiyon özelliğini yükler. 31 numaralı kod satırında **"print"** komutu kullanarak **"y_test"** veri setindeki veriler ile **"y_pred"** işlemi ile modelin doğruluğunun konsol ekranında %82,9 başarım oranında tahminlendiğini görür.



Şekil 3.16. Bayes öğrenme ile araç satış durumunu için a) model doğruluk belirleme kod ekranı b) model doğruluk sonucu

Eğitmene Not

Eğitmen, burada elde edilmiş olan başarım değerinin nasıl artırılacağını öğrencilerle tartışır ve kodlar üzerinde düzenlemelerle daha yüksek başarı değerleri yakalanmaya çalışılır.

PYTHON KODLARI:

```
import numpy as np
import pandas as pd

train =pd.read_csv("mitbih_train.csv")
X_train=np.array(train)[:,:187]
y_train=np.array(train)[:,187]

test =pd.read_csv("mitbih_test.csv")
X_test=np.array(test)[:,:187]
y_test=np.array(test)[:,187]

from sklearn.naive_bayes import CategoricalNB
gnb = CategoricalNB()
gnb.fit(X_train, y_train)
y_pred = gnb.predict(X_test)

from sklearn.metrics import confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt
cm= confusion_matrix(y_test,y_pred)
index = ['No','S','V','F','Q']
columns = ['No','S','V','F','Q']
cm_df = pd.DataFrame(cm,columns,index)
plt.figure(figsize=(10,6))
sns.heatmap(cm_df, annot=True,fmt="d",cmap="YlGnBu")
```

```
from sklearn import metrics
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

Düşün, tartış…

Makine Öğrenmesi'nin çalışma şekli insanların öğrenme şekline ne kadar benzerdir? Bu soruya yönelik öğrencilerin düşünceleri sınıf ortamında tartışılabilir ya da öğrenciler görüşleri hakkında notlar oluşturarak eğitmenin ve diğer öğrencilerin görüşleri ile karşılaştırabilir.

6. İLAVE ETKİNLİK

Şekil 3.17'de gösterildiği gibi hayvanat bahçesindeki verilen bilgilere göre hayvanın cinsini tahminleyen Bayes öğrenme modeli kurunuz (İlgili kod Github platformu Hafta 3 klasörü altında H3_bayes_hayvanat-bahcesi.py adlı dosya ile sunulmuş durumdadır).

Şekil 3.17. Hayvanat bahçesinden görüntü

VERİ SETİ İÇİN İNDİRME LİNKİ

https://github.com/deneyapyz/lise/Hafta3/hayvanatbahcesi.csv

Eğitmene Not

Eğitmen, öğrencilerden ilgili indirme linkini kullanmak suretiyle, orijinal veri setinin (https://archive.ics.uci.edu/ml/datasets/zoo) Türkçe'ye dönüştürülmüş versiyonunu indirmelerini ister:

sac varligi	Tuy varligi	yumurta	sut	havada yasami	suda yasami	yirtici	dis varligi
Omurga	nefes alimi	zehirli mi	yuzgecler	bacak sayisi	kuyruk	yerli	kedi boyu mu

Eğitmen, çizelgede verilen giriş değişkenlerin (bacak sayısı hariç) hayvanlarda var ise "1" yok ise "0" olduğunu öğrencilere açıklar. Bacak sayısı ise 0,2,4,6,8 şeklinde olduğunu belirtir.

Eğitmen öğrencilere, ilgili çizelgede verilen çıkış **"sınıf"** parametresindeki hayvan türlerine ait sınıfları açıklar:

1	2	3	4	5	6	7
Yer domuzu, antilop, ayı, domuz, bufalo, buzağı, tüylü kemirgen, çita, geyik, yunus, fil, meyve-yarasa, zürafa, kız, keçi, goril, hamster, tavşan, leopar, aslan, vaşak, vizon, köstebek, firavun faresi, keseli sıçan, ornitorenk, sansar, midilli, liman-yunusu, puma, kedi, rakun, ren geyiği, fok, fok balığı, sincap, vampir, tarla faresi, ufak kanguru, kurt	Tavuk, karga, güvercin, Ördek, flamingo, martı, Şahin, Kivi kuşu, tarlakuşu, devekuşu, muhabbet kuşu, penguen, Sülün, Amerika devekuşu, Deniz süpürücüsü, Korsanmartı, serçe, kuğu, akbaba, çalıkuşu	çukur engerek, deniz yılanı, yavaş solucan, kaplumbağa, tuatara	bas, sazan, yayın balığı, Şap, köpek balığı, mezgit balığı, ringa balığı, Turna, Pirana, Denizatı, Dilbalığı, Vatoz, Ton balığı	kurbağa, semender, kara kurbağası	pire, tatarcık, bal arısı, karasinek, uğur böceği, güve, termit, yaban arısı	deniz tarağı, yengeç, kerevit, ıstakoz, ahtapot, akrep, deniz arısı, sümüklü böcek, deniz yıldızı, solucan

Eğitmene Not

Eğitmen, ilgili hafta eğitimi hakkında öğrencilerin ilgisini arttırmak ve yeni öğrenilen bilgileri önceden öğrenilen bilgiler ile karşılaştırmalarını sağlamak için aşağıda maddeler halinde verilen soruları tartışır.

- İlgili haftada ele alınan yapay zekâ uygulamalarına uygun örnek veri setleri neler olabilir?
- İlgili haftada ele alınan veri setlerinin yapay zekâ teknik ve kütüphanelerinde ne derecede başarım etkisi olmuştur?
- İlgili haftada öğretilen yapay zekâ tekniği ile uyumlu ve uyumsuz olabilecek açık erişimli veri seti çeşitleri sunan (kaagle, github vs.) internet sitelerinden elde edilecek farklı veri çeşitleri ile yapay zekâ tekniği uyumluluğunu sorar.

PYTHON KODLARI:

```
import numpy as np
import pandas as pd
veri =pd.read_csv("hayvanatbahcesi.csv",encoding='unicode_escape')

girisler=np.array(veri.drop(["sinifi"],axis=1))
cikis=np.array(veri["sinifi"])

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(girisler,cikis,
test_size=0.35,random_state=109)

from sklearn.naive_bayes import CategoricalNB

gnb = CategoricalNB()
gnb.fit(X_train, y_train)
y_pred = gnb.predict(X_test)

from sklearn.metrics import confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt
cm= confusion_matrix(y_test,y_pred)
```

```
index = ['1','2','3','4','5','6','7']
columns =  ['1','2','3','4','5','6','7']
cm_df = pd.DataFrame(cm,columns,index)
plt.figure(figsize=(10,6))
sns.heatmap(cm_df, annot=True,fmt="d")

from sklearn import metrics
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

Kaynakça

Bilgin, M. (2017). Gerçek Veri Setlerinde Klasik Makine Öğrenmesi Yöntemlerinin Performans Analizi. Breast, 2(9), 683.

Çevik, K. K., & Kayakuş, M. (2020). Bilişim Teknolojileri Departmanında Kullanıcıların Taleplerine Cevap Verme Süresinin Makine Öğrenmesi İle Tahmin Edilmesi. Mühendislik Bilimleri Ve Tasarım Dergisi, 8(3), 728-739.

Şahinarslan, F. V. (2019). Makine Öğrenmesi Algoritmaları İle Nüfus Tahmini: Türkiye Örneği (Doctoral dissertation, Sosyal Bilimler Enstitüsü).

Web Kaynağı 3.1: https://blogs.sap.com/2015/01/20/why-airlines-need-to-keep-planes-in-the-air/

Web Kaynağı 3.2: https://www.domo.com/blog/data-never-sleeps-3-0/

Web Kaynağı 3.3: https://erdicenger.medium.com/big-data-nedir-fcd4a4a09d1

Web Kaynağı 3.4: https://isibilenesor.files.wordpress.com/2021/10/news-aus-1.jpg?w=848

Web Kaynağı 3.5: https://www.business2community.com/small-business/value-product-recommendation-marketing-small-businesses-0985412?hcb=1

Web Kaynağı 3.6: https://bilimgenc.tubitak.gov.tr/makale/robotlar-otizmli-cocuklarin-sosyal-becerilerini-gelistirmeye-yardimci-olabilir

Web Kaynağı 3.7: https://tr.euronews.com/green/2020/10/12/bozulan-iklimimizi-yapay-zeka-yard-m-yla-duzeltebilir-miyiz

Web Kaynağı 3.8: https://www.kaggle.com/shayanfazeli/heartbeat

4. Hafta: Karar Ağaçları ile Tutarlı Kararlar

Ön Bilgi:

- Karar ağaçları temelleri, Karar ağaçları tekniği, Karar ağaçları tekniğinde çözüm modelleri, Karar ağaçları ile pirinç yaprağı hastalıklarının teşhis uygulaması.
- Python kodlamada döngüler, fonksiyonlar ve kütüphane yapılarının kullanımı (Bu konuda Python anlatım videoları: 10-11. Döngüler, 13. Fonksiyonlar, 14-16. Python kütüphaneleri serisi yardımcı araç olarak kullanılabilecektir).

Haftanın Kazanımları:

- Öğrenciler, karar ağaçları algoritmasının temel yapısını kavrar.
- Öğrenciler, verileri karar ağaçları ile gerçek hayat problemlerine uygular.
- Öğrenciler karar ağaçları algoritmaları ile tahmin, sınıflandırma yöntemlerini bilir.
- Öğrenciler, Karar ağaçları üzerindeki koşul ifadeleri için 'düğüm' ve 'kök düğüm', Düğümler arası bağlantı için 'kenar' ve koşul belirtmeyen ifadeler için ise 'Yaprak' kavramlarını bilir.
- Öğrenciler Python programlama dilini kullanarak aracın özelliklerine göre pirinç yaprağı hastalıklarının teşhisi için program kodunu oluşturur.

Haftanın Amacı:

Bu haftanın amacı, "karar ağaçları algoritması" kavramının tüm öğrenciler tarafından doğru bir şekilde anlaşılmasını sağlamaktır. Haftanın bir diğer amacı, karar ağaçları algoritması kullanılarak farklı örnekler ile öğrencilerin ilgisini çekerek etkinliklere yönelik ilgilerini artırmaktır. Ayrıca, öğrencilere Python programı kullanarak karar ağaçları algoritmaları ile örnek modelleme ve çözüm üretme yetenekleri kazandırılacaktır.

Kullanılacak Malzemeler:

Bilgisayar, kâğıt, kalem, Python kod editörü, örnekler için çıktı görselleri

Haftanın İşlenişi:

Algıla: Öğrenciler karar ağaçları tekniğine ilişkin temel kavramları tanımlayabilir.

Tasarla: Öğrenciler karar ağaçları ile pirinç yaprak hastalıklarının teşhisi probleminin giriş ve çıkış parametrelerini tasarlayabilir. Bu hafta için Github platformu (https://github.com/deneyapyz/lise/) ile etkileşim Tasarla aşaması ile başlatılabilmektedir.

Harekete Geç: Öğrenciler Python programlama dili ile veri setindeki görüntü değerlerini sayısallaştırıp eğitim ve test verilerini ayırarak karar ağaçları tabanlı modeller kurabilir.

Yürüt: Eğitmenler öğrenciler ile etkileşimli bir biçimde karar ağaçları modelini eğitir ve pirinç yaprak hastalıklarının teşhisi için tahmin durumunu belirler. Öğrenciler ilgili problem doğrultusunda genel kodlama ve model yürütme süreçlerini uygulayabilir.

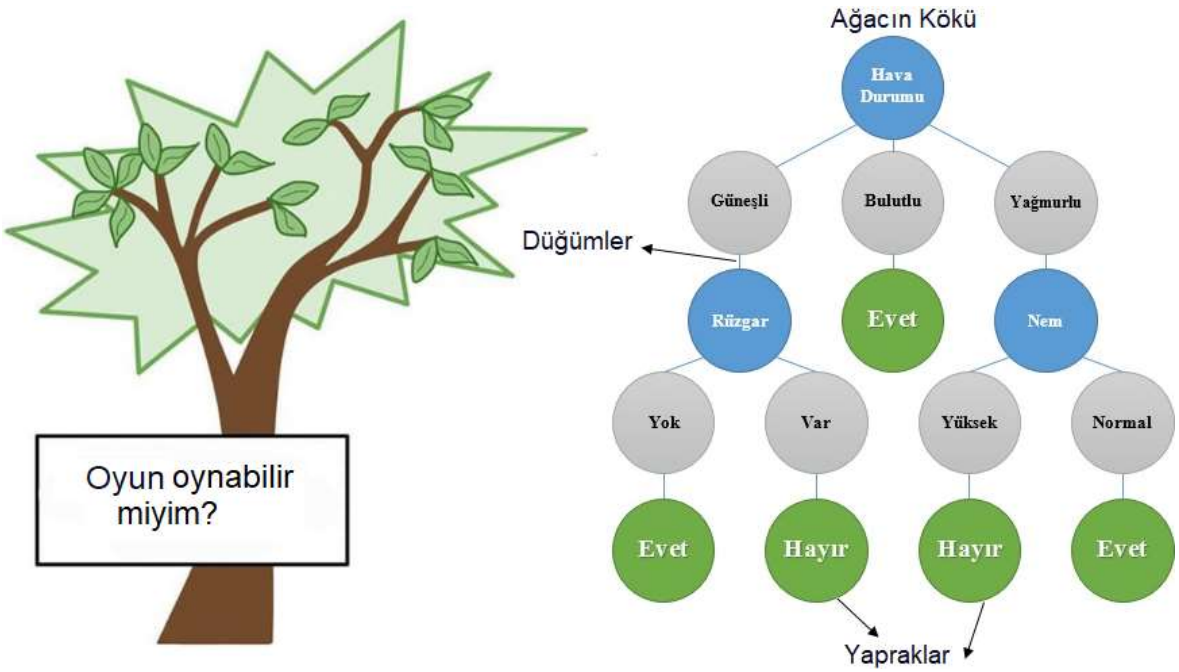
Karar Ver: Öğrenciler alternatif yapay zekâ modelini düzenleyerek / kodlayarak çözüm üretebilir. Öğrenciler özellikle görsel veriler üzerindeki uygulamaların işleyişini inceleyip ve tartışabilir.

1. ALGILA

1.1. Karar Ağaçları Algoritması Temelleri

Karar ağaçları, sınıfları bilinen örnek veriden karar düğümleri (decision nodes) ve yaprak düğümleri (leaf nodes) oluşturarak ağaç şekilli bir karar akışı çıktısı veren yapay zekâ algoritmasıdır (SPSS, 1999). Karar ağaçları algoritması, veri setini bölüp küçülterek geliştirilen bir yöntemdir. Karar düğümleri bir veya birden fazla daldan meydana gelebilir. İlk düğüme kök düğüm (root node) denir. Karar ağacı algoritmaları hem metinsel hem de sayısal verilerden oluşabilir (Web Kaynağı 4.1).

Şekil 4.1'de gösterildiği gibi havanın durumuna göre öğrencilerin okulun bahçesinde oyun oynayıp oynayamayacağını belirlemek için karar ağaçları yapısı gösterilmiştir. Burada karar ağacının algoritmasının kökü, hava durumudur. Karar ağacının düğümleri ise güneş, bulut ve yağmurdur. Rüzgâr ve havanın nemi de koşul ifadeleridir. Karar ağacının yaprakları ise öğrencilerin okulun bahçesinde oyun oynayıp oynamayacağını belirtir.



Şekil 4.1. Karar ağaçlarının yapısı (Web Kaynağı 4.1)

Karar ağaçları avantaj ve dezavantajları şu şekilde ifade edilebilir:

Avantajlar

- Ağaç yapılarının yorumlanması ve görselleştirilmesi kolaydır.
- Hem metinsel hem de sayısal veriler analiz edilebilir.

- Karar ağaçları, giriş parametrelerine bağlı çarpanlardan oluşan denklem şeklinde değil, koşullu bir çıkış modeli verir. Bu nedenle modelin çalışması hızlıdır.
- Yüksek miktarda veriye ihtiyaç duymadan model eğitimi gerçekleştirilebilir.
- Birden fazla çıkış parametresine sahip problemleri çözebilir.

Dezavantajlar

- Verilerin analizi yapılırken otomatik oluşan büyük ağaç yapılarının aşırı karmaşık olmasından dolayı ağaç dallarının takibi zordur.
- Aşırı öğrenme (over fitting) yaşanabilir.

Eğitmene Not

Eğitmen, öğrencilere aşırı öğrenme (over fitting) kavramını şu şekilde açıklar:

Algoritmanın eğitim verisi üzerinden veriyi ezberlemesine verilen tekniğin adıdır. Aşırı öğrenmede eğitim verilerinden elde edilen sonuç yüksek olurken, eğitim verisinden farklı bir veri modele verildiğinde hatalı sonuçlar üretir. Kısaca model kararlı bir yapıya sahip değildir. Model underfitting olmuşsa, verilerin altında yatan mantığı kavrayamamış demektir. Model bu veriler ile ne yapacağını bilemez ve doğru olmayan sonuçlar verir (Web Kaynağı 4.2).

Dünyadan Haberler

Mucit Yapay Zekâ

Surrey Üniversitesinde profesör olan Ryan Abbot tarafından 2019'da başlatılan *The Artificial Inventor* adlı proje, yapay zekâ ve yasanın yollarının kesişmesine odaklanıyordu. DABUS olarak adlandırılan bir sistem için projenin başlamasıyla birlikte 2 patent başvurusunda bulunulmuştu. Ayarlanabilir bir yemek kabı ve acil durum işaret ışığı olan icatların patentleri, mucit olarak DABUS'u gösteriyordu.

Dr. Stephen Thaler tarafından geliştirilen DABUS adlı sistem, sahip olduğu trilyonlarca ağ sayesinde büyük fikir ve icatlar öne sürebiliyordu. Fakat Amerika başta olmak üzere birçok ülke, yapay zekânın mucit olamayacağını söyleyerek patent tekliflerini geri çevirdi. Bunun üzerine davalar açan Thaler ve Abbott, son gelişmelerle birlikte bekledikleri sonuca yavaş yavaş ulaşacak gibi görünüyorlar.

Başlangıçta Avustralya da bu fikre karşı çıkıyordu. Fakat Avustralya Federal Mahkemesinden Jonathan Beach, bu durum karşısındaki tavrın değişmesine karar verdi. Beach'in vardığı karara göre DABUS, mucit olarak görülecek, fakat patent için aday olamayacak veya başvuramayacak. Bu noktada patentin sahibi, DABUS'un geliştiricisi Thaler olacak.

Beach, yaptığı açıklamada kendi görüşüne göre yapay zekâ sisteminin veya cihazın mucit olarak tanınabileceğini söylüyor: "İkimiz de yaratıldık ve yaratıyoruz. Bizim yarattıklarımız neden yaratamasın? (Web Kaynağı 4.3)

Biliyor musunuz?

Karar Ağaçları ile karar verme aşamasında hangi durumların birbirini takip edeceğine Entropi adı verilen bilgi kazancı hesabıyla karar verilir. Bizi karara ulaştırabilen her durumdan hangisinin Entropi değeri yüksek çıkarsa ağaçta sıradaki düğüme o eklenir.

Entropi formülü şu şekildedir:

$$I(x) = \log \frac{1}{P(x)} = -\log P(x)$$

P ile gösterilen değerimiz, örneğin spor oynama kararını belirleyen sıcaklık, nem ve hava tahmini adlı giriş durumlarından her biri için yerine konularak ayrı ayrı *I(x)* değerleri bulunur. Hangi durumun değeri yüksek çıkarsa ağaç düğümüne o durum eklenir. Ardından eklenen durumun herhangi bir alt değeri/sınıfı (örneğin sıcaklık durumu için düşük, orta ve yüksek şeklindeki sınıflardan biri) için veri seti filtrelenerek, elimizde kalan alt veri kümesi için Entropi hesabı tekrarlanır. İşlemler ve algoritma süreci, değerlerini hesaplayacağımız durum ve sınıflar kalmayıncaya kadar devam eder.

1.2. Karar Ağacı Tekniği

Karar ağacı tekniğini kullanarak verinin sınıflanması, öğrenme ve sınıflama olmak üzere iki basamaklı bir işlemdir. Yaygın olarak bilinen karar ağacı algoritmaları şu şekildedir:

1.2.1. ID3 Karar Ağacı Algoritması

ID3, yapay zekâ alanında çalışmaya ilk başlayan öğrencilerin karar ağaçlarının temel çalışma şeklini kolayca öğrenebilecekleri ideal bir algoritmadır. ID3 karar ağaç algoritmasının C4.5 ve C5.0 isminde iki tane versiyonu sıklıkla kullanılmaktadır. ID3 karar ağacı algoritmasında her düğümden çıkan dallar ile karar ağacı oluşmaktadır. Ağaçtaki dalların sayısı algoritmada tahmin edilecek sınıf sayısına eşittir. Karar ağacı algoritmasında yapraktaki hata (error) oranına göre budama işlemi yapılır.

Eğitmene Not

Eğitmen, öğrencilere karar ağaçları ile ilgili aşağıda verilmiş olan örneği tartışarak çözer. Bir şirketin A, B, C ürünleri ile ilgili satış detayları Aşağıdaki Tablo'da verilmiştir. Bu ürünler ile aşağıda belirlenen özelliklerde ürünler ile aksesuar alma kararı da sette tanımlanmıştır. Tabloya uygun olarak basit bir karar ağaç yapısı oluşturunuz.

No	Tip	Ürün Sayısı	Ciro	Yurtiçi	Aksesuar
1	A	<=5	düşük	evet	hayır
2	A	<=5	düşük	evet	hayır
No	**Tip**	**Ürün Sayısı**	**Ciro**	**Yurtiçi**	**Aksesuar**
3	B	5…10	orta	evet	evet

4	A	<=5	orta	evet	hayır
5	A	<=5	yüksek	evet	hayır
6	B	>10	orta	evet	evet
7	A	<=5	orta	evet	evet
8	A	<=5	orta	evet	hayır
9	C	<=5	orta	evet	evet

1.2.2. C&RT Karar Ağacı Algoritması

Gini indeksi (dizini) veya Gini katsayısı, İtalyan istatistikçi Corrado Gini tarafından 1912'de geliştirilen istatistiksel bir ölçüdür. Gini'ye dayalı ikili bölme işlemine göre çalışan bir karar ağacı algoritmasıdır. Bu algoritmada en son veya uçta olmayan her bir düğümde iki adet dal vardır. Hem Sınıflandırma hem de regresyon (sayısal sonuç) uygulamalarında kullanılır. Budama işlemi oluşturulan karar ağacı yapısına göre değişiklik gösterir.

1.2.3. CHAID Karar Ağacı Algoritması

Karar ağacı CHAID algoritması, istatistik tabanlı olarak G. V. Kass tarafından 1980'de geliştirilmiştir. Sınıflandırma ve regresyon uygulamalarında tercih edilir. CHAID algoritması, bağımsız değişkenlerin, birbirleriyle olan etkileşimini bulan bir tekniktir. CHAID algoritması dallanma kriterinde bağımlı değişken kategorik ise iki ya da daha çok grup arasında fark olup olmadığını tespit eden Ki-kare testine göre bölme işlemini gerçekleştirir.

1.2.4. SPRINT Karar Ağacı Algoritması

SPRINT algoritması 1996 yılında Shafer, Agrawal ve Mehta tarafından geliştirilip entropiye dayanmaktadır. SPRINT karar ağaçları algoritması büyük veri kümeleri için ideal bir algoritmadır. Ağaç yapısında en iyi dallanma için her bir değişkene ait özellikleri bir kez sıraya dizer ve karar ağaçı yapısı bu şekilde oluşur. Bu algoritmada her bir değişken için ayrı bir değişken listesi hazırlanır. Bölme işlemi tek bir özelliğin değerine göre saptanır.

1.2.5. SLIQ Karar Ağacı Algoritması

SLIQ karar ağacı algoritması 1996 yılında Agrawal, Mehta ve Rissanen tarafından geliştirilmiştir. Bu algoritma Gini tekniği ile nicel ve nitel veri tipleri kullanılabilmektedir. Ayrıca verilerin sıralanması aşamasında en iyi dallara ayırma tekniğini uygulamaktadır. Bu algoritma hızlı ölçüm yapan bir sınıflandırıcıya ve hızlı ağaç budama algoritmasına sahiptir.

Düşün, tartış...

Yapay Zekâ'nın karar verme süreçlerinde biz insanları desteklemesi her zaman faydalı olur mu? Olası riskleri var mıdır? Sorulara yönelik öğrenci görüşleri sınıf ortamında tartışılabilir ve bu konuda Web ortamındaki alternatif düşünceler araştırılarak yorumlanabilir.

2. TASARLA

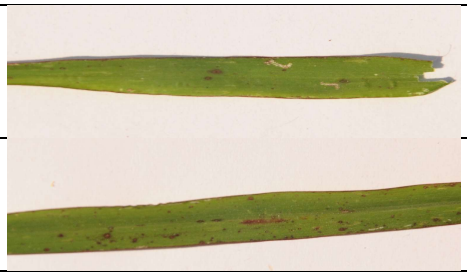
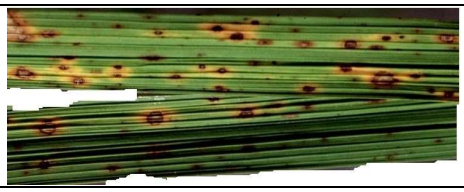
Öğrenciler, ilk olarak karar ağaçları ile pirinç yaprak hastalıkları değerlendirilmesi problemini anlar. Şekil 4.2'de gösterilen pirinç yaprak hastalıkları görüntüsüne göre hastalık türünü modeller.

Şekil 4.2. Pirinç yaprak hastalığına ait örnek görüntü

Öğrenciler Çizelge 4.1'de verilen karar ağaçları algoritması ile pirinç yaprak hastalıkları için giriş çıkış parametresini belirler.

Çizelge 4.1. Karar ağaçları algoritması için giriş ve çıkış parametreleri

Veri Sayısı	GİRİŞ PARAMETRESİ	ÇIKIŞ PARAMETRESİ
	Pirinç yaprak hastalıkları görüntüsü	Hastalık Türü
1		0 (Bakteriyel yaprak yanıklığı)
2		0 (Bakteriyel yaprak yanıklığı)
…	…	…
40		0 (Bakteriyel yaprak yanıklığı)
41		1 (Kahverengi nokta)
42		1 (Kahverengi nokta)
…	…	…
80		1 (Kahverengi nokta)

81		2 (Yaprak isi)
82		2 (Yaprak isi)
...		
120		2 (Yaprak isi)

Bu bölümde hastalara ait 120 adet veri setinde (Kaggle, 2021) pirinç yaprak hastalıkları görüntüsü giriş parametrelerine göre hastalık türü sınıfının karar ağaçları algoritması ile tahminlenmesi amaçlanmıştır.

Eğitmene Not

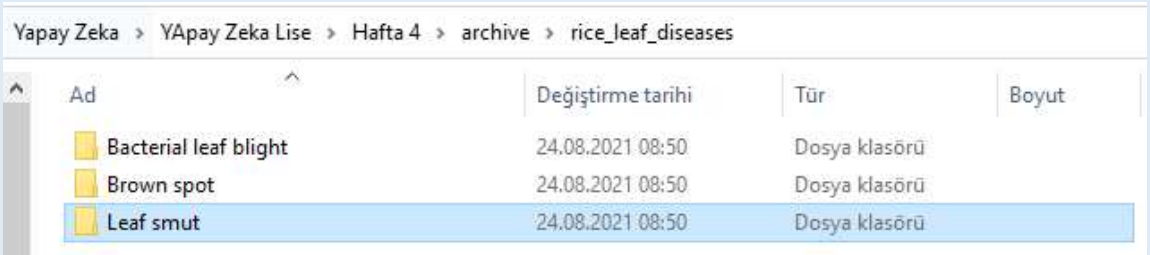
Eğitmen, öğrencilere Kaggle, Github gibi açık erişimli internet sitelerinin tüm dünyada veri seti olarak ortak kullanıma açılmış olduğunu açıklar. Bu uygulamada kullanılacak veri seti Kaggle ortamında yer aldığı gibi aynı zamanda Github platformuna da eklenmiştir. Eğitmen uygulamalarda bu tür platformlarla etkileşimi vurgular.

VERİ SETİ İÇİN İNDİRME LİNKİ

https://www.kaggle.com/vbookshelf/rice-leaf-diseases/download

Eğitmene Not

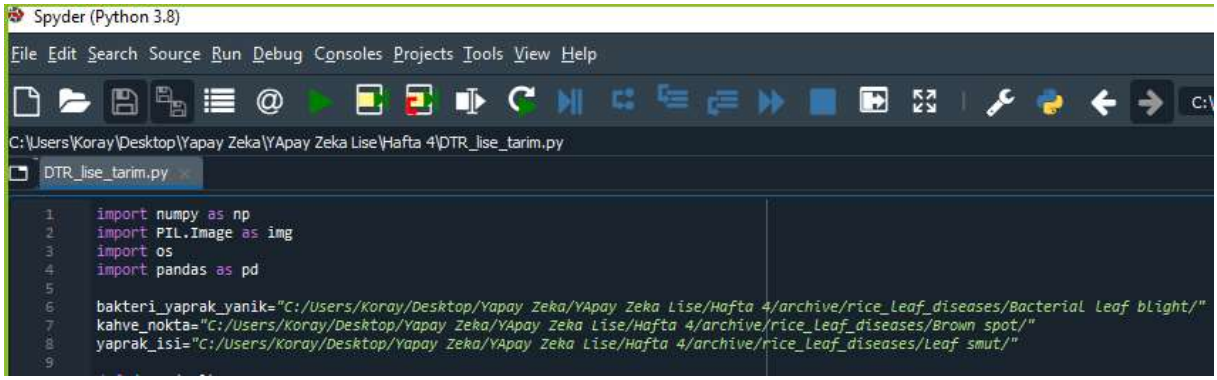
Eğitmen, Şekil 4.3'te gösterildiği gibi öğrencilere veri setini anlatırken veri seti indirme linkinden indirilen verilerin Bacterial leaf blight (Bakteriyel yaprak yanıklığı), Brown spot (Kahverengi nokta) ve Leaf smut (Yaprak isi) verileri olarak ayrı bir şekilde dikkate alındığını anlatır (Bu noktada, anlatılan kodlarla uyumlu olması için indirilen veriler bitki_veri-seti adlı bir klasör altında toplanabilir).



Şekil 4.3. Veri seti klasörleri

3. HAREKETE GEÇ

Öğrenciler, ilgili veri seti dosyasını basit bir karar ağaçları algoritması ile pirinç yaprak hastalıkları teşhisine yönelik tahminleme işlemi yapar. Şekil 4.4'te gösterildiği gibi "veri seti" dosyasını yüklemek için gerekli kütüphaneleri ve komutları yazar.



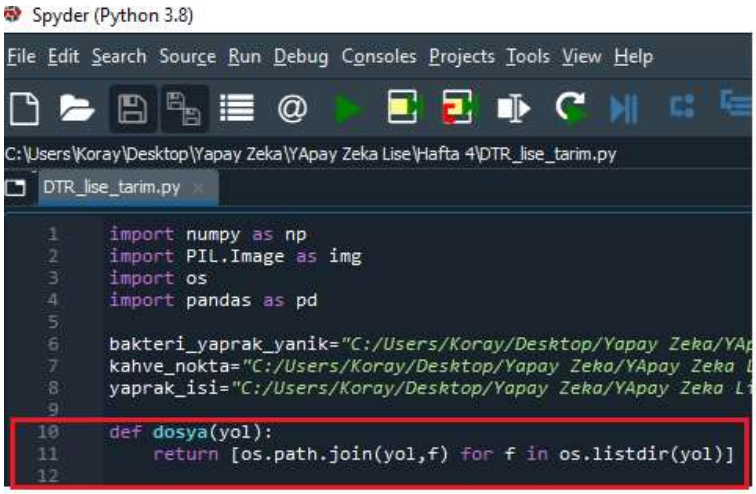
Şekil 4.4. Gerekli kütüphaneleri ve komutların kod gösterimi

Eğitmene Not

Eğitmen, Python programlama dilinde 6., 7. ve 8. satırda kod yazarken veri seti klasörü konumunun her bir öğrencinin bilgisayarında farklılık göstereceğini bilir. Bu nedenle eğitmen, öğrencilere Şekil 4.5'te gösterildiği gibi veri setinin konum adresini belirlemeyi öğretir.

Şekil 4.5. Veri seti klasör konumu

Öğrenciler Şekil 4.6'da görüldüğü gibi, **operating system** kütüphanesini kullanmak suretiyle, ilgili klasördeki pirinç yaprak hastalığı görüntülerinin okunmasını sağlar.



Şekil 4.6. Pirinç hastalık görüntülerinin konumları liste halinde çağırımı kod ekranı

Şekil 4.7'de gösterildiği gibi pirinç yaprağı hastalık türleri verilerini dönüştürmek için **"klasor_adi"** ve **"sinif_adi"** parametrelerini içeren **"veri_donusturme"** isminde bir fonksiyon tanımlar.

15. kod satırında pirinç yaprağı hastalık türleri görüntülerinin bulunduğu klasörlerden tek tek okumak için **"goruntuler"** isminde bir değişken oluşturur.

17. Kod satırında pirinç yaprağı hastalık türleri görüntüleri etiketlemek için **"goruntu_sinif"** isminde boş bir dizi oluşturur.

18. kod satırda pirinç yaprağı hastalık türleri tüm görüntüleri işlemek için bir "**for**" döngüsü oluşturur.

19. Satırda tüm pirinç yaprağı hastalık türleri görüntüleri okunarak **".convert('L')"** özelliği kullanılarak gri tonlamalı görüntülere dönüştürülür.

20. Kod satırında **"goruntu_boyutlandirma"** değişkeni kullanılarak tüm pirinç yaprağı hastalık türleri görüntüleri **".resize"** komutu kullanılarak 28x28 piksel boyutuna küçültülür.

Eğitmene Not

Eğitmen, eğitim sürecini hızlı bir şekilde gerçekleştirmek için pirinç yaprağı hastalık türleri görüntü boyutunu 28x28 boyutuna düşürdüğünü öğrenciyle paylaşır.

21. Kod satırında iki boyutlu olan pirinç yaprağı hastalık türleri görüntülerini karar ağaçları ile eğitebilmek için **".flatten"** özelliği kullanılarak görüntüler 784 elemanlı tek boyutlu bir diziye (vektör) dönüştürülür.

Eğitmene Not

Eğitmen, öğrenciye 784 rakamını nasıl elde edildiğini sorar ve tartışır. 784 rakamı 28x28 görüntü boyutuna düşürülen iki boyutlu bir görüntü dizisinin tek boyuta düşürülmesi ile 28x28=784 elde edilir.

22.-33. kod satırları arasında, pirinç yaprağı hastalık türleri görüntüleri etiketlenmiştir. Etikletleme işleminde Bacterial Leaf Blight (Bakteriyel Yaprak Yanıklığı) hastalığı olanlar **"0"**, Brown Spot (Kahverengi Nokta) hastalığı olanlar **"1"** ve Leaf Smut (Yaprak İsi) hastalığı olanlar ise **"2"** şeklinde etiketlenmiştir. İlgili üç hastalık haricinde olan görüntülerde ise etiketleme yapılmamıştır.

35. Kod satırında ise etiketleme yapılan değişken olan **"goruntu_sinif"** değişkeni ile etiketleme işlemi tamamlanmıştır.

```
8   yaprak_isi="C:/Users/Koray/Desktop/Yapay Zeka/YApay Zeka Lise/Hafta 4/d
9
10  def dosya(yol):
11      return [os.path.join(yol,f) for f in os.listdir(yol)]
12
13  def veri_donusturme(klasor_adi,sinif_adi):
14
15      goruntuler=dosya(klasor_adi)
16
17      goruntu_sinif=[]
18      for goruntu in goruntuler:
19          goruntu_oku= img.open(goruntu).convert('L')
20          gorunu_boyutlandirma=goruntu_oku.resize((28,28))
21          goruntu_donusturme=np.array(gorunu_boyutlandirma).flatten()
22          if sinif_adi=="bakteri_yaprak_yanik":
23              veriler=np.append (goruntu_donusturme, [0])
24
25          elif sinif_adi=="kahve_nokta":
26              veriler=np.append (goruntu_donusturme, [1])
27
28          elif sinif_adi=="yaprak_isi":
29              veriler=np.append (goruntu_donusturme, [2])
30
31          else:
32              continue
33          goruntu_sinif.append(veriler)
34
35      return goruntu_sinif
36
```

Şekil 4.7. Veri setindeki metin değerlerinin sayısallaştırılması kod ekranı

Öğrenciler Şekil 4.8'de gösterilen 38. kod satırlarında **"yanik_veri"** veri değişkeni ile oluşturulmuş durumdaki **"veri_donusturme"** fonksiyonuna giderek bakteriyel yaprak yanıklığı olarak etiketlenen verileri **"yanik_veri"** değişkenine aktarır.

39. kod satırında ise bakteriyel yaprak yanıklığı verileri pandas kütüphanesinin **".DataFrame"** özelliği ile daha sade ve anlaşılabilir hale dönüştürülmüştür.

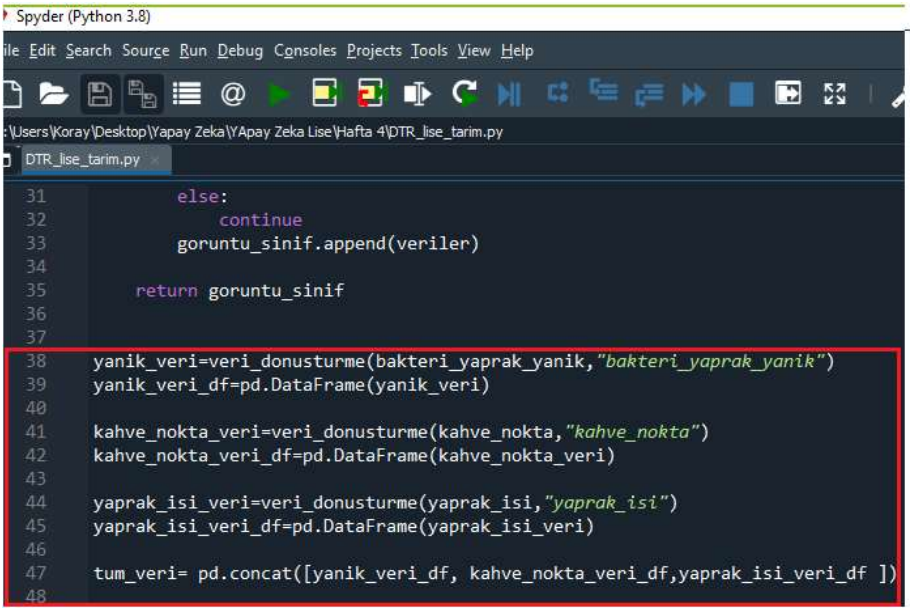
Eğitmene Not

Eğitmen, öğrencilere Pandas'daki iki temel veri yapısı olan, Seriler (1 boyutlu) ve DataFrame (2 boyutlu)'in, mühendislik gibi birçok alanda sıklıkla kullanıldığını aktarır. Pandas kütüphanesi, NumPy'ın üzerine inşa edilmiş bir kütüphanedir. Maddeler halinde verilen şu işlemleri gerçekleştirmekte kullanılır (Web Kaynağı 4.4):

- Kayıp verilerin tespit ve analizini kolaylaştırması,
- DataFrame'ler kullanılarak büyük boyutta sütunlar eklenerek veya kaldırılarak boyutların değiştirilmesi,
- Gruplama özelliği kullanılarak verilerin toplanması ve dönüştürülmesi işlemlerinde kolaylıklar sunması,

41. ve 42 kod satırlarında ve 44., 45. kod satırlarında yazılan kodlar 38. ve 39. kod satırlarında bakteriyel yaprak yanıklığı verileri için yapılan işlemler kahverengi nokta ve yaprak isi hastalıkları için de gerçekleştirilmiştir.

47. Kod satırında ise [38,39], [41,42] ve [44,45] kod satırlarındaki bakteriyel yaprak yanıklığı, kahverengi nokta ve yaprak isi hastalık verileri Pandas kütüphanesinin **".concat"** özelliği kullanılarak birleştirilerek **"tüm_veri"** değişkenine aktarılmıştır.



Şekil 4.8. Pirinç yaprağı hastalık görüntülerinin etiketlenerek birleştirilmesi

Öğrenciler 50. kod satırında pandas kütüphanesi çağırır.

51. Kod satırında "sklearn" kütüphanesi içerisinde yer alan "**DecisionTreeClassfier**" ile karar ağaçları algoritmaları ile eğitim gerçekleştirebilmek için **"sklearn.tree"** kütüphanesini çağırır.

52. kod satırında ise, veri setindeki verileri eğitim ve test olarak ayırabilmek için **"sklearn.model_selection"** kütüphanesinde yer alan **"train_test_split"** kütüphanesini çağırır.

53. Kod satırında "sklearn" kütüphanesi içerisinde yer alan "metrics" kütüphanesini kullanarak modeli değerlendirir.

54. Kod satırında ise, "**sklearn.model_selection"** kütüphanesinde yer alan **"GridSearchCV"** algoritmasını çağırır.

Eğitmene Not

Eğitmen, öğrenciye **"GridSearchCV"** algoritması hakkında bilgi sahibi olup olmadığını sorar ve tartışır.
"GridSearchCV" algoritması modelin hiper parametreleri değiştirilerek model doğruluğunu arttırmak için kullanılan bir algoritmadır.

56. ve 57. Kod satırlarında ise "**.flatten**" özelliği ile 784 elemanlı tek boyutlu bir vektöre dönüştürülen görüntüler **"Giris"** ve **"Cikis"** değişkenlerine aktarılmıştır.

58. satırda ise, veri setindeki eğitim ve test verilerini %80 eğitim, %20 test olacak şekilde **"test_size=0,2"** komutu kullanarak rastgele ayırır. Burada "**random_state=109"** ifadesi her eğitim tekrarında alınacak verilerin aynı kalmasını sağlar.

```
50  import pandas as pd
51  from sklearn.tree import DecisionTreeClassifier
52  from sklearn.model_selection import train_test_split
53  from sklearn import metrics
54  from sklearn.model_selection import GridSearchCV
55
56  Giris=np.array(tum_veri)[:,:784]
57  Cikis=np.array(tum_veri)[:,784]
58  Giris_train, Giris_test, Cikis_train, Cikis_test = train_test_split(Giris, Cikis, test_size=0.2, random_state=109)
59
```

Şekil 4.9. Veri setinin eğitim ve test olarak rastgele ayrılması için kod ekranı.

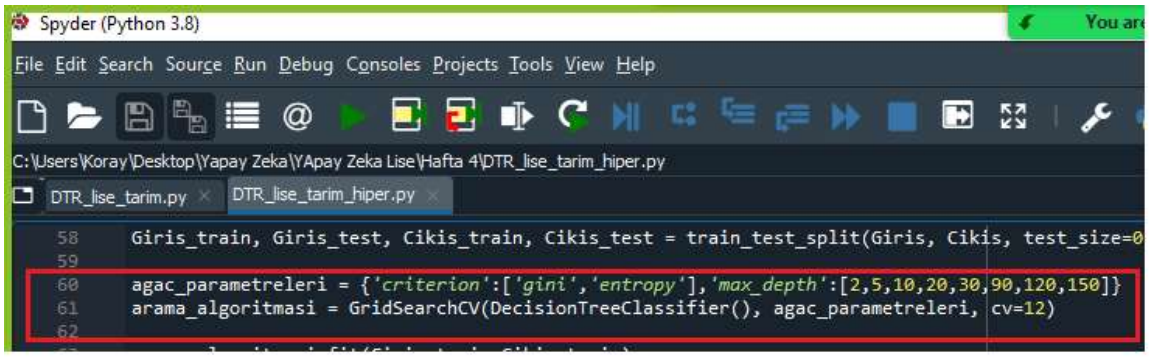
Kararlar Veren Makinelerin Çağı!

Düşün, tartış…

Yapay Zekâ sağlık alanında başarılı teşhisleriyle biliniyor… Sizce Yapay Zekâ'nın kararlarını doğrudan uygulamalı mıyız yoksa son karar verici insanlar mı olmalı? Soruya yönelik öğrenci görüşleri sınıf ortamında tartışılabilir ve bu konuda Web ortamındaki alternatif düşünceler araştırılarak yorumlanabilir.

4. YÜRÜT

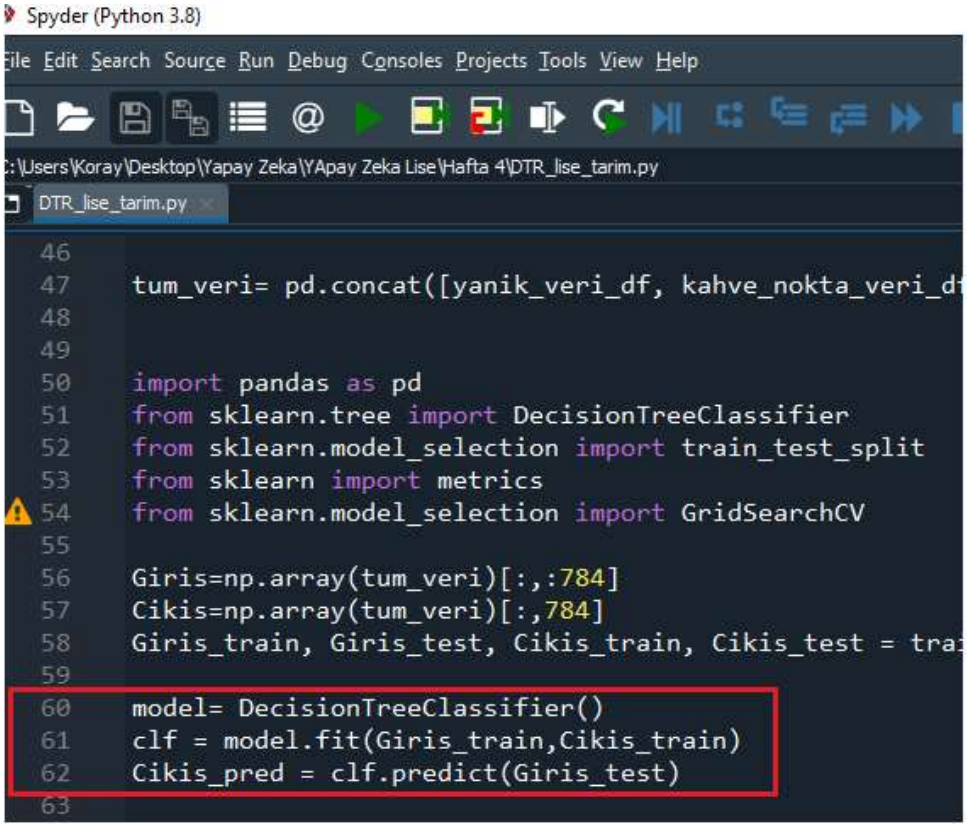
Öğrenciler Şekil 4.10'da gösterildiği gibi 60. kod satırında karar ağaçları algoritmasını **"DecisionTreeClassifier"** sınıflandırıcı ile model değişkenine aktarır.



Şekil 4.10. Karar ağaçları modelinin kurulması için kod ekranı

61. Kod satırında ise pirinç yaprak hastalığı görüntülerine göre hastalık türünün tahmini durumu için oluşturmuş olduğu karar ağaçları algoritmasını kullanabilmek için **"model.fit"** komutunu yazarak giriş eğitim verilerine göre çıkış eğitim verileri için eğitim gerçekleştirerek sonucu **"clf"** değişkenine aktarır.

62. Kod satırında ise karar ağaçları modelinden elde edilen eğitim sonuçlarını **"Giris_test"** verilerine göre **".predict"** özelliği ile tahmin edip, sonucu **"Cikis_pred"** değişkenine aktarır.



Şekil 4.11. Karar ağaçları algoritmasında eğitim ve tahminleme işlemleri için kod ekranı

5. KARAR VER

5.1. KARAR AĞACI MODELİN BAŞARIM ORANI

Öğrenciler yürüt aşamasında eğitmiş oldukları karar ağaçları modeline ait doğruluk sonucunu ve grafiklerini bu aşamada gerçekleştirir (İlgili kodun tamamı Github platformunda, Hafta 4 altında H4_kararagaci_bitki_1.py dosyası olarak paylaşılmış durumdadır). 64. Kod satırında karar ağaçları modeli ile eğitilen pirinç yaprağı hastalık görüntülerinin tahmin sonuçlarını konsol ekranında görmek için kodu yazar.

66. kod satırında grafik çizmek için **"matplotlib.pyplot"** kütüphanesini çağırır.

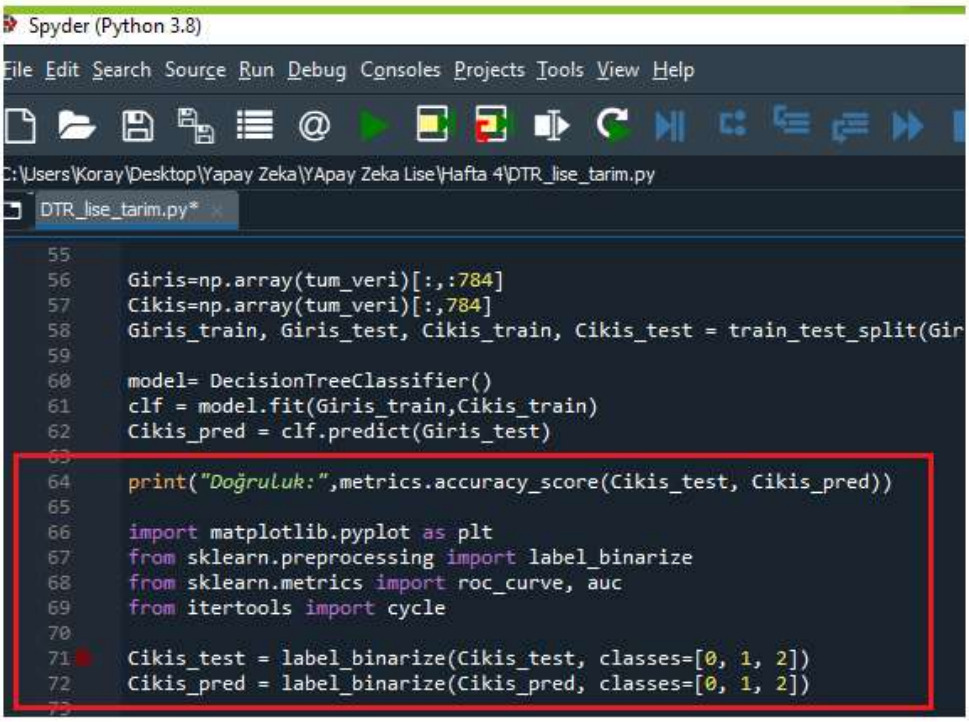
67. kod satırında sklearn kütüphanesinden pirinç yaprağı hastalığı tür sonuçlarını ikili sayı sistemine kodlamak için **"label_binarize"** alt sınıfını çağırır.

68. kod satırında sklearn kütüphanesinden ROC eğrisi ve ROC eğrisi altında kalan alanı hesaplayarak çizebilmek için **"roc, curve, auc"** alt sınıfını çağırır.

69. kod satırında, grafikte kullanılacak olan renkleri diziye dönüştürmek için **itertools** kütüphanesindeki **"cycle"** alt sınıfı çağırılır.

71. ve 72. kod satırlarının görevi, hastalık türlerine göre kodlama yöntemi oluşturmaktır. Buna göre Bacterial Leaf Blight (Bakteriyel Yaprak Yanıklığı) hastalığı olan ve **"0"** şeklinde temsil edilenler [0,0,1] olarak, Brown Spot (Kahverengi Nokta) hastalığı olan ve **"1"** şeklinde temsil

edilenler [0,1,0] olarak ve Leaf Smut (Yaprak İsi) hastalığı olup **"2"** şeklinde temsil edilenler de 1,0,0] şeklinde kodlanır. Bu kodlama yöntemi verilerin ROC eğrisinde ayrı ayrı çizilmesini sağlar.



Şekil 4.12. Modelin sınıflarının binary (ikili) kodlanması

74. Kod satırında ise 60x40 boyutunda ve 150 dpi çözünürlükte yeni bir boş çizim penceresi **"plt.figure"** komutu ile oluşturulur.

75. Kod satırında çalışmadaki grafik çizimi için problemdeki toplam sınıf sayısı olan 3 değeri **"n_classes"** değişkenine aktarılır.

76-78 kod satırlarında 80 ve 81. Kod satırlarında elde edilen değerleri kaydedilebilmek için **"fpr", "tpr"** ve **"roc_auc"** sözlükleri tanımlanmıştır.

79-80 kod satırlarında oluşturulan sınıf sayısı kadar dönen for döngüsü (her bir sınıf için ayrı ayrı) ile **"Cikis_test"** verilerine göre **"Cikis_pred"** ile karşılaştırıp ROC eğrisini çizebilmek için **"roc_curve"** özelliği kullanılarak yanlış pozitif oranı (fpr) , doğru pozitif oranı (tpr) değerlerini hesaplar.

Eğitmene Not

Eğitmen öğrencilere, yapay zekâ ile yapılan sınıflandırma uygulamalarında modellerin performansını ölçmek için doğruluk, kesinlik, duyarlılık, f-puanı ve alıcı işletim karakteristiği eğrisi (Receiver Operating Characteristic-ROC) gibi araçlar kullanıldığını anlatır. Bu aşamada, sınıflandırma modellerinde kullanılan dört farklı durum şu şekildedir:

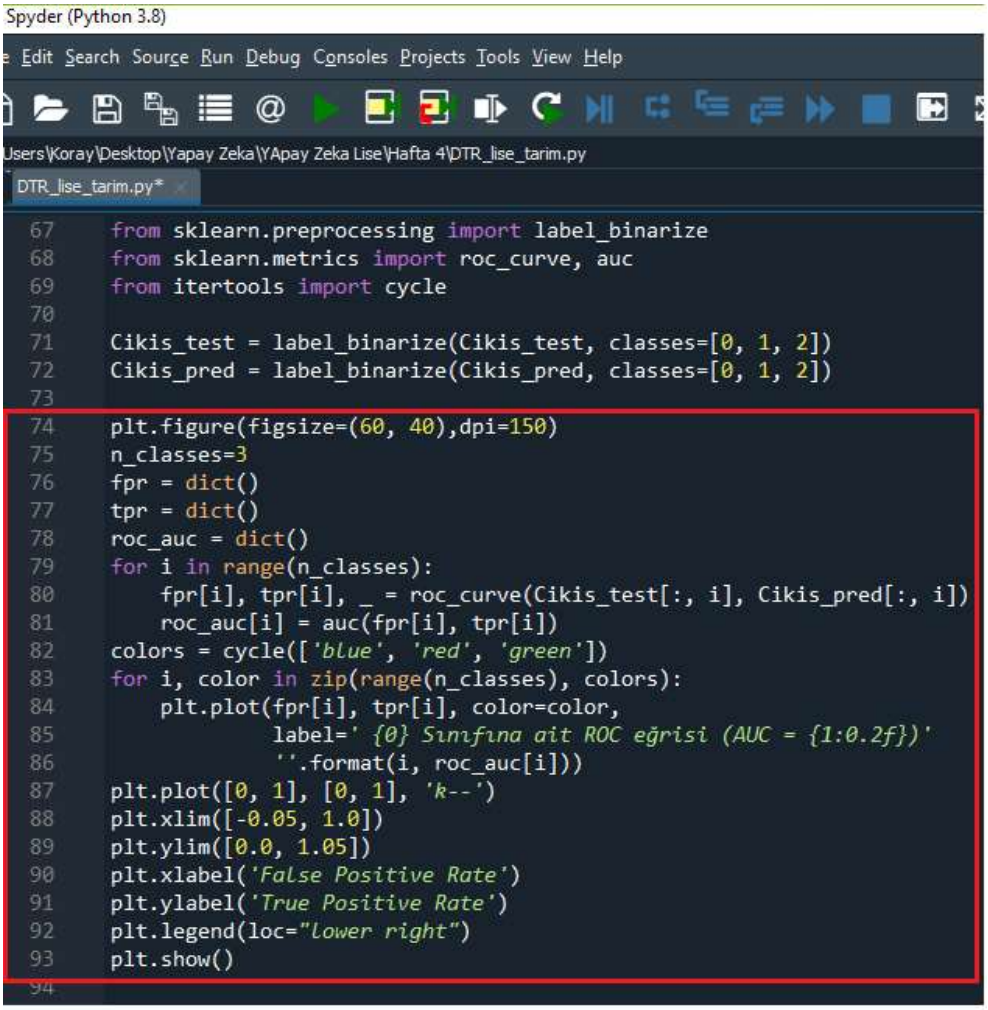
- Doğru Pozitif Oranı (tpr): Doğru olarak tahmin edilen pozitif sınıfların sayısı,
- Doğru Negatif Oranı (tnr): Doğru olarak tahmin edilen negatif sınıfların sayısı,
- Yanlış Pozitif Oranı (fpr): Yanlış olarak tahmin edilen pozitif sınıfların sayısı,
- Yanlış Negatif Oranı (fnr): Yanlış olarak tahmin edilen negatif sınıfların sayısı.

81. Kod satırında ise her bir sınıf için ayrı ayrı metrics kütüphanesinin **"auc"** özelliğini kullanarak yanlış pozitif oranı (fpr), doğru pozitif oranı (tpr) değerlerini hesaplayarak **"roc_auc"** sözlüğüne aktarır.

82. kod satırında ise, oluşturulan her bir sınıf için çizim renkleri belirlenmiştir. Bacterial leaf blight (Bakteriyel yaprak yanıklığı) hastalığı olanlar **"0" için mavi**, Brown spot (Kahverengi nokta) hastalığı olanlar ise **"1" için kırmızı** ve Leaf smut (Yaprak isi) hastalığı olanlar ise **"2" için yeşil** renkler belirlenmiştir.

83-86. kod satırlarında ise sınıf sayısı kadar çalışan for döngüsü sayesinde, her bir sınıfa yönelik ROC eğrileri **"plt.plot"** komutu ile (farklı renkler kullanmak suretiyle) çizilmiştir.

Öğrenciler, 87-93. satırlar arasında ROC eğrisi çizimine yönelik komutları yazar. Buna göre, eğrinin başlığı, yanlış pozitif oranın (fpr) doğru pozitif oranına (tpr) göre çizimi ve etiketlenmesi, hangi renkte çizileceği, hangi sayısal aralıkta çizileceği ve x-y eksen isimleri çeşitli parametreler yardımıyla belirtilir.

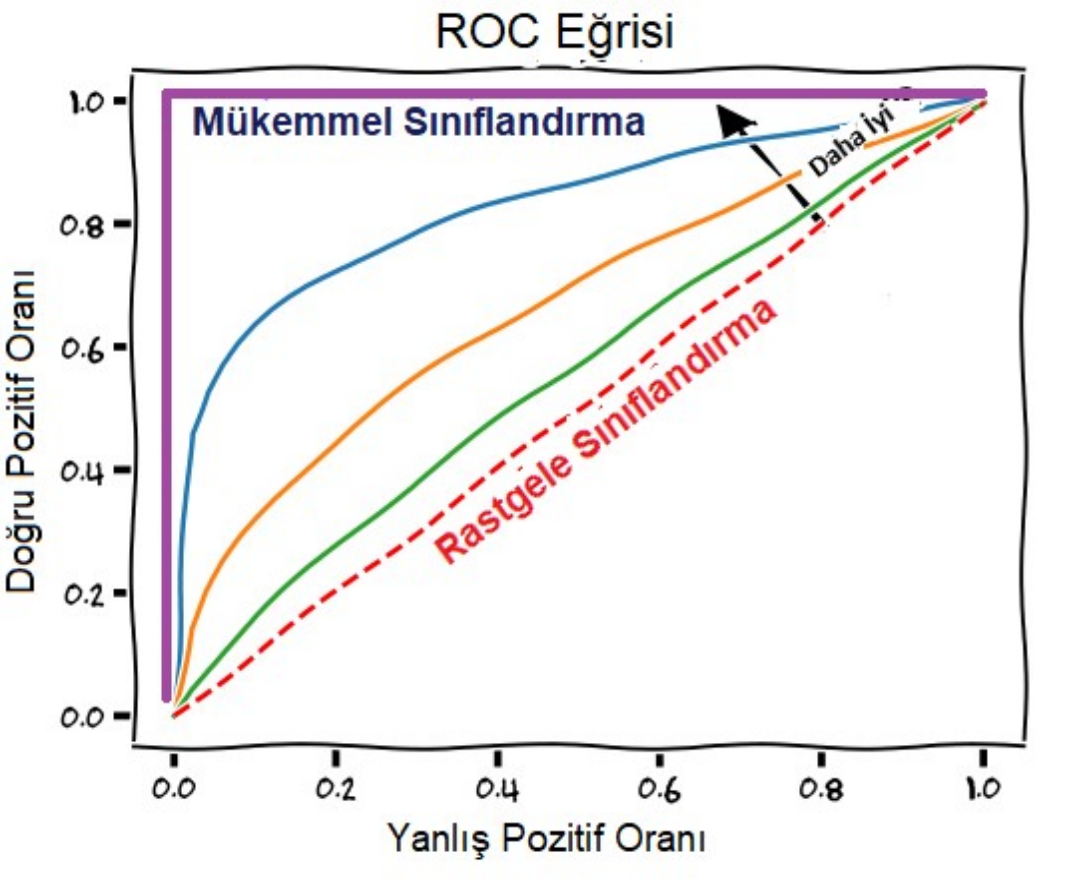


Şekil 4.13. ROC eğrisinin çizimi için kod ekranı

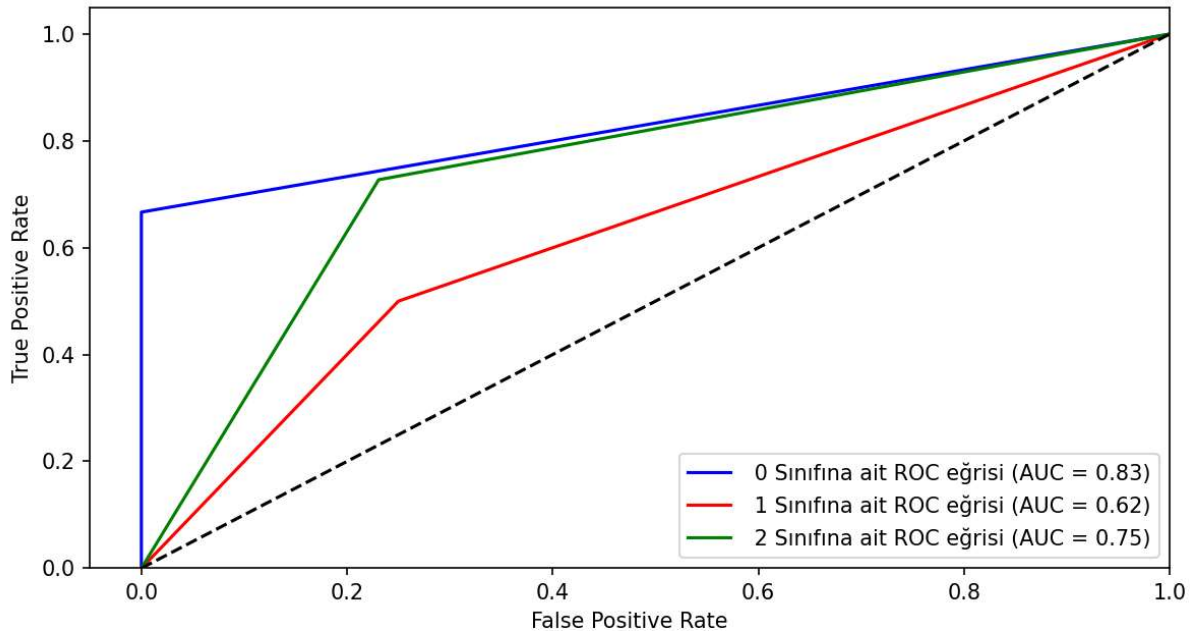
Öğrenciler detayları takip eden satırlarda açıklanan pirinç yaprağı hastalığı türünü karar ağaçları algoritması ile tahminleme örneği için Python kodlarını yazar. Öğrenciler hazırlamış oldukları Python kodlarını çalıştırdıklarında Şekil 4.14'te ve Şekil 4.15'te gösterildiği gibi ROC eğrisini ve modelin başarısını (doğruluk oranı) görürler. Öğrenciler karar ağaçları algoritması kullanarak 100 görüntü içerisinde 66 görüntünün doğru başarım oranı (%66) ile tahminlendiğini görürler.

Eğitmene Not

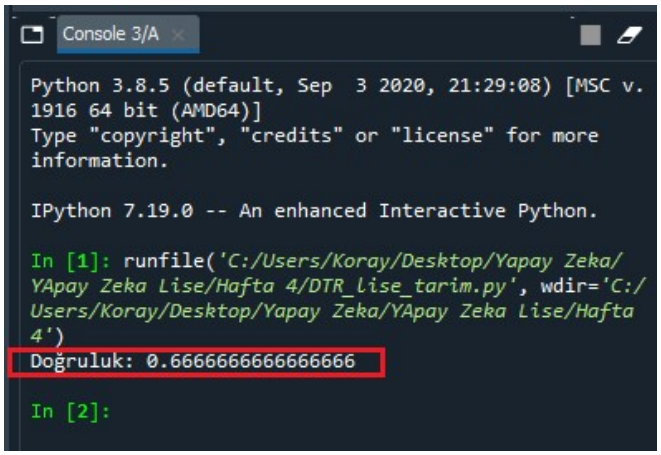
Eğitmen, Şekil 4.14'te gösterildiği gibi öğrenciye çalışma karakteristik eğrisi (Receiver Operating Characteristic-ROC) hakkında açıklama yapar.



Şekil 4.14. ROC Eğrisi gösterimi



Şekil 4.15. Model başarısı eğrisi



Şekil 4.16. Model başarısı sonucu

Biliyor musunuz?

Karar Ağaçları, Yapay Sinir Ağları ile Yapay Zekâ'nın Makine Öğrenmesi alanı kapsamındaki en eski tekniklerinden biridir. Tabi ki birçok teknik gibi bu teknik de zamanla birçok çeşitlere ayrılmış ve güncelliğini yitirmemiştir.

Dünyadan Haberler

Yapay Zekâyla COVID19 (Koronavirüs) Teşhisi

COVID19 pandemisinin başladığı günden bu yana en büyük problemlerden biri de PCR testlerinde bazen %60'lara varan oranda yalancı negatif sonuçların çıkması oldu. Türkiye'de bu sorun büyük ölçüde, belirtisi olan ama PCR'ı negatif olan hastaya, bilgisayarlı tomografi (BT) ile kesin tanı konarak aşıldı. Ancak radyologların "insan gözüyle" onlarca akciğer BT görüntüsünü okuyup raporlaması, binlerce hastanın söz konusu olduğu bir salgında, sağlık çalışanlarına ekstra yük de getirdi.

Pandeminin başladığı ilk günlerden itibaren yapay zekâ ile hastalığın teşhisi için çalışmalara başladıklarını söyleyen İstanbul İl Sağlık Müdürlüğü Görüntüleme Hizmetleri Koordinatörlüğü, Sağlık Bilimleri Üniversitesi ve Ankara Üniversitesi Teknokentleri iş birliği ile akciğer tomografisinden COVID19 teşhisini birkaç saniye içinde "yüzde 99,9 doğrulukla" koyabilen yapay zekâ geliştirildi (Web Kaynağı 4.5).

PYTHON KODLARI (Doğruluk %66,6):

```
import numpy as np
import PIL.Image as img
import os
import pandas as pd

bakteri_yaprak_yanik="bitki_veri-seti/rice_leaf_diseases/Bacterial leaf blight/"
kahve_nokta="bitki_veri-seti/rice_leaf_diseases/Brown spot/"
yaprak_isi="bitki_veri-seti/rice_leaf_diseases/Leaf smut/"

def dosya(yol):
    return [os.path.join(yol,f) for f in os.listdir(yol)]

def veri_donusturme(klasor_adi,sinif_adi):

    goruntuler=dosya(klasor_adi)

    goruntu_sinif=[]
    for goruntu in goruntuler:
```

```
        goruntu_oku= img.open(goruntu).convert('L')
        gorunu_boyutlandirma=goruntu_oku.resize((28,28))
        goruntu_donusturme=np.array(gorunu_boyutlandirma).flatten()
        if sinif_adi=="bakteri_yaprak_yanik":
            veriler=np.append (goruntu_donusturme, [0])

        elif sinif_adi=="kahve_nokta":
            veriler=np.append (goruntu_donusturme, [1])

        elif sinif_adi=="yaprak_isi":
            veriler=np.append (goruntu_donusturme, [2])

        else:
            continue
        goruntu_sinif.append(veriler)
    return goruntu_sinif

yanik_veri=veri_donusturme(bakteri_yaprak_yanik,"bakteri_yaprak_yanik")
yanik_veri_df=pd.DataFrame(yanik_veri)

kahve_nokta_veri=veri_donusturme(kahve_nokta,"kahve_nokta")
kahve_nokta_veri_df=pd.DataFrame(kahve_nokta_veri)

yaprak_isi_veri=veri_donusturme(yaprak_isi,"yaprak_isi")
yaprak_isi_veri_df=pd.DataFrame(yaprak_isi_veri)

tum_veri= pd.concat([yanik_veri_df, kahve_nokta_veri_df,yaprak_isi_veri_df ])

import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
```

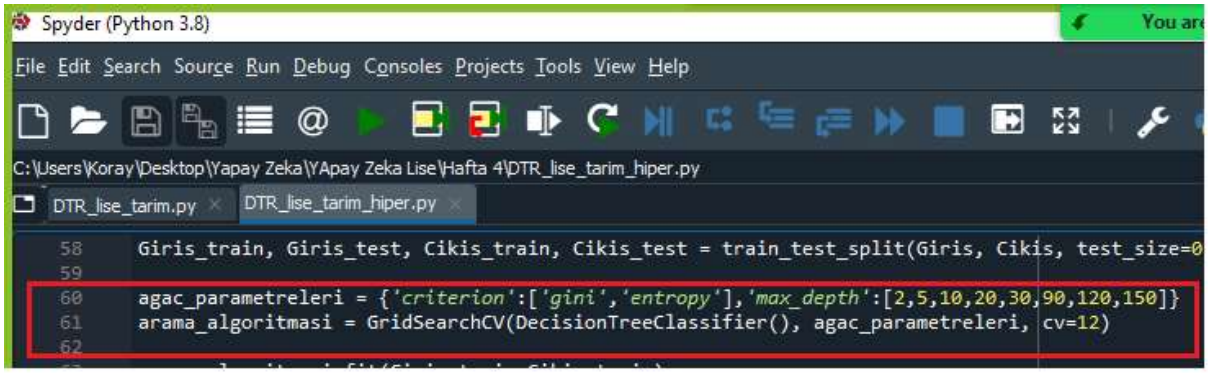
```
from sklearn import metrics
from sklearn.model_selection import GridSearchCV

Giris=np.array(tum_veri)[:,:784]
Cikis=np.array(tum_veri)[:,784]
Giris_train, Giris_test, Cikis_train, Cikis_test = train_test_split(Giris, Cikis,
test_size=0.2, random_state=109)

model= DecisionTreeClassifier()
clf = model.fit(Giris_train,Cikis_train)
Cikis_pred = clf.predict(Giris_test)

print("Doğruluk:",metrics.accuracy_score(Cikis_test, Cikis_pred))

import matplotlib.pyplot as plt
from sklearn.preprocessing import label_binarize
from sklearn.metrics import roc_curve, auc
from itertools import cycle

Cikis_test = label_binarize(Cikis_test, classes=[0, 1, 2])
Cikis_pred = label_binarize(Cikis_pred, classes=[0, 1, 2])

plt.figure(figsize=(60, 40),dpi=150)
n_classes=3
fpr = dict()
tpr = dict()
roc_auc = dict()
for i in range(n_classes):
    fpr[i], tpr[i], _ = roc_curve(Cikis_test[:, i], Cikis_pred[:, i])
    roc_auc[i] = auc(fpr[i], tpr[i])
colors = cycle(['blue', 'red', 'green'])
```

```
for i, color in zip(range(n_classes), colors):
    plt.plot(fpr[i], tpr[i], color=color,
             label=' {0} Sınıfına ait ROC eğrisi (AUC = {1:0.2f})'
             ''.format(i, roc_auc[i]))
plt.plot([0, 1], [0, 1], 'k--')
plt.xlim([-0.05, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend(loc="lower right")
plt.show()
```

5.2.MODELİN BAŞARIM ORANININ YÜKSELTİLMESİ

Önceki uygulamada elde edilen başarım oranını yükseltmek için kod üzerinde birtakım güncellemeler gerçekleştirilir (İlgili kodun tamamı Github platformunda, Hafta 4 altında H4_kararagaci_bitki_2-h.py dosyası olarak paylaşılmış durumdadır.). Öğrenciler 60. kod satırında **"gini"** ve **"entropy"** kriterlerine göre karar ağaçları algoritmasında maksimum derinlik **"max_depth"** özelliği 2 ile 150 arasında sekiz (8) farklı değer alarak sonuçlar **"agac_parametreleri"** sözlüğüne aktarılmıştır. Burada **"gini"** ve **"entropy"** kriterleri sekiz farklı maksimum derinlik değeri için toplam onaltı kez eğitilmiştir. Maksimum derinlik değerinin 2 ile 150 arasında sınırlandırılmasının nedeni ağaç yapısının karmaşık bir yapıya dönüşmeden eğitimin hızlı biçimde gerçekleştirilmesini ve over fitting'i (aşırı öğrenme) engellemektir.

Öğrenciler 61. kod satırında ise, GridSearchCV algoritması kullanılarak **"ağaç_parametreleri"** sözlüğünde yer alan verileri 12 parçaya ayrırarak (yani bütün veri %100 olarak kabul edilirse, verileri rastgele %0-8, %9-17, …, %93-100 oranlarında parçalara bölerek) en iyi sonucu veren parametre değerleri karar ağaçları algoritmasının eğitiminde kullanılır.

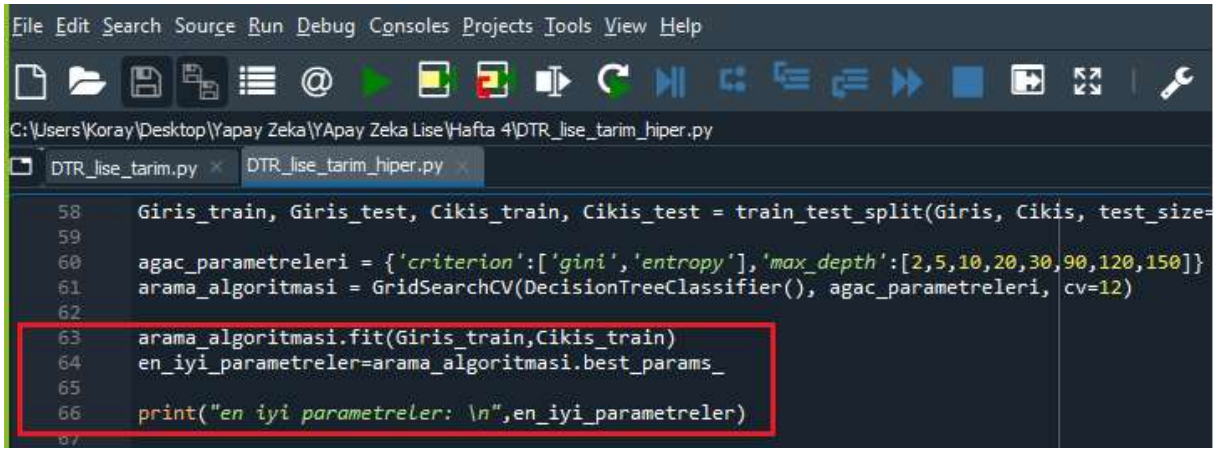


Şekil 4.17. Karar ağacı hiper parametrelerin tanımı ve GridSearchCV algoritmasının uygulanması

Öğrenciler Şekil 4.18'de gösterildiği gibi 63. Kod satırında pirinç yaprağı hastalığı görüntülerine göre hastalık türünün tahmini durumu için oluşturmuş olduğu karar ağaçları algoritmasını kullanabilmek için **"arama_algoritması.fit"** komutu yazarak giriş eğitim verilerine göre çıkış eğitim verileri için eğitim gerçekleştirir.

64. kod satırında ise, GridSearchCV algoritmasının **".best_params_"** özelliğini kullanılarak en iyi hiper parametreleri "**en_iyi_parametreler"** değişkenine aktarır.

65. Kod satırında ise GridSearchCV algoritmasındaki en iyi hiper parametreleri konsol ekrana yazdırır.

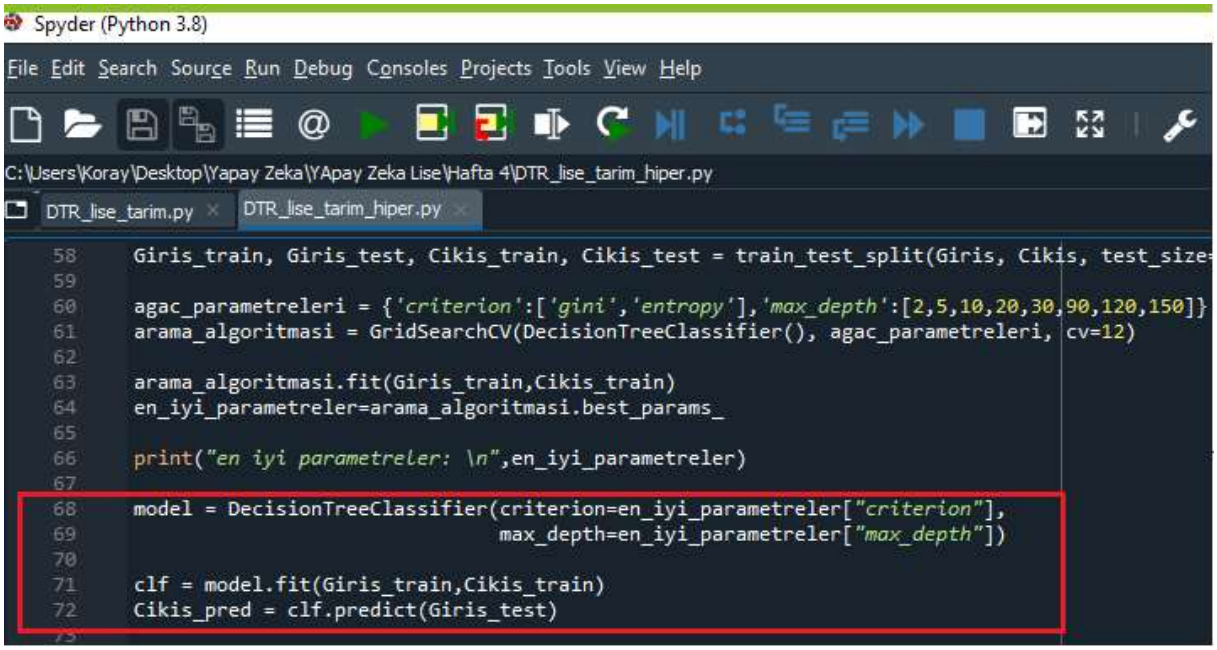


Şekil 4.18. GridSearchCV algoritması ile karar ağaçları modeli için en uygun belirlenen hiper parametrelerin belirlenmesi kod ekranı

Öğrenciler Şekil 4.19'da gösterildiği gibi 68. Ve 69. Kod satırında belirlenen kriterler (Gini, Entropy) ve maksimum derinlik (max_depth) parametrelerine göre karar ağaçları modeli oluşturur.

70. Kod satırında ise pirinç yaprağı hastalığı görüntülerine göre hastalık türünün tahmini durumu için oluşturmuş olduğu karar ağaçları algoritmasını kullanabilmek için **"model.fit"** komutu yazarak giriş eğitim verilerine göre çıkış eğitim verileri için eğitim gerçekleştirip sonucu **"clf"** değişkenine aktarır.

72. Kod satırında ise karar ağaçları modelinden elde edilen eğitim sonuçlarını **"Giris_test"** verilerine göre **".predict"** özelliği kullanarak tahmin edip, sonucu **"Cikis_pred"** değişkenine aktarır.



Şekil 4.19. Karar ağaçları algoritmasında eğitim ve tahminleme işlemleri için kod ekranı

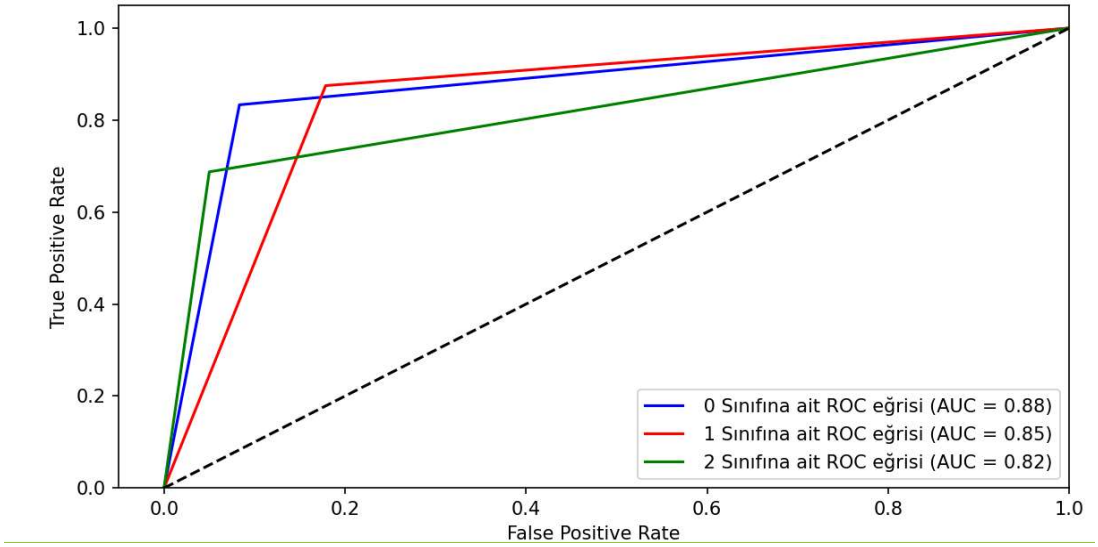
PYTHON KODLARI (Doğruluk %77,7):

```
import numpy as np
import PIL.Image as img
import os
import pandas as pd

bakteri_yaprak_yanik="bitki_veri-seti/rice_leaf_diseases/Bacterial leaf blight/"
kahve_nokta="bitki_veri-seti/rice_leaf_diseases/Brown spot/"
yaprak_isi="bitki_veri-seti/rice_leaf_diseases/Leaf smut/"

def dosya(yol):
    return [os.path.join(yol,f) for f in os.listdir(yol)]

def veri_donusturme(klasor_adi,sinif_adi):

    goruntuler=dosya(klasor_adi)
    goruntu_sinif=[]
```

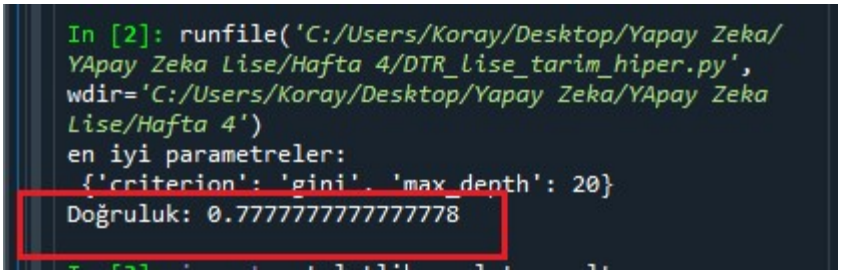
```
    for goruntu in goruntuler:
        goruntu_oku= img.open(goruntu).convert('L')
        gorunu_boyutlandirma=goruntu_oku.resize((28,28))
        goruntu_donusturme=np.array(gorunu_boyutlandirma).flatten()
        if sinif_adi=="bakteri_yaprak_yanik":
            veriler=np.append (goruntu_donusturme, [0])

        elif sinif_adi=="kahve_nokta":
            veriler=np.append (goruntu_donusturme, [1])

        elif sinif_adi=="yaprak_isi":
            veriler=np.append (goruntu_donusturme, [2])

        else:
            continue
        goruntu_sinif.append(veriler)
    return goruntu_sinif

yanik_veri=veri_donusturme(bakteri_yaprak_yanik,"bakteri_yaprak_yanik")
yanik_veri_df=pd.DataFrame(yanik_veri)

kahve_nokta_veri=veri_donusturme(kahve_nokta,"kahve_nokta")
kahve_nokta_veri_df=pd.DataFrame(kahve_nokta_veri)

yaprak_isi_veri=veri_donusturme(yaprak_isi,"yaprak_isi")
yaprak_isi_veri_df=pd.DataFrame(yaprak_isi_veri)

tum_veri= pd.concat([yanik_veri_df, kahve_nokta_veri_df,yaprak_isi_veri_df ])

import pandas as pd
from sklearn.tree import DecisionTreeClassifier
```

```
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.model_selection import GridSearchCV

Giris=np.array(tum_veri)[:,:784]
Cikis=np.array(tum_veri)[:,784]
Giris_train, Giris_test, Cikis_train, Cikis_test = train_test_split(Giris, Cikis,
test_size=0.3, random_state=109)

agac_parametreleri =
{'criterion':['gini','entropy'],'max_depth':[2,5,10,20,30,90,120,150]}
arama_algoritmasi = GridSearchCV(DecisionTreeClassifier(), agac_parametreleri,
cv=12)

arama_algoritmasi.fit(Giris_train,Cikis_train)
en_iyi_parametreler=arama_algoritmasi.best_params_

print("en iyi parametreler: \n",en_iyi_parametreler)

model = DecisionTreeClassifier(criterion=en_iyi_parametreler["criterion"],
                               max_depth=en_iyi_parametreler["max_depth"])

clf = model.fit(Giris_train,Cikis_train)
Cikis_pred = clf.predict(Giris_test)

print("Doğruluk:",metrics.accuracy_score(Cikis_test, Cikis_pred))

import matplotlib.pyplot as plt
from sklearn.preprocessing import label_binarize
from sklearn.metrics import roc_curve, auc
from itertools import cycle
```

```
Cikis_test = label_binarize(Cikis_test, classes=[0, 1, 2])
Cikis_pred = label_binarize(Cikis_pred, classes=[0, 1, 2])

plt.figure(figsize=(60, 40),dpi=150)
n_classes=3
fpr = dict()
tpr = dict()
roc_auc = dict()
for i in range(n_classes):
    fpr[i], tpr[i], _ = roc_curve(Cikis_test[:, i], Cikis_pred[:, i])
    roc_auc[i] = auc(fpr[i], tpr[i])
colors = cycle(['blue', 'red', 'green'])
for i, color in zip(range(n_classes), colors):
    plt.plot(fpr[i], tpr[i], color=color,
             label=' {0} Sınıfına ait ROC eğrisi (AUC = {1:0.2f})'
             ''.format(i, roc_auc[i]))
plt.plot([0, 1], [0, 1], 'k--')
plt.xlim([-0.05, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend(loc="lower right")
plt.show()
```



Şekil 4.20. ROC eğrisi



Şekil 4.21. Karar ağaçları modelin en iyi kriterler ve maksimum derinlik hiper parametrelerine göre doğruluk oranı

Eğitmene Not

Karar ağaçları algoritması ile eğitimi esnasında modellerin doğrulukları birbirinden farklı değerler çıkabilir. Birbirinden farklı değerler çıkması genellikle veri seti ile ilgili bir durumdur. Kod satırında yer alan "random_state=109" kod satırında 109 rakamını değiştirerek (örneğin "0", "1", vs.) modelin doğruluğunun değiştiğiniz gözlemleyiniz ve tartışınız. Eğitmen, en yüksek doğruluğu elde edilinceye kadar modeli eğitebilir.

6. İLAVE ETKİNLİK

Şekil 4.22'de gösterildiği gibi orman yangınlarındaki görüntülerde yer alan duman veya yangın görüntülerine göre orman yangınının olup olmadığını karar ağaçları algoritması kullanarak tahminleyen modeli kurunuz (Bu modelin ormanları tarayan bir drone içerisine entegre edileceğini düşünün. (İlgili uygulamanın kodu Github platformu Hafta4 klasörü altında H4_kararagaci_orman-yangini.py dosyası ile paylaşılmış durumdadır.)

Şekil 4.22. Orman yangını görüntüsü

VERİ SETİ İÇİN İNDİRME LİNKİ

https://github.com/aiformankind/wildfire-dataset/archive/refs/heads/master.zip

Eğitmene Not

Eğitmen, ilgili indirme linkinden veri setini indirir. Çizelge 4.2'de gösterildiği gibi negative (yangın olmayan görüntüler) ve positive (yangın olan görüntüler) isimli dosyayı ilave etkinlikte kullanılmak üzere öğrencilere verir. Eğitmen, öğrencilere veri setinde yer alan giriş parametresi görüntülere göre çıkış parametresi olan yangın yok ise negatif (0), yangın var ise pozitif (1) sınıfı temsil ettiğini anlatır (Burada anlatılan kodlara uyum sağlamak için ilgili veriler, orman-yangini adlı bir klasör altında tutulabilir.).

Çizelge 4.2. Veri setinin giriş çıkış parametreleri

Veri Sayısı	GİRİŞ PARAMETRESİ	ÇIKIŞ PARAMETRESİ
	Görüntü	Yangın olup olmadığı
1		0 (Negatif)

2		0 (Negatif)
…	…	…
97		0 (Negatif)
98		1 (Pozitif)
…	…	…
302		1 (Pozitif)

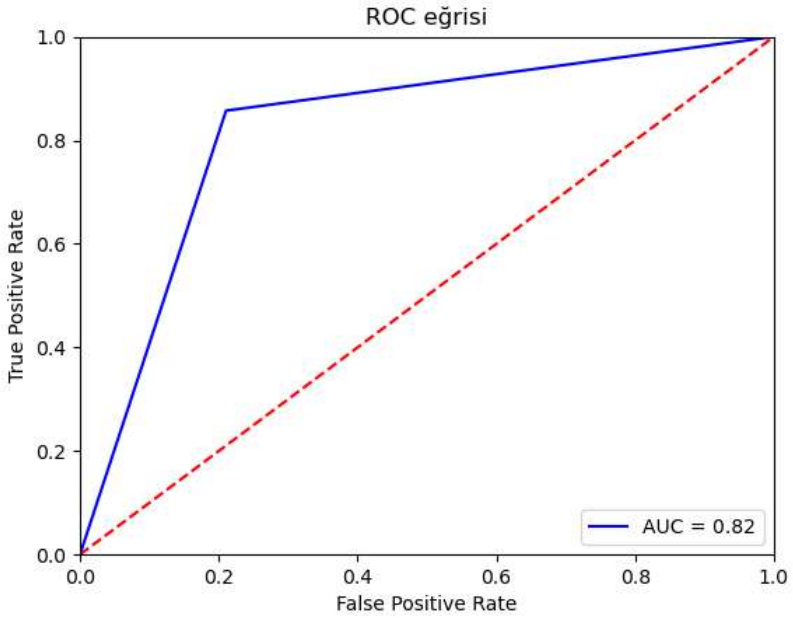
PYTHON KODLARI:

```
import numpy as np
import PIL.Image as img
import os
import pandas as pd
```

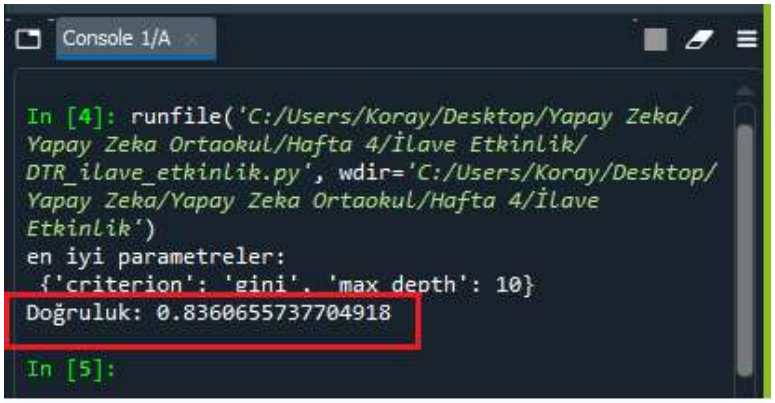
```
duman_yangin_yok="orman-yangini/negative/"
duman_yangin_var="orman-yangini/positive/"

def dosya(yol):
    return [os.path.join(yol,f) for f in os.listdir(yol)]

def veri_donusturme(klasor_adi,sinif_adi):

    goruntuler=dosya(klasor_adi)

    goruntu_sinif=[]
    for goruntu in goruntuler:
        goruntu_oku= img.open(goruntu).convert('L')
        gorunu_boyutlandirma=goruntu_oku.resize((28,28))
        goruntu_donusturme=np.array(gorunu_boyutlandirma).flatten()
        if sinif_adi=="yok":
            veriler=np.append (goruntu_donusturme, [0])

        elif sinif_adi=="var":
            veriler=np.append (goruntu_donusturme, [1])

        else:
            continue
        goruntu_sinif.append(veriler)

    return goruntu_sinif

duman_yangin_yok_veri=veri_donusturme(duman_yangin_yok,"yok")
duman_yangin_yok_df=pd.DataFrame(duman_yangin_veri_yok)
```

```
duman_yangin_var_veri=veri_donusturme(duman_yangin_var,"var")
duman_yangin_var_df=pd.DataFrame(duman_yangin_var_veri)

tum_veri= pd.concat([duman_yangin_yok_df, duman_yangin_var_df])

import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.model_selection import GridSearchCV

Giris=np.array(tum_veri)[:,:784]
Cikis=np.array(tum_veri)[:,784]
Giris_train, Giris_test, Cikis_train, Cikis_test = train_test_split(Giris, Cikis,
test_size=0.2, random_state=1)

agac_parametreleri =
{'criterion':['gini','entropy'],'max_depth':[2,5,10,20,30,90,120,150]}

arama_algoritmasi = GridSearchCV(DecisionTreeClassifier(), agac_parametreleri,
cv=5)
arama_algoritmasi.fit(Giris_train,Cikis_train)
en_iyi_parametreler=arama_algoritmasi.best_params_

print("en iyi parametreler: \n",en_iyi_parametreler)

model = DecisionTreeClassifier(criterion=en_iyi_parametreler["criterion"],
                               max_depth=en_iyi_parametreler["max_depth"])

clf = model.fit(Giris_train,Cikis_train)
```

```
Cikis_pred = clf.predict(Giris_test)
print("Doğruluk:",metrics.accuracy_score(Cikis_test, Cikis_pred))

import matplotlib.pyplot as plt
fpr, tpr, threshold = metrics.roc_curve(Cikis_test, Cikis_pred)
roc_auc = metrics.auc(fpr, tpr)
plt.title('ROC eğrisi')
plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc_auc)
plt.legend(loc = 'lower right')
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
```



Çizelge 4.23. ROC eğrisi sonucu



Çizelge 4.24. Doğruluk değeri sonucu

Eğitmene Not

Eğitmen, bu hafta eğitimine yönelik öğrencilerin ilgisini arttırmak ve yeni öğrenilen bilgileri önceden öğrenilen bilgiler ile karşılaştırmalarını sağlamak için aşağıda maddeler halinde verilen soruları tartışır.

- İlgili haftada ele alınan yapay zekâ uygulamalarına uygun örnek veri setleri neler olabilir?
- İlgili haftada ele alınan veri setlerinin yapay zekâ teknik ve kütüphanelerinde ne derecede başarım etkisi olmuştur?
- İlgili haftada öğretilen yapay zekâ tekniği ile uyumlu ve uyumsuz olabilecek açık erişimli veri seti çeşitleri sunan (kaagle, github vs.) internet sitelerinden elde edilecek farklı veri çeşitleri ile yapay zekâ tekniği uyumluluğunu sorar.

Eğitmene Not

Eğitmen, öğrencilerin 3. ve 4. Hafta içerisinde elde ettikleri bilgi-becerileri değerlendirmek, aktif katılımlarını sağlamak ve onları gelecek haftalara motive etmek için 'Kahoot' uygulamasını ya da benzeri bir Web 2.0 uygulamasını kullanarak 'bilgi yarışması' düzenleyebilir.

3. ve 4. Hafta bağlamında sorulabilecek sorular; Python ile Makine Öğrenmesi temel çözüm süreçleri, Bayes Öğrenmesi temelleri/uygulamaları ve Karar Ağaçları temelleri/uygulamaları yönünde olabilir.

Kaynakça

SPSS. (1999). AnwerTree Algorithm Summary, SPSS White Paper, USA.

Web Kaynağı 4.1: https://erdincuzun.com/makine_ogrenmesi/decision-tree-karar-agaci-id3-algoritmasi-classification-siniflama/

Web Kaynağı 4.2: https://www.datasciencearth.com/underfitting-ve-overfitting/

Web Kaynağı 4.3: https://www.webtekno.com/avustralya-mahkeme-yapay-zeka-mucit-olabilecegini-onayladi-h113004.html

Web Kaynağı 4.4: https://teknoloji.org/pandas-nedir-nasil-kullanilir-python-kutuphanesi/

Web Kaynağı 4.5: https://www.cnnturk.com/video/turkiye/yapay-zekayla-koronavirus-teshisi

5. Hafta: Yapay Sinir Hücresi ve Yapay Sinir Ağları

Ön Bilgi:

- Yapay sinir ağları temelleri, Çok katmanlı algılayıcı modeli, Alternatif yapay sinir ağları modelleri, Yapay sinir ağları ile duygu tanıma uygulaması
- Python kodlamada döngüler, fonksiyonlar ve kütüphane yapılarının kullanımı (Bu konuda Python anlatım videoları: 10-11. Döngüler, 13. Fonksiyonlar, 14-16. Python kütüphaneleri serisi yardımcı araç olarak kullanılabilecektir).

Haftanın Kazanımları:

- Öğrenciler, yapay sinir ağları temel yapısını kavrar.
- Öğrenciler, yapay sinir ağları ile insan beyninin öğrenme yapısını modelleyerek uygular.
- Öğrenciler yapay sinir ağları ile matematiksel modelleme denklemlerini yorumlamayı kavrar.
- Öğrenciler yapay sinir ağları ile bina enerji verimliliği sistemini modelleme ve çözüm üretme yetenekleri kazanır.
- Öğrenciler, yapay sinir ağları üzerindeki bağımlı değişken, bağımsız değişken, ağırlık değeri ve bias terimlerini kavrar.
- Öğrenciler Python programlama dilini kullanarak bina enerji verimliliği için program kodunu yazabilir.
- Öğrenciler Python programlama dilinde yapay zekâ modelleri için hazırlanan keras kütüphanesinin fonksiyonlarını kullanma yeteneği kazanır.
- Öğrenciler, bina enerji verimliliğine ait verileri giriş ve çıkış parametresine göre belirler.
- Öğrenciler yapay sinir ağlarında veriye uygun model tasarlar, analiz eder ve modeli eğitir.
- Öğrenciler test verileri üzerinde modeli değerlendirerek yorumlar.
- YSA modeli ile bina enerji verimliliği için konsol ekranında %82 ve %77 olarak yüksek bir başarım oranı ile model oluşturma yeteneği kazanır.
- Yapay Sinir Ağları ile farklı verilerle yanlış öğrenmenin gerçekleşip gerçekleşmeyeceğini kavrar.
- Türkiye'de 2021-2025 yılları Ulusal Yapay Zekâ Stratejisi hakkında bilgi sahibi olur.

Haftanın Amacı:

Bu haftanın amacı, "yapay sinir ağları" kavramının tüm öğrenciler tarafından doğru bir şekilde anlaşılmasını sağlamaktır. Haftanın bir diğer amacı, yapay sinir ağlarının kullanıldığı farklı örnekler ile öğrencilerin dikkatini çekmek ve etkinliklere yönelik ilgilerini artırmaktır. Ayrıca, öğrenciler Python programlama dili kullanarak yapay sinir ağları ile örnek modelleme ve çözüm üretme yetenekleri kazandırılacaktır.

Kullanılacak Malzemeler:

Bilgisayar, kâğıt, kalem, Python kod editörü, örnekler için çıktı görselleri

Haftanın İşlenişi:

Algıla: Öğrenciler yapay sinir ağları tekniği bağlamındaki temel kavramları tanımlayıp yorumlayabilir.

Tasarla: Öğrenciler yapay sinir ağları ile bina enerji verimliliğine tanıma probleminin giriş ve çıkış parametrelerini tasarlayabilir. Bu hafta için Github platformu (https://github.com/deneyapyz/lise/) ile etkileşim Tasarla aşaması ile başlatılabilmektedir.

Harekete Geç: Öğrenciler Python programlama dili ile veri setindeki görüntü değerlerini sayısallaştırıp eğitim ve test verilerini ayırıp yapay sinir ağları tabanlı modeller kurabilir.

Yürüt: Eğitmen, öğrenciler ile etkileşimli olarak yapay sinir ağları modelini eğitir ve bina enerji verimliliği sistemi için sonuçları elde eder. Öğrenciler ise yapay sinir ağları odağında genel model tasarımı ve çözüm üretme süreçlerine yönelik kodlama ve yürütme işlemlerini gerçekleştirebilir.

Karar Ver: Öğrenciler alternatif yapay zekâ modelini düzenleyerek / kodlayarak çözüm üretebilir.

1. ALGILA

1.1. Yapay Sinir Ağları Temelleri

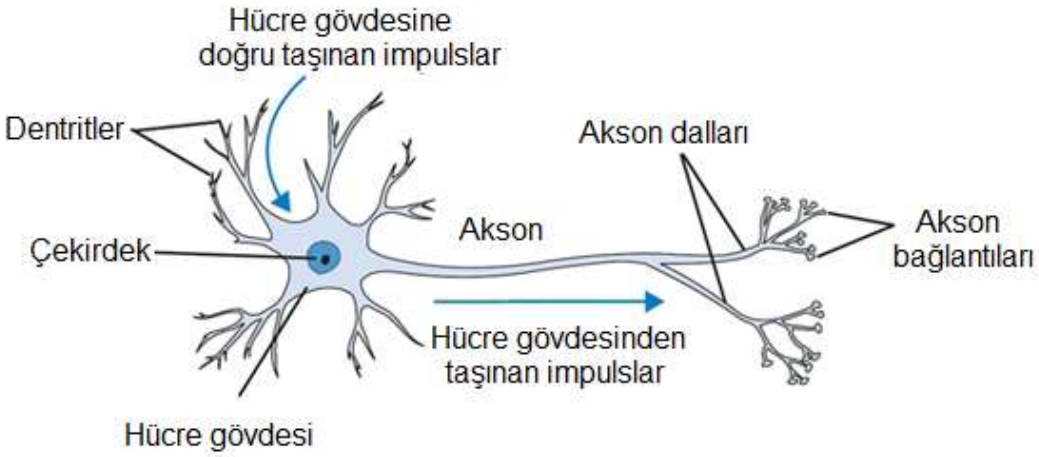
Eğitmene Not

Eğitmen, öğrencilere "**sibernetik**", "**yapay sinir ağları**", "**modelleme**" "**aktivasyon fonksiyonu**" "**yapay sinir ağları katmanları**" ve "**nöron**" kavramları hakkında bilgi sahibi olup olmadıklarını sorar ve tartışır. Böylece öğrencilerin konuya dikkatini çeker.

Canlıların davranışlarını inceleyerek, matematiksel olarak modelleyip, benzer yapay modellerin üretilmesine "**sibernetik"** denir. Eğitilebilir, adaptif, kendi kendine organize olup öğrenebilen ve değerlendirme yapabilen yapay sinir ağları ile insan beyninin öğrenme yapısının modellenmesi amaçlanmıştır. Bu bakımdan Yapay Sinir Ağları tekniği, içerisindeki esnek matematiksel yapı sayesinde, Yapay Zekâ'nın en güçlü tekniği olma unvanını elinde tutmaktadır. Yapay sinir ağları vasıtasıyla tıpkı insanoğlunda olduğu gibi makinelerin eğitilmesi, öğrenmesi ve karar vermesi amaçlanmaktadır. Şekil 5.1'de insan sinir hücresinin (nöron) örnek görseli verilmiştir.

Eğitmene Not

Eğitmen, öğrencilere Şekil 5.1'de verilen insan sinir hücresinin yapısını, çalışma prensibini ve özelliklerini detaylı biçimde anlatarak açıklar.

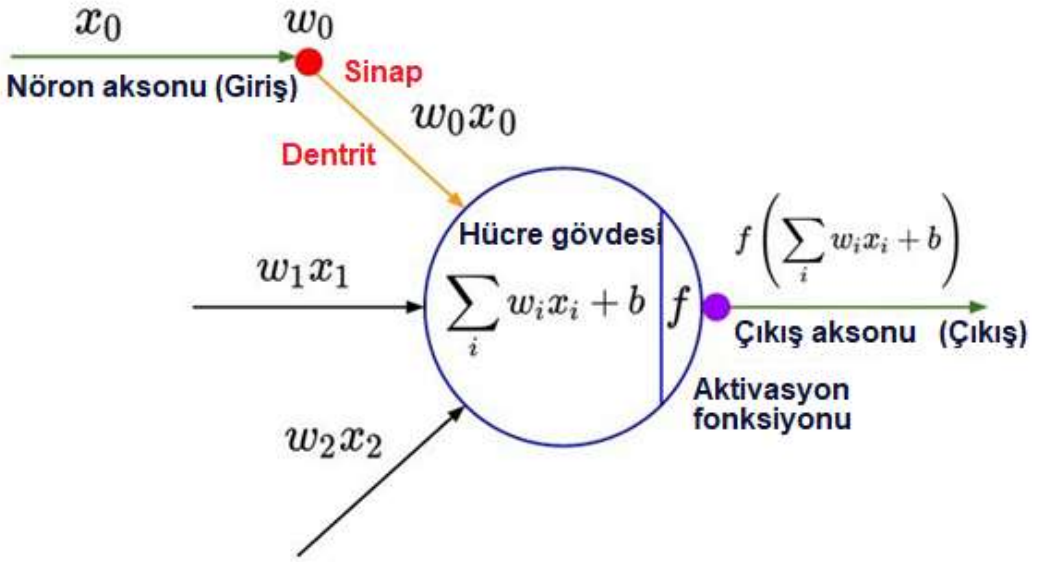


Şekil 5.1. İnsan sinir hücresinin yapısı

Şekil 5.2'de gösterildiği gibi Yapay sinir ağları insan sinir hücrelerine benzer bir yapıda giriş değerlerine göre (x0, x1, x2 vs.) ağırlıkları belirlenip (w0, w1, w2,…vs.) çarpılarak elde edilen sonuçların toplanması ilkesine dayanmaktadır. Yine yapay sinir ağlarında sabit bir bias değeri de kullanılabilmekte; bu değer ile aktivasyon fonksiyonlarının sınıflandırma ya da regresyon yönündeki hesaplamaları yönlendirilebilmektedir.

Eğitmene Not

Eğitmen, Yapay Sinir Ağları tekniğindeki temel mantığı, üzerinde çok çalıştığımız bir konuyu pekiştirme yaptıkça hatırlamamızı sağlayan, beyin hücreleri arasındaki artan elektriksel yüke benzeterek somutlaştırmaya çalışır. Bu noktada üzerinde çok düşünülmeyen konuların hücreler arası elektriksel yüklerin azalması nedeniyle unutulacağını da ifade eder. Yine beyin hücreleri arasındaki elektriksel bağların, Yapay Sinir Ağları içerisindeki hücrede ağırlık değerleriyle karşılandığını açıklayarak, doğal ve yapay sinir hücresi arası esin kaynaklarını irdeler.



Şekil 5.2. Temel yapay sinir hücresinin matematiksel modeli

Temel bir yapay sinir ağı modelinin matematiksel denklemi Şekil 5.3'te verilmiştir. Burada;

- y: x'e bağlı bağımlı değişken olup, giriş parametrelerine göre modelden elde edilen doğruluk sonucunu verir.
- x: Bağımsız giriş parametresidir.
- w: Her bir giriş parametresinin ağırlık değeridir.
- b: Sabit bir bias değeridir.

Yapay sinir ağı modellerinde w ve b parametreleri değiştirilerek en iyi doğruluk sonucu elde edilinceye kadar model eğitilir.

Şekil 5.3. Yapay sinir ağlarının matematiksel denklemi

Eğitmene Not

Eğitmen dilerse farklı kaynaklardan yapay sinir ağlarının matematiksel gösterimi ile ilgili daha detaylı bilgileri öğrencilerle paylaşabilir.

1.2. Tek ve Çok Katmanlı Yapay Sinir Ağı Modelleri

Yapay sinir ağları (YSA) tek ve çok katmanlı olmak üzere iki farklı model olarak tasarlanabilir. İlk olarak yapay sinir ağı modelleri tek katmanlı yapıya sahip olarak tasarlanmıştır. 1960 yılında Widrow ve Hoff çok gizli katmanlı yapıya geçen ilk çalışmayı yapmışlardır. Şekil 5.4'te tek ve çok katmanlı yapay sinir ağları modellerinin görseli verilmiştir. Tek gizli katmanlı YSA modelinde giriş katmanlarından sonra oluşturulan gizli katmandaki nöron sayısına göre modeller eğitilerek çıkış katmanına gönderilir. Tek gizli katmanlı YSA modelinde 4+2=6 nöron (çıkış ve gizli katmandaki nöron sayısı) bulunmaktadır. Öğrenilecek olan parametre sayısı;

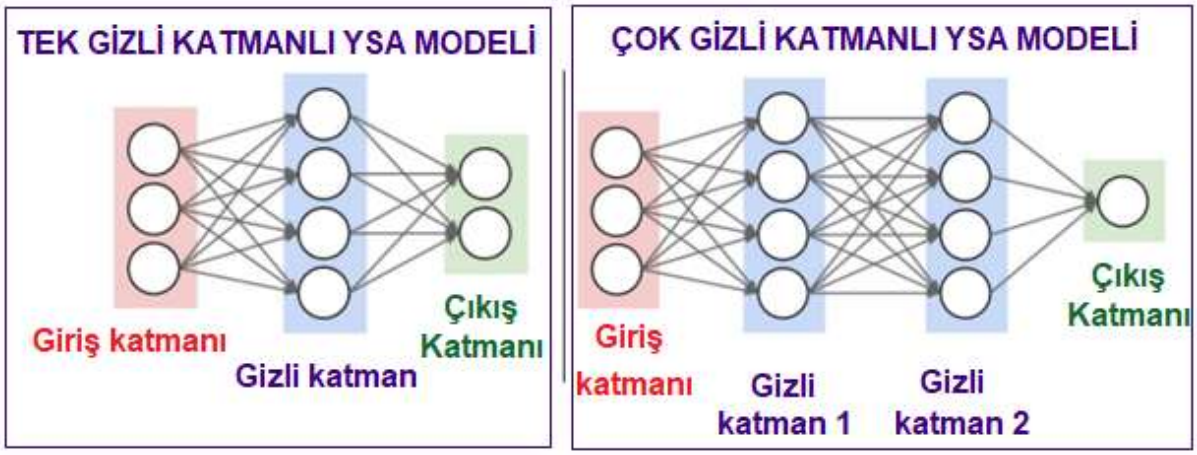
Giriş katmanındaki nöron sayısı (3) X gizli katmandaki nöron sayısı (4) + gizli katmandaki nöron sayısı (4) x çıkış katmanındaki nöron sayısı (2) = 20 ağırlık değeri elde edilir.

Elde edilen ağırlık değeri gizli katmandaki nöron sayısı (4) + çıkış katmandaki nöron sayısının (2) toplamı ile elde edilen 6 (altı) değeri bias değeri olup eklenerek toplam 26 adet öğrenilmesi gereken parametre hesaplanır.

Çok gizli katmanlı YSA modelinde gizli katmandaki nöron sayısı-1 (4) + gizli katmandaki nöron sayısı-2 (4) + çıkış katmanındaki nöron sayısı (1) olmak üzere 9 nörondan (çıkış ve gizli katmanlardaki nöron sayısı) oluşmaktadır.

Giriş katmandaki nöron sayısı (3) X gizli katman-1 nöron sayısı (4) + gizli katman-1 nöron sayısı (4) x gizli katman-2 nöron sayısı (4) + gizli katman-2 nöron sayısı (4) x çıkış katmanındaki nöron sayısı (1) = 32 ağırlık değeri elde edilir.

Elde edilen ağırlık değeri gizli katman-1 nöron sayısı (4) + gizli katma-2 nöron sayısı (4) + çıkış katmandaki nöron sayısının (1) toplamı ile elde edilen değer 9 (dokuz) değeri bias değeri olup eklenerek toplam 41 adet öğrenilmesi gereken parametre hesaplanır.



Şekil 5.4. Tek ve çok gizli katmanlı yapay sinir ağı model yapıları.

Biliyor musunuz?

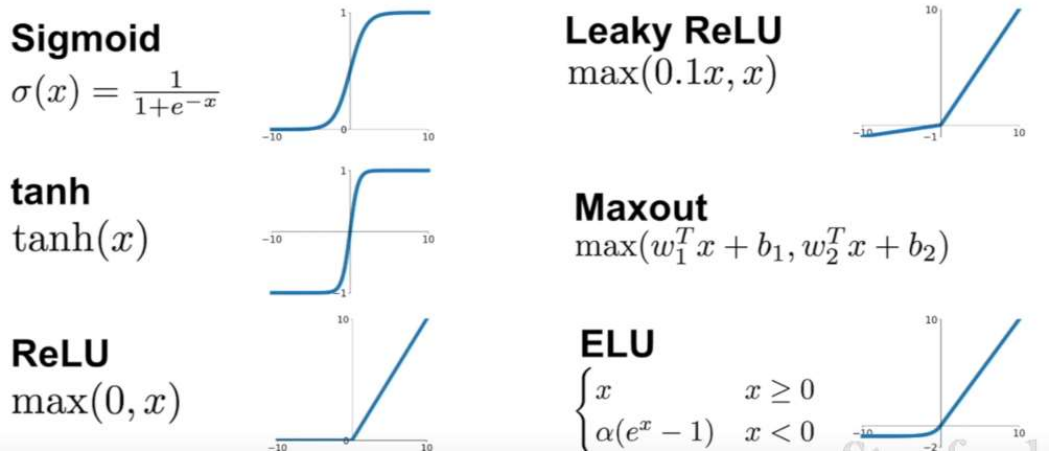
Yapay Sinir Ağları tekniğinin oluşmasına sebep olan önemli gelişmelerden biri de ilk aşamada katmansız kullanılan nöron yapılarının XOR adı verilen bir mantık kapısı problemine (giriş değerleri farklıysa sonuç 1: Doğru, aksi halde 0: Yanlış'tır) iyi sonuç üretememesidir. Yapılan çalışmalar katmanlar altında tasarlanan ağ yapısı halindeki nöronların bu problemi çözebildiğini ('birlikten kuvvet doğar.') göstermiş ve dolayısıyla Yapay Sinir Ağları çağı başlamıştır…

1.3. Yapay Sinir Ağlarında Nöronların Kullanımı

Yapay sinir ağında katmanlar içinde kullanılan nöronların birbirleri ile ilişkisi yoktur. En önemli görevleri sistemde olan bilgiyi bir sonraki katmana ya da çıkış katmanına aktarmaktır. Art arda gelen iki katmandaki nöronlar farklı aktivasyon değerlerine göre YSA modelin öğrenme seviyesini belirleyip aktarım işlemini gerçekleştirir. Yapay sinir ağlarında nöron sayısı, giriş parametresi ve katman sayısı modelin önemli parametrelerindendir. Pek çok hiper parametre kullanılarak çıkış performansı arttırılabilir. W ağırlık vektörü ise düğüm/nöron sayısı (hücre), bias (b) değerleri de gelecek katmandaki düğüm sayısı kadar olmalıdır.

1.4. Yapay Sinir Ağlarında Aktivasyon Fonksiyonları

Yapay sinir ağı modelinde regresyon veya sınıflandırma probleminin çözümü için kurulan modelde farklı aktivasyon fonksiyonları kullanılabilir (Şekil 5.5).



Şekil 5.5. Yapay sinir ağlarında sıklıkla kullanılan aktivasyon fonksiyonları

1.4.1. Sigmoid Fonksiyonu

YSA modellerinde sıklıkla kullanılan aktivasyon fonksiyonlarından birisi de Sigmoid fonksiyonudur. Sigmoid fonksiyonu [0,1] arasında değerler alabilen ve sınıflandırma problemlerinde kullanılan bir fonksiyondur. Bu fonksiyon, modelden elde edilen sonucun hangi sınıfa ait olduğunu öğrenen, olasılıksal bir değer üretir.

1.4.2. Hiperbolik Tanjant (tanh) Fonksiyonu

YSA modellerinde kullanılan hiperbolik tanjant (tanh) fonksiyonu sigmoid fonksiyonuna oldukça benzeyen ve sınıflandırma için kullanılan fonksiyondur. Aktivasyon fonksiyon çıkış değeri [-1,1] aralığında doğrusal olmayan bir fonksiyondur.

1.4.3. Rectified Linear Unit/Rektifiye Doğrusal Birim (ReLu) Fonksiyonu

YSA modellerinde ReLU aktivasyon fonksiyonu çok gizli katmanlarda sıklıkla kullanılıp aynı anda tüm nöronları aktive etmez. ReLU aktivasyon fonksiyonunda negatif değerler üreten nöronlar sıfır değeri kabul edilir. Böylece ReLU aktivasyon fonksiyonu daha verimli ve hızlı eğitir.

1.4.4. Leaky ReLu Fonksiyonu

YSA modellerinde Leaky ReLU aktivasyon fonksiyonu ReLU fonksiyonuna benzer biçimdedir. Ancak Leaky ReLU fonksiyonunda negatif değerler ReLU fonksiyonunda olduğu gibi tam sıfır yerine sıfıra yakın negatif bir değer alır. Böylece öğrenmenin çift yönlü yönü olarak da gerçekleşmesi sağlayan bir aktivasyon fonksiyonudur.

1.4.5. Maxout Fonksiyonu

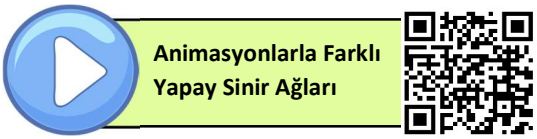
Maxout fonksiyonu öğrenilebilen, hesaplamalı olarak ucuz ve yalnızca etken giriş parametresini dikkate alan YSA modellerinde kullanılan bir aktivasyon fonksiyonudur. Böylece en etkili giriş parametresi seçilerek elde edilen sonuçlar çıkışa gönderir.

1.4.6. Üst Lineer Birim (Exponential Linear Unit)-ELU Fonksiyonu

YSA modellerinde kullanılan ELU fonksiyonu negatif girdiler hariç ReLU fonksiyonuna oldukça benzerdir. ELU fonksiyonunda orta nokta "0" sıfırdır. Böylece ölü nöronların önüne geçerek modelin daha hızlı yakınsaması sağlanabilir.

1.5. Yapay Sinir Ağları Kullanım Alanları

Günümüzde bilgisayarlar ve bilgisayar sistemleri yaşamımızın vazgeçilmez bir parçası haline gelmiştir. Yapay zekânın alt dallarından birisi olan Yapay Sinir Ağları, bilgisayarların derin öğrenme yöntemlerinde sıklıkla kullanılan bir yapıdır. YSA temel alınarak farklı birçok derin öğrenme mimarisi tasarlanmıştır. YSA modelleri sağlık, medikal ve tıp sektörleri başta olmak üzere makine imalat sanayisi, otomotiv, elektronik, havacılık ve uzay sanayisi, bankacılık ve askeri alanlarda etkili biçimde kullanılmaktadır. Ayrıca insansı robotlarda sıklıkla kullanılmaktadır (Kaya, 2018).



Eğitmene Not

Eğitmen, '*Animasyonlarla Farklı Yapay Sinir Ağları*' videosunu öğrencilere izleterek, sinir hücresinin farklı ağ modelleri geliştirmede nasıl araç olabileceğini vurgular; insan beyni ve hatta iletişim ağları (bilgisayar ağları, sosyal ağlar…vs.) üzerinden yola çıkarak iş birliği içerisindeki bilgi aktarımının Yapay Zekâ'daki öğrenme süreçleri için ne kadar önemli bir esin kaynağı olduğunu açıklar.

2. TASARLA

Eğitmene Not

Eğitmen, öğrencilere yapay sinir ağları ile farklı bina şekillerine göre binanın enerji verimliliği analizi için rölatif sıkılık, binanın yüzeyi, duvar ve çatı alanı, binanın yüksekliği, bina yerleşimi (oryantasyon), cam alanı ve camın alan dağılımı giriş parametrelerine göre binanın enerji verimliliği, ısıtma ve soğutma yükü ile ilgili şu ifadeleri aktarır:

"*Binayı, ısıtma, havalandırma ve iklimlendirme (HVAC) sistemleri ve etkin dış ortam koşullarını kapsayan bir sistemler bütünü olarak ele almak ve çevresel etkenler/gereklilikler, binaların gittikçe karmaşıklaşması, geleceğe yönelik esneklik arayışları, kalite anlayışının yükselmesi, inşaat sektörünün gittikçe daha büyük bir oranda endüstriyel ve uluslararası bir konuma oturması gibi bir çok bir biriyle ilişkili ekonomik, teknik, politik ve sosyal gelişmelere bağlı olarak gittikçe önem kazanmaktadır.*

*İnşaat sektöründe **Yapay sinir ağları** ile oluşturulan yapay zekâ modeli sayesinde en yüksek seviyede enerji verimliliği analizi yapabilmek mümkün olmaktadır. Bu durum binanın ısıtma ve soğutma açısından daha yüksek verimliliğe ulaşmasını sağlamaktadır. YSA ile bina enerji verimliliği analizi sayesinde binaların enerji tüketim israfı önlenmektedir.*"

Öğrenciler, ilk olarak yapay sinir ağları ile farklı binalara ait rölatif sıkılık, binanın yüzey, duvar ve çatı alanı, binanın yüksekliği, bina yerleşimi (oryantasyon), cam alanı ve camın alan dağılımı giriş parametrelerine göre ısıtma ve soğutma yükü değerlerini hesaplayıp değerlendirilmesi problemini anlar. Şekil 5.6'da bina enerji verimliliğine ait görüntü verilmiştir.

Şekil 5.6. Bina enerji verimliliği -temsili (Web Kaynağı 5.1).

Öğrenciler Çizelge 5.1'de verilen yapay sinir ağları ile bina enerji verimliliği analizi için giriş çıkış parametresini belirler.

Çizelge 5.1. Yapay sinir ağları için giriş ve çıkış parametreleri

Veri Sayısı	GİRİŞ PARAMETRELERİ								ÇIKIŞ PARAMETRESİ	
	X1	X2	X3	X4	X5	X6	X7	X8	Y1	Y2
1	0,98	514,50	294,00	110,25	7,00	2	0,00	0	15,55	21,33
2	0,98	514,50	294,00	110,25	7,00	3	0,00	0	15,55	21,33
3	0,98	514,50	294,00	110,25	7,00	4	0,00	0	15,55	21,33
4	0,98	514,50	294,00	110,25	7,00	5	0,00	0	15,55	21,33
5	0,90	563,50	318,50	122,50	7,00	2	0,00	0	20,84	28,28
…	…	…	…	…	…	…	…	…	…	…
766	0,62	808,50	367,50	220,50	3,50	3	0,40	5	16,44	17,11
767	0,62	808,50	367,50	220,50	3,50	4	0,40	5	16,48	16,61
768	0,62	808,50	367,50	220,50	3,50	5	0,40	5	16,64	16,03

Bu bölümde farklı binalara ait 768 adet veri setinde (Tsanas ve Xifara, 2012) rölatif sıkılık, binanın yüzey, duvar ve çatı alanı, binanın yüksekliği, bina yerleşimi (oryantasyon), cam alanı ve camın alan dağılımı giriş parametrelerine göre ısıtma ve soğutma yükü değerlerinin yapay sinir ağları ile tahminlenmesi amaçlanmıştır.

VERİ SETİ İÇİN İNDİRME LİNKİ

https://archive.ics.uci.edu/ml/machine-learning-databases/00242/
https://github.com/deneyapyz/lise/Hafta5/ENB2012_data.xlsx

Eğitmene Not

Eğitmen veri setini anlatırken veri seti indirme linkinden indirilen farklı binalara ait verileri giriş ve çıkış parametrelerine ait şu düzenlemeleri öğrencilere aktarır:

- X1: Rölatif sıkılık
- X2: Yüzey Alanı
- X3: Duvar Alanı
- X4: Çatı Alanı
- X5: Genel Yükseklik
- X6: Bina yönlendirme
- X7: Cam Alanı
- X8: Cam Alan Dağılımı

- Y1: Isıtma Yükü
- Y2: Soğutma Yükü

Düşün, tartış…

Problem çözümünde yapay sinir ağları kullanmanın avantajları neler olabilir? Diğer makine öğrenme algoritmaları tercih edilse sonuçlar daha iyi olabilir mi? Sorulara yönelik öğrenci görüşleri sınıf ortamında tartışılabilir. Yine ilgili konu bağlamında Web ortamındaki alternatif düşünceler araştırılarak yorumlanabilir.

3. HAREKETE GEÇ

Eğitmene Not

Eğitmen, öğrencilere ilk önce kod satırlarını tek tek yazdırarak her kod satırında öğrenciye kod satırının ne anlama geldiği ile ilgili sorular sorar ve tartışır.

1 nolu kod satırında yapay zekâ kütüphanesi olan **keras** kütüphanesinin **Sequential** fonksiyonu ile boş bir yapay sinir ağı modeli oluşturmak için gerekli olan sınıf çağrılır.

2 ve 3 nolu kod satırında yapay zekâ kütüphanesi olan **keras** kütüphanesinin **Dense, Input ve Dropout** fonksiyonu ile yapay sinir ağı tam bağlı katmanları tanımlamak için gerekli olan sınıf çağrılır.

4 nolu kod satırında matematiksel dizi işlemlerini gerçekleştirmek için **numpy** kütüphanesi çağrılır.

5 nolu kod satırında veri okuma işlemini gerçekleştirmek için **pandas** kütüphanesi çağrılır.

6 nolu kod satırında veri setini bölmek için **sklearn.model_selection** kütüphanesinden **train_test_split** fonksiyonu çağrılır.

7 nolu kod satırında model eğitiminden elde edilen sonuçları çizdirmek için **matplotlib.pyplot** kütüphanesi çağrılır.

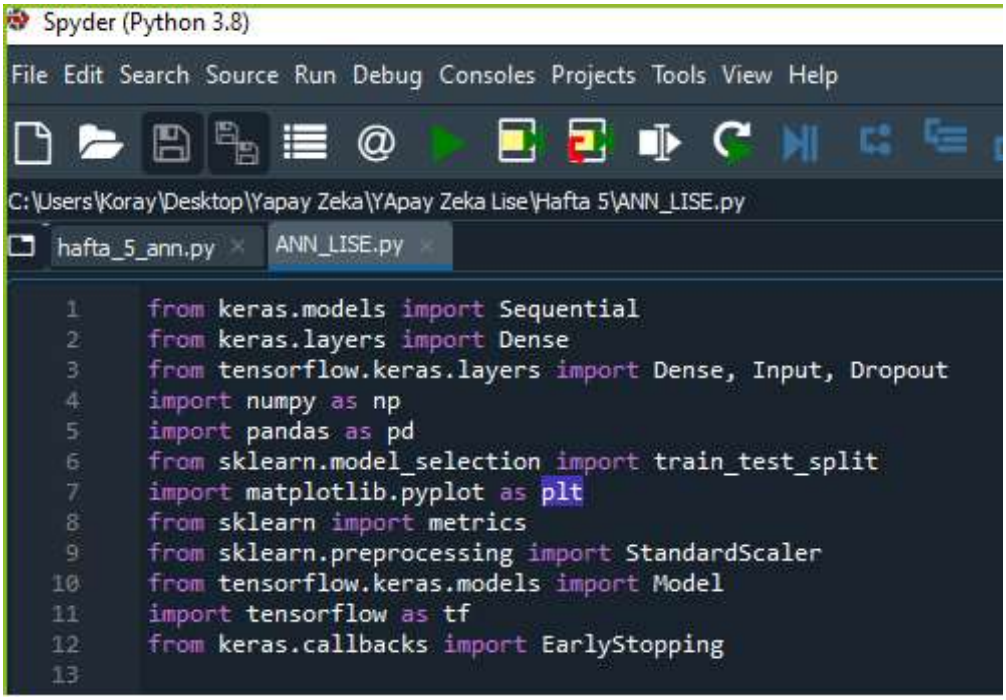
8 nolu kod satırında ise **sklearn** kütüphanesinden **metrics** fonksiyonu ile modelin değerlendirebilmesi için gerekli fonksiyon çağrılır.

9 nolu kod satırında ise veri ölçeklendirme işlemi için **sklearn.preprocessing** kütüphanesinden **StantardScaler** fonksiyonu çağrılır.

10 nolu kod satırında ise yapay sinir ağı modeli oluşturmak için **tensorflow.keras.models** kütüphanesinden **Model** fonksiyonu çağrılır.

11 nolu kod satırında ise yapay zekâ kütüphanesi olan **tensorflow** kütüphanesi çağrılır.

12 nolu kod satırında ise yapay sinir ağı modeli eğitilirken istenilen doğruluk oranına ulaşıldığında eğitimin durması için **keras.callbacks** kütüphanesinden **EarlyStopping** fonksiyonu çağrılır.



Şekil 5.7. Gerekli kütüphanelerin kod gösterimi

Öğrenciler Şekil 5.8'de görüldüğü gibi veri seti ile ilgili düzenlemeleri 14.-19. kod satırları arasında gerçekleştirir.

14. kod satırında pandas kütüphanesinin **read_excel** özelliğini kullanarak veri seti ile aynı konumda olan kod dosyasının içerisindeki 'ENB2012_data.xlsx' veri dosyasını okuyarak dataset değişkenine aktarır.

15. kod satırında **values** fonksiyonu ile excel veri seti dosyasındaki başlıklar hariç verileri okur ve iki boyutlu diziye dönüştürür.

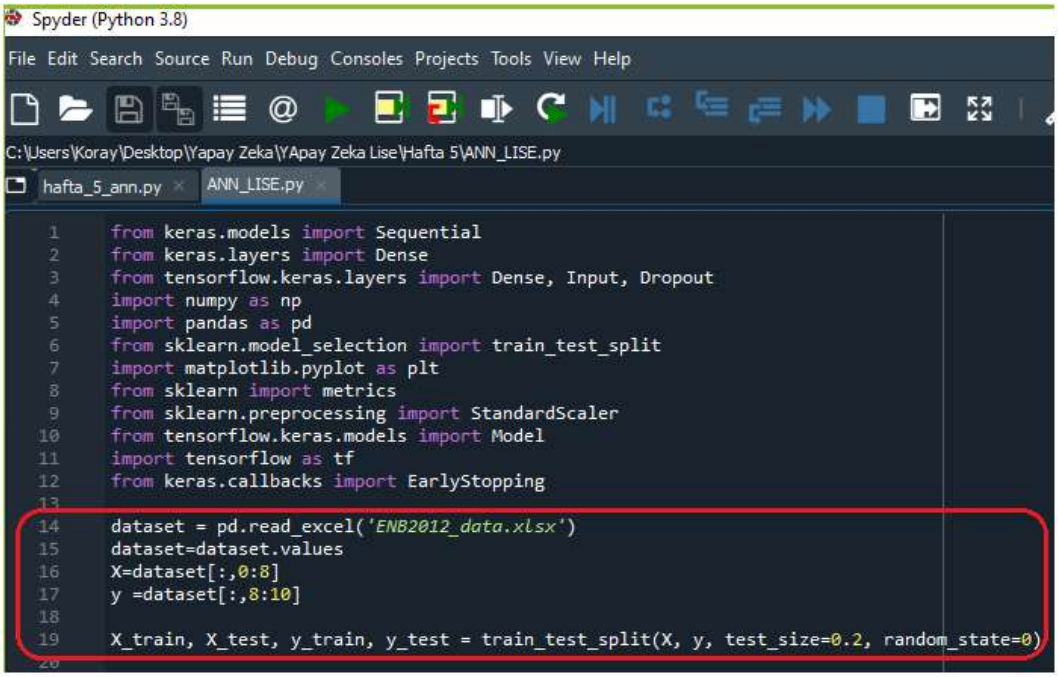
16. kod satırında dataset değişkenindeki veri setindeki ilk sekiz sütunu ([:,0:8]) giriş parametresi olarak ayarlayarak "**X**" değişkenine aktarır.

Eğitmene Not

Kod satırındaki ([:,0:8]) ifadesinde virgülden önce **":"** karakteri yazılmasının sebebi, veri setindeki tüm satırları okumak içindir. **"0:8"** ifadesi ise veri setinde yer alan ilk sekiz sütunun giriş parametresi olarak dikkate alınmasını sağlar.

17. kod satırında dataset değişkeninde yer alan veri setindeki dokuzuncu ve onuncu sütunu ([:,8:10]) çıkış parametresi olarak ayarlayarak "**y**" değişkenine aktarır.

19. kod satırında veri setindeki eğitim ve test verilerini %80 eğitim, %20 test olacak şekilde **"test_size=0,2"** komutu kullanarak rastgele ayırır.



Şekil 5.8. Veri setinin giriş, çıkış ve eğitim için bölünmesi kod ekranı.

Öğrenciler 21-29 kod satırları arasında veri setinin ölçeklendirilmesi işlemini gerçekleştirir.

21. kod satırında 9. Kod satırında çağrılan **StandardScaler** fonksiyonunu tanımlayarak sc değişkenine aktarır.

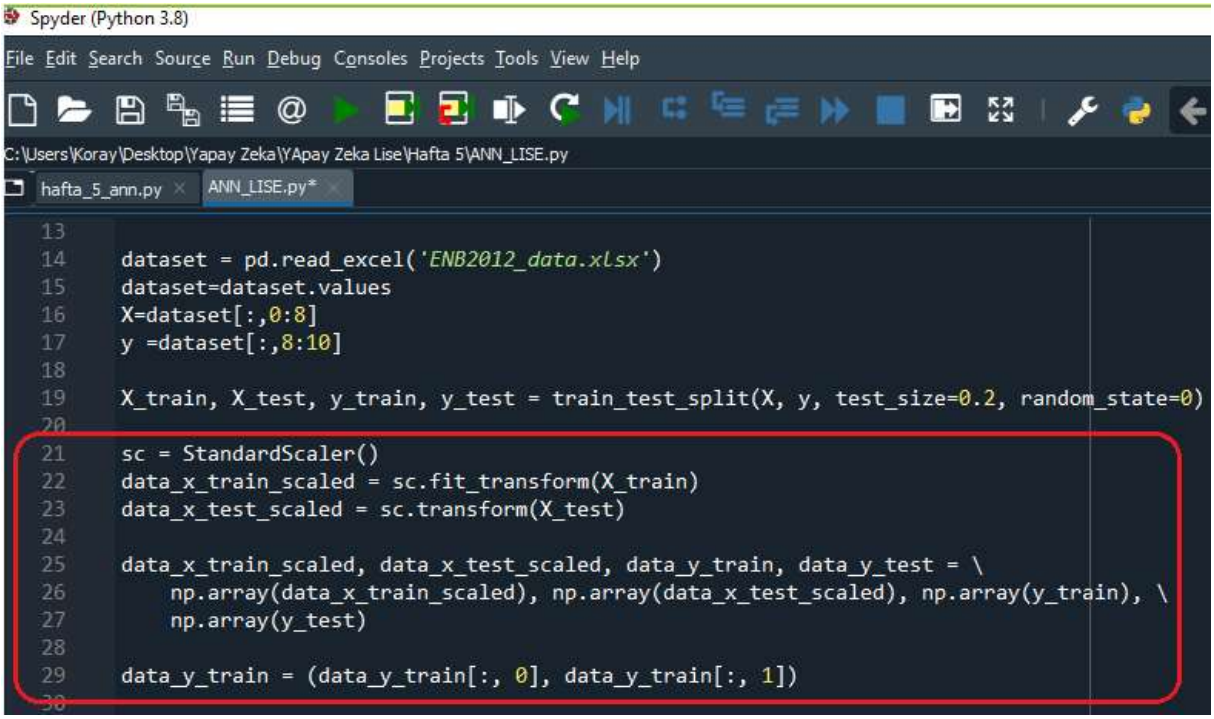
22. ve 23. kod satırında eğitim ve test giriş verilerinin **"fit_transform"** ve **"transform"** özellikleri kullanılarak ölçeklendirilir.

25.-27. kod satırında ölçeklendirilen giriş ve çıkış verileri **numpy** kütüphanesi kullanılarak YSA ile eğitim yapabilmek için **array** formatına dönüştürülür.

29. kod satırında ise YSA ile eğitilecek modelden iki adet çıkış parametresi olacağı için **"tuple"** formatına dönüştürülmüştür.

Eğitmene Not

Eğitmen tuple kavramını şu şekilde açıklar:
Kod satırındaki ([:,0]) ifadesinde virgülden önce **":"** karakteri yazılarak çıkışlardan ilk sütun alınır, dolayısıyla ([:,1]) ifadesi ile de çıkışlardan ikinci sütun alınıp eşleştirme yapılmış olur.



Şekil 5.9. Veri setinin ölçeklendirme

Şekil 5.12'de gösterildiği gibi 31-39 kod satırları arasında yer alan YSA modelinin yapısının oluşumu ile ilgili katman bilgileri verilmiştir.

31. kod satırında giriş parametresi ile eşit nöron sayılı giriş katmanı oluşturulur.

32. kod satırında 128 nöronlu ve Relu aktivasyon fonksiyonlu ilk gizli katmanı oluşturularak giriş katmanına bağlanır.

33. kod satırında **"Dropout"** ilk gizli katmandaki [0,3] ifadesi ile %30 oranında önemsiz nöronlar dışarıya bırakılır (yani kullanılmaz). Dolayısıyla YSA modeli eğitim süreci hızlanır, aşırı öğrenme (overfitting) engellenir.

34. kod satırında 128 nöronlu ve Relu aktivasyon fonksiyonlu ikinci gizli katman oluşturularak ilk gizli katmana bağlanır.

35. kod satırında "Dropout" ikinci gizli katmandaki [0,3] ifadesi ile %30 oranında önemsiz nöronlar dışarıya bırakılır (yani kullanılmaz). Dolayısıyla YSA modeli eğitim süreci hızlanır, aşırı öğrenme (overfitting) engellenir.

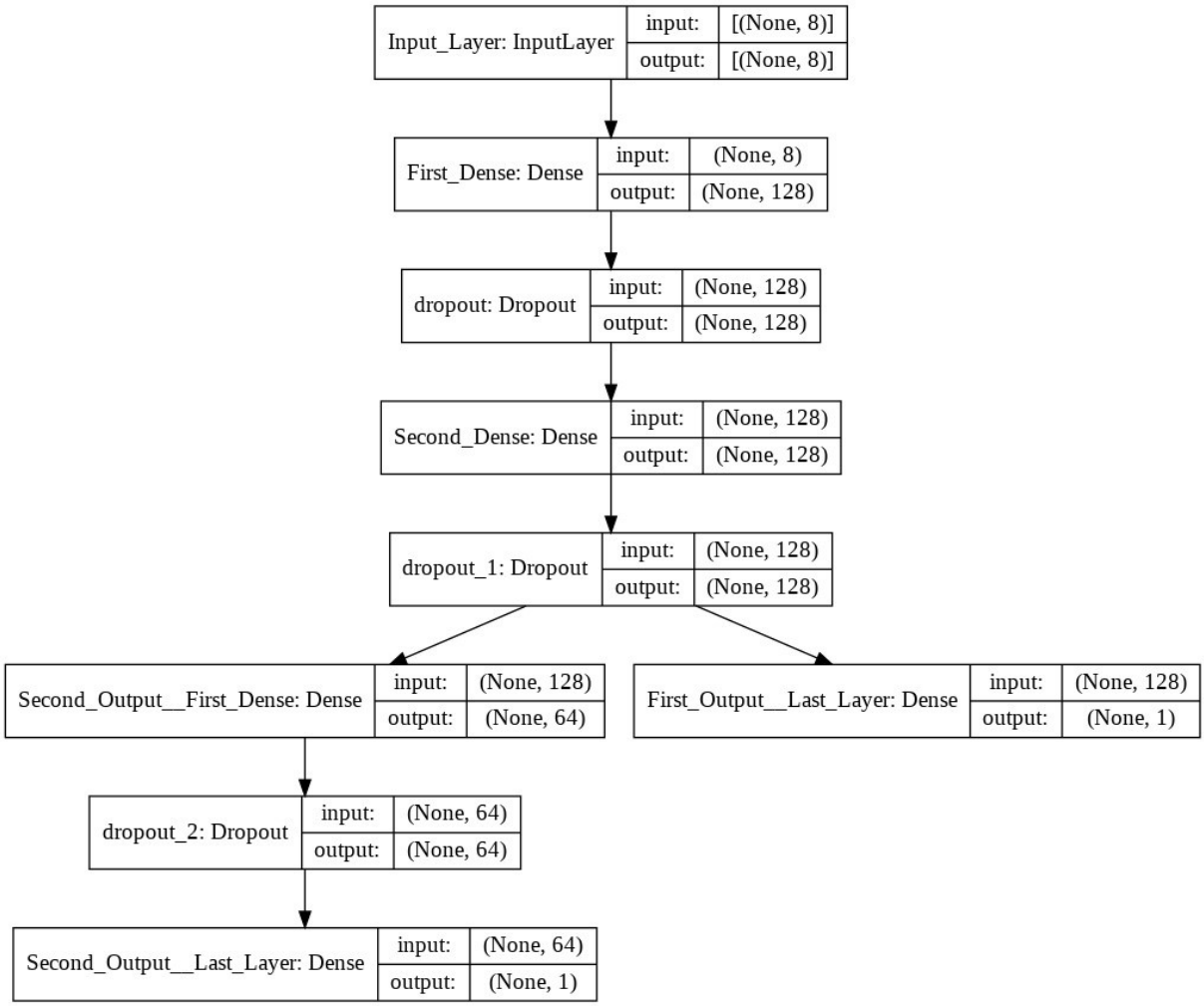
36. kod satırında birinci çıkış parametresi için tek (1) nöronlu son katmanı alır.

37. kod satırında ikinci çıkış için kullanmak üzere 64 nöronlu gizli katmanı ve Relu aktivasyon fonksiyonlu son gizli katmanı oluşturur.

38. kod satırında "Dropout" son gizli katmandaki [0,3] ifadesi ile %30 oranında önemsiz nöronlar dışarıya bırakılır (yani kullanılmaz). Dolayısıyla YSA modeli eğitim süreci hızlanır, aşırı öğrenme (overfitting) engellenir.

39. kod satırında ikinci çıkış parametresi için tek (1) nöronlu son katmanı alır.

41. kod satırında "**Model"** fonksiyonu kullanılarak YSA modelin giriş ve çıkış katmanları tanımlanarak **"model"** değişkenine aktarılır.

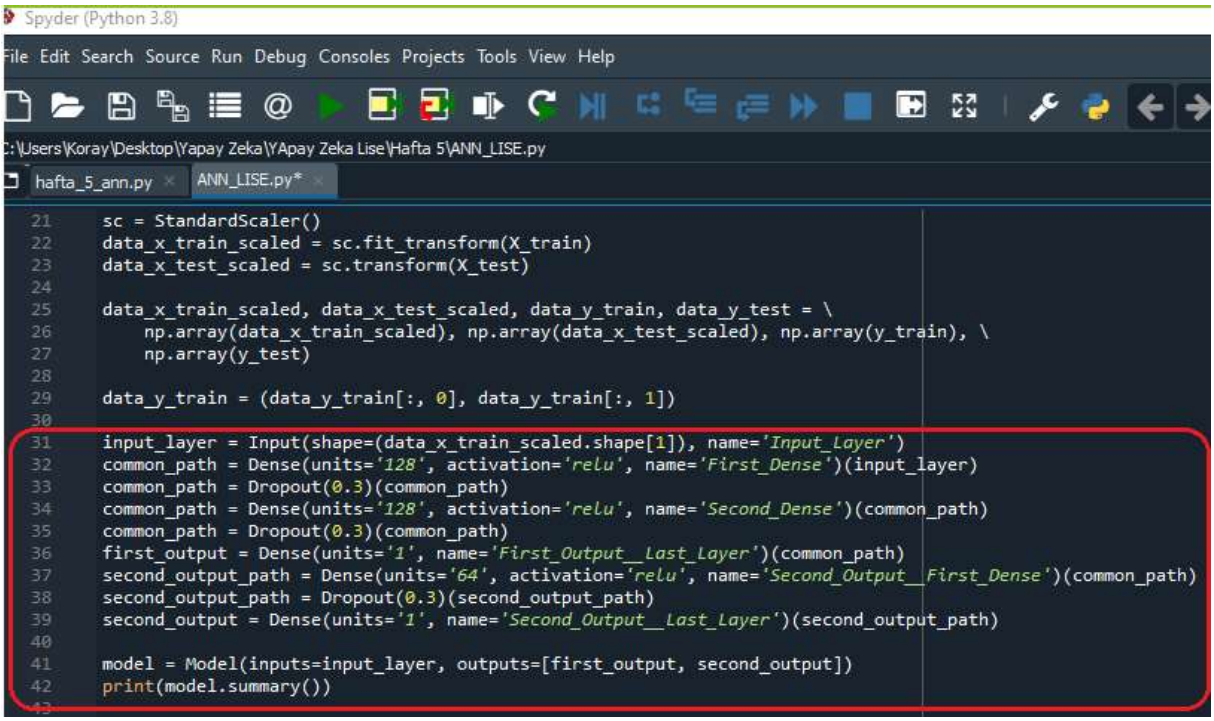


Şekil 5.10. YSA modelin yapısının algoritmik gösterimi

Şekil 5.12'de gösterildiği gibi 42. kod satırında oluşturulan yapay sinir ağı modelindeki nöronların, giriş çıkış parametrelerine ait özellikleri **".summary"** fonksiyonu ile görür.

```
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
Input_Layer (InputLayer)        [(None, 8)]          0
__________________________________________________________________________________________________
First_Dense (Dense)             (None, 128)          1152        Input_Layer[0][0]
__________________________________________________________________________________________________
dropout (Dropout)               (None, 128)          0           First_Dense[0][0]
__________________________________________________________________________________________________
Second_Dense (Dense)            (None, 128)          16512       dropout[0][0]
__________________________________________________________________________________________________
dropout_1 (Dropout)             (None, 128)          0           Second_Dense[0][0]
__________________________________________________________________________________________________
Second_Output__First_Dense (Den (None, 64)           8256        dropout_1[0][0]
__________________________________________________________________________________________________
dropout_2 (Dropout)             (None, 64)           0           Second_Output__First_Dense[0][0]
__________________________________________________________________________________________________
First_Output__Last_Layer (Dense (None, 1)            129         dropout_1[0][0]
__________________________________________________________________________________________________
Second_Output__Last_Layer (Dens (None, 1)            65          dropout_2[0][0]
==================================================================================================
Total params: 26,114
Trainable params: 26,114
Non-trainable params: 0
```

Şekil 5.11. YSA modelin özeti



Şekil 5.12. Yapay sinir ağı modelinin kurulum için kod ekranı

44. kod satırında **"keras"** kütüphanesini **"optimizers"** sınıfı ile SGD optimizasyon yöntemi kullanılarak öğrenme oranı (learning rate) değeri yüz binde bir (0,00001) olarak belirlenip "**optimizer"** değişkenine aktarılmıştır.

Eğitmene Not

Makine öğrenmesinde hata oranını en aza indirgemek için kullanılan yöntemlerinden birisi de SGD optimizasyon yöntemidir (Seyyarer vd., 2020). SGD (Stochastic Gradient Descent) tüm gradient'ler kullanmak yerine rastgele bir kısım gradient'le ağırlıkları değiştirerek optimizasyon yapmaktadır. Mevcut gradient'i ($\partial L\ \partial wt$ /), öğrenme katsayısı (a) ile çarparak mevcut ağırlığı (wt) güncellemektedir (Yazan ve ark., 2017; Zeiler, 2012; Web Kaynağı 5.2, Web Kaynağı 5.3, Web Kaynağı 5.4, Alpaydın, 2018).

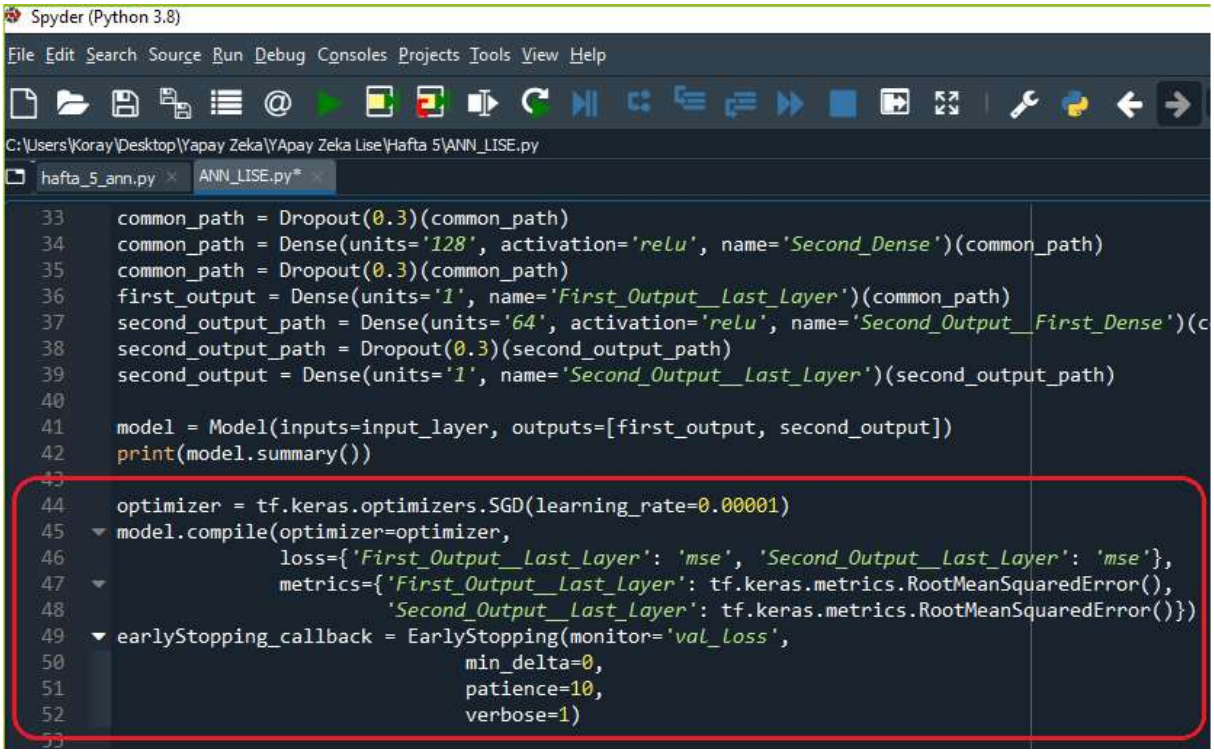
$$w_{t+1} = w_t - a\frac{\partial L}{\partial w_t}$$

45-48 kod satırında **"compile"** fonksiyonu ile kayıp veri değerlerini **"mse (mean square error)",** optimizasyon yöntemi olarak **"SGD"** ve ölçüm metriği olarak **"RootMeanSquaredError (RMSE)"** parametrelerine göre derler.

Eğitmene Not

Eğitmen, kod satırında yer alan **"mse**", ifadesinin makine öğrenmesi modelindeki ortalama kare hatayı hesaplamak için kullanıldığını açıklar. Benzer şekilde, **"RootMeanSquaredError (RMSE)"** ifadesini anlatırken, makine öğrenmesi modelinin tahmin ettiği değerler ile eğitim verisindeki gerçek değerler arasındaki uzaklığın bulunduğunu vurgular.

49-52 kod satırında **Earlystopping** özelliği ile modelin eğitimi esnasında "rmse" değerinin stabil (durağan) olduğu durumda eğitimi durdurur. Kod satırlarında yer alan **"min_delta=0"** ifadesi modelin stabil kaldığı anlamına gelir. **"patience=10"** ifadesi ise eğitim esnasında 10 epoch (tekrar) boyunca modelin doğruluğu değişmez ise eğitim durdurulur. **"verbose=1"** ise eğitim esnasında eğitim sonuçlarını ekranda gösterir.



Şekil 5.13. Yapay sinir ağı modelinin parametrelerinin tanımlanması

4. YÜRÜT

Öğrenciler 54-61 satırları arasında farklı binalara ait enerji verimliliği analizi için kurulan yapay sinir ağı modelinin eğitimini gerçekleştirir.

54-55. kod satırında **"model.fit"** komutu yazarak ölçeklendirilmiş giriş eğitim verilerine göre çıkış eğitim verileri için yapay sinir ağları ile eğitim gerçekleştirir. Burada her bir eğitim için **"batch_size"** fonksiyonuna aktarılan 10 değeri ile eğitime alınacak veri sayısı belirlenmektedir. Yine (**epochs=500**), **"verbose=0"** ifadeleri ile 500 adet eğitim gerçekleştirileceği ve her bir eğitimde elde edilecek sonuçların konsolda gösterilmeyeceği belirtilmektedir. Eğitilen modelin doğrulaması eğitim verilerinin %30'u alınarak (**validation_split=0,3**) gerçekleştirilir. Ayrıca callbacks özelliği kapsamında earlystopping_callback tanımlanarak eğitimin erken durma özelliği aktif hale getirilmiştir.

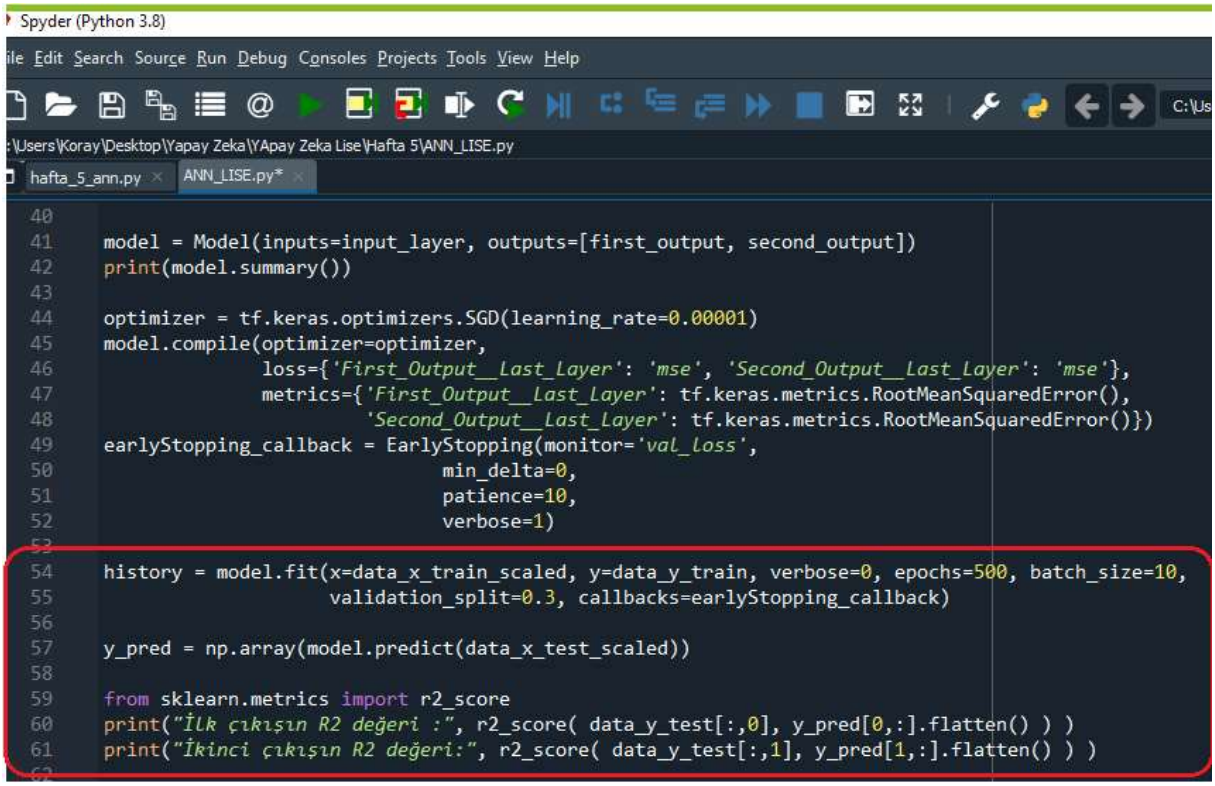
57 kod satırında ise YSA ile eğitim işlemi sonunda giriş test verilerine göre **".predict"** fonksiyonu ile ısıtma ve soğutma yükü için tahminleme işlemini gerçekleştirir.

59. kod satırında **"sklearn.metrics"** kütüphanesinden **"r2_score"** özelliğini yükler.

60-61. kod satırında YSA modelinden elde edilen tahmin sonuçları ile gerçek sonuçlar karşılaştırılarak elde edilen iki boyutlu dizi sonuçları **".flatten"** fonksiyonu kullanılarak tek boyutlu diziye çevrilir, sonuçlar R^2 performans ölçüt kriterine göre doğruluk değerleri **"print"** komutu kullanılarak konsola yazdırılır.

Eğitmene Not

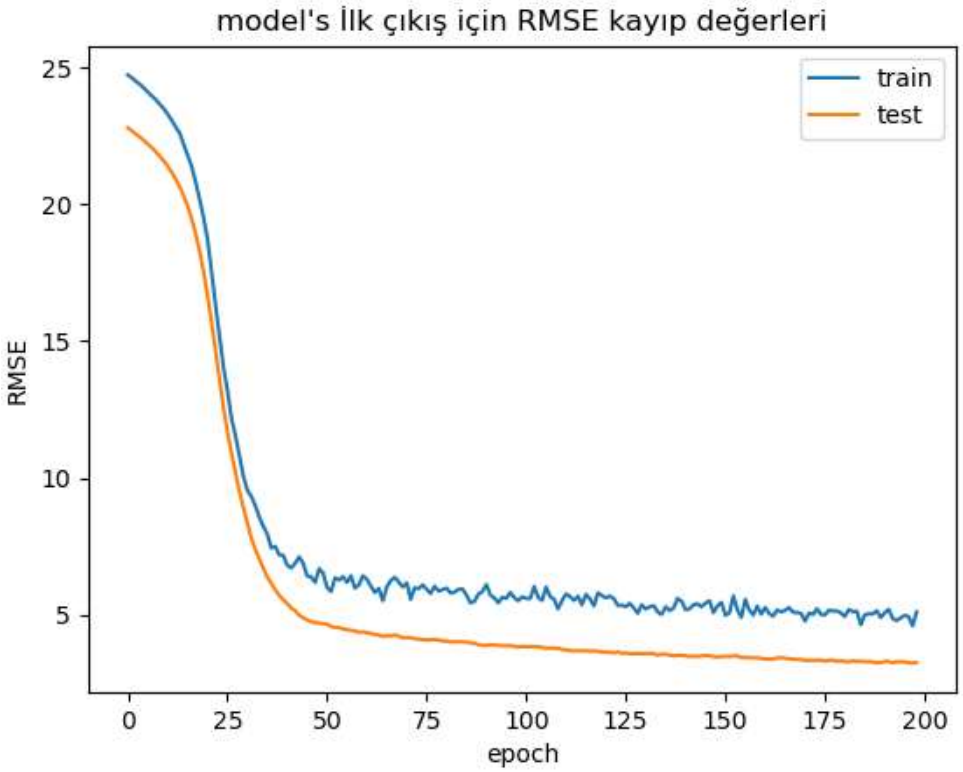
Eğitmen öğrencilere regresyon analizlerinde, deneysel verilerin doğrusal bir eğriye ne kadar uyumlu olduğunu belirlemede "**Determinasyon katsayısı/R^2"** değerinin kullanıldığını ifade eder. Yapay zekâ işlemlerinde elde edilen R^2 değerlerini öğrencilerle birlikte tartışır. $R^2 = 1$ olmasının, deneysel verilerin kusursuz bir doğrusal eğri sağladığının kanıtı olduğunu öğrencilerle paylaşır.



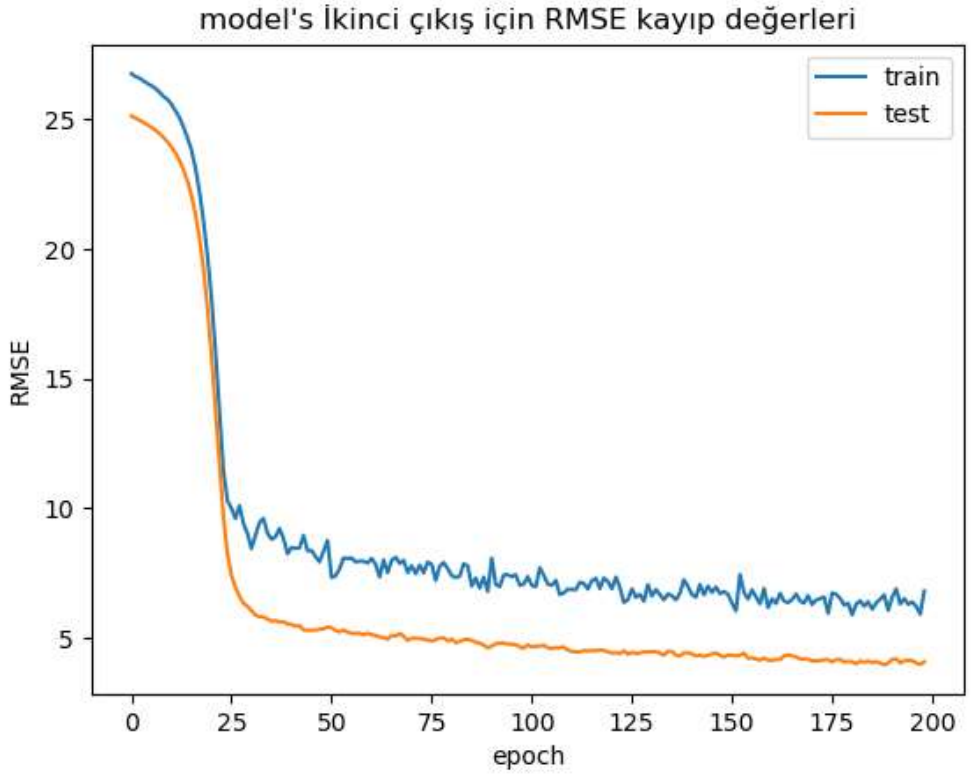
Şekil 5.14. Yapay sinir ağları modelinde eğitim ve tahminleme işlemleri için kod ekranı

5. KARAR VER

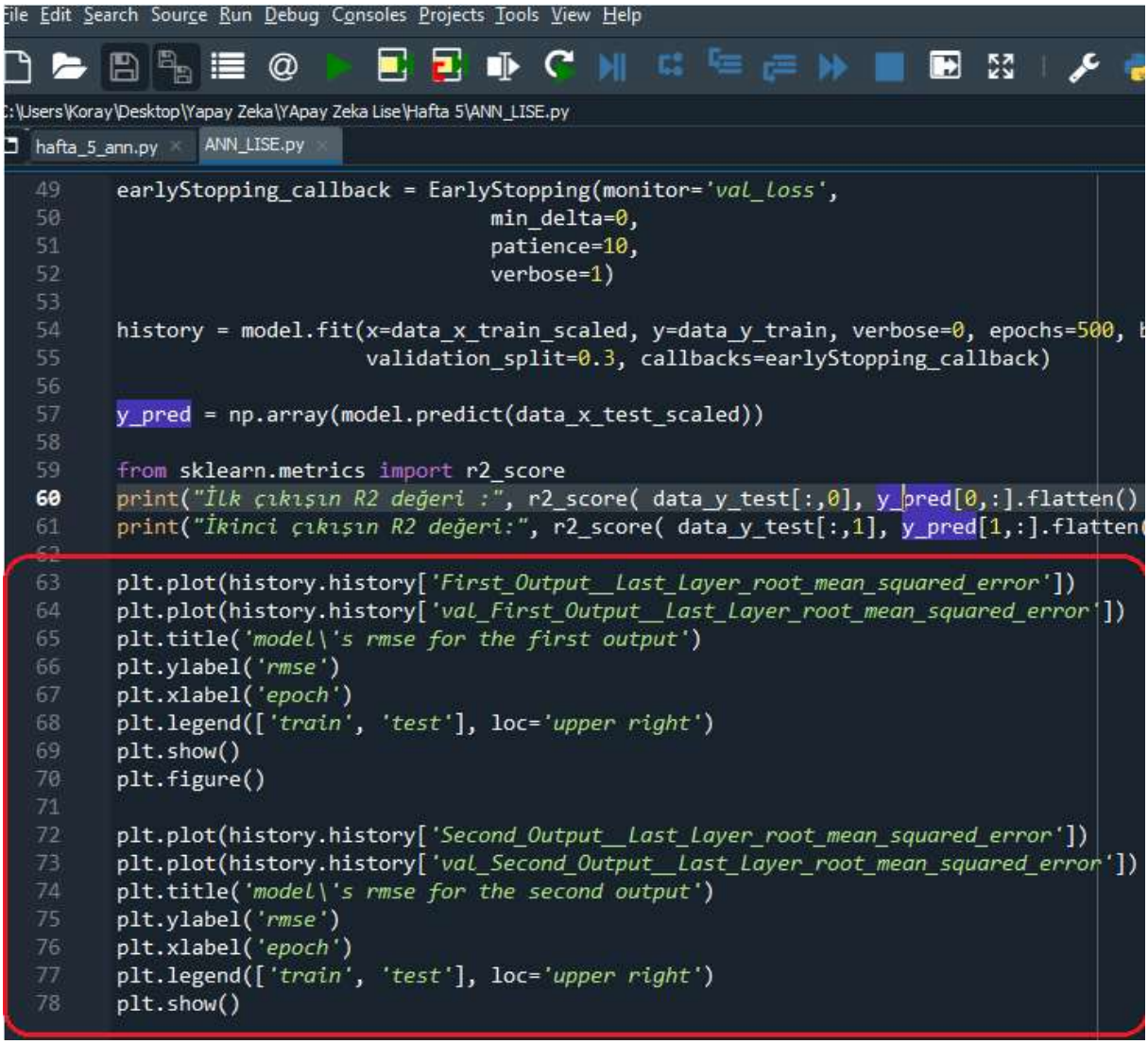
63.-70. ve 72-79. Kod satırlarında ısıtma ve soğutma yüküne ait kayıp çıkış verilerine göre eğitim ve doğrulama grafikleri çizdirilmiştir (İlgili kodun tamamı Github platformunda, Hafta 5 altında H5_ysa_bina-enerji.py dosyası olarak paylaşılmış durumdadır). Şekil 5.15'te ilk çıkış için RMSE kayıp değerleri, Şekil 5.16'da ise ikinci çıkış için RMSE kayıp değerlerine ait grafikler çizdirilmiştir. Grafikler incelendiğinde eğitim ve test verilerindeki kayıp veri oranlarının azalması modelin doğru biçimde eğitildiğinin göstergesidir.



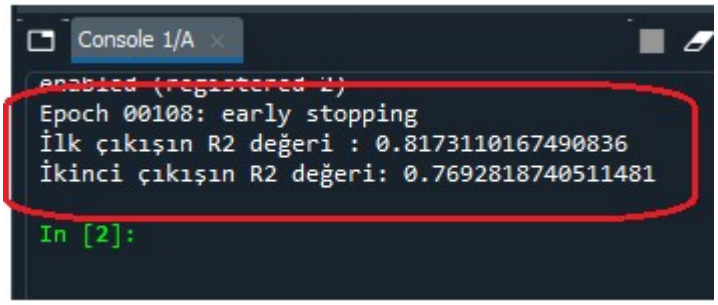
Şekil 5.15. İlk çıkış için RMSE kayıp değerleri grafiği



Şekil 5.16. İkinci çıkış için RMSE kayıp değerleri grafiği



Şekil 5.17. YSA modelinden kayıp çıkış verilerine göre eğitim ve doğrulama grafiklerinin çizilmesi



Şekil 5.18. YSA modeli ile pamuk tarlası sulama sistemi için doğruluk sonucu

Eğitmene Not

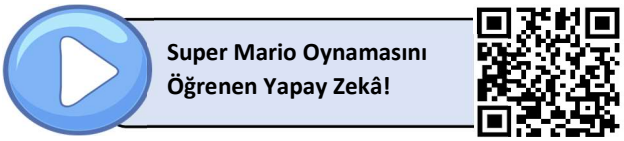
Eğitmen, öğrencilere YSA tekniği ile farklı binalara ait enerji verimliliği analizi amacıyla kurulan model için Erken durdurma (Early stopping) 108. Epoch (eğitim) gerçekleştiğini göstererek anlatır.

PYTHON KODLARI:

```
from keras.models import Sequential
from keras.layers import Dense
from tensorflow.keras.layers import Dense, Input, Dropout
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
from sklearn import metrics
from sklearn.preprocessing import StandardScaler
from tensorflow.keras.models import Model
import tensorflow as tf
from keras.callbacks import EarlyStopping

dataset = pd.read_excel('ENB2012_data.xlsx')
dataset=dataset.values
X=dataset[:,0:8]
y =dataset[:,8:10]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)

sc = StandardScaler()
data_x_train_scaled = sc.fit_transform(X_train)
data_x_test_scaled = sc.transform(X_test)

data_x_train_scaled, data_x_test_scaled, data_y_train, data_y_test = \
   np.array(data_x_train_scaled), np.array(data_x_test_scaled), np.array(y_train), \
   np.array(y_test)

data_y_train = (data_y_train[:, 0], data_y_train[:, 1])
```

```
input_layer = Input(shape=(data_x_train_scaled.shape[1]), name='Input_Layer')
common_path = Dense(units='128', activation='relu',
name='First_Dense')(input_layer)
common_path = Dropout(0.3)(common_path)
common_path = Dense(units='128', activation='relu',
name='Second_Dense')(common_path)
common_path = Dropout(0.3)(common_path)
first_output = Dense(units='1', name='First_Output__Last_Layer')(common_path)
second_output_path = Dense(units='64', activation='relu',
name='Second_Output__First_Dense')(common_path)
second_output_path = Dropout(0.3)(second_output_path)
second_output = Dense(units='1',
name='Second_Output__Last_Layer')(second_output_path)

model = Model(inputs=input_layer, outputs=[first_output, second_output])
print(model.summary())

optimizer = tf.keras.optimizers.SGD(learning_rate=0.00001)
model.compile(optimizer=optimizer,
        loss={'First_Output__Last_Layer': 'mse', 'Second_Output__Last_Layer':
'mse'},
        metrics={'First_Output__Last_Layer':
tf.keras.metrics.RootMeanSquaredError(),
                'Second_Output__Last_Layer':
tf.keras.metrics.RootMeanSquaredError()})
earlyStopping_callback = EarlyStopping(monitor='val_loss',
                    min_delta=0,
                    patience=10,
                    verbose=1)
```

```
history = model.fit(x=data_x_train_scaled, y=data_y_train, verbose=0, epochs=500,
batch_size=10,
                validation_split=0.3, callbacks=earlyStopping_callback)

y_pred = np.array(model.predict(data_x_test_scaled))

from sklearn.metrics import r2_score
print("İlk çıkışın R2 değeri :", r2_score( data_y_test[:,0], y_pred[0,:].flatten() ) )
print("İkinci çıkışın R2 değeri:", r2_score( data_y_test[:,1], y_pred[1,:].flatten() ) )

plt.plot(history.history['First_Output__Last_Layer_root_mean_squared_error'])
plt.plot(history.history['val_First_Output__Last_Layer_root_mean_squared_error'])
plt.title('model\'s İlk çıkış için RMSE kayıp değerleri')
plt.ylabel('RMSE')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper right')
plt.show()
plt.figure()

plt.plot(history.history['Second_Output__Last_Layer_root_mean_squared_error'])
plt.plot(history.history['val_Second_Output__Last_Layer_root_mean_squared_error'])
plt.title('model\'s İkinci çıkış için RMSE kayıp değerleri')
plt.ylabel('RMSE')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper right')
plt.show()
```



Düşün, tartış…

Bir Yapay Sinir Ağları'nın farklı verilerle karşılaşarak yanlış öğrenme gerçekleştirmesini sağlamak mümkün olabilir mi? Soru bağlamında öğrenci görüşleri sınıf ortamında tartışılabilir ve yine Web'te bulunabilecek alternatif düşünceler araştırılarak yorumlanabilir.

Dünyadan Haberler

Türkiye'nin Ulusal Yapay Zekâ Stratejisi

Cumhurbaşkanlığı Dijital Dönüşüm Ofisi Başkanlığı ile Sanayi ve Teknoloji Bakanlığı iş birliğinde ve ilgili tüm paydaşların etkin katılımıyla hazırlanan "*Ulusal Yapay Zekâ Stratejisi 2021-2025*"'e ilişkin 2021/18 sayılı Cumhurbaşkanlığı Genelgesi 20/08/2021 tarihli ve 31574 sayılı Resmî Gazete'de yayımlanarak yürürlüğe girmiştir.

Yapay zekâ alanında ülkemizin ilk ulusal strateji belgesi olma özelliğini taşıyan Ulusal Yapay Zekâ Stratejisi (UYZS) ile Türkiye, yapay zekâ (YZ) stratejisini yayımlayan ülkeler arasında yerini almıştır. UYZS, On Birinci Kalkınma Planı ile Cumhurbaşkanlığı Yıllık Programları doğrultusunda, "Dijital Türkiye" vizyonu ve "Milli Teknoloji Hamlesi" ile uyumlu olarak hazırlanmıştır. Vizyonu "*müreffeh bir Türkiye için çevik ve sürdürülebilir yapay zekâ ekosistemiyle küresel ölçekte değer üretmek*" olan Strateji, 6 stratejik öncelik etrafında tasarlanmıştır: (1) YZ Uzmanlarını Yetiştirmek ve Alanda İstihdamı Artırmak, (2) Araştırma, Girişimcilik ve Yenilikçiliği Desteklemek, (3) Kaliteli Veriye ve Teknik Altyapıya Erişim İmkânlarını Genişletmek, (4) Sosyoekonomik Uyumu Hızlandıracak Düzenlemeleri Yapmak, (5) Uluslararası İş Birliklerini Güçlendirmek, (6) Yapısal ve İşgücü Dönüşümünü Hızlandırmak.

UYZS ile 2021-2025 yılları arasında ülkemizde yürütülen YZ alanındaki çalışmaları ortak bir zemine oturtacak tedbirler ve bu tedbirleri hayata geçirmek üzere oluşturulacak yönetişim mekanizması ortaya konulmaktadır (Web Kaynağı 5.5).

6. İLAVE ETKİNLİK

Şekil 5.19'da gösterildiği gibi MNIST (Modified National Institue of Standarts) veri seti, içerisinde yer alan 60.000 adet 28x28 piksel boyutuna sahip farklı insanların el yazmalarından kesilmiş resimlerden oluşmaktadır. Bunlara karşılık gelen 60.000 tane sayı (skaler) vardır. MNIST veri setini yapay sinir ağları kullanarak veri seti içerisinde yer alan resmin hangi rakama ait olduğunu tahminleyen bir model oluşturunuz (Veri seti keras kütüphanesi içerisinden otomatik

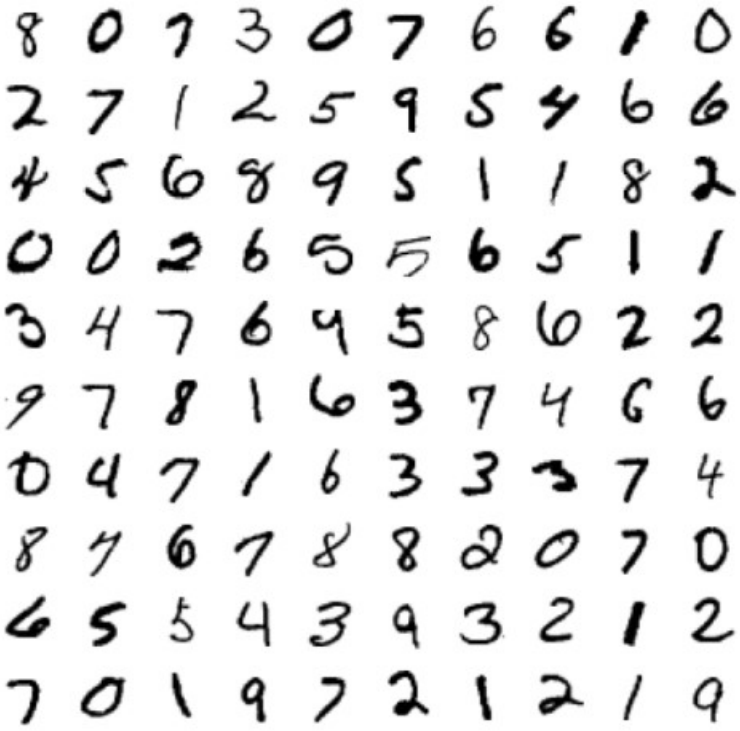
çekilmektedir. İlgili kodun tamamı Github platformunda, Hafta 5 altında H5_ysa_mnist.py dosyası olarak paylaşılmış durumdadır).

Eğitmene Not

Eğitmen, öğrencilere Python içinde vektörize edilmiş bir biçimde, yani resimlerin piksel değerleriyle sayısallaştırılmış halinin hazır bulunduğunu açıklar. Çizelge 5.2'de gösterildiği gibi 28x28 boyutundaki el yazması görüntülerin giriş parametresini, bu görüntülere ait sayısallaştırmış değerin çıkış sınıfını temsil ettiğini anlatır.

Çizelge 5.2. Veri setinin giriş çıkış parametreleri

Veri Sayısı	GİRİŞ PARAMETRESİ	ÇIKIŞ PARAMETRESİ
	Görüntü	Sınıf
1		8
2		0
…	…	…
59000		7
60000		6

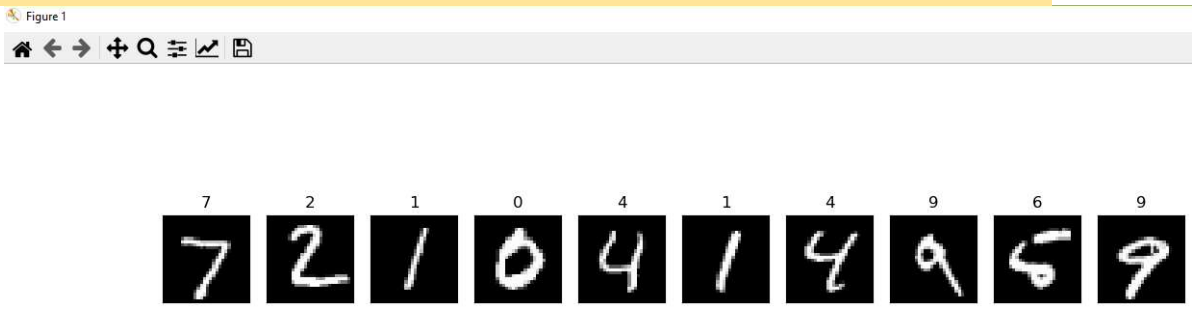


Şekil 5.19. El yazısı görüntüsü

PYTHON KODLARI:

```
import numpy as np
import matplotlib.pyplot as plt
from keras.layers import Dense, Flatten
from keras.models import Sequential
from tensorflow.keras.utils import to_categorical
from keras.datasets import mnist

(X_train, y_train), (X_test, y_test) = mnist.load_data()
print(X_train.shape)
print(X_test.shape)

temp = []

for i in range(len(y_train)):
    temp.append(to_categorical(y_train[i], num_classes=10))
y_train = np.array(temp)
# Convert y_test into one-hot format
temp = []
for i in range(len(y_test)):
    temp.append(to_categorical(y_test[i], num_classes=10))
y_test = np.array(temp)

model = Sequential()
model.add(Flatten(input_shape=(28,28)))
model.add(Dense(16, activation='sigmoid'))
model.add(Dense(10, activation='softmax'))
model.summary()

model.compile(loss='categorical_crossentropy',
```

```
            optimizer='adam',
            metrics=['acc'])

model.fit(X_train, y_train, epochs=15,
          validation_data=(X_test,y_test))

predictions = model.predict(X_test)
predictions = np.argmax(predictions, axis=1)

fig, axes = plt.subplots(ncols=10, sharex=False,
                         sharey=True, figsize=(20, 4))
for i in range(10):
    axes[i].set_title(predictions[i])
    axes[i].imshow(X_test[i], cmap='gray')
    axes[i].get_xaxis().set_visible(False)
    axes[i].get_yaxis().set_visible(False)
plt.show()
```



Şekil 5.20. El yazısı genel tanımlama görüntüleri

```
Epoch 1/15
1875/1875 [==============================] - 4s 2ms/step - loss: 1.0922 - acc: 0.6976 - val_loss: 0.6564 - val_acc: 0.8255
Epoch 2/15
1875/1875 [==============================] - 3s 2ms/step - loss: 0.5876 - acc: 0.8316 - val_loss: 0.5526 - val_acc: 0.8375
Epoch 3/15
1875/1875 [==============================] - 3s 2ms/step - loss: 0.5009 - acc: 0.8514 - val_loss: 0.4441 - val_acc: 0.8717
Epoch 4/15
1875/1875 [==============================] - 3s 2ms/step - loss: 0.4859 - acc: 0.8538 - val_loss: 0.4770 - val_acc: 0.8582
Epoch 5/15
1875/1875 [==============================] - 3s 2ms/step - loss: 0.4550 - acc: 0.8642 - val_loss: 0.4353 - val_acc: 0.8689
Epoch 6/15
1875/1875 [==============================] - 3s 2ms/step - loss: 0.4414 - acc: 0.8662 - val_loss: 0.4198 - val_acc: 0.8703
Epoch 7/15
1875/1875 [==============================] - 3s 2ms/step - loss: 0.4269 - acc: 0.8720 - val_loss: 0.4199 - val_acc: 0.8782
Epoch 8/15
1875/1875 [==============================] - 3s 2ms/step - loss: 0.4331 - acc: 0.8705 - val_loss: 0.3994 - val_acc: 0.8855
Epoch 9/15
1875/1875 [==============================] - 3s 2ms/step - loss: 0.4315 - acc: 0.8734 - val_loss: 0.4424 - val_acc: 0.8638
Epoch 10/15
1875/1875 [==============================] - 4s 2ms/step - loss: 0.4144 - acc: 0.8763 - val_loss: 0.4027 - val_acc: 0.8851
Epoch 11/15
1875/1875 [==============================] - 5s 3ms/step - loss: 0.3985 - acc: 0.8798 - val_loss: 0.3734 - val_acc: 0.8877
Epoch 12/15
1875/1875 [==============================] - 5s 3ms/step - loss: 0.4034 - acc: 0.8796 - val_loss: 0.4140 - val_acc: 0.8717
Epoch 13/15
1875/1875 [==============================] - 5s 3ms/step - loss: 0.3992 - acc: 0.8813 - val_loss: 0.3721 - val_acc: 0.8944
Epoch 14/15
1875/1875 [==============================] - 5s 3ms/step - loss: 0.3813 - acc: 0.8880 - val_loss: 0.3669 - val_acc: 0.8968
Epoch 15/15
1875/1875 [==============================] - 4s 2ms/step - loss: 0.3783 - acc: 0.8887 - val_loss: 0.3883 - val_acc: 0.8897
```

Şekil 5.21. Epoch bazlı genel akış

Eğitmene Not

Eğitmen, bu hafta eğitimi hakkında öğrencilerin ilgisini arttırmak ve yeni öğrenilen bilgileri önceden öğrenilen bilgiler ile karşılaştırmalarını sağlamak için aşağıda maddeler halinde verilen soruları tartışır.

- İlgili haftada ele alınan yapay zekâ uygulamalarına uygun örnek veri setleri neler olabilir?
- İlgili haftada ele alınan veri setlerinin yapay zekâ teknik ve kütüphanelerinde ne derecede başarım etkisi olmuştur?
- İlgili haftada öğretilen yapay zekâ tekniği ile uyumlu ve uyumsuz olabilecek açık erişimli veri seti çeşitleri sunan (kaagle, github vs.) internet sitelerinden elde edilecek farklı veri çeşitleri ile yapay zekâ tekniği uyumluluğunu sorar.

Kaynakça

Alpaydın, E., (2018). “Yapay Öğrenme”. Boğaziçi Üniversitesi Yayınevi, İstanbul.

Tsanas, A., ve Xifara, A. (2012). Accurate quantitative estimation of energy performance of residential buildings using statistical machine learning tools. *Energy and Buildings*, 49, 560-567.

Seyyarer, E., Ayata, F., Uçkan, T., & Karcı, A. (2020). Derin öğrenmede kullanılan optimizasyon algoritmalarinin uygulanması ve kıyaslanması. *Computer Science*, 5(2), 90-98.

Yazan, E., ve Talu, M.F., (2017). "Comparison of the stochastic gradient descent based optimization techniques." Artificial Intelligence and Data Processing Symposium (IDAP), 2017 International. IEEE,

Zeiler, M. D. (2012). Adadelta: an adaptive learning rate method. arXiv preprint arXiv:1212.5701.

Web Kaynağı 5.1: http://velar.com.tr/enerji-verimliligi/binalarda-enerji-verimliligi/

Web Kaynağı 5.2: https://ruder.io/optimizing-gradient-descent/

Web Kaynağı 5.3: https://medium.com/deep-learning-turkiye/gradient-descent-nedir-3ec6afcb9900

Web Kaynağı 5.4: https://towardsdatascience.com/10-gradient-descent-optimisation-algorithms-86989510b5e9

Web Kaynağı 5.5: https://cbddo.gov.tr/uyzs

6. Hafta: Etmen Tabanlı Modelleme

Ön Bilgi:

- Etmen tabanlı Yapay Zekâ kavramının temelleri, etmen tabanlı modellemenin temelleri
- Python kodlamada döngüler, fonksiyonlar ve kütüphane yapılarının kullanımı (Bu konuda Python anlatım videoları: 13. Fonksiyonlar, 14. Python Kütüphane Kullanımı yardımcı araç olarak kullanılabilecektir).

Haftanın Kazanımları:

- Öğrenciler etmen tabanlı modelleme kavramını kavrar.
- Öğrenciler, etmen modelleme üzerine problem çözümleri tasarlayarak uygular.
- Öğrenciler etmen tabanlı modellemenin çözüm süreçlerini yorumlamayı kavrar.
- Öğrenciler birbirleriyle etkileşen etmenlerle problem çözümü üretme yeteneği kazanır.
- Öğrenciler, etmen, çevre, dönüt, çoklu-etmen gibi kavramları anlar.
- Öğrenciler Python programlama dilini kullanarak tipik bir etmen tabanlı çözüme yönelik program kodunu yazabilir.
- Öğrenciler Python programlama dilinde etmen tabanlı modelleme için kullanılan kütüphane fonksiyonlarını kullanma yeteneği kazanır.
- Öğrenciler, örnek bir probleme göre etmen tabanlı modelleme yeteneği kazanır.
- Öğrenciler etmen tabanlı çözüm gerektiren problem için model tasarlar ve bu modeli çözüm yönünde uygular.
- Öğrenciler modelin davranışlarını değerlendirerek yorumlar.
- Öğrenciler etmen modelli programlama mantığının Yapay Zekâ alanı açısından uygulamasını yapar.

Haftanın Amacı:

Bu haftanın amacı, “etmen (agent/ajan)” kavramının tüm öğrenciler tarafından doğru bir şekilde anlaşılmasını sağlamaktır. Haftanın bir diğer amacı, etmen tabanlı modelleme yaklaşımını izleyerek alternatif problem yapıları ile öğrencilerin ilgisini çekecek etkinlikleri gerçekleştirmektir. Yine bu bağlamda, öğrencilerin Python programlama dilini kullanarak etmen tabanlı örnek problem modelleme ve çözüm üretme konularında beceri kazanması sağlanacaktır.

Kullanılacak Malzemeler:

Bilgisayar, kâğıt, kalem, Python kod editörü, örnekler için çıktı görselleri

Haftanın İşlenişi:

Algıla: Öğrenciler etmen ve etmen tabanlı modelleme kavramlarını tanımlayabilir.

Tasarla: Öğrenciler etmen tabanlı modelleme üzerinden örnek problemler için çözüm süreçleri tasarlayabilir. Bu hafta için Github platformu (https://github.com/deneyapyz/lise/) ile etkileşim Tasarla aşaması ile başlatılabilmektedir.

Harekete Geç: Öğrenciler Python programlama dili ile etmen tabanlı bir model yapısı oluşturup uygulayabilir.

Yürüt: Eğitmenler öğrenciler ile etkileşimli bir biçimde etmen modelini çalıştırır ve genel çözüm akışını değerlendirir. Öğrenciler etmen tabanlı yaklaşım doğrultusunda kodlama ve çözüm üretme süreçlerini yürütebilir.

Karar Ver: Öğrenciler alternatif yapay zekâ modelini düzenleyerek / kodlayarak çözüm üretebilir. Öğrenciler ayrıca etmen tabanlı uygulamalar ile elde edilen genel sonuçları inceleyip, tartışabilir.

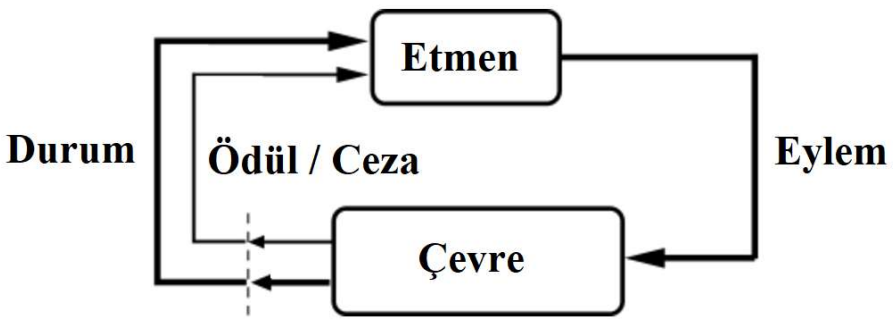
1. ALGILA

1.1. Etmen Tabanlı Yapay Zekâ Modelleme Temelleri

Eğitmene Not

Eğitmen, öğrencilere "**etmen**" ve "**etmen tabanlı yapay zekâ**" kavramları hakkında bilgi sahibi olup olmadıklarını sorar ve tartışır. Böylece öğrencilerin konuya dikkatini çeker.

Yapay Zekâ kapsamında çevresel faktörlerle etkileşim gerektiren problem çözümlemeleri için tasarlanmış çeşitli yaklaşımlar bulunmaktadır. Bu yaklaşımların özünde şu ana dek değinilen farklı Yapay Zekâ algoritmalarının etkileşimsel fonksiyonları içerecek şekilde tasarlanması yer almaktadır. Buna göre, çevreyle etkileşen Yapay Zekâ unsurları kısaca **etmen** ya da İngilizce karşılığından esinlenerek **ajan** (**agent**) olarak adlandırılmaktadır. Şekil 6.1'de tipik bir etmen modeli gösterilmiştir.



Şekil 6.1. Tipik bir etmenin genel yapısı (Web Kaynağı 6.1)

Şekil 6.1'den anlaşılacağı üzere bir etmen, çevresiyle etkileşime girip girdiği etkileşim sonucunda elde ettiği bulgulara göre dönütlerde (yani eylemlerde) bulunabilecek şekilde tasarlanmaktadır. Bir etmenin eylemlerine karar verme aşamasında içerisinde bulunduğu durum ve çevreden aldığı ödül/ceza dönütleri etkili olmakta, bunlar gelecek durumları da yönlendirmektedir. Buradan hareketle, özellikle kolektif, çok sayıda unsurun bir araya gelerek çözüm üretmesi gereken problem durumlarında birden fazla etmen de kullanılabilmektedir. Genel olarak bu yöndeki Yapay Zekâ tabanlı çözüm yaklaşımlarına **etmen tabanlı Yapay Zekâ**, bu yöndeki çözüm süreçlerinin tasarımına da **etmen tabanlı modelleme** adı verilmektedir.

Eğitmene Not

Eğitmen öğrencilere, etmen tabanlı modellemenin aslında robotik sistemlere temel teşkil eden bir konu olduğunu ifade edecektir.

Benzer şekilde; bir insan olarak da çevreyle etkileşimlerimiz neticesinde kararlar alabilen ve eylemler gerçekleştiren unsurlar olarak etmen tabanlı modellemeyle olan benzerliklerimiz konusunda örnekler (örneğin markette kasiyer ile olan iletişimimiz, bir çocuğun sıcak bir unsura elini değdirdiği zaman ulaştığı tecrübe: 'sıcağa dokunmamalıyım'…vs.) vererek, etmen kavramına ilişkin pekiştirici örnekleri de vurgulayarak, öğrencilerin kavramları daha iyi anlamasını sağlayacaktır.

1.2. Etmen Tabanlı Modellemede Problem Tasarımı

Etmen tabanlı çözümlerin çalışacağı problemlerde, etkin çözüm elde etmek adına birtakım faktörlerin/bileşenlerin dikkate alınması ve bunlar altında tanımlamalar yapılması gerekmektedir.

Eğitmene Not

Eğitmen öğrencilere insan özellikleri ve eylemlerinden örnekler verir. Bu bağlamda şöyle bir açıklama yapar: Örneğin, bir insanın göz rengi, boyu, kilosu gibi özellikler net değerler alan değişkenlerdir. Yürümek, koşmak, konuşmak gibi eylemler ise dinamik çözüm yolları; yani fonksiyonlar gibidir. Bu doğrultuda düşünerek, özellikler ve eylemleri birleştirmek suretiyle problemlere çözümler üretiriz.

Eğitmen ayrıca robot süpürgesi örneğini de irdeler. Robot süpürgeyi bir etmen olarak kabul edersek; kamera, kızılötesi gibi sensörler ile çevre algılama sağlanırken, dinamik tepki vericiler olarak çeşitli motorlar, fırçalar, mekanik kollar vs. ile problem çözümü (çevreyi temizlemek/süpürmek) sağlanır.

İlgili faktörleri/bileşenleri genel olarak şöyle ifade edebiliriz:

- **Etmen Sayısı:** Çözümde kullanılacak etmenlerin sayısı, tek bir etmenin mi yoksa çok sayıda etmenin mi (kolektif bir şekilde) çalışacağı belirlenmelidir. Bazı problemler tek bir etmen ile kolayca çözülebilirken, bazı problemler de çok sayıda etmenin hem çevreyle hem de birbirleriyle iletişim halinde olarak çalışmasını ve çözüme ulaşmasını gerekli kılmaktadır.

- **Etmen Nitelikleri:** Çözümde yer alacak etmenlerin sahip olacakları muhtemel parametreler aynı zamanda etmen nitelikleri olarak bilinmektedir. Bu parametreler, etmenlerin eylemlerine, muhakemelerine ve çevreyle ya da diğer unsurlarla (örneğin diğer etmenler) etkileşimlerine yön verecek, düzenlenebilir değerler olmaktadır.
- **Etmen Eylemleri:** Etmen eylemleri, ilgili etmenlerin çevreyle etkileşimleri ve geçerli nitelik değerleri üzerinden karar süreçlerini işletmelerini ve hatta eylemde bulunmalarını içermektedir. Bu eylem yapıları aslında birer fonksiyon olarak tanımlanmaktadır.
- **Çevre:** Etmenlerin içerisinde bulundukları çevrenin, en iyi etmen etkileşimleri için sınırları ve karakteristik nitelikleri ile tanımlanmaları gerekmektedir.

Eğitmene Not

Eğitmen öğrencilere 'etmen odaklı problemler başka neler olabilir?' şeklinde bir soru yönlendirir. Cevabı örneklerle öğrenciler ile tartışır.

İfade edilen unsurlar ile etmenlerin içerisinde bulunacağı problemin de tanımlanması gerekmektedir. Problemle ilgili olarak şu tanım faktörleri önemlidir:

- **Problem Kısıtları:** Problem ile ilgili çözümü temsil eden parametrelerin hangi kısıtlar altında olacağı, etmen davranışları neticesindeki çıktıların da gidişatını belirlemektedir.
- **Çevredeki Dinamik Unsurlar:** Etmenlerin eylemleri neticesinde durumları değişebilecek dinamik çevrenin söz konusu olacağı gibi, çevrede yer alacak ve etmenlerin gelecek davranışlarını şekillendirecek başka dinamik unsurlar da söz konusu olabilmektedir. Özellikle çok etmenli modellemelerde başka etmenler de dinamik unsurlar olarak kabul edilmektedir. Yine tek veya çok etmenli problem çözümlerinde çevreyle bağlantılı değişken parametreli unsurlar da problem çözümünü benzetim odaklı problemler için daha uyumlu hale getirebilmektedir.

Eğitmene Not

Eğitmen öğrencilere robot süpürgelerdeki problem tanım faktörlerinin neler olduğunu sorar ve şu doğrultuda tartışmayı yönlendirir: Evlerdeki robot süpürgeler hareket ettikleri sürece odaların haritasını oluşturabilmekte, aksi halde engellerle oda/ev sınırlarının inşa edilmesini sağlamaktadır. Sınırlar, dinamik unsurlar ve ödül/ceza fonksiyonları sayesinde problem kısıtları olarak algılanmakta ve robot süpürgenin davranışları neticesinde hem temizlik/süpürme sağlanmakta hem de harita oluşturularak gelecek eylemler için öğrenilmiş planlamalar yapılmaktadır.

- **Ödül/Ceza Fonksiyonları:** Etmenlerin eylemleri sonrasında gelecek kararlarını ve eylemlerini düzenleyecek birtakım çevresel ödüller veya cezalar dönüt olarak verilebilmektedir. Bu ödül ve cezalar çevredeki bazı unsurlarla etkileşimden doğabileceği gibi, etmenler için tanımlanan kurallarla ya da etmenin çevrede hareket halinde olduğu her aşamada uygulanabilmektedir.

Biliyor musunuz?

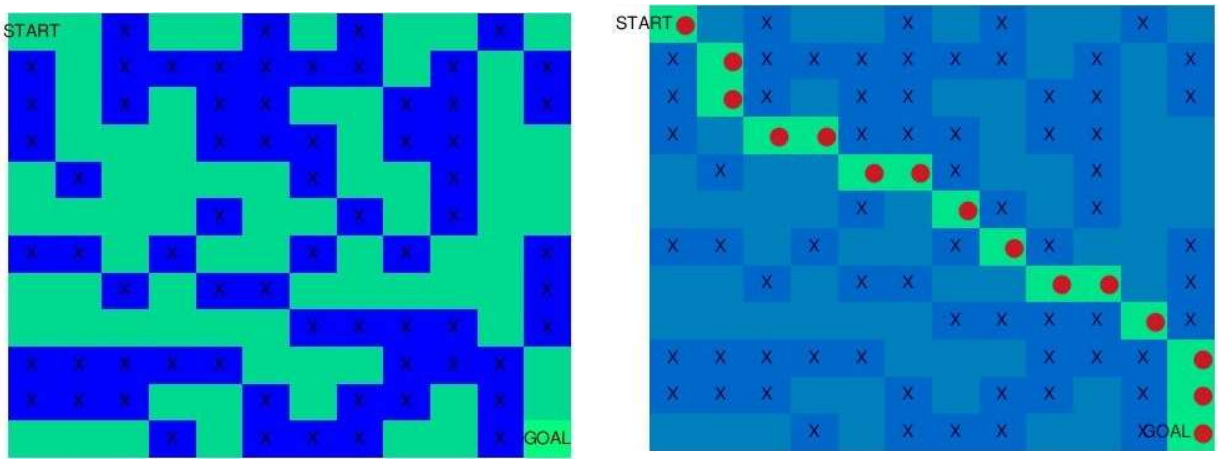
Etmen tabanlı modelleme anlaşılacağı üzere; insanların ve diğer canlıların dünya üzerinde çevreyle etkileşimine oldukça benzemektedir. Esasında bu benzetime dayanarak tasarlanan etmen tabanlı modelleme Yapay Zekâ'nın robotik uygulamalarında da oldukça önemli bir rol sahibidir. Yine Yapay Zekâ'daki bir başka alan olan zeki optimizasyon kapsamı da etmen tabanlı tasarımla ilişkilendirilebilmektedir. Buna göre, doğadaki canlı davranışlarının zeki optimizasyonda kullanılması etmen tabanlı tasarıma benzemektedir.

1.3. Q-Öğrenme ile Öğrenen Etmen Modelleme

Etmen tabanlı çözümlerde yapay zekânın makine öğrenmesi alanında kullanılan öğrenme yöntemi **takviyeli öğrenme (reinforcement learning)** olarak bilinmektedir. Takviyeli öğrenme gerçek hayattaki **yaşayarak öğrenme** gibidir. Bu öğrenmede genel olarak şu hususlar söz konusudur:

- Öğrenmede ödül/ceza mantığı vardır.
- Ödül olarak algılanacak değerler daha yüksekken, ceza olarak algılanan değerler düşük değerler olarak tasarlanır.
- Takviyeli öğrenme yapan unsurun ödül/ceza değerini belirleyen belli eylemler vardır (duvara çarparsan eksi 1 puan, çarpmazsan -yoluna devam edersen- her zaman artı 1 puan gibi).
- Takviyeli öğrenme yapacak unsur içerisinde bulunduğu ortam/problem için rastgele eylemlerde bulunur ve çok sayıda eylemlerle en iyi tecrübenin kazanılması sağlanır.

Etmen tabanlı çözümlerde takviyeli öğrenme için kullanılan temel tekniklerden biri de **Q-Öğrenme (Q-Learning)** olarak bilinmektedir. Q-Öğrenme'de daha önceden tasarlanmış ödül/ceza puanlarının yer aldığı bir tablo kullanılmak suretiyle, başlangıçta içi boş olan yeni bir Q tablosunun en uygun değerlerle doldurulması sağlanır. Şekil 6.2a'da gösterildiği gibi; başlangıç noktası (start) ve bitiş noktası (goal) gösterilen bir problemde, engellerin olduğu her kareye negatif, engelsiz karelere ise pozitif puan verdiği düşünülürse; Q-Öğrenme ile gerçekleştirilen işlem sonucunda, etmenin Şekil 6.2b'de görüldüğü gibi kendisini amaca götüren en uygun yolu bulması sağlanmaktadır. **Bu örnekte soldaki görüntü önceden tasarlanmış ödül/ceza (puan) tablosuyken, sağdaki görüntü başlangıçta içi boş olan ama daha sonra en uygun yoldaki değerlerin diğer karelere göre daha yüksek olduğu Q tablosudur.**



(a) (b)

Şekil 6.2. Q-Öğrenme ile örnek bir problem akışı.

Q-Öğrenme'de etmen her seferinde rastgele adımları denemekte ve bu paragrafı takiben verilen formül sayesinde mevcut ödül/ceza tablosunu kullanmak suretiyle gelecek eylemlerinden gelebilecek maksimum değerleri (tıpkı satrançta gelecek hamleleri düşünmek gibi) dikkate almakta; yine öğrenmesini etkileyen diğer parametreleri ve yeni tablodaki ödül/ceza değerini harmanlayarak adım attığı eylemlerdeki değerlerin güncellenmesini sağlamaktadır. Böylelikle iyi eylemler zamanla daha fazla değer kazanırken, kötü eylemlerin değerlerinin zamanla azalması sağlanmaktadır.

$$Q^{\text{yeni}}(s_t, a_t) \leftarrow (1-\alpha) \cdot \underbrace{Q(s_t, a_t)}_{\text{eski değer}} + \underbrace{\alpha}_{\text{öğrenme oranı}} \cdot \overbrace{\left(\underbrace{r_t}_{\text{ödül}} + \underbrace{\gamma}_{\text{azalma değeri}} \cdot \underbrace{\max_a Q(s_{t+1}, a)}_{\text{gelecekteki tahmini maks. değer}} \right)}^{\text{eylem sonucunda öğrenilen değer}}$$

Eğitmene Not

Eğitmen öğrencilere Q-Öğrenme formülünü şu akışa benzer bir şekilde anlatır:

1. Q-Öğrenme tekniği çalışmadan önce elimizde eylemlerin puanlarını taşıyan bir tablomuz olur. Bununla birlikte içi boş değerlerden (sıfır) oluşan puan tablosuyla aynı satır-sütundan oluşan bir Q tablosu tasarlarız.

2. Q-Öğrenme için öğrenme oranı ve azalma değeri olarak iki reel sayı belirleriz. Bu sayılar öğrenme gücünü ve şans faktörünü etkiler.

3. Etmenimiz probleme göre ilk adımını atar.

4. Formülde eylem sonucunda öğrenilen değer kısmını hesaplamak için etmenin gelecek olası adımlarının bütün kombinasyonlarından (ilk başta boş tasarladığımız ama zamanla dolacak Q tablosundan) gelecek en büyük puanı azalma değeri ile çarpar ve eylem puanlarını taşıyan tablodan etmenin attığı adıma tekabül eden ödül değeriyle topladıktan sonra elde ettiğimiz değeri öğrenme oranı ile çarparız.

5. Dördüncü adım ile elde ettiğimiz değeri Q tablosunda mevcut olan değerin üzerine ekler, ardından (1 – öğrenme oranı) değeriyle çarparız.

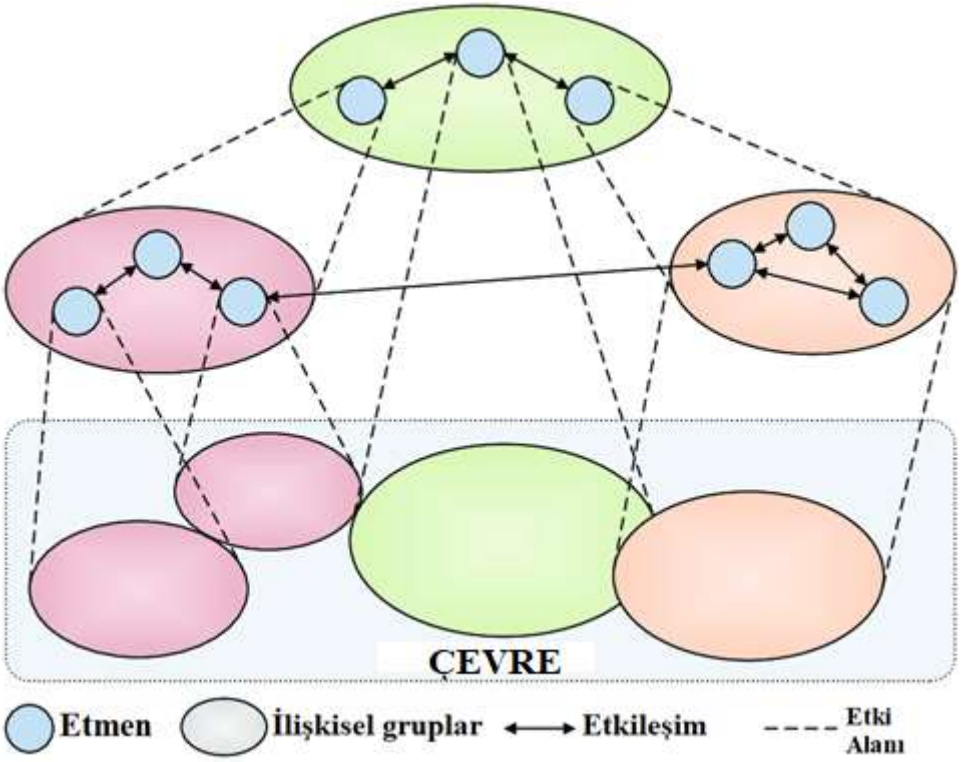
6. Yeni elde ettiğimiz değer, etmenin adım attığı Q tablosu hücresindeki yeni değer olur.

7. Üçüncü adımdan itibaren bütün işlem adımlarını bir durma sayısı/kriterine kadar tekrarlar, böylece Q tablosunu döngüsel/iteratif bir şekilde güncelleriz.

1.4. Çok Etmenli Etkileşimler

Birden fazla etmenin varlığı çok etmenli modellemeleri ortaya çıkarmaktadır. Bu modelleme özellikle günümüz karmaşık problemlerinin çözümünde daha etkin sonuçlar üretebilmektedir. Çok etmenli problem modellemelerinde etmenler arası ilişkiler, her bir etmenin çevreyle etkileşimi ve

bağlı oldukları çevresel düzenlemeler farklılıklar içerebilmekte, böylece karmaşık düzendeki problemlere adaptasyon daha kolay sağlanabilmektedir.



Şekil 6.3. Çok etmenli problem ortamı (Web Kaynağı 6.2)

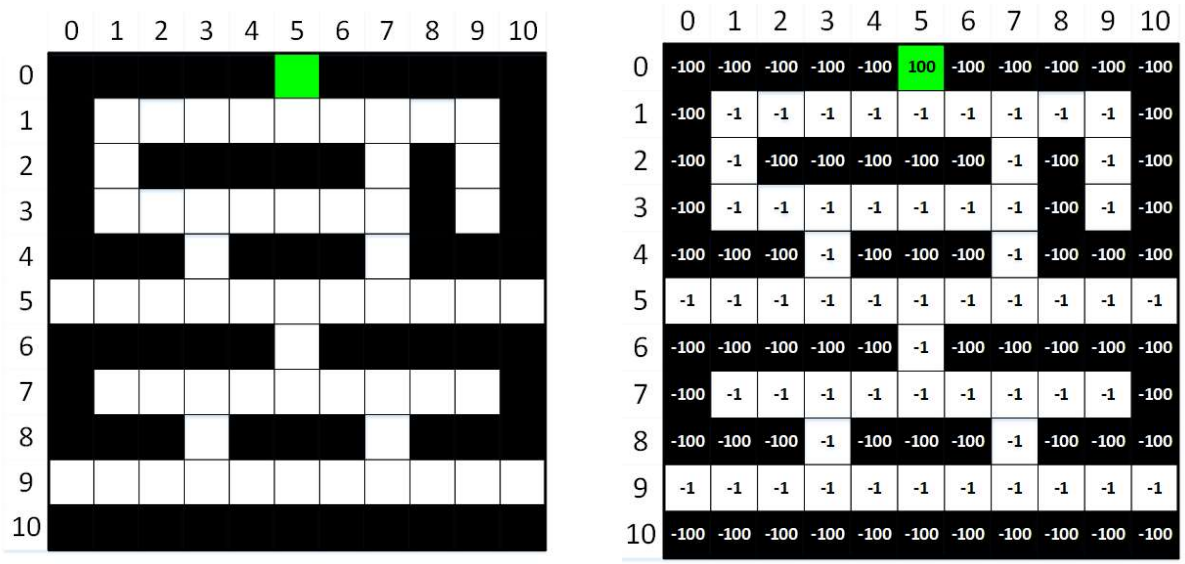
Eğitmene Not

Eğitmen, çok etmenli modellemeye örnek olarak farklı karakterlerin birbiriyle etkileşimde olduğu video oyunlarını örnek verir. Bu noktada, Braw stars ya da platform tabanlı video oyunlarda (örneğin, Mario), karakterlerin nitelikleri, eylemleri, çevrenin düzenlenmesi ve hatta çok etmenli mantıkta farklı karakterler arası ilişkileri etmen tabanlı modelleme odağında öğrenciler ile tartışır.

2. TASARLA

Eğitmene Not

Eğitmen, bir kargo robotunun en uygun yol üzerinden kargo indirmesine yardımcı olmak için bir problem modellemesi yapacaklarını anlatır. Buna göre, Şekil 6.4'te sol tarafta verilen ortam içerisinde, beyaz karelerin izlenmesi suretiyle yeşil kareye kargo indirilmesi gerekmektedir. Buna göre robot beyaz karelerle gösterilen herhangi bir yere konumlandığında yeşil kareye olan uygun yolun rotasını çıkarabilmeli, siyah karelerin temsil ettiği kargo depolarına takılmamalıdır. Bunu sağlamak için ortamın ödül/ceza değerli modellemesi aynı şekilde sağ tarafta verilmiştir.



Şekil 6.4. Kargo robotu etmen ve Q-Öğrenme modellemesi.

3. HAREKETE GEÇ

Öğrenciler Şekil 6.5'teki gibi ortamı satır-sütun düzeninde modelleyip yeşil, beyaz ve siyah kareler için değerleri oluşturan kodları yazar (Kodun tamamı Github platformunda Hafta6 klasörü altında H6_etmen_kargo-problemi.py dosyası kapsamında paylaşılmış durumdadır.):

```
import numpy as np
ortam_satir_sayisi = 11
ortam_sutun_sayisi = 11
q_degerleri = np.zeros((ortam_satir_sayisi, ortam_sutun_sayisi, 4))
hareketler = ['yukari', 'sag', 'asagi', 'sol']
oduller = np.full((ortam_satir_sayisi, ortam_sutun_sayisi), -100.)
oduller[0, 5] = 100.
gecitler = {}
gecitler[1] = [i for i in range(1, 10)]
gecitler[2] = [1, 7, 9]
gecitler[3] = [i for i in range(1, 8)]
gecitler[3].append(9)
gecitler[4] = [3, 7]
gecitler[5] = [i for i in range(11)]
gecitler[6] = [5]
gecitler[7] = [i for i in range(1, 10)]
gecitler[8] = [3, 7]
gecitler[9] = [i for i in range(11)]

for satir_indeks in range(1, 10):
  for sutun_indeks in gecitler[satir_indeks]:
    oduller[satir_indeks, sutun_indeks] = -1.

for satir in oduller:
  print(satir)
```

Şekil 6.5. İlgili problem için model ortamının kodlanması ve ekranda gösterilmesi

24. ve 25. satırlardaki kodlar sayesinde modellenen ortam ekranda gösterilmiş olur:

```
[-100. -100. -100. -100. -100.  100. -100. -100. -100. -100. -100.]
[-100.   -1.   -1.   -1.   -1.   -1.   -1.   -1.   -1.   -1. -100.]
[-100.   -1. -100. -100. -100. -100. -100.   -1. -100.   -1. -100.]
[-100.   -1.   -1.   -1.   -1.   -1.   -1.   -1. -100.   -1. -100.]
[-100. -100. -100.   -1. -100. -100. -100.   -1. -100. -100. -100.]
[-1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1.]
[-100. -100. -100. -100. -100.   -1. -100. -100. -100. -100. -100.]
[-100.   -1.   -1.   -1.   -1.   -1.   -1.   -1.   -1.   -1. -100.]
[-100. -100. -100.   -1. -100. -100. -100.   -1. -100. -100. -100.]
[-1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1.]
[-100. -100. -100. -100. -100. -100. -100. -100. -100. -100. -100.]
```

Şekil 6.6. Modellenen ortam bilgisi

Ardından robotun her hareketinde beyaz karede kalıp kalmadığını belirleyen, her yeni hareket döngülerinde başlangıç noktası atamasını yapan ve her yeni karede bir sonraki hareket noktasını yeni Q tablosu ile belirleyen fonksiyonlar tanımlanır:

```
27 def engel_mi(gecerli_satir_indeks, gecerli_sutun_indeks):
28   if oduller[gecerli_satir_indeks, gecerli_sutun_indeks] == -1.:
29     return False
30   else:
31     return True
32
33 def baslangic_belirle():
34   gecerli_satir_indeks = np.random.randint(ortam_satir_sayisi)
35   gecerli_sutun_indeks = np.random.randint(ortam_sutun_sayisi)
36   while engel_mi(gecerli_satir_indeks, gecerli_sutun_indeks):
37     gecerli_satir_indeks = np.random.randint(ortam_satir_sayisi)
38     gecerli_sutun_indeks = np.random.randint(ortam_sutun_sayisi)
39   return gecerli_satir_indeks, gecerli_sutun_indeks
40
41 def sonraki_hareket_belirle(gecerli_satir_indeks, gecerli_sutun_indeks, epsilon):
42   if np.random.random() < epsilon:
43     return np.argmax(q_degerleri[gecerli_satir_indeks, gecerli_sutun_indeks])
44   else:
45     return np.random.randint(4)
```

Şekil 6.7. Robot hareket mekanizmalarının kodlanması

Bu noktaya kadar yazılan kodlar kapsamında;

27. satır itibariyle yazılan **engel_mi** fonksiyonu robotun hareket etmesi sonrası gelinen hücrenin engel (yani kargo indirilen siyah ya da hedef yeşil kare) olup olmadığını tespit etmektedir.

33. satır itibariyle yazılan **baslangic_belirle** fonksiyonu robotun her yeni öğrenme döngüsünde rastgele beyaz karelerden birinden başlamasını sağlanmaktadır. Bunun için 36. satırda daha önce 27. satır itibariyle yazılan **engel_mi** fonksiyonundan da faydalanılmaktadır.

41. satır itibariyle yazılan **sonraki_hareket_belirle** fonksiyonu robotun mevcut konumundan sonra hareket edebileceği sonraki konumu eğitim sonunda elde edilen Q tablosundan belirlemektedir. Bu noktada buradaki kod yapısına özel olarak belirlenen **epsilon** değeri de şans faktörü olarak dikkate alınmaktadır. Buna göre, epsilon değerinden küçük olan rastgele bir değer üretilirse puan

tablosundan maksimum değerin, eşit ya da büyük rastgele bir değer üretilirse de bir başka rastgele değerin (puan tablosu dikkate alınmadan) döndürülmesi sağlanmaktadır.

Robotun mevcut konumundan bir sonraki noktaya gitmesi için ayrı bir fonksiyon tanımlanır:

```
47 def sonraki_noktaya_git(gecerli_satir_indeks, gecerli_sutun_indeks, hareket_indeks):
48   yeni_satir_indeks = gecerli_satir_indeks
49   yeni_sutun_indeks = gecerli_sutun_indeks
50   if hareketler[hareket_indeks] == 'yukari' and gecerli_satir_indeks > 0:
51     yeni_satir_indeks -= 1
52   elif hareketler[hareket_indeks] == 'sag' and gecerli_sutun_indeks < ortam_sutun_sayisi - 1:
53     yeni_sutun_indeks += 1
54   elif hareketler[hareket_indeks] == 'asagi' and gecerli_satir_indeks < ortam_satir_sayisi - 1:
55     yeni_satir_indeks += 1
56   elif hareketler[hareket_indeks] == 'sol' and gecerli_sutun_indeks > 0:
57     yeni_sutun_indeks -= 1
58   return yeni_satir_indeks, yeni_sutun_indeks
```

Şekil 6.8. Robot gelecek hareket mekanizması kodlanması

47. satır itibariyle yazılan **sonraki_noktaya_git** fonksiyonu sayesinde, robotun bir sonraki hareketi satır sütun ekseninde belirlenmekte; yeni hareket noktasına göre robotun yukarı hareketinde satır değeri azaltılıp, aşağı hareketinde artırılmakta, yine sağa harekette sütun değeri artırılıp, sola harekette azaltılmaktadır.

4. YÜRÜT

Bu aşamaya kadar yazılan kodları dikkate alarak, Robotun Q-Öğrenme ile eğitimi sonrasında belirlenmiş olan en uygun yolu verecek fonksiyon yazılır:

```
60 def en_kisa_mesafe(basla_satir_indeks, basla_sutun_indeks):
61   if engel_mi(basla_satir_indeks, basla_sutun_indeks):
62     return []
63   else:
64     gecerli_satir_indeks, gecerli_sutun_indeks = basla_satir_indeks, basla_sutun_indeks
65     en_kisa = []
66     en_kisa.append([gecerli_satir_indeks, gecerli_sutun_indeks])
67     while not engel_mi(gecerli_satir_indeks, gecerli_sutun_indeks):
68       hareket_indeks = sonraki_hareket_belirle(gecerli_satir_indeks, gecerli_sutun_indeks, 1.)
69       gecerli_satir_indeks, gecerli_sutun_indeks = sonraki_noktaya_git(gecerli_satir_indeks,
70                                                                        gecerli_sutun_indeks, hareket_indeks)
71       en_kisa.append([gecerli_satir_indeks, gecerli_sutun_indeks])
72     return en_kisa
```

Şekil 6.9. Uygun yol tespiti fonksiyonu

Son olarak, robotun takviyeli öğrenmesini sağlayacak; Q-Öğrenme ana fonksiyonu ve öğrenme parametreleri kodlanır:

```
74 epsilon = 0.9
75 azalma_degeri = 0.9
76 ogrenme_orani = 0.9
77
78 for adim in range(1000):
79   satir_indeks, sutun_indeks = baslangic_belirle()
80   while not engel_mi(satir_indeks, sutun_indeks):
81     hareket_indeks = sonraki_hareket_belirle(satir_indeks, sutun_indeks, epsilon)
82     eski_satir_indeks, eski_sutun_indeks = satir_indeks, sutun_indeks
83     satir_indeks, sutun_indeks = sonraki_noktaya_git(satir_indeks, sutun_indeks, hareket_indeks)
84     odul = oduller[satir_indeks, sutun_indeks]
85     eski_q_degeri = q_degerleri[eski_satir_indeks, eski_sutun_indeks, hareket_indeks]
86     fark = odul + (azalma_degeri * np.max(q_degerleri[satir_indeks, sutun_indeks])) - eski_q_degeri
87     yeni_q_degeri = eski_q_degeri + (ogrenme_orani * fark)
88     q_degerleri[eski_satir_indeks, eski_sutun_indeks, hareket_indeks] = yeni_q_degeri
89 print('Eğitim tamamlandı.')
```

Şekil 6.10. Q-Öğrenme süreci kodları

Son kod bölümünde 78. satır itibariyle Q-Öğrenme döngüsü tanımlanmakta, buna göre **range** fonksiyonu içerisinde verilen değer maksimum öğrenme adım sayısı olmaktadır.

80. satır itibariyle öğrenme esnasında **engel_mi** fonksiyonundan faydalanılmaktadır. Yine her yeni harekette **sonraki_hareket_belirle** fonksiyonu çağırılarak robotun hareket ettirilmesi sağlanmaktadır.

84. satır ile 88. satırlar arasındaki kodlarla Q-Öğrenme formülü üzerinden hesaplama yapılarak, başlangıçta boş tasarlanan **q_degerleri** adıyla temsil edilen Q tablosunda güncellemeler yapılmaktadır.

Eğitim sona erdiğinde 89. satırdaki kod ile ekrana 'Eğitim tamamlandı.' ibaresi yansıtılmaktadır.

5. KARAR VER

5.1. Sonuçların Gözlemlenmesi

Şekil 6.11'de gösterildiği gibi, eğitilen robot Q tablosunda kendisini yeşil kareye götürecek uygun rotaları belirlemiş olduğu için kullanıcıdan herhangi bir başlangıç noktası (satır ve sütun indeks değerine göre) istenerek kargonun indirilmesi için izlenecek yolun/rotanın ekrana yazdırılması sağlanır:

```
91 ogr_sonrasi_satir = input('Robotun harekete başlayacağı satır indeksini giriniz:')
92 ogr_sonrasi_sutun = input('Robotun harekete başlayacağı satır indeksini giriniz:')
93
94 print('Kargo noktasına giden rota:',
95       en_kisa_mesafe(int(ogr_sonrasi_satir), int(ogr_sonrasi_sutun)))
```

Şekil 6.11. Başlangıç noktasına göre uygun yol/rota tespiti

91. satırda kullanıcıdan başlangıç satır indeksi istenirken, 92. satırda ise başlangıç sütun indeksi istenmektedir.

94. satırda, kullanıcıdan alınan bilgiler ışığında, **en_kisa_mesafe** fonksiyonu kullanılmak suretiyle robotun yeşil karede kargo indirmek için izleyeceği rota ekrana yazdırılmaktadır.

Örneğin satır indeksi 5, sütun indeksi 0 olması durumunda izlenecek rota şu şekilde görüntülenebilmektedir:

```
Robotun harekete başlayacağı satır indeksini giriniz:5

Robotun harekete başlayacağı satır indeksini giriniz:0
Kargo noktasına giden rota: [[5, 0], [5, 1], [5, 2], [5, 3], [5, 4], [5, 5],
[5, 6], [5, 7], [4, 7], [3, 7], [2, 7], [1, 7], [1, 6], [1, 5], [0, 5]]
```

Şekil 6.12. Örnek yol/rota tespiti

PYTHON KODLARI:

```
import numpy as np
ortam_satir_sayisi = 11
ortam_sutun_sayisi = 11
q_degerleri = np.zeros((ortam_satir_sayisi, ortam_sutun_sayisi, 4))
hareketler = ['yukari', 'sag', 'asagi', 'sol']
oduller = np.full((ortam_satir_sayisi, ortam_sutun_sayisi), -100.)
oduller[0, 5] = 100.
gecitler = {}
gecitler[1] = [i for i in range(1, 10)]
gecitler[2] = [1, 7, 9]
gecitler[3] = [i for i in range(1, 8)]
gecitler[3].append(9)
gecitler[4] = [3, 7]
gecitler[5] = [i for i in range(11)]
gecitler[6] = [5]
gecitler[7] = [i for i in range(1, 10)]
gecitler[8] = [3, 7]
gecitler[9] = [i for i in range(11)]

for satir_indeks in range(1, 10):
  for sutun_indeks in gecitler[satir_indeks]:
    oduller[satir_indeks, sutun_indeks] = -1.
```

```
for satir in oduller:
  print(satir)

def engel_mi(gecerli_satir_indeks, gecerli_sutun_indeks):
  if oduller[gecerli_satir_indeks, gecerli_sutun_indeks] == -1.:
    return False
  else:
    return True

def baslangic_belirle():
  gecerli_satir_indeks = np.random.randint(ortam_satir_sayisi)
  gecerli_sutun_indeks = np.random.randint(ortam_sutun_sayisi)
  while engel_mi(gecerli_satir_indeks, gecerli_sutun_indeks):
    gecerli_satir_indeks = np.random.randint(ortam_satir_sayisi)
    gecerli_sutun_indeks = np.random.randint(ortam_sutun_sayisi)
  return gecerli_satir_indeks, gecerli_sutun_indeks

def sonraki_hareket_belirle(gecerli_satir_indeks, gecerli_sutun_indeks, epsilon):
  if np.random.random() < epsilon:
    return np.argmax(q_degerleri[gecerli_satir_indeks, gecerli_sutun_indeks])
  else:
    return np.random.randint(4)

def sonraki_noktaya_git(gecerli_satir_indeks, gecerli_sutun_indeks,
hareket_indeks):
  yeni_satir_indeks = gecerli_satir_indeks
  yeni_sutun_indeks = gecerli_sutun_indeks
  if hareketler[hareket_indeks] == 'yukari' and gecerli_satir_indeks > 0:
    yeni_satir_indeks -= 1
```

```
  elif hareketler[hareket_indeks] == 'sag' and gecerli_sutun_indeks <
ortam_sutun_sayisi - 1:
    yeni_sutun_indeks += 1
  elif hareketler[hareket_indeks] == 'asagi' and gecerli_satir_indeks <
ortam_satir_sayisi - 1:
    yeni_satir_indeks += 1
  elif hareketler[hareket_indeks] == 'sol' and gecerli_sutun_indeks > 0:
    yeni_sutun_indeks -= 1
  return yeni_satir_indeks, yeni_sutun_indeks

def en_kisa_mesafe(basla_satir_indeks, basla_sutun_indeks):
  if engel_mi(basla_satir_indeks, basla_sutun_indeks):
    return []
  else:
    gecerli_satir_indeks, gecerli_sutun_indeks = basla_satir_indeks,
basla_sutun_indeks
    en_kisa = []
    en_kisa.append([gecerli_satir_indeks, gecerli_sutun_indeks])
    while not engel_mi(gecerli_satir_indeks, gecerli_sutun_indeks):
      hareket_indeks = sonraki_hareket_belirle(gecerli_satir_indeks,
gecerli_sutun_indeks, 1.)
      gecerli_satir_indeks, gecerli_sutun_indeks =
sonraki_noktaya_git(gecerli_satir_indeks,
                                                gecerli_sutun_indeks, hareket_indeks)
      en_kisa.append([gecerli_satir_indeks, gecerli_sutun_indeks])
    return en_kisa

epsilon = 0.9
azalma_degeri = 0.9
ogrenme_orani = 0.9
```

```
for adim in range(1000):
  satir_indeks, sutun_indeks = baslangic_belirle()
  while not engel_mi(satir_indeks, sutun_indeks):
    hareket_indeks = sonraki_hareket_belirle(satir_indeks, sutun_indeks, epsilon)
    eski_satir_indeks, eski_sutun_indeks = satir_indeks, sutun_indeks
    satir_indeks, sutun_indeks = sonraki_noktaya_git(satir_indeks, sutun_indeks,
hareket_indeks)
    odul = oduller[satir_indeks, sutun_indeks]
    eski_q_degeri = q_degerleri[eski_satir_indeks, eski_sutun_indeks,
hareket_indeks]
    fark = odul + (azalma_degeri * np.max(q_degerleri[satir_indeks, sutun_indeks]))
- eski_q_degeri
    yeni_q_degeri = eski_q_degeri + (ogrenme_orani * fark)
    q_degerleri[eski_satir_indeks, eski_sutun_indeks, hareket_indeks] =
yeni_q_degeri
print('Eğitim tamamlandı.')

ogr_sonrasi_satir = input('Robotun harekete başlayacağı satır indeksini giriniz:')
ogr_sonrasi_sutun = input('Robotun harekete başlayacağı satır indeksini giriniz:')

print('Kargo noktasına giden rota:',
     en_kisa_mesafe(int(ogr_sonrasi_satir), int(ogr_sonrasi_sutun)))
```

Düşün, tartış...

Sürücüsüz otomobilleri etmen tabanlı modelleme ile kazaları ve trafik problemlerini önlemek için yeniden tasarlayabilir miyiz? Soruya yönelik öğrenci görüşleri sınıf ortamında paylaşılıp tartışılabilir. Ayrıca Web ortamındaki alternatif düşünceler de araştırılarak yorumlanabilir.

Dünyadan Haberler

İnsansı robotlar bilimi sevdirecek

Cumhurbaşkanlığı Millet Kütüphanesi Teknoloji Atölyesi'nde 13-15 yaş grubundaki çocukları bilim ve teknoloji çalışmalarına yönlendirmek amacıyla 'insansı robot' eğitimi verilmeye başlandı.

Atölyelerde altı kişilik yapılan etkinlikte, insansı gövde ve eklem yapısına sahip robotların çalışma prensipleri ile işlevleri ele alınıyor. Cumhurbaşkanlığı Millet Kütüphanesi Eğitim Merkezi ve Atölyeler Koordinatörü Ayhan Bozkurt, TÜBİTAK tarafından kurulan bilim ve teknoloji atölyelerinde çocuklara, robotların ne olduğunu anlatacaklarını belirtti. Bozkurt, şöyle devam etti:

"*İki tane insansı robot üzerinden gözlem yapacaklar. Bizim sorularımız olacak. Onlara, 'biz robot olsaydık bizi nasıl programlarlardı, hangi direktifleri vereceklerdi' gibi sorularla insansı robotlara yönelik farkındalık oluşturmak istiyoruz. Amacımız, çocukların bilim ve teknolojiye olan meraklarını artırmak, onların o alanda çalışmalar yapmalarını sağlamak ve tetiklemektir. Onların gerçek anlamda bir robot görerek bu tarz çalışmalarla bilime ve teknolojiye yönelmelerini sağlamaktır.*" Daha önceki eğitimlerde, çocukların robotlara yönelik ilgilerinin fazla olduğunu gördüklerini aktaran Bozkurt, etkinliklerin devam edeceğini bildirdi (Web Kaynağı 6.3).

6. İLAVE ETKİNLİK

Eğitmene Not

Eğitmen, öğrencilere ceza/ödül fonksiyonuna sahip olmayan; basit yapıda çok etmenli bir model tasarımı geliştireceklerini belirtir.

Bu örnekte etmenlerin aralarında sayısal değer (varlık) paylaşımı yaptığı bir akış kurgulanmıştır.

Öğrenciler, ilk olarak **pip install mesa** komutu ile etmen tabanlı modelleme için kullanılan, mesa adlı bir başka Python kütüphanesini kurarlar (Web Kaynağı 6.4).

Ardından bu paragraf akabindeki komutları kodlayarak standart bir etmen modelini tasarlarlar (Söz konusu uygulama, Github platformu Hafta6 klasörü altındaki iki dosya altında: H6_etmen_2_model-kodu.py ve H6_etmen_2_model-ornegi.py ile paylaşılmış durumdadır):

Şekil 6.13'te gösterildiği gibi öğrenciler gerekli kodları yazar.

1 nolu kod satırında yapay zekâ kütüphanesi olan **mesa** kütüphanesinin **Agent ve Model** sınıflarını çağırır.

2 nolu kod satırında **"Etmen"** isminde bir sınıf oluşturur.

4-5-6 nolu kod satırında **"Etmen"** sınıfın içerisine **def__init__** isminde bir fonksiyon ile her bir etmen için id tanımlanır.

7 nolu kod satırında **"Cevre"** isminde bir sınıf oluşturur.

9-10 nolu kod satırında **"Cevre"** sınıfın içerisine **def__init__** isminde bir fonksiyon ile N adet etmen sayısı tanımlanır.

12-13 nolu kod satırlarında oluşturulan bir for döngüsü ile Etmen sınıfı çağırılarak etmen tanımlamaları yapılmak suretiyle a değişkenine aktarılır.

```
1 from mesa import Agent, Model
2 class Etmen(Agent):
3     """Başlangıç varlık değerine sahip bir etmen."""
4     def __init__(self, unique_id, model):
5         super().__init__(unique_id, model)
6         self.varlik = 1
7 class Cevre(Model):
8     """Etmenlerin yer alacağı çevre."""
9     def __init__(self, N):
10        self.num_agents = N
11        # etmenlerin oluşturulması
12        for i in range(self.num_agents):
13            a = Etmen(i, self)
```

Şekil 6.13. Başlangıç varlığı ve etmenlerin yer alacağı çevre kod ekranı

Eğitmene Not

Eğitmen öğrencilere, önceki aşamada yazılan etmen ve çevre kodlarını takiben, bu not akabinde gösterilen kod satırlarını entegre ettirerek, her kod satırında kod satırının ne anlama geldiği ile ilgili sorular sorar ve tartışır.

```python
from mesa import Agent, Model
from mesa.time import RandomActivation
class Etmen(Agent):
    """Başlangıç varlık değerine sahip bir etmen."""
    def __init__(self, unique_id, model):
        super().__init__(unique_id, model)
        self.varlik = 1
    def step(self):
        # Etmenin her adımda gerçekleştireceği eylem
        print ("Merhaba, ben etmen " + str(self.unique_id) +".")

class Cevre(Model):
    """Etmenlerin yer alacağı çevre."""
    def __init__(self, N):
        self.num_agents = N
        # etmenlerin oluşturulması
        for i in range(self.num_agents):
            a = Etmen(i, self)
            self.schedule.add(a)
    def step(self):
        '''Etmene bir adım attır.'''
        self.schedule.step()
```

Şekil 6.14. Etmen ve çevre modeli kodlanması

Yeni eklenen kodlarda **RandomActivation** kütüphane bileşeni ile etmenlerin çevre içerisinde bir adımlık eylem gerçekleştirmeleri sağlanabilmektedir. Eylemlerde her adım **step** fonksiyonu ile karşılanmakta, her adımda herhangi bir etmen kendi sıra numarasını söylemektedir.

Eğitmene Not

Eğitmen, sonraki aşamada öğrencilere yeni bir Python kod dosyası açtırarak, önceki aşamada kodlanan etmen modelinin kaydedilip, yeni kod ortamına **import** edilmesini sağlar.

```python
from etmen_modeli import Etmen
yeni_model = Etmen(10)
yeni_model.step()
```

Şekil 6.15. Adımlar eşliğinde etmen modellenmesi

Bu kod örneği ile birlikte rastgele 10 etmenin otomatik olarak oluşturulması sağlanır.

Çalıştırılan kod örneği ile kendilerini ifade eden rastgele 10 etmenin oluşturulduğu gözlenir (Şekil 6.16).

```
Merhaba, ben etmen 2.
Merhaba, ben etmen 9.
Merhaba, ben etmen 5.
Merhaba, ben etmen 3.
Merhaba, ben etmen 7.
Merhaba, ben etmen 0.
Merhaba, ben etmen 4.
Merhaba, ben etmen 6.
Merhaba, ben etmen 8.
Merhaba, ben etmen 1.

In [9]:
```

Şekil 6.16. Çalıştırılan kod örneği sonucunda çalışan farklı etmenlerin çalışması

Sonraki aşamada, kendi değerini (varlık) kontrol eden ve eğer varsa rastgele başka bir etmene bağış yapan bir fonksiyon yapısı, önceden yazılan ve etmen model dosyası içerisinde yer alan **step** fonksiyonu düzenlenerek eklenir:

```
20    def step(self):
21        if self.varlik == 0:
22            return
23        baska_etmen = self.random.choice(self.model.schedule.agents)
24        baska_etmen. varlik += 1
25        self. varlik -= 1
```

Şekil 6.17. Etmenler arası etkileşimin kodlanması

Söz konusu ekleme sonrası modelin import edildiği yeni kod dosyası içerisinde 10 adet etmen oluşturma koduna ek olarak etmenlerin 10 adımlık eylemde bulunmasını sağlayan kod yapısı eklenir:

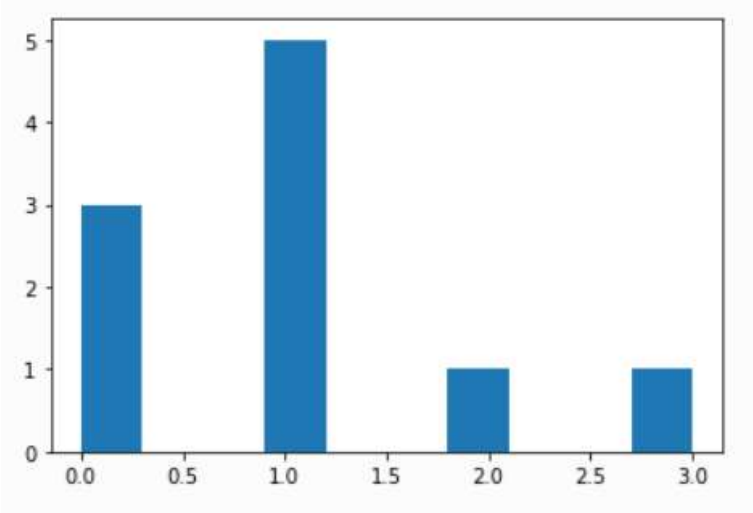
```
1 from etmen_modeli import Etmen
2 yeni_model = Etmen(10)
3 for i in range(10):
4     yeni_model.step()
```

Şekil 6.18. Bütün etmenler için döngüsel akışın kodlanması

Öğrenciler, yeni **step** fonksiyonu neticesinde çalıştırılan kod yapısı ile her bir etmenin 10 adım sonunda sahip oldukları değerlerin grafiksel gösterimini görüntüler. Bunun için **matplotlib.pyplot** kütüphanesi ile birlikte etmenlerin son değerlerinin bir araya toplandığı değişken üzerinden görüntüleme yapılır.

```
 9 etmen_varlik = [a.varlik for a in yeni_model.schedule.agents]
10 plt.hist(etmen_varlik)
```

Şekil 6.19. Etmen hareketlerinin görselleştirilmesi



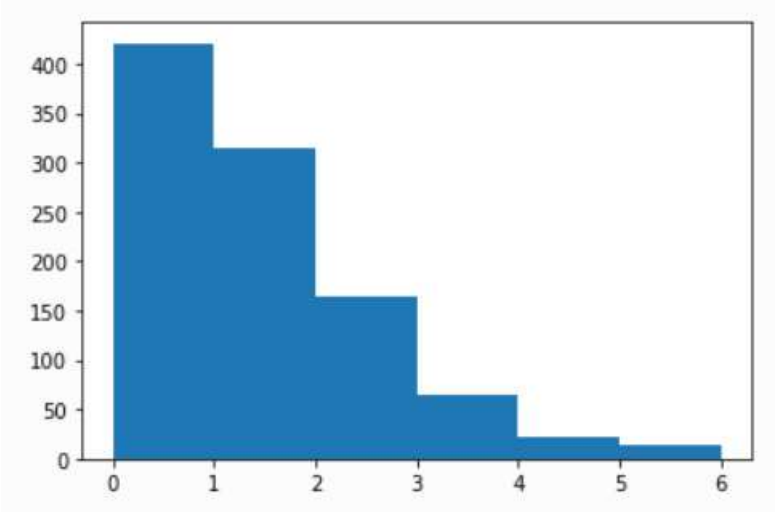
Şekil 6.20. 10 adımlık etkileşim sonrasında değerlerin dağılımı

Söz konusu bulguların daha geniş ölçüye taşınması için her 10 adımlık sürecin toplamda 100 farklı senaryoda meydana geldiği bir yapı tasarlanır. Bunun için Şekil 6.21'de olduğu gibi, **butun_varliklar** adı altında bir değişken oluşturulup, etmen oluşturma kodlarının adımlık bir döngüye alınması ve ekrana çizdirilen grafiğin **butun_varliklar** değişkeni üzerinden yeniden çizdirilmesi sağlanır:

```
from etmen_modeli import Etmen

butun_varliklar = []

for j in range(100):
    yeni_model = Etmen(10)
    for i in range(10):
        yeni_model.step()

    for etmen in yeni_model.schedule.agents:
        butun_varliklar.append(etmen.wealth)

import matplotlib.pyplot as plt

plt.hist(butun_varliklar, bins=range(max(butun_varliklar)+1))
```

Şekil 6.21. Genel değer (varlık) dağılımının görselleştirilmesi

Kodun çalıştırılması neticesinde önceki grafikten daha genel bir histogram dağılımı görüntülenmiş olur:



Şekil 6.22. 10 adımlık etkileşimin 100 döngü (etkileşim senaryosu) sonrasına tekabül eden değerler dağılımı

PYTHON KODLARI:

```
#model kod yapısı
from mesa import Agent, Model
from mesa.time import RandomActivation
class Etmen(Agent):
    """Başlangıç varlık değerine sahip bir etmen."""
    def __init__(self, unique_id, model):
        super().__init__(unique_id, model)
        self.varlik = 1
    def step(self):
        # Etmenin her adımda gerçekleştireceği eylem
        print ("Merhaba, ben etmen " + str(self.unique_id) +".")
class Cevre(Model):
    """Etmenlerin yer alacağı çevre."""
    def __init__(self, N):
        self.num_agents = N
```

```
        # etmenlerin oluşturulması
        for i in range(self.num_agents):
            a = Etmen(i, self)
            self.schedule.add(a)
    def step(self):
        if self.varlik == 0:
            return
        baska_etmen = self.random.choice(self.model.schedule.agents)
        baska_etmen. varlik += 1
        self. varlik -= 1
```

```
#örnek problem çözümü (10 etmen tanımlı)
from etmen_modeli import Etmen
yeni_model = Etmen(10)
for i in range(10):
    yeni_model.step()
import matplotlib.pyplot as plt
etmen_varlik = [a.varlik for a in yeni_model.schedule.agents]
plt.hist(agent_varlik)
```

```
#örnek problem çözümü (10 etmen ve 100 döngü tanımlı)
from etmen_modeli import Etmen
butun_varliklar = []
for j in range(100):
    yeni_model = Etmen(10)
    for i in range(10):
        yeni_model.step()

    for etmen in yeni_model.schedule.agents:
        butun_varliklar.append(etmen.wealth)
import matplotlib.pyplot as plt
plt.hist(butun_varliklar, bins=range(max(butun_varliklar)+1))
```

Eğitmene Not

Eğitmen, bu haftaki eğitim hakkında öğrencilerin ilgisini arttırmak ve yeni öğrenilen bilgileri önceden öğrenilen bilgiler ile karşılaştırmalarını sağlamak için aşağıda maddeler halinde verilen soruları tartışır.

- İlgili haftada ele alınan yapay zekâ uygulamalarına uygun örnek veri setleri neler olabilir?
- İlgili haftada ele alınan veri setlerinin yapay zekâ teknik ve kütüphanelerinde ne derecede başarım etkisi olmuştur?
- İlgili haftada öğretilen yapay zekâ tekniği ile uyumlu ve uyumsuz olabilecek açık erişimli veri seti çeşitleri sunan (kaagle, github vs.) internet sitelerinden elde edilecek farklı veri çeşitleri ile yapay zekâ tekniği uyumluluğunu sorar.

Eğitmene Not

Eğitmen, öğrencilerin 5. ve 6. Hafta içerisinde elde ettikleri bilgi-becerileri değerlendirmek, aktif katılımlarını sağlamak ve onları gelecek haftalara motive etmek için 'Kahoot' uygulamasını ya da benzeri bir Web 2.0 uygulamasını kullanarak 'bilgi yarışması' düzenleyebilir.

5. ve 6. Hafta bağlamında sorulabilecek sorular; Yapay Sinir Ağları temelleri/uygulamaları, Etmen Tabanlı Modelleme yaklaşımı ve ve Etmen Tabanlı Modelleme ile uygulamalar hakkında olabilir.

Kaynakça

Web Kaynağı 6.1: http://www.incompleteideas.net/book/ebook/node28.html

Web Kaynağı 6.2: https://www.researchgate.net/publication/324808821_Self-Reconfigurable_Manufacturing_Control_based_on_Ontology-Driven_Automation_Agents

Web Kaynağı 6.3: https://www.hurriyet.com.tr/yerel-haberler/ankara/insansi-robotlar-bilimi-sevdirecek-41850836

Web Kaynağı 6.4: https://mesa.readthedocs.io/en/master/tutorials/intro_tutorial.html

7. Hafta: Zeki Optimizasyon

Ön Bilgi:

- Optimizasyon kavramının temelleri öğrenebilecektir.
- Python kodlamada döngüler, fonksiyonlar ve kütüphane yapılarının kullanımı (Bu konuda Python anlatım videoları: 10-11. Döngüler, 13. Fonksiyonlar, 14. Python Kütüphane Kullanımı yardımcı araç olarak kullanılabilecektir).

Haftanın Kazanımları:

- Öğrenciler optimizasyon kavramını kavrar.
- Öğrenciler optimizasyon tabanlı problem modelleri tasarlayarak uygular.
- Öğrenciler optimizasyona dayanan çözüm süreçlerini yorumlamayı kavrar.
- Öğrenciler yapay zekâ alanı kapsamındaki zeki optimizasyon yöntemini anlar.
- Öğrenciler Python programlama dilini kullanarak genetik algoritma program kodunu yazabilir.
- Öğrenciler Python programlama dilini kullanarak karınca koloni optimizasyonu program kodunu yazabilir.
- Öğrenciler, örnek bir probleme göre zeki optimizasyon çözümü gerçekleştirme yeteneği kazanır.
- Öğrenciler zeki optimizasyon algoritmalarının davranışlarını değerlendirerek yorumlar.
- Öğrenciler zeki optimizasyon süreçlerinin Yapay Zekâ alanı açısından önemini kavrar.

Haftanın Amacı:

Bu haftanın amacı, “zeki optimizasyon” kavramının tüm öğrenciler tarafından doğru bir şekilde kavranmasını sağlamaktır. Haftanın bir diğer amacı da zeki optimizasyona temel teşkil eden doğa esinli kolektif davranışlarla zeki optimizasyon tasarlama mantığını yorumlama noktasında istendik bilgi birikimini aktarmaktır. Bu kapsam içerisinde öğrencilerin Python programlama dilini kullanarak Genetik Algoritma ve Karınca Koloni Optimizasyonu olarak bilinen önemli zeki optimizasyon algoritmalarının kullanımı konusunda yeterlikler elde etmesi de sağlanacaktır.

Kullanılacak Malzemeler:

Bilgisayar, kâğıt, kalem, Python kod editörü, örnekler için çıktı görselleri

Haftanın İşlenişi:

Algıla: Öğrenciler optimizasyon ve zeki optimizasyon kapsamındaki temel kavramları tanımlayıp yorumlayabilir.

Tasarla: Öğrenciler farklı zeki optimizasyon algoritmalarıyla örnek problemler için çözüm süreçleri tasarlayabilir. Bu hafta için Github platformu (https://github.com/deneyapyz/lise/) ile etkileşim Tasarla aşaması ile başlatılabilmektedir.

Harekete Geç: Öğrenciler Python programlama dili ile zeki optimizasyon tabanlı çözüm yaklaşımı oluşturup uygulayabilir.

Yürüt: Eğitmenler öğrenciler ile etkileşimli bir biçimde farklı zeki optimizasyon algoritmalarını çalıştırır ve genel çözüm akışını değerlendirir. Öğrenciler ise ilgili doğrultuda zeki optimizasyon süreçlerini kodlayıp çözüm süreçlerini uygulayabilir.

Karar Ver: Öğrenciler alternatif yapay zekâ modelini düzenleyerek / kodlayarak çözüm üretebilir.

1. ALGILA

1.1. Optimizasyon Kavramı

Eğitmene Not

Eğitmen, öğrencilere "**optimizasyon**" kavramının ne anlama geldiğini sorar ve tartışır. Gerçek hayattan örneklerle bilinen birtakım çözümlerin aslında optimizasyonla ilişkili olduğunu açıklar. Böylece öğrencilerin konuya dikkatini çeker.

Optimizasyon ya da başka bir deyişle 'en iyileme' kavramı, genel olarak bir problemin çözümünde en uygun parametrelerin tespit edilmesi olarak bilinmektedir. Optimizasyon aslında günlük hayatta sayısal tahminlerde bulunurken ya da aklımızdan planlamalar yaparken gerçekleştirdiğimiz çözümlerin bilimsel ifade ediliş şeklidir. Daha genel anlamda optimizasyon kavramının matematik ile bağlantısı, değeri bilinmeyen denklem değişkenlerinin bulunması şeklinde olmaktadır.

Matematiksel anlamda optimizasyon, denklemlerle modellenmesi yapılmış bir problem için bilinmeyen değerlerin bulunmasına karşılık gelmektedir.

Optimizasyonu matematiksel olarak ifade etmek için fonksiyon gösterimleri kullanılmaktadır. Örneğin, y ile gösterilen yazılı sınavı puanı ve s ile gösterilen sözlü sınavı puanı dikkate alınmak üzere, y'nin %60'ı ve s'nin %40'ının alınması hesaplanacak ders başarı puanı DB şu şekilde gösterilebilmektedir:

DB(y, s) = (0,6 * y) + (0,4 * s)

DB fonksiyonunda bir dersten başarılı olmak için başarı puanının en az 60 olması gerektiğini kabul edersek, yazılıdan (y) 50 alan bir öğrencinin, sözlüden (s) en az kaç alması gerektiğini bulmak, optimizasyona tekabül etmektedir. Burada denklemdeki 0,6 ve 0,4 **katsayı** olarak bilinmekte, y ve

s ise **değişken** olarak ifade edilmektedir. Eğer fonksiyona +10 puan kanaat puanı eklenecek olursa bu değer de **sabit** olarak isimlendirilmektedir.

Eğitmene Not

Eğitmen öğrencilere, 50 yazılı puanı karşısında, sözlüden en az 75 puan alınırsa başarılı olunacağını, 75'ten daha fazla puanların da bu kapsamda kabul edilebileceğini açıklar. Bununla birlikte öğrencilerden benzeri bir matematiksel model tasarlayıp tasarlayamayacaklarını sorar ve tartışır.

DB fonksiyonunun hesaplanması aklımızdan çözebileceğimiz bir optimizasyon problemidir. Bununla birlikte gerçek dünyadaki birçok problem optimizasyon modellenmesiyle çözümlenebilmektedir. Örnek olarak:

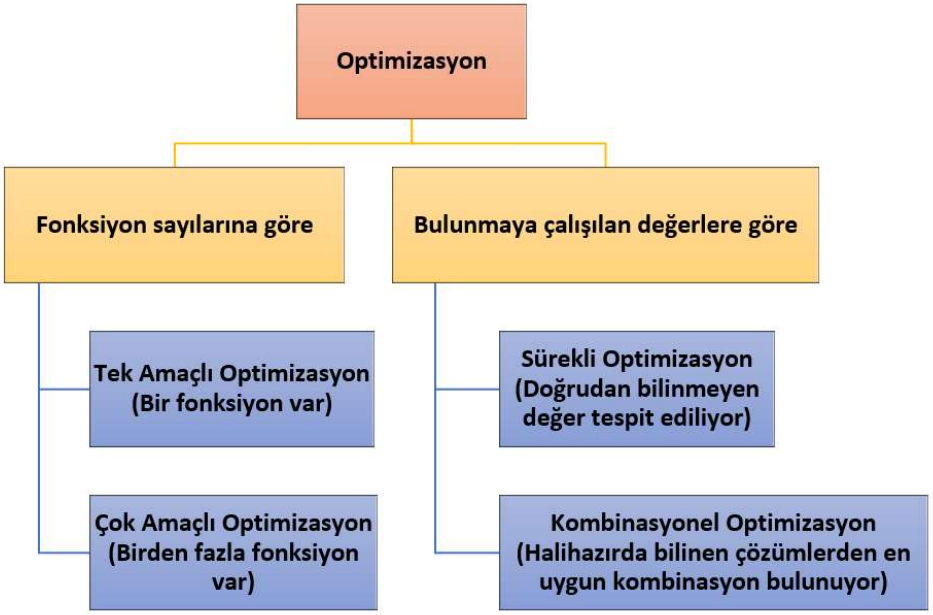
- Eldeki hammaddeleri en az kayıpla kullanıp istenilen ölçülerde ürün elde etmek bir optimizasyon problemidir.
- Bir inşaatta istenilen yükseklik ve genişlikteki odalarda, en iyi güneş ışığı oranını elde etmek için belli sayıdaki pencerelerin hangi noktalara konumlanması gerektiği optimizasyon problemi olmaktadır.
- Belli özellikleri baskın kimyasal bir maddeyi elde etmek için yüzlerce bileşenden kaçar gram karıştırılması gerektiği optimizasyon çözümlemesini gerekli kılmaktadır.
- 10 farklı şehirden hangilerinin ziyaret edilmesi neticesinde en kısa mesafe ile en az yakıt tüketiminin sağlanacağı sorusu bir optimizasyon çözümünü işaret etmektedir.

Özellikle mühendislik tabanlı optimizasyon problemleri daha yüksek dereceli olabilmekte (yani değişkenler daha yüksek üstel sayılarda temsil edilmekte), aynı anda çok daha fazla değerin bulunmasını gerekli kılabilmekte ve daha karmaşık fonksiyonlara tekabül edebilmektedir. Böyle durumlarda **türev** gibi daha ileri düzey matematiksel çözümler kullanılmaktadır.

Optimizasyon problemleri modellenirken şu hususlar da oldukça önemli olmaktadır:

- Değeri bulunması gerekli olan değişkenlerin sınır değerleri [Örneğin, DB probleminde yazılı (y) ve sözlü sınav (s) değişkenleri en az 0 (sıfır), en fazla 100 (yüz) olmalıdır.] dikkate alınmalıdır.
- Aynı anda birbirleriyle ilişkili birden fazla fonksiyonun dikkate alınması gerekebilmektedir. Örneğin bir şirkette K fonksiyonu ile karı yükselten değişkenler bulmaya çalışılırken, aynı anda Z fonksiyonu ile zararı en az tutmaya çalışan değişkenler bulunabilmekte; aynı anda iki fonksiyonda da yer alan personel sayısı şeklindeki bir değişken için her iki fonksiyonu da sağlayan en uygun (optimum) değer bulunabilmektedir.
- Bazı optimizasyon problemleri doğrudan bilinmeyen değerlerin bulunmasına odaklanırken (sürekli optimizasyon) bazıları bilinen çözümlerin hangi kombinasyonlarının daha iyi olacağını bulmak için (kombinasyonel optimizasyon) modellenmektedir.

Şekil 7.1'de optimizasyonun modellenmesi ile ilgili problemlere yönelik farklı sınıflar temel karakteristikleri ile kısaca ifade edilmiştir.



Eğitmene Not

Eğitmen öğrencilere farklı optimizasyon modelleme şekillerine yönelik örnekler vererek, konunun irdelenmesini sağlar. Bu konuda öğrencilerden de örnekler istenerek konu tartışılır.

Şekil 7.1. Farklı optimizasyon modellemeleri

1.2. Zeki Optimizasyon Kavramı

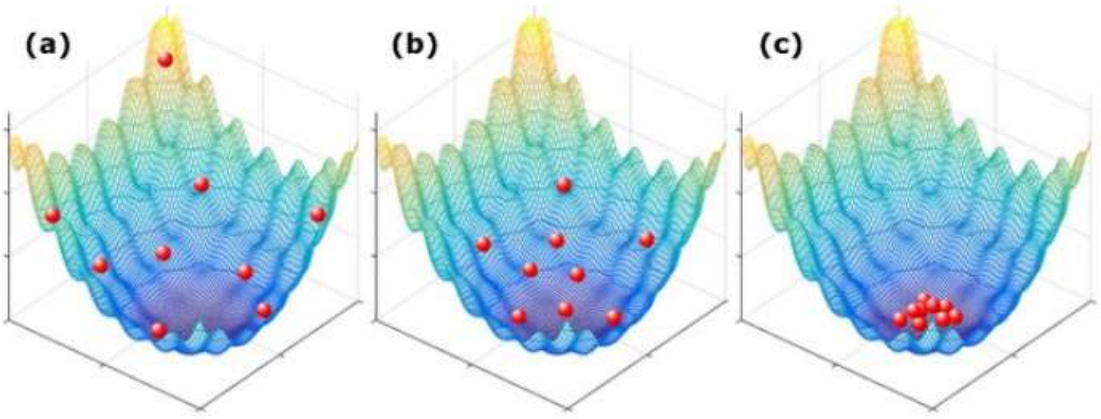
Matematikte bilinen klasik optimizasyon yöntemleri, karmaşıklığı yüksek ve oldukça fazla değişken içeren problem modellerinde başarılı sonuç verememektedir. Bunun neticesinde temellerini doğadaki canlı sürülerinin iş birliği içerisindeki, şans faktörü içeren eylemlerinden alan yapay zekâ tabanlı optimizasyon algoritmaları geliştirilmiştir.

Zeki optimizasyon, makine öğrenmesi kapsamında olduğu gibi öğrenme süreci gerektirmemektedir. Daha çok döngüler yardımıyla en uygun değerlerin tespit edilmeye çalışılmasına dayanmaktadır. Bunu sağlamak için de algoritmalar tasarlanırken şu mekanizmalar işletilmektedir:

- Tıpkı etmen tabanlı modellemede olduğu gibi *N* sayıda etmenin tanımı yapılmaktadır. Bu noktada etmen tabanlı modellemeden farklı olarak bu çözüm unsurları sadece sayı ya da kombinasyon aramakla sorumludur. Her etmen algoritma başlangıcında rastgele değerlerle eşlenerek aramaya başlamaktadır. Bu süreç tıpkı çok sayıda kişinin aynı problemi çözmek için uğraşıp elde ettikleri sonuçlara göre birbirleriyle yardımlaşmaları gibidir.

- Problemle bağlantılı olarak *N* sayıda etmen rastgele hareketlerle problem modeline (denkleme/fonksiyona) konulacak değerler türetmekte ve her bir etmenin hesapladığı nihai değerlerle en uygun (en küçük/minimum ya da en büyük/maksimum) sonucu veren değerler tespit edilmektedir.
- *N* adet etmen arasında belli bir aşamada en uygun değeri verene diğerleri sayısal olarak yaklaştırılmaya çalışılmakta, bu süreç belli bir döngü sayısı boyunca işletilmektedir.
- Zeki optimizasyonu sağlamak için farklı matematiksel mekanizmalarla tasarlanan ve doğadaki canlılardan (kuşlar, böcekler vs.), insan topluluklarından ve hatta dinamik olgulardan (mevsimsel değişimler, fizik kanunlarıyla bağlantılı olaylar, çiçek tozlaşması, akarsuların akışı vs.) esinlenen, bilinmeyen değerleri arama yolunda kendi iç farklılıklarını içeren algoritmalar tasarlanabilmektedir. Farklı algoritmalarda *N* adet çözüm unsurlarını tarif etmek için algoritma esin kaynağına göre parçacık, birey, arı, karınca gibi isimler kullanılabilmektedir.

Optimizasyon içerisinde tasarlanan modeller görsel olarak incelendiğinde Şekil 7.2'dekine benzer bir çözüm süreci elde edilmiş olmaktadır. Şekilde görüldüğü gibi, bulunmaya çalışılan en uygun düşük değeri bulmak soldan sağa doğru belli bir süreci gerekli kılmaktadır. Bu sırada istenilen değere ulaşma sırasında çeşitli engeller ve yanlış tespitlerde bulunmak da olasıdır. Bu nedenle zeki optimizasyon garanti çözüm sunmayan, ancak mümkün olduğunca etkili çözümlerle gerçek hayattaki problemlere başarılı sonuçlar üreten algoritmalar içermektedir.



Şekil 7.2. Görsel olarak tipik bir optimizasyon süreci (Web Kaynağı 7.1)

1.3. Genetik Algoritma ile Zeki Optimizasyon

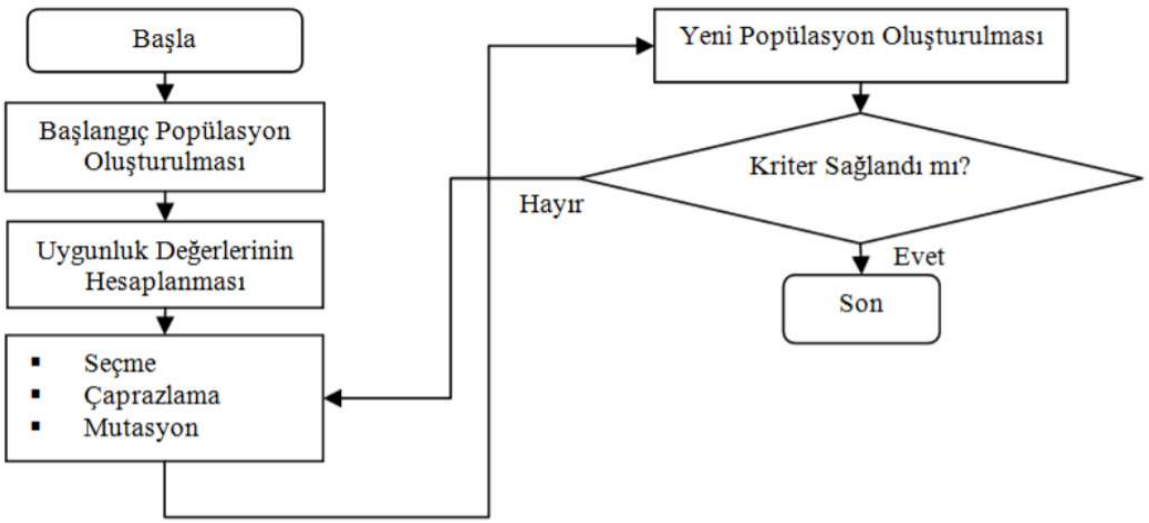
Zeki optimizasyon konusunda yüzlerce algoritma bulunmaktadır. Ancak bunlardan bazıları konuyu anlamak ve başlangıç için daha idealdir. Genetik Algoritma da yapısı itibariyle buna en uygun algoritmalardan biridir. Genetik Algoritma, güçlü ve başarılı olan çözüm unsurlarının özelliklerinin yeni çözüm unsurlarına aktarılarak başarı şansının artırıldığı bir zeki optimizasyon algoritmasıdır. Buna göre mevcut popülasyonlardan (çözüm unsurlarından) problem fonksiyonu için nispeten daha iyi değer üretenler birbirleriyle eşleştirilerek daha başarılı yeni nesil bireylerden

popülasyonlar elde edilmektedir. Hedeflenen döngü adımları boyunca nesillerin aynı şekilde türetilmesine devam edilmektedir. Genetik Algoritma orijinal yapısı itibariyle bir sürekli optimizasyon algoritmasıdır.

Genetik Algoritma'da bireylerin her biri problem fonksiyona ait aranılan değerleri tutmak adına gen adı verilen bileşenlerle tanımlanmaktadır. Genel olarak kodlama adı verilen bu süreç ile örneğin beş bilinmeyen değişkeni olan bir problem için *N* tane bireyin her birinde beşer adet değer/gen tutulmaktadır. Buradan hareketle her birey bir tür kromozom ile temsil edilir. Genetik Algoritma'nın her yeni döngüsünde, ilgili bireylerin taşıdığı değerler (genler) problem fonksiyonuna konularak fonksiyon sonuçları (uygunluk fonksiyonu değerleri) üretilmekte, belli sayıdaki en iyi değerleri bulan bireyler arası gen değişimleri (çaprazlama/crossover) ve belli genlerin de yeni değerlerle değiştirilmesi (mutasyon) suretiyle eski popülasyon yerine yeni nesil popülasyon (ebeveynlerin yerine çocuklar) elde edilerek bir sonraki döngüye geçilmektedir. Genetik Algoritma başarılı olan çözümleri gelecek süreçlere taşıyabilmesi nedeniyle Evrimsel Algoritmalar arasında kabul edilmektedir. Şekil 7.3 bu çerçevede Genetik Algoritma adımlarını kısaca görselleştirmektedir.

Eğitmene Not

Eğitmen öğrencilere Genetik Algoritma'daki kodlama, çaprazlama, çaprazlama için birey (etmen) eşleştirmeleri gibi süreçlerin birçok farklı şekilde yapılabildiğini açıklayarak, Genetik Algoritma'nın temellerindeki evrimsel süreçlere vurgu yapar. Ayrıca bu mekanizmaları canlılar arasındaki kalıtımsal özelliklerin aktarılmasına bağlayarak konunun pekişmesini sağlar.



Şekil 7.3. Genetik Algoritma'nın genel akışı (Web Kaynağı 7.2)

Düşün, tartış…

Doğadaki canlılar (örneğin kuşlar, arılar) problemlerini çözme konusunda ne şekillerde iş birlikleri yaparlar? Bunların zeki optimizasyona yansıması nasıl olabilir? Sorulara dair öğrenci görüşleri sınıf ortamında tartışılabilir ve bu konuda Web ortamındaki alternatif düşünceler incelenerek yorumlanabilir.

1.4. Karınca Koloni Optimizasyonu ile Zeki Optimizasyon

Karınca Koloni Optimizasyonu, kombinasyonel optimizasyon söz konusu olduğunda, konuyu anlamak için en uygun algoritmadır. Söz konusu algoritma, karıncaların yiyecek kaynaklarına giden yolları birbirlerine işaret etmek için kullandıkları biyolojik mekanizmalara dayanan bir zeki optimizasyon algoritmasıdır. Buna göre karıncalar, gerçek dünyada yiyecek yollarını işaretlemek için feremon (pheremone) adı verilen bir madde salgılamaktadır. Bu yolla yiyeceklere giden en uygun yollar, sürüler için algılanır hale gelmektedir. Benzer şekilde kombinasyonal optimizasyon çözümleri için algoritma içerisinde tanımlanmış değişkenler birer feremon görevi görmekte, feremon değişken değerleri yüksek olan çözüm kombinasyonları bir fonksiyon üzerinden tespit edilebilmektedir.

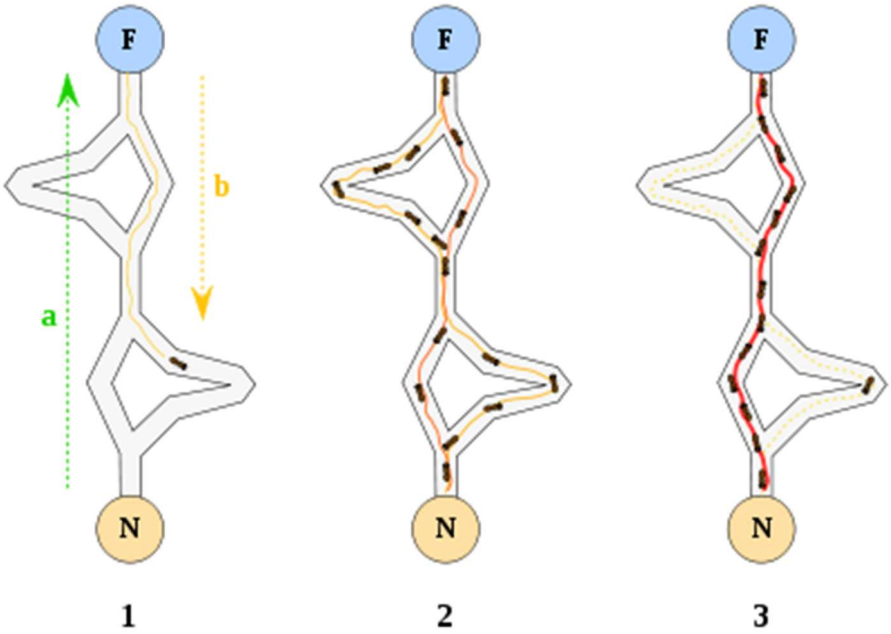
Eğitmene Not

Eğitmen öğrencilere kombinatöryal ve sürekli optimizasyon arasındaki farklılıkları, çözüm unsurları halihazırda bilinen; gezgin satıcı problemi (TSP: Travelling Salesman Problem) üzerinden örneklendirerek açıklar.

Karınca Koloni Optimizasyonunda karınca adı verilen her bir parçacık ve kurulan problem modeli, şu mekanizmalara dayanarak optimizasyon çözümünü desteklemektedir:

- *N* adet her bir karıncanın taşıdığı feremon değerleri vardır. Benzer şekilde, problem kapsamında aralarında potansiyel bağ olduğu düşünülen unsurlar da (örneğin en kısa yol bulunması istenen farklı şehirler arası yollar) feremon değerleriyle temsil edilmektedir.
- Her döngü adımında karıncalar kombinasyonel adımlarla tespit ettikleri potansiyel unsurları problem fonksiyonundaki uygunluk/başarı değerini hesaplamak için kullanırlar.
- Uygun değer üreten karıncaların ve potansiyel çözüm unsurlarının feremon değerleri geliştirilir.
- Bir karıncanın aynı anda birden fazla potansiyel çözüm unsuruna yönelme durumu varsa, olasılıksal hesaplamalarla hangi tarafı tercih edeceği belirlenir.
- Algoritma tamamen çalışıp sona erdiğinde en çok feremon değeriyle desteklenen (en uygun) çözüm problemin çözümü olmaktadır.

Karınca Koloni Optimizasyonu'nun genel çözüm süreci Şekil 7.4'tekine benzer bir akışı ortaya çıkarmaktadır. Şekilde N ve F arası en uygun rota zamanla olgunlaşan; optimum çözümü oluşturmaktadır.



Şekil 7.4. Karınca Koloni Optimizasyonu ile temsili problem çözümü (Web Kaynağı 7.3)

Karınca Koloni Optimizasyonu kapsamında farklı potansiyel çözüm unsurları arası geçişlerde feremon değerlerinin güncellenmesi konusunda farklı matematiksel hesaplamalar gerçekleştirilmektedir (Şekil 7.5). Bu hesaplamalar doğrultusunda karıncalar problem çözümünde tıpkı etmen tabanlı modellemede olduğu gibi yeni adımlar atabilmektedir.

$$P_{ij}^{k} = \begin{cases} \dfrac{\tau_{ij}^{\alpha}\,\eta_{ij}^{\beta}}{\sum_{c_{il}\in N(s^p)} \tau_{il}^{\alpha}\,\eta_{il}^{\beta}} & c_{il} \in N(s^p)\, ise \\ 0 & aksi\ taktirde \end{cases} \quad (1)$$

τ_{ij} : (i, j) köşelerindeki feromon iz miktarıdır.

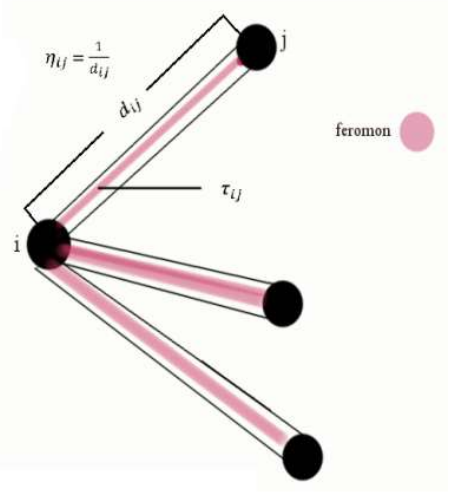
η_{ij} : (i, j) köşeleri arasındaki görünürlük (visibility) veya bağlantının amaç fonksiyonunun özelliğidir.

$\eta_{ij} = \frac{1}{d_{ij}}$

d_{ij} = i ve j noktaları arasındaki uzaklıktır.

$$\tau_{ij} \leftarrow (1-\rho)\,\tau_{ij} + \sum_{k=1}^{m} \Delta\tau_{ij}^{k} \quad (2)$$

ρ: Feromon izinin buharlaşma oranı. ($0 < \rho < 1$)



Şekil 7.5. Karınca Koloni Optimizasyonu içerisinde atılacak yeni adımların hesaplanması

Biliyor musunuz?

Zeki optimizasyon içerisinde yüzlerce farklı algoritma vardır. Araştırıp incelenebilecek alternatif bazı algoritmalar şu şekildedir:

- Parçacık Sürü Optimizasyonu
- Guguk Kuşu Algoritması
- Ateş Böceği Algoritması
- Yapay Arı Kolonisi Algoritması
- Girdap Optimizasyon Algoritması
- Akıllı Su Damlaları Algoritması
- Diferansiyel Evrim Algoritması

Simülasyon ile Zeki Optimizasyon

2. TASARLA

Genetik Algoritma kullanmak suretiyle bu paragrafı takiben verilen fonksiyonu 44 değerine eşitleyen x1, x2, x3, x4, x5, x6 değerlerini tespit etmek istenmektedir (İlgili kodun tamamı Github platformu Hafta7 klasörü altında H7_genetik-algoritma.py dosyası ile sunulmuş durumdadır):

f(x1, x2, x3, x4, x5, x6) = 4x1 − 2x2 + 3,5x3 + 5x4 − 11x5 − 4,7x6

Öğrenciler, ilk olarak **pip install pygad** komutu ile Genetik Algoritma için kullanılan bir Python kütüphanesini kurarlar.

Eğitmen öğrencilerle birlikte Şekil 7.6'da gösterilen kodları yazar:

```
1 import pygad
2 import numpy
3
4 function_inputs = [4,-2,3.5,5,-11,-4.7]
5 desired_output = 44
6
7 def fitness_func(solution, solution_idx):
8     output = numpy.sum(solution*function_inputs)
9     fitness = 1.0 / numpy.abs(output - desired_output)
10    return fitness
11
12 fitness_function = fitness_func
13
14 num_generations = 100
15 num_parents_mating = 7
16
17 sol_per_pop = 50
18 num_genes = len(function_inputs)
19
20 last_fitness = 0
```

Şekil 7.6. Genel fonksiyon ve başlangıç parametrelerinin kodlanması

4. satırda fonksiyon değişken katsayıları tanımlanmakta, 5. satırda ise fonksiyonun vermesi istenen değer belirlenmektedir.

7.-10. satır arası fonksiyonun tanımlandığı Python kod bloğudur.

14. satırda optimizasyon toplam döngü sayısı 100 nesil sayısı ile tanımlanmaktadır.

15. satırda tanımlandığı üzere her döngüde en iyi 7 birey çaprazlama, mutasyon gibi işlemlerle yeni nesil popülasyon oluşturmak için seçilmektedir.

17. satırda tanımlandığı üzere, toplamda 50 adet bireyden oluşan bir popülasyon söz konusudur.

3. HAREKETE GEÇ

Problem ve algoritma tanımları sonrası her döngüde (nesilde) en iyi çözümleri ekrana yansıtan nesil_ozeti fonksiyon kod bloğu yazılır ve **pygad.GA** çağrısı ile Genetik Algoritma uygulaması tanımlanır (Şekil 7.7):

```
def nesil_ozeti(ga_instance):
    global last_fitness
    print("Nesil = {generation}".format(generation=ga_instance.generations_completed))
    print("Fonksiyon Sonucu = {fitness}".format(fitness=ga_instance.best_solution()[1]))
    print("Degisim = {change}".format(change=ga_instance.best_solution()[1] - last_fitness))
    last_fitness = ga_instance.best_solution()[1]

ga_instance = pygad.GA(num_generations=num_generations,
                       num_parents_mating=num_parents_mating,
                       fitness_func=fitness_function,
                       sol_per_pop=sol_per_pop,
                       num_genes=num_genes,
                       on_generation=nesil_ozeti)
```

Şekil 7.7. Genetik Algoritma çözümünün tanımlanması

4. YÜRÜT

Tanımlanan algoritma ve uygulama düzeni için **run** fonksiyonu ile optimizasyon süreci başlatılır. Optimizasyon sonucundaki en iyi/uygun değerler üç farklı değişken (en iyi çözüm değişken değerleri için **solution,** en iyi çözüm fonksiyonu sonucu için **solution_fitness** ve en iyi çözümü veren birey için **solution_idx**) altında toplanır:

```
ga_instance.run()

solution, solution_fitness, solution_idx = ga_instance.best_solution()
```

Şekil 7.8. En iyi çözüm tespitinin kodlanması

En uygun/iyi çözüme ilişkin değerler ekrana yazdırılır:

```
print("En uygun cozum degisken degerleri : {solution}".format(solution=solution))
print("En uygun cozumu veren birey indeks no.: {solution_idx}".format(solution_idx=solution_idx))

prediction = numpy.sum(numpy.array(function_inputs)*solution)
print("En uygun cozum ile fonksiyon sonucu : {prediction}".format(prediction=prediction))

if ga_instance.best_solution_generation != -1:
    print("En uygun cozum {best_solution_generation} nesil sonra elde edildi."
          .format(best_solution_generation=ga_instance.best_solution_generation))
```

Şekil 7.9. En iyi çözümün ekrana yazdırılması

5. KARAR VER

5.1. Sonuçların Gözlemlenmesi

Algoritma sürecinin başlatılması ile ekrana her döngüde/nesilde en uygun sonuca ilişkin bilgiler Şekil 7.10'da olduğu gibi yansıtılmaktadır:

```
Nesil = 86
Fonksiyon Sonucu = 903.3083029984443
Degisim = 0.0
Nesil = 87
Fonksiyon Sonucu = 903.3083029984443
Degisim = 0.0
Nesil = 88
Fonksiyon Sonucu = 903.3083029984443
Degisim = 0.0
Nesil = 89
Fonksiyon Sonucu = 903.3083029984443
Degisim = 0.0
```

Şekil 7.10. Nesillere göre en iyi çözümlerin ekrana yazdırılması

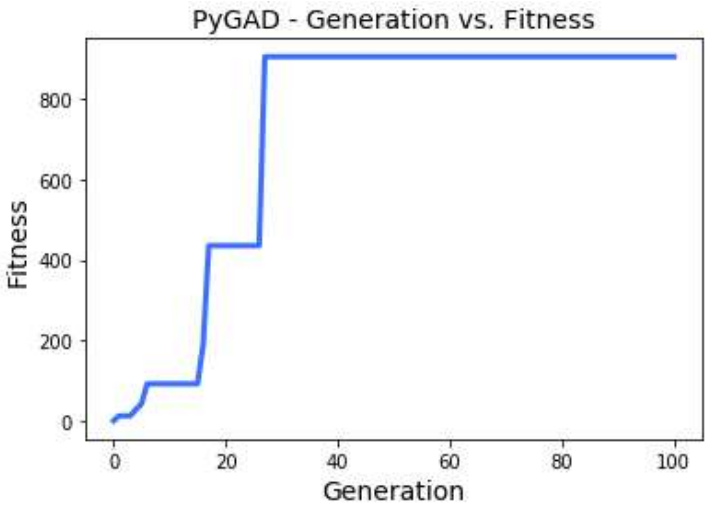
Ekrana yansıtılan sonuçlarda öncelikli olarak ilgili nesil sırası bilgisi, ardından o nesil içerisinde elde edilen en uygun değer (Fonksiyon Sonucu) yazdırılmaktadır. Bir önceki nesilde bulunan en iyi sonuç ile arada değişim olup olmadığı ise Degisim ibaresi altında gösterilmektedir.

Optimizasyon süreci bitmesiyle birlikte en iyi/uygun sonucu veren altı değişkenin değerleri, bu değerleri veren bireyin indeks numarası, fonksiyon sonucu ve en iyi sonucun ilk görülmeye başlandığı nesil sırası da ekranda belirtilmektedir (Şekil 7.11):

```
En uygun cozum degisken degerleri : [-1.16498213  1.98641117 -2.33159974  3.01034528
-3.70851693 -1.05251707]
En uygun cozumu veren birey indeks no.: 0
En uygun cozum ile fonksiyon sonucu : 43.99889295825503
En uygun cozum 27 nesil sonra elde edildi.
```

Şekil 7.11. Optimizasyon süreci sonucu çözüm

İlgili kodlara ek olarak **ga_instance.plot_fitness()** komutu yardımıyla en iyi/uygun değerin değişim grafiği de gösterilebilmektedir:



Şekil 7.12. En iyi/uygun değerin değişim grafiği

Eğitmene Not

Eğitmen öğrencilere elde edilen grafik üzerinde en iyi/uygun değerin değişiminin nasıl aşama aşama farklılaştığını anlatır. Ardından algoritmayı tekrar değiştirerek her zaman için tıpatıp aynı grafiğin elde edilemeyebileceği, bu durumun optimizasyonunun doğasında olduğunu açıklar. Farklı parametreler (nesil sayısı, en iyi birey sayısı vs.) ile alternatif senaryoların öğrenciler tarafından denenmesini sağlar.

PYTHON KODLARI:

```
import pygad
import numpy

function_inputs = [4,-2,3.5,5,-11,-4.7]
desired_output = 44

def fitness_func(solution, solution_idx):
    output = numpy.sum(solution*function_inputs)
    fitness = 1.0/numpy.abs(output - desired_output)
    return fitness
fitness_function = fitness_func
num_generations = 100
num_parents_mating = 7
sol_per_pop = 50
num_genes = len(function_inputs)
last_fitness = 0
def nesil_ozeti(ga_instance):
    global last_fitness
    print("Nesil =
{generation}".format(generation=ga_instance.generations_completed))
    print("Fonksiyon Sonucu =
{fitness}".format(fitness=ga_instance.best_solution()[1]))
    print("Degisim = {change}".format(change=ga_instance.best_solution()[1] -
last_fitness))
```

```
    last_fitness = ga_instance.best_solution()[1]
ga_instance = pygad.GA(num_generations=num_generations,
                       num_parents_mating=num_parents_mating,
                       fitness_func=fitness_function,
                       sol_per_pop=sol_per_pop,
                       num_genes=num_genes,
                       on_generation=nesil_ozeti)
ga_instance.run()

solution, solution_fitness, solution_idx = ga_instance.best_solution()
print("En uygun cozum degisken degerleri : {solution}".format(solution=solution))
print("En uygun cozumu veren birey indeks no.:
{solution_idx}".format(solution_idx=solution_idx))

prediction = numpy.sum(numpy.array(function_inputs)*solution)
print("En uygun cozum ile fonksiyon sonucu :
{prediction}".format(prediction=prediction))
if ga_instance.best_solution_generation != -1:
    print("En uygun cozum {best_solution_generation} nesil sonra elde edildi."
          .format(best_solution_generation=ga_instance.best_solution_generation))
ga_instance.plot_fitness()
```

Dünyadan Haberler

Yapay zekâ, iklim değişikliğini tahmin edebilecek güçlü bir araç olabilir mi?

Yapay zekânın iddialı bir hedefi, Dünya'nın bir "dijital ikizi"ni, yani gezegenimizdeki sistemleri ve süreçleri taklit eden bir kopyasını oluşturmak. Dr. Mathieu, "*Bahsedilen 'dijital ikiz', dünyamızı yansıtan sayısal bir laboratuvar niteliğindedir. Burada çeşitli denemeler yapabilir, doğacak sonuçları değerlendirebilir ve edinilen bulgular ışığında gerekli politikaları oluşturabiliriz.*" diyor. BAS çevresel veri bilimcisi Dr. Scott Hosking, "*Doğal ortamların dijital ikizlerini geliştirmek için yapay zekâmız gerekli yapı taşlarına sahiptir ve bunlardan yola çıkarak nihayetinde Dünya'nın da bir dijital ikizini de oluşturabileceğiz.*" diye konuşuyor. "*Gezegenimizde değişen faktörlerin her birini gerekli detay seviyesinde izleyemiyoruz. Doğal ortamların dijital ikizlerini oluşturarak, kutup bölgeleri gibi erişimi zor olan bölgeler hakkında daha sağlıklı veriler oluşturabiliriz. Bu bilgiler, ölçüm yapacak insansız hava araçlarını ve denizaltılarını ölçüm yapmaları için daha hedefli olarak yönlendirebilmemizi sağlayabilir.*"

Fakat yapay zekâ hâlen kusursuz sonuçlar sunan bir teknoloji değil. Uzmanlar, elimizdeki verilerin iklim öngörülerinde bulunacak algoritmaları geliştirme konusunda yetersiz olduğuna dair uyarıda bulunuyor. Atmosferik ve Çevresel Araştırmalar (AER) mevsimsel tahmin direktörü ve MIT'de iklim bilimcisi olarak görev yapan Dr. Judah Cohen, bu konudaki görüşlerini şöyle ifade ediyor: "*Uyduların yaygınlaştığı 1979'dan beri toplanan verilerden faydalanıyoruz; fakat buna rağmen elimizde yapay zekâdan optimum sonuçlar alabilmemize yetecek kadar bilgi bulunmuyor. Belki oluşturduğumuz modeller temelinde yapay veriler oluşturabiliriz; fakat bu verilerin geçmişe dayalı gerçek veriler kadar sağlıklı olup olmayacağı konusu belirsiz*"(Web Kaynağı 7.4).

6. İLAVE ETKİNLİK

Birbirine yollarla bağlı beş şehir arasında atılabilecek en kısa mesafeyi bulmak adına Karınca Koloni Optimizasyonu algoritması kullanılmak istenmektedir. (Uygulamanın tamamı Github platformu Hafta7 klasörü altında iki dosya altında: H7_karinca-kolonisi.py ve H7_karinca-kolonisi_ornek.py dosyaları ile sunulmuş durumdadır).

Söz konusu probleme göre, Python ortamında farklı indekslerle temsil edilen şehirler arasındaki mesafe değerleri Çizelge 7.1'de gösterildiği şekilde tanımlanmalıdır.

Çizelge 7.1. Şehirler arasındaki mesafe değerlerinin tanımı.

	0	1	2	3	4
0	inf	2	2	5	7
1	2	inf	4	8	2
2	2	4	inf	1	3
3	5	8	1	inf	2
4	7	2	3	2	inf

Söz konusu kombinatöryal optimizasyon problemi gösteriminden yola çıkılarak Karınca Koloni Optimizasyonu algoritmasının öncelikli olarak temel parametrelerinin tanımlanması sağlanır:

```python
import random as rn
import numpy as np
from numpy.random import choice as np_choice

class AntColony(object):

    def __init__(self, distances, n_ants, n_best, n_iterations, decay, alpha=1, beta=1):
        self.distances  = distances
        self.pheromone = np.ones(self.distances.shape) / len(distances)
        self.all_inds = range(len(distances))
        self.n_ants = n_ants
        self.n_best = n_best
        self.n_iterations = n_iterations
        self.decay = decay
        self.alpha = alpha
        self.beta = beta

```

Şekil 7.13. Karınca Koloni Optimizasyonu algoritmasının temel parametreleri

Ardından algoritmanın çalışmasını ve şehirler arası yollara feremon değerlerinin atanmasını sağlayan fonksiyonlar, Şekil 7.14'te olduğu gibi tanımlanır.

```python
    def run(self):
        shortest_path = None
        all_time_shortest_path = ("placeholder", np.inf)
        for i in range(self.n_iterations):
            all_paths = self.gen_all_paths()
            self.spread_pheronome(all_paths, self.n_best, shortest_path=shortest_path)
            shortest_path = min(all_paths, key=lambda x: x[1])
            print (shortest_path)
            if shortest_path[1] < all_time_shortest_path[1]:
                all_time_shortest_path = shortest_path
            self.pheromone * self.decay
        return all_time_shortest_path

    def spread_pheronome(self, all_paths, n_best, shortest_path):
        sorted_paths = sorted(all_paths, key=lambda x: x[1])
        for path, dist in sorted_paths[:n_best]:
            for move in path:
                self.pheromone[move] += 1.0 / self.distances[move]
```

Şekil 7.14. Genel algoritma akış kodları

Her bir karıncanın geçtiği yolların mesafelerinin toplandığı ve yine her karıncanın geçtiği yolların kayıt altına alındığı fonksiyonlar kodlanır:

```
37      def gen_path_dist(self, path):
38          total_dist = 0
39          for ele in path:
40              total_dist += self.distances[ele]
41          return total_dist
42
43      def gen_all_paths(self):
44          all_paths = []
45          for i in range(self.n_ants):
46              path = self.gen_path(0)
47              all_paths.append((path, self.gen_path_dist(path)))
48          return all_paths
```

Şekil 7.15. Karıncaların yol değerlerinin kayıt altına alınması

Karıncaların daha önce ziyaret etmedikleri şehirlere uğramalarını ve bu sırada yol ayrımlarında geldiklerinde, gidilebilecek bir sonraki şehri seçmelerini sağlayan fonksiyonlar tanımlanır:

```
50      def gen_path(self, start):
51          path = []
52          visited = set()
53          visited.add(start)
54          prev = start
55          for i in range(len(self.distances) - 1):
56              move = self.pick_move(self.pheromone[prev], self.distances[prev], visited)
57              path.append((prev, move))
58              prev = move
59              visited.add(move)
60          path.append((prev, start))
61          return path
62
63      def pick_move(self, pheromone, dist, visited):
64          pheromone = np.copy(pheromone)
65          pheromone[list(visited)] = 0
66
67          row = pheromone ** self.alpha * (( 1.0 / dist) ** self.beta)
68
69          norm_row = row / row.sum()
70          move = np_choice(self.all_inds, 1, p=norm_row)[0]
71          return move
```

Şekil 7.16. Karıncaların olasılıksal şehir seçiminin kodlanması

Eğitmene Not

Eğitmen bu noktada öğrencilerden algoritma ana kod bloğunu ayrı bir dosya olarak kaydetmelerini ister ve yeni bir kod dosyası içerisinde problemin çözümleneceği kodların yazılmasını sağlar.

Yeni bir Python kod dosyası açıldıktan sonra, öncelikli olarak **numpy** ve kodlanan algoritmanın çağrısı yapılır. Ardından şehirler arası mesafelerin tanımlandığı dizi veri yapısı tanımlanır:

```
import numpy as np

from ant_colony import AntColony

distances = np.array([[np.inf, 2, 2, 5, 7],
                      [2, np.inf, 4, 8, 2],
                      [2, 4, np.inf, 1, 3],
                      [5, 8, 1, np.inf, 2],
                      [7, 2, 3, 2, np.inf]])

```

Şekil 7.17. Şehir yapısının kodlanması

Problemin tanımı sonrasında toplamda 500 döngüde, 10 karınca üzerinden algoritma çağrısı yapılır:

```
ant_colony = AntColony(distances, 10, 1, 500, 0.95, alpha=0.05, beta=1)
shortest_path = ant_colony.run()
print ("En kısa yol: {}".format(shortest_path))
```

Şekil 7.18. Algoritma çağrısının kodlanması

Çağrı sonrasında **run** fonksiyonu ile başlatılan optimizasyon süreci sırasında her döngüde o ana kadar bulunan en kısa gidiş rotası ekrana yazdırılır; döngüsel süreç sona erdiğinde ise en kısa mesafeyi veren nihai yol/rota ifade edilir:

```
([(0, 1), (1, 4), (4, 3), (3, 2), (2, 0)], 9.0)
([(0, 2), (2, 3), (3, 4), (4, 1), (1, 0)], 9.0)
([(0, 1), (1, 4), (4, 3), (3, 2), (2, 0)], 9.0)
([(0, 2), (2, 3), (3, 4), (4, 1), (1, 0)], 9.0)
([(0, 2), (2, 3), (3, 4), (4, 1), (1, 0)], 9.0)
([(0, 2), (2, 3), (3, 4), (4, 1), (1, 0)], 9.0)
([(0, 1), (1, 4), (4, 3), (3, 2), (2, 0)], 9.0)
([(0, 1), (1, 4), (4, 3), (3, 2), (2, 0)], 9.0)
En kısa yol: ([(0, 2), (2, 3), (3, 4), (4, 1), (1, 0)], 9.0)
```

Şekil 7.19. Genel algoritma sonuçlarının akışı

En kısa yol, soldan sağa doğru hücre indeks değerleri dikkate alınmak suretiyle; örneğin (0, 2) ifadesi ile 0 indeksli şehirden 2 indeksli şehre, ardından (2, 3) ifadesi ile 2 indeksli şehirden de 3 indeksli şehre gidildiğini ve sürecin bu şekilde 9 birimlik mesafe ile en kısa yola tekabül ettiğini göstermektedir.

PYTHON KODLARI:

```
#karinca koloni algoritmasi kod bloğu
import random as rn
import numpy as np
from numpy.random import choice as np_choice

class AntColony(object):

   def __init__(self, distances, n_ants, n_best, n_iterations, decay, alpha=1,
beta=1):
      self.distances  = distances
      self.pheromone = np.ones(self.distances.shape)/len(distances)
      self.all_inds = range(len(distances))
      self.n_ants = n_ants
      self.n_best = n_best
      self.n_iterations = n_iterations
      self.decay = decay
      self.alpha = alpha
      self.beta = beta

   def run(self):
      shortest_path = None
      all_time_shortest_path = ("placeholder", np.inf)
      for i in range(self.n_iterations):
         all_paths = self.gen_all_paths()
         self.spread_pheronome(all_paths, self.n_best,
shortest_path=shortest_path)
         shortest_path = min(all_paths, key=lambda x: x[1])
         print (shortest_path)
         if shortest_path[1] < all_time_shortest_path[1]:
```

```
            all_time_shortest_path = shortest_path
        self.pheromone * self.decay
    return all_time_shortest_path

def spread_pheronome(self, all_paths, n_best, shortest_path):
    sorted_paths = sorted(all_paths, key=lambda x: x[1])
    for path, dist in sorted_paths[:n_best]:
        for move in path:
            self.pheromone[move] += 1.0/self.distances[move]

def gen_path_dist(self, path):
    total_dist = 0
    for ele in path:
        total_dist += self.distances[ele]
    return total_dist

def gen_all_paths(self):
    all_paths = []
    for i in range(self.n_ants):
        path = self.gen_path(0)
        all_paths.append((path, self.gen_path_dist(path)))
    return all_paths

def gen_path(self, start):
    path = []
    visited = set()
    visited.add(start)
    prev = start
    for i in range(len(self.distances) - 1):
        move = self.pick_move(self.pheromone[prev], self.distances[prev], visited)
        path.append((prev, move))
```

```
            prev = move
            visited.add(move)
        path.append((prev, start))
        return path

    def pick_move(self, pheromone, dist, visited):
        pheromone = np.copy(pheromone)
        pheromone[list(visited)] = 0

        row = pheromone ** self.alpha * (( 1.0/dist) ** self.beta)

        norm_row = row/row.sum()
        move = np_choice(self.all_inds, 1, p=norm_row)[0]
#örnek rota probleminin uygulanması
import numpy as np

from ant_colony import AntColony

distances = np.array([[np.inf, 2, 2, 5, 7],
                      [2, np.inf, 4, 8, 2],
                      [2, 4, np.inf, 1, 3],
                      [5, 8, 1, np.inf, 2],
                      [7, 2, 3, 2, np.inf]])

ant_colony = AntColony(distances, 10, 1, 500, 0.95, alpha=0.05, beta=1)
shortest_path = ant_colony.run()
print ("En kısa yol: {}".format(shortest_path))
```

Eğitmene Not

Eğitmen, bu haftanın eğitimine yönelik öğrencilerin ilgisini arttırmak ve yeni öğrenilen bilgileri önceden öğrenilen bilgiler ile karşılaştırmalarını sağlamak için aşağıda maddeler halinde verilen soruları tartışır.

- İlgili haftada ele alınan yapay zekâ uygulamalarına uygun örnek veri setleri neler olabilir?
- İlgili haftada ele alınan veri setlerinin yapay zekâ teknik ve kütüphanelerinde ne derecede başarım etkisi olmuştur?
- İlgili haftada öğretilen yapay zekâ tekniği ile uyumlu ve uyumsuz olabilecek açık erişimli veri seti çeşitleri sunan (kaagle, github vs.) internet sitelerinden elde edilecek farklı veri çeşitleri ile yapay zekâ tekniği uyumluluğunu sorar.

Kaynakça

Web Kaynağı 7.1: https://www.researchgate.net/publication/320531598_Adaptive_optics_stochastic_optical_reconstruction_microscopy_AO-STORM_by_particle_swarm_optimization

Web Kaynağı 7.2: https://dergipark.org.tr/tr/download/article-file/353860

Web Kaynağı 7.3: https://www.biyologlar.com/karinca-surusu-optimasyonu

Web Kaynağı 7.4: https://tr.euronews.com/green/2020/10/12/bozulan-iklimimizi-yapay-zeka-yard-m-yla-duzeltebilir-miyiz

8. Hafta: Derin Öğrenme ile İleri Düzey Çözümler

Ön Bilgi:

- Derin öğrenme kavramının temelleri, derin öğrenme modellemenin temelleri
- Python kodlamada döngüler, fonksiyonlar ve kütüphane yapılarının kullanımı (Bu konuda Python anlatım videoları: 14-16. Python kütüphaneleri serisi yardımcı araç olarak kullanılabilecektir).

Haftanın Kazanımları:

- Öğrenciler derin öğrenme model kavramını anlar.
- Öğrenciler, derin öğrenme modellerini büyük veri setlerine tasarlayarak uygular.
- Öğrenciler, derin öğrenme modellerinden elde edilen sonuçları yorumlayarak kavrar.
- Öğrenciler, derin öğrenme tabanlı problem çözebilme yetenekleri kazanır.
- Öğrenciler Python programlama dilinde derin öğrenme mimarileri için kullanılan kütüphane fonksiyonlarını kullanma yeteneği kazanır.
- Öğrenciler, Konvolüsyonel sinir ağları (CNN) derin öğrenme modelinin çalışma yöntemini kavrar.
- Öğrenciler CNN tabanlı AlexNet, ResNet, GoogleNet gibi mimarilerin yapısını kavrar.
- Öğrenciler ResNet tabanlı örnek bir problem üzerinde modeli tasarlar ve bu modeli çözüm yönünde uygular.
- Öğrenciler ResNet modelinden elde edilen sonuçları değerlendirerek yorumlar.
- Öğrenciler Yapay Zekâ alanı açısından derin öğrenme tekniğini uygular.

Haftanın Amacı:

Bu haftanın amacı, “derin öğrenme (deep learning)” kavramının tüm öğrenciler tarafından doğru bir şekilde anlaşılmasını sağlamaktır. Haftanın bir diğer amacı, sıklıkla kullanılan derin öğrenme tabanlı ResNet modeli kullanılarak kedi köpek görüntüleri üzerinde sınıflandırma işlemini başarılı bir biçimde gerçekleştirmektedir. Böylece, öğrencilerin Python programlama dilini kullanarak derin öğrenme tabanlı örnek problem modelleme ve çözüm üretme konusunda beceri kazanması sağlanacaktır.

Kullanılacak Malzemeler:

Bilgisayar, kâğıt, kalem, Python kod editörü, örnekler için çıktı görselleri

Haftanın İşlenişi:

Algıla: Öğrenciler derin öğrenme temellerine ilişkin kavramları ve genel mimari yapıları tanımlayabilir.

Tasarla: Öğrenciler derin öğrenme modelleme üzerinde görüntü verillerini (kedi ve köpek görüntülerini) ayırt etmek için çözüm süreçleri tasarlayabilir.

Harekete Geç: Öğrenciler Python programlama dili ile derin öğrenme tabanlı bir ResNet mimari yapısı oluşturup uygulayabilir. Bu hafta için Github platformu (https://github.com/deneyapyz/lise/) ile etkileşim Harekete Geç aşaması ile başlatılabilmektedir.

Yürüt: Eğitmenler öğrenciler ile etkileşimli bir biçimde derin öğrenme tabanlı ResNet modelini çalıştırır ve genel çözüm akışını değerlendirir. Öğrenciler aynı doğrultuda ResNet modeli özelinde model kodlayıp çözüm süreçlerini işletebilir.

Karar Ver: Öğrenciler alternatif yapay zekâ modelini düzenleyerek / kodlayarak çözüm üretebilir. Ayrıca öğrenciler, ResNet modelinden elde edilen sonuçları değerlendirerek tartışabilir.

1. ALGILA

1.1. Yapay Sinir Ağları Temelinde Derin Öğrenme'ye Geçiş

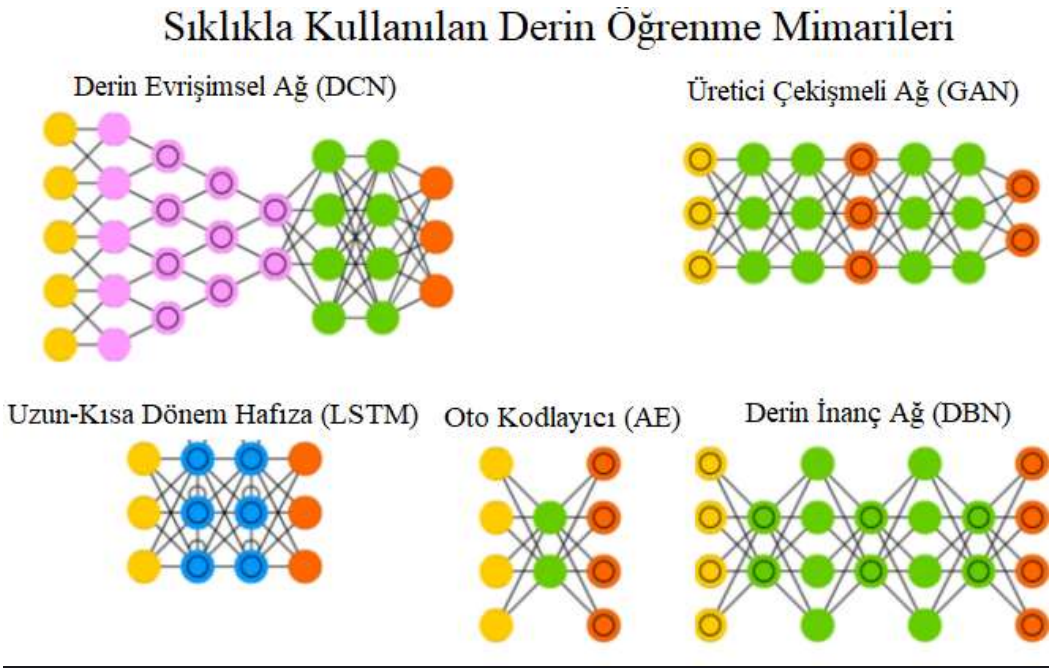
Eğitmene Not

Eğitmen, öğrencilere "**derin öğrenme**", "**derin öğrenme mimarisi**" ve "**derin öğrenmenin önemi**" hakkında bilgi sahibi olup olmadıklarını sorar ve tartışır. Böylece öğrencilerin konuya dikkatini çeker.

Derin öğrenme, sayısal sistemlerin yapılandırılmamış, etiketlenmemiş verilere göre öğrenmesi ve kararlar alması için yapay sinir ağlarını kullanan makine öğrenmesinin bir alt dalıdır (Şekil 8.1). Yapay zekâ alanında çalışan uzmanlar, büyük veri olarak adlandırılan ve genellikle metin, ses veya resimlere dayalı veri kümelerini daha hızlı ve doğru bir şekilde analiz etmek için derin öğrenme mimarilerini kullanırlar.

Şekil 8.1. Yapay zekâ ve alt dalları

Yapay sinir ağlarındaki katman sayılarının artırılmasıyla kurulan çok farklı türde derin öğrenme mimarileri bulunmaktadır. Şekil 8.2'de sıklıkla kullanılan derin öğrenme mimarileri verilmiştir.



Şekil 8.2. Derin öğrenme mimarileri (Web Kaynağı 8.1)

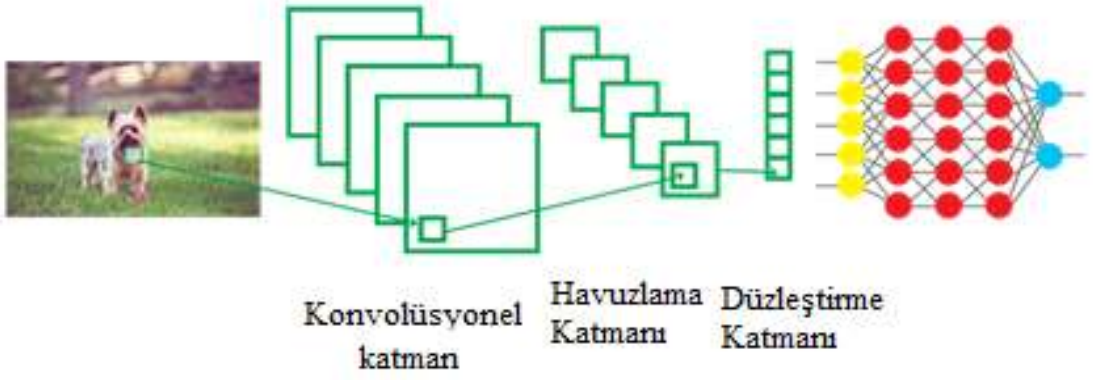
1.2. Evrişimsel Sinir Ağları Tekniği

Eğitmene Not

Eğitmen öğrencilere derin öğrenme mimarileri ve uygulama alanlarına ilişkin şu bilgileri aktararak konuyu pekiştirir:

Derin öğrenme mimarileri ile tıp, robotik, görüntü işleme, havacılık ve uzay sanayi, bilgisayarlı görü, görüntüden nesne tespiti, yüz tanıma, ses işleme-tanıma, finans gibi pek çok farklı alanda çözümler üretilmektedir (Doğan ve Türkoğlu., 2019).

Evrişimsel sinir ağı (**ConvNet/Convolutional neural networks -CNN**), girdi olarak alınan bir görüntüyü analiz ederek görünüşlerini veya görüntü içerisindeki görüntüleri birbirinden ayırt etmede sıklıkla kullanılan bir derin öğrenme mimarisidir. CNN derin öğrenme mimarisi Şekil 8.3'te görüldüğü gibi evrişim katmanı, doğrusal olmayan katman, havuzlama katmanı, düzleştirme katmanı ve tamamen bağlı katman yapılarından oluşmaktadır.



Şekil 8.3. CNN mimarisi (Web Kaynağı 8.2)

Eğitmene Not

Eğitmen öğrencilere CNN mimarilerinde kullanılan katmanların özelliklerini örnek vererek aktarır.

- **Konvolüsyonel Katman –** Hayvanın cinsi, ten rengi, şekli, çevre vb. fiziksel özellikleri saptamak için kullanılır.
- **Havuzlama Katmanı**– Havuzlama katmanı, görüntü üzerindeki boyutsallığı azaltmak için kullanılır. Böylece işlem sayısı azalırken, yakalanan gereksiz özellikler yok sayılır. Örneğin görüntüdeki köpeğin dışında kalan çevre gereksiz özellik olarak yok sayılır.
- **Düzleştirme Katmanı**– Tam bağımlı katmanda verileri analiz etmek için hazırlamada kullanılan katmandır. Örneğin önemli özellikleri verilen köpeğin görüntü halindeki iki boyutlu verilerini tek boyutlu veriye dönüştürerek tam bağımlı katmanda analiz edilmesini sağlar.
- **Tam Bağımlı Katman**– Tam bağımlı katman, CNN mimarisinin son ve en önemli katmanlardan birisidir. Tam bağımlı katman ile sınıflandırma veya regresyon işlemleri için eğitimler gerçekleştirilir. Örneğin görüntüden çıkartılan özelliklerin kedi köpek vs. gibi hangi hayvana ait olduğunu sınıflandırılmak için kullanılır.

Eğitmene Not

Eğitmen öğrencilere CNN tabanlı örnek mimariler hakkında şu derin öğrenme algoritmalarını örnek verip pekiştirerek aktarır.

- **LeNet** – Bu ağ, Konvolüsyonel Sinir Ağlarının ilk başarılı uygulaması sayılır. 1990'lı yıllarda Yann LeCun tarafından geliştirilmiş ve posta kodlarını, basit basamakları, banka çeklerindeki rakamları vb. okumak için kullanılmıştır.
- **AlexNet –** Bu ağ, 2012'de Endüstri 4.0 devriminin önemli bileşenlerinden biri olan yapay zekânın bir alt bileşini olan derin öğrenmenin ve Konvolüsyonel sinir ağ modellerinin tekrar popüler hale gelmesini sağlamıştır. Alex Krizhevsky, Ilya Sutskever ve Geoffrey Hinton tarafından geliştirilmiştir. Uzaktan algılama, yabancı otların birbirinden ayırt edilmesi, araçların sınıflandırılması örneklerinde sıklıkla kullanılır.

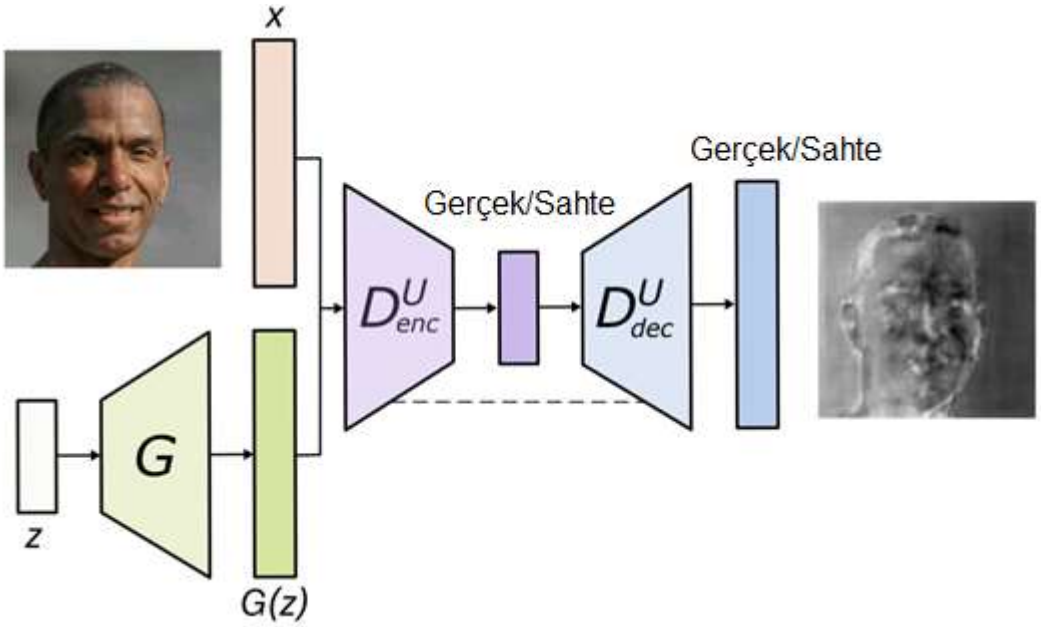
- **GoogLeNet** –2014 yılında Google tarafından geliştirilen, hesaplama maliyetinin yüksek olduğu yapay zekâ uygulamalarında hız ve başarımı oldukça yüksek olan bir CNN mimarisidir. Yüz tanıma ve algılama sistemlerinde sıklıkla kullanılır.
- **VGGNet –** Bu ağ, VGG Group (Oxford) tarafından geliştirilmiştir. VGG mimarisi, AlexNet mimarisinde kullanılan yüksek işlevci (kernel) boyutlarının azaltılması ile oluşturulmuştur. VGG mimarisi konvolüsyonel sinir ağının başarımının arttırılması daha da derinleştirilmesi tekniği ile oluşturulmuştur. Nesne tanıma, beyin tümör tespiti, kuş türlerinin sınıflandırılması vb. gibi birçok alanda kullanılır.

Biliyor musunuz?

COVID-19 ile savaşmak için, derin öğrenmenin araştırmalara yardımcı olduğunu biliyor musunuz? En başarılı makine öğrenme algoritmalarından biri olan derin öğrenme için, biyoinformatik alanında DNA, RNA, protein dizileri ve Nanopore sinyali sekans verileri kaynak oluşturmaktadır. Elde edilen verilerde gözlenmiş veya daha önce bilinmeyen motifleri, modelleri ve alanları tespit etmede ve tanımlamada derin öğrenme kullanılır. Derin öğrenme ile belirli hastalık ağları, gen birlikte ifade ağları, hücre sistemi hiyerarşi gibi grafik verileriyle başa çıkılabilir (Web Kaynağı 8.3).

1.3. Üretici Çekişmeli Ağ (GAN)

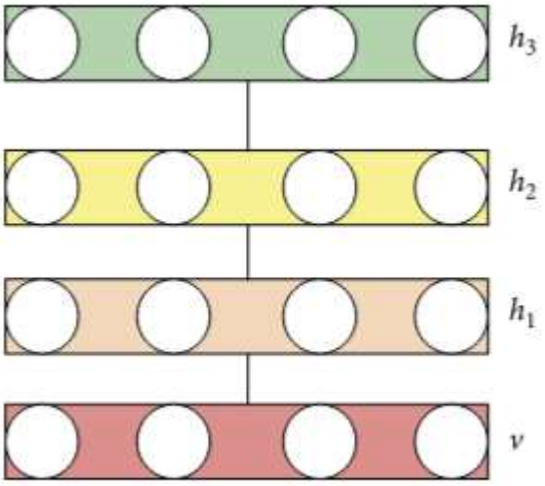
Çekişmeli üretici ağlar (GAN), 2014 yılında Ian Goodfellow ve çalışma grubu tarafından NIPS tarafından sunulmuştur. Çekişmeli üretken ağlar olarak adlandırılan derin öğrenme ağı İngilizce yazılışının baş harfleriyle "GAN" olarak ifade edilmektedir. GAN modelleme denetimsiz öğrenme türü olup, giriş verilerindeki kalıpları, orijinal veri kümesinden uygun bir şekilde çıkararak yeni örnekler oluşturabilmek için otomatik olarak keşfetmeyi ve öğrenmeyi içermektedir. Diğer derin öğrenme yapay sinir ağ modellerinden farklı olarak bir üretici (generative, G) ve bir ayırıcıya (discriminator, D) sahip iki farklı derin ağ vardır. GAN mimarisinde bu iki ağın çekişmeli olarak çalışmasıyla öğrenme işlemi gerçekleşir. Temel bir GAN mimarisinin yapısı Şekil 8.4'te verilmiştir.



Şekil 8.4. GAN model şeması (Web Kaynağı 8.4)

1.4. Derin İnanç Ağları Tekniği

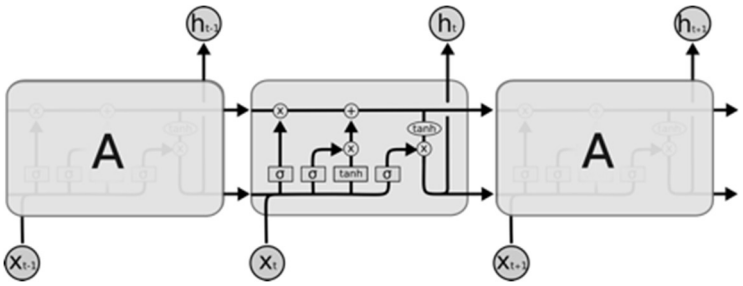
Derin İnanç Ağları (Deep Belief Network-DBN) genel olarak birden fazla gizli düğüm katmanından oluşan, düğümler yerine katmanlar arasında bağlantıların yer aldığı bir derin sinir ağı türüdür. Temel bir DBN mimarisinin yapısı Şekil 8.5'te verilmiştir.



Şekil 8.5. DBN model şeması (Dai et al., 2020)

1.5. Uzun-Kısa Dönem Hafıza Tekniği

Sepp Hochreiter ve Juergen Schmidhuber 1997 yılında vanishing gradient problemini çözmek için LSTM'i geliştirdiler. Daha sonra birçok kişinin katkısıyla düzenlenen ve popülerleşen LSTM şu anda geniş bir kullanım alanına sahiptir. LSTM tekniği, yeni bir kanal açma yoluyla sabit bir hata değeri sağlayarak recurrent ağların öğrenme adımlarının devam edebilmesini sağlamaktadır. Şekil 8.6'da LSTM tekniğinin ağ yapısı verilmiştir.

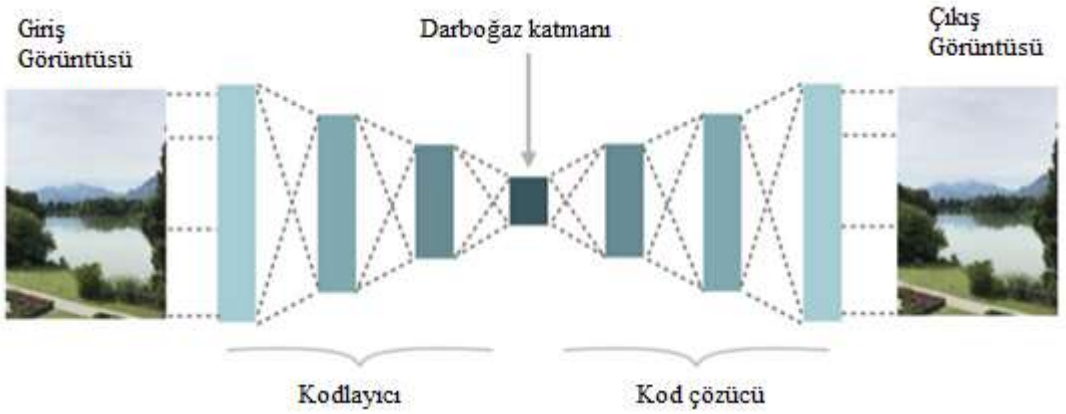


Şekil 8.6. LSTM Network Yapısı (Web Kaynağı 8.5)

LSTM yapısında tekrar eden modülün farkı tek bir yapay ağ katmanı yerine, özel bir şekilde bağlı kapı olarak adlandırılan dört katmandan oluşmasıdır. Normal akışın dışında dışarıdan bilgi alan bir yapıdır. Bu bilgiler depolanabilir, hücreye yazılabilir, okunabilir. Hücre neyi depolayacağına, ne zaman okumasına, yazmasına veya silmesine izin vereceğine kapılar sayesinde karar verir. Bu kapılarda bir ağ yapısı ve aktivasyon fonksiyonu bulunmaktadır. Aynı nöronlarda olduğu gibi gelen bilgiyi ağırlığına göre geçirir veya durdurur.

1.6. Oto kodlayıcı Tekniği

Oto kodlayıcı tekniği (auto encoder) derin öğrenme yöntemleri alanındaki en popüler modellerden birisidir. Oto kodlayıcı tekniği veriyi koda dönüştürerek otomatik olarak öğrenmeyi sağlar. Encoder (kodlayıcı) ve dekoder (kod çözücü) olarak iki kısımdan oluşur. Eğitim aşamasında tek bir modelmiş gibi beraber eğitilir (Şekil 8.7). Örneğin göndericiye kodlayıcı, alıcıya ise kod çözücü olarak çalışılarak kişisel görüntülerimizin güvenli ve yüksek doğruluk oranında taşınması sağlanabilir.



Şekil 8.7. Oto kodlayıcı tekniği şematiği (Web Kaynağı 8.6)

Biliyor musunuz?

Hayatımızı kolaylaştıran onlarca mobil uygulamaya ek olarak bazı uygulamaları çoğu zaman eğlenmek için kullanıyoruz. Yayına alındığı ilk 2 haftada 1 milyon iPhone kullanıcısıyla buluşan FaceApp uygulaması yapay sinir ağlarını makine öğrenmesi metodu ile bağdaştırıyor ve bu filtrelerle oldukça gerçek sonuçlar elde etmenizi sağlıyor. Facebook'un yüz tanıma özelliği, Apple cihazlarda bulunan Siri ve FaceApp gibi günlük hayatımızda sık sık kullandığımız uygulamalarını başarıya götüren Derin Öğrenme (Deep Learning) olduğunu biliyor muydunuz?

Ayrıca Snapchat, FaceApp, Instagram gibi yüz tanımaya dayalı katmanlar kullanan, Google görseller gibi Derin Öğrenme ile desteklenen uygulamalar önümüzdeki yıllarda içerik üretme konusunda işletmeler için oldukça büyük faydalar sağlayacak gibi görünüyor (Web Kaynağı 8.7)

2. TASARLA

CIFAR-10 veri seti, 10 sınıftan ve 3 kanalda 32 x 32 pikselden oluşan Python içerisinde hazır olan bir veri sınıfıdır. Rastgele veri setine ait örnek görsel Şekil 8.8'de gösterilmiştir.

Eğitmene Not

Eğitmen, CIFAR-10 veri setinde yer alan sınıflara (İngilizce ifadeler) ait Türkçe karşılıklarını öğrencilere aktarır.

Şekil 8.8. Veri setindeki örnek görüntü

VERİ SETİ İÇİN İNDİRME LİNKİ

https://www.cs.toronto.edu/~kriz/cifar.html

3. HAREKETE GEÇ

Öğrenciler, verilen veri seti dosyası için **"ayrimci" ve "üretici"** isimli derin öğrenme modeli tanımlayarak CIFAR10 görüntüleri üzerinden 10 sınıfa ait derin görüntüler oluşturmaya çalışır. Şekil 8.9'da gösterildiği gibi derin öğrenme eğitiminde kullanılacak kütüphane komutlarını yazar (Kodun tamamı ilgili Github platformundaki Hafta8 klasörü altında, H8_derinogrenme_gan-cifar.py dosyası ile sunulmuş durumdadır.).

1-15 nolu kod satırında çalışma için gerekli kütüphaneleri çağırır. Burada 6-14 numaralı kod satırında derin öğrenme tekniği ile ilgili kütüphaneleri çağırır. 5 numaralı kod satırında ise CIFAR10 veri setini yüklemek için kullanır.

```
1   from numpy import zeros
2   from numpy import ones
3   from numpy.random import randn
4   from numpy.random import randint
5   from keras.datasets.cifar10 import load_data
6   from keras.optimizers import Adam
7   from keras.models import Sequential
8   from keras.layers import Dense
9   from keras.layers import Reshape
10  from keras.layers import Flatten
11  from keras.layers import Conv2D
12  from keras.layers import Conv2DTranspose
13  from keras.layers import LeakyReLU
14  from keras.layers import Dropout
15  from matplotlib import pyplot
```

Şekil 8.9. Kütüphanelerin kod satırı

Öğrenciler, Şekil 8.10'da gösterilen 18-34 kod satırları arasında ayrımcı model tanımlama işlemini gerçekleştirir.

18 kod satırında; **"ayrimci"** isimli fonksiyonu oluşturarak fonksiyonun girdi değişkeni olarak 32x32x3 şeklinde görüntü boyutunu belirler.

19. kod satırında boş bir model oluşturur.

20.-27. kod satırlarında ayrımcı model ile aşağı yönlü örnekleme için **"Conv2D" ve "LeakyReLU"** olarak katmanlar ekleyerek modelin katmanlarını oluşturur.

28. kod satırı düzleştirme katmanıdır.

29. kod satırında %40 oranında veri azaltma (önemi az olan) işlemi gerçekleştirilmiştir.

30. kod satırında **"sigmoid"** aktivasyon fonksiyonu ile full_connected katmanı oluşturulmuştur.

31. kod satırında **"Adam"** optimizasyon yöntemiyle öğrenme oranı (lr=0,0002) ile optimizasyon yöntemi tanımlanmıştır.

32. kod satırında model derlenmiştir.

33. kod satırında derlenen model geriye gönderilmiştir.

```
17  # ayrımcıyı model tanımlama
18  def ayrimci(in_shape=(32,32,3)):
19      model = Sequential()
20      model.add(Conv2D(64, (3,3), padding='same', input_shape=in_shape))
21      model.add(LeakyReLU(alpha=0.2))
22      model.add(Conv2D(128, (3,3), strides=(2,2), padding='same'))
23      model.add(LeakyReLU(alpha=0.2))
24      model.add(Conv2D(128, (3,3), strides=(2,2), padding='same'))
25      model.add(LeakyReLU(alpha=0.2))
26      model.add(Conv2D(256, (3,3), strides=(2,2), padding='same'))
27      model.add(LeakyReLU(alpha=0.2))
28      model.add(Flatten())
29      model.add(Dropout(0.4))
30      model.add(Dense(1, activation='sigmoid'))
31      opt = Adam(lr=0.0002, beta_1=0.5)
32      model.compile(loss='binary_crossentropy', optimizer=opt, metrics=['accuracy'])
33      return model
34
```

Şekil 8.10. Ayrımcı modeli tanımlamak için kod satırları

Öğrenciler, Şekil 8.11'de gösterilen 36-49 kod satırları arasında üretici model tanımlama işlemini gerçekleştirir.

36. kod satırında; **"uretici"** isimli fonksiyonu oluşturarak "**son_boyut"** girdi değişkeni olarak alınmıştır.

37. kod satırında boş bir model oluşturur.

38. kod satırında "**n_nodes"** isiminde bir değişken tanımlanmıştır.

39. kod satırında full_connected katmanı oluşturulmuştur.

40-48. kod satırlarında üretici model ile yukarı yönlü örnekleme için **"Conv2D" ve "LeakyReLU"** olarak katmanlar ekleyerek modelin katmanlarını oluşturur.

49. kod satırında derlenen model geriye gönderilmiştir.

```
35  # üretici model tanımlama
36  def uretici(son_boyut):
37      model = Sequential()
38      n_nodes = 256 * 4 * 4
39      model.add(Dense(n_nodes, input_dim=son_boyut))
40      model.add(LeakyReLU(alpha=0.2))
41      model.add(Reshape((4, 4, 256)))
42      model.add(Conv2DTranspose(128, (4,4), strides=(2,2), padding='same'))
43      model.add(LeakyReLU(alpha=0.2))
44      model.add(Conv2DTranspose(128, (4,4), strides=(2,2), padding='same'))
45      model.add(LeakyReLU(alpha=0.2))
46      model.add(Conv2DTranspose(128, (4,4), strides=(2,2), padding='same'))
47      model.add(LeakyReLU(alpha=0.2))
48      model.add(Conv2D(3, (3,3), activation='tanh', padding='same'))
49      return model
```

Şekil 8.11. Üretici modeli tanımlamak için kod satırları

Öğrenciler, Şekil 8.12'de gösterilen 52-59 kod satırları arasında üretici ile ayrımcı modelleri birleştirilerek GANs modelini oluşturur.

52. kod satırında; **"GANs"** isimli fonksiyonu oluşturarak **"g_model" ve "d_model"** girdi değişkenleri olarak alınmıştır.

53. kod satırında d_model değişkenin eğitilebilir özelliği kapatılmıştır.
54. kod satırında boş bir model oluşturur.
55. ve 56. kod satırlarında **"g_model" ve "d_model"** değişkenlerini modele ekler.
57. kod satırında **"Adam"** optimizasyon yöntemiyle öğrenme oranı (lr=0,0002) ile optimizasyon yöntemi tanımlanmıştır.
58. kod satırında model derlenmiştir.
59. kod satırında derlenen model geriye gönderilmiştir.

```
51  # üretici ve ayrımcı modellerin birleştirme
52  def GANs(g_model, d_model):
53      d_model.trainable = False
54      model = Sequential()
55      model.add(g_model)
56      model.add(d_model)
57      opt = Adam(lr=0.0002, beta_1=0.5)
58      model.compile(loss='binary_crossentropy', optimizer=opt)
59      return model
60
```

Şekil 8.12. Üretici ve ayrımcı modellerin birleştirilmesi kod satırları

Öğrenciler, Şekil 8.13'te gösterilen 62-66 kod satırları arasında CIFAR10 veri setinin yükleme işlemini gerçekleştirmiştir.

```
60
61  # veri seti yükleme
62  def Veri_set_yukleme():
63      (trainX, _), (_, _) = load_data()
64      X = trainX.astype('float32')
65      X = (X - 127.5) / 127.5 #[-1,1]
66      return X
67
```

Şekil 8.13. Veri seti yükleme kod satırları

Öğrenciler, Şekil 8.14'te gösterilen 69-73 kod satırları arasında CIFAR10 veri setinden örnek veriler seçmek için gerekli kod satırlarını yazar.

```
67
68  # örnek veri seçme
69  def ornek_veri(dataset, n_samples):
70      ix = randint(0, dataset.shape[0], n_samples)
71      X = dataset[ix]
72      y = ones((n_samples, 1))
73      return X, y
74
```

Şekil 8.14. Örnek veri seçme kod satırları

Öğrenciler, Şekil 8.15'te gösterilen 76-79 kod satırları arasında görüntü boyutunda rastgele gizli veriler oluşturmak için gerekli kod satırlarını yazar.

```
74
75  # gizli noktalar oluşturmak
76  def noktalar_olusturma(son_boyut, n_samples):
77      x_input = randn(son_boyut * n_samples)
78      x_input = x_input.reshape(n_samples, son_boyut)
79      return x_input
80
```

Şekil 8.15. Gizli noktalar oluşturma kod satırları

4. YÜRÜT

Öğrenciler, Şekil 8.16'da gösterilen 82-86 kod satırları arasında sahte nesneler oluşturarak gerçek görüntüler ile karşılaştırır.

```
80
81  #sahte nesneler oluşturma
82  def sahte_nesne_olusturma(g_model, son_boyut, n_samples):
83      x_input = noktalar_olusturma(son_boyut, n_samples)
84      X = g_model.predict(x_input)
85      y = zeros((n_samples, 1))
86      return X, y
87
```

Şekil 8.16. Sahte nesneler oluşturma kod satırları

Öğrenciler, 89-97 kod satırları arasında sahte nesnelerin çizilmesi için gerekli kodları yazar:

```
87
88  # çizim
89  def cizim(examples, epoch, n=7):
90      examples = (examples + 1) / 2.0
91      for i in range(n * n):
92          pyplot.subplot(n, n, 1 + i)
93          pyplot.axis('off')
94          pyplot.imshow(examples[i])
95      filename = 'oluşturulan_ornek %03d.png' % (epoch+1)
96      pyplot.savefig(filename)
97      pyplot.close()
98
```

Şekil 8.17. Çizim oluşturma kod satırları

5. KARAR VER

Öğrenciler, Şekil 8.18'de gösterilen 100-108 kod satırları arasında oluşturmuş oldukları GANs modeli değerlendirerek, oluşturdukları bu modeli sisteme kayıt ederek doğruluk sonuçlarını belirlerler.

```
99  # model değerlendirme
100 def degerlendir(epoch, g_model, d_model, dataset, son_boyut, n_samples=150):
101     X_gercek, y_gercek = ornek_veri(dataset, n_samples)
102     _, acc_gercek = d_model.evaluate(X_gercek, y_gercek, verbose=0)
103     x_sahte, y_sahte = sahte_nesne_olusturma(g_model, son_boyut, n_samples)
104     _, acc_sahte = d_model.evaluate(x_sahte, y_sahte, verbose=0)
105     print('gerçek doğruluğu %.0f%%, sahte doğruluğu: %.0f%%' % (acc_gercek*100, acc_sahte*100))
106     cizim(x_sahte, epoch)
107     filename = 'oluşturulan_model_%03d.h5' % (epoch+1)
108     g_model.save(filename)
109
```

Şekil 8.18. Model değerlendirme kod satırları

Öğrenciler, Şekil 8.19'da gösterilen 111-126 kod satırları arasında oluşturmuş olduğu GANs modelini eğitmek için oluşturulan fonksiyonları tek tek çağırarak kayıp değerlerine göre modeli değerlendirir.

```
110  # model eğitimi
111  def train(g_model, d_model, gan_model, dataset, son_boyut, n_epochs=30, n_batch=128):
112      bat_per_epo = int(dataset.shape[0] / n_batch)
113      half_batch = int(n_batch / 2)
114      for i in range(n_epochs):
115          for j in range(bat_per_epo):
116              X_gercek, y_gercek = ornek_veri(dataset, half_batch)
117              d_loss1, _ = d_model.train_on_batch(X_gercek, y_gercek)
118              X_sahte, y_sahte = sahte_nesne_olusturma(g_model, son_boyut, half_batch)
119              d_loss2, _ = d_model.train_on_batch(X_sahte, y_sahte)
120              X_gan = noktalar_olusturma(son_boyut, n_batch)
121              y_gan = ones((n_batch, 1))
122              g_loss = gan_model.train_on_batch(X_gan, y_gan)
123              print('>%d, %d/%d, dr=%.3f, ds=%.3f g=%.3f' %
124                    (i+1, j+1, bat_per_epo, d_loss1, d_loss2, g_loss))
125          if (i+1) % 5 == 0:
126              degerlendir(i, g_model, d_model, dataset, son_boyut)
127
```

Şekil 8.19. Model eğitimi kod satırları

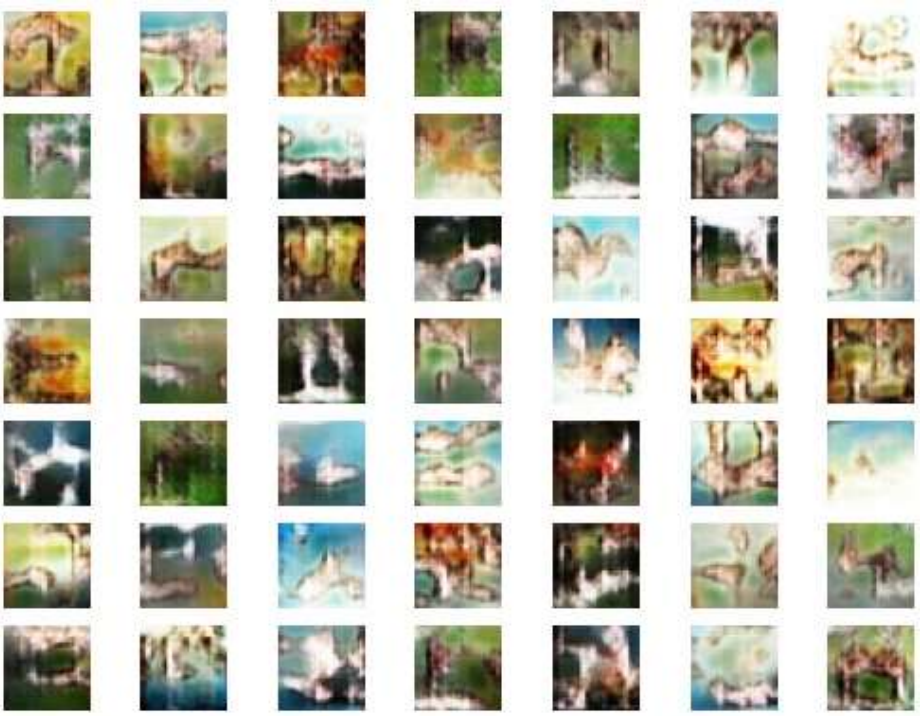
İlgili kodlar sonrasında, "**ayrimci", "üretici", "GANs", "Veri_set_yukleme"** fonksiyonları çağırılır:

```
127
128  son_boyut = 100
129  d_model = ayrimci()
130  g_model = uretici(son_boyut)
131  gan_model = GANs(g_model, d_model)
132  dataset = Veri_set_yukleme()
133  train(g_model, d_model, gan_model, dataset, son_boyut)
```

Şekil 8.20. Fonksiyonların çağrılması kod satırları

Şekil 8.21. Model eğitimi esnasında elde edilen örnek görüntüler (epoch_sayısı=5)

Şekil 8.22. Model eğitimi esnasında elde edilen örnek görüntüler (epoch_sayısı=10)



Şekil 8.23. Model eğitimi esnasında elde edilen örnek görüntüler (epoch_sayısı=15)

Eğitmene Not

Eğitmen, öğrencilere ilgili kod satırlarını yazdıktan sonra eğitim süresinin çok uzun olduğunu ifade eder. Öğrenciler kod yazımını tamamladıktan sonra yaklaşık her bir eğitimin 15 saat sürdüğünü göz önünde bulundurarak eğitim sürecini evde yapmalarını belirtir.

Eğitmene Not

Eğitmen, öğrencilerle 1. Hafta'da Python yapay zekâ kütüphaneleri içerisinde hazır olarak bulunan **"İRİS ÇİÇEĞİ"** uygulamasını gerçekleştirirken, 3. Hafta'da Python yapay zekâ kütüphanelerinde hazır bir veri seti olarak bulunmayan açık erişimli internet sitesinden alınan **"EKG sinyali"** mitbih_train.csv ve mitbih_test.csv dosyaları üzerinden iki farklı yapay zekâ uygulaması gerçekleştirilmiştir. Böylece hem Python içerisinde gömülü veri seti ile gömülü olmayan veri seti üzerinde uygulama yapma imkânı bulmuşlardır. 8. Haftada ise gömülü hazır **büyük veri tabanlı bir veri seti** ile görüntülerin sınıflandırma işlemi gerçekleştirilmişlerdir.

6. UYGULAMANIN PYTHON KODLARI

```
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
import tensorflow as tf
from tqdm import tqdm
import os
from sklearn.metrics import classification_report,confusion_matrix
from PIL import Image
from tensorflow.keras.applications import ResNet152V2
from sklearn.model_selection import train_test_split
from warnings import filterwarnings
filterwarnings('ignore')
labels = ['Cat','Dog']
image_size = 64
X_train = []
y_train = []
for i in labels:
  folderPath = os.path.join('D:/Yapay Zekâ/Yapay Zekâ Lise/Hafta
8/archive/PetImages/'+i)
   klasor=folderPath+"/"
   for j in tqdm(os.listdir(folderPath)):
      yol=os.path.join(klasor,j)
      try:
         img=np.array(Image.open(yol).convert('RGB').resize((image_size, image_size),
Image.ANTIALIAS))
         X_train.append(img)
         y_train.append(i)
      except:
         continue
```

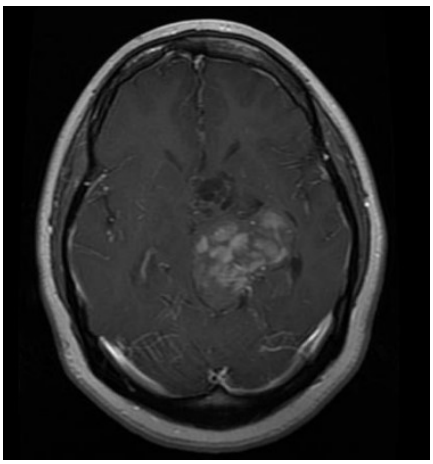
```
X_train=np.array(X_train)
y_train=np.array(y_train)
X_train,X_test,y_train,y_test = train_test_split(X_train,y_train,
test_size=0.1,random_state=101)
def kodlama (y_t):
    y_new = []
    for i in y_t:
        y_new.append(labels.index(i))
    return tf.keras.utils.to_categorical(y_new)
y_train=kodlama(y_train)
y_test=kodlama(y_test)
resnet = ResNet152V2(weights='imagenet',
include_top=False,input_shape=(image_size,image_size,3))
model = resnet.output
model = tf.keras.layers.GlobalAveragePooling2D()(model)
model = tf.keras.layers.Dropout(rate=0.5)(model)
model = tf.keras.layers.Dense(2,activation='softmax')(model)
model = tf.keras.models.Model(inputs=resnet.input, outputs = model)
model.summary()
model.compile(loss='categorical_crossentropy',optimizer = 'Adam', metrics= ['accuracy'])
model.fit(X_train,y_train,validation_split=0.1, epochs =5, verbose=1, batch_size=64)

pred = model.predict(X_test)
pred = np.argmax(pred,axis=1)
y_test_new = np.argmax(y_test,axis=1)

print(classification_report(y_test_new,pred))
fig,ax=plt.subplots(1,1,figsize=(14,7))
sns.heatmap(confusion_matrix(y_test_new,pred),ax=ax,xticklabels=labels,yticklabels=labels,a
nnot=True)
plt.show()
```

Yapay Zekâ'da Derin Öğrenme Çağı!

7. İLAVE ETKİNLİK

Beyin tümörü, çocuklar ve yetişkinler arasında ölümcül hastalıklardan biri olarak kabul edilir. Her yıl yaklaşık 11.700 kişiye beyin tümörü teşhisi konulmaktadır. Beyin tümörü olan kişiler için genel olarak ortalama 5 yıllık bir yaşam süresi öngörülmektedir. Bu öngörü de erkeklerin yaklaşık yüzde 34'ü, kadınların ise yüzde 36'sı için yapılabilmektedir. Hastaların yaşam beklentisini iyileştirmek için doğru tedavi, planlama ve doğru teşhis uygulanmalıdır. Beyin tümörlerini saptamak için en iyi teknik Manyetik Rezonans Görüntüleme (MRI) tekniğidir. Her ne kadar MRI görüntüleri radyolog tarafından incelense de beyin tümörlerinde yer alan karmaşıklık düzeyi ve özellikleri nedeniyle manuel muayene sık sık hatalara neden olabilmektedir.

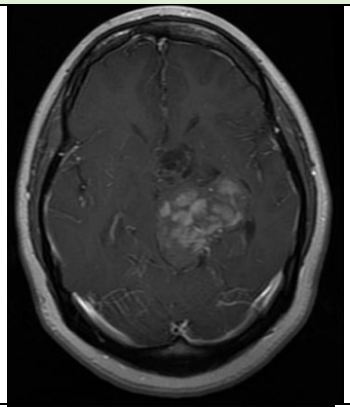
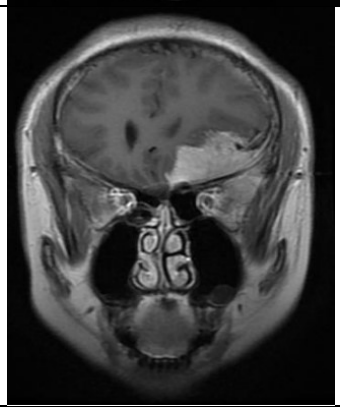
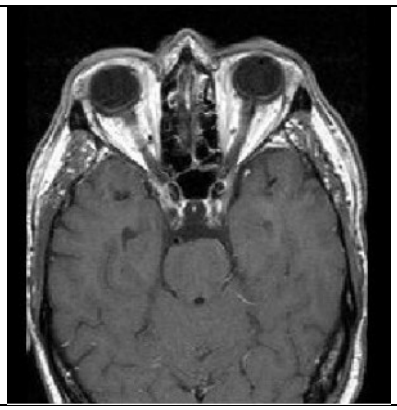
Derin öğrenme teknikleri kullanılarak insanlardan kaynaklanan hataları minimize etmek dünyanın her yerindeki doktorlara yardımcı olacaktır. Bu nedenle, açık erişimli internet sitesinden (Kaggle.com) alınan veri seti Konvolüsyonel Sinir Ağı olan ResNet152_V2 derin öğrenme yöntemi kullanılarak beyin tümörleri sınıflandırılmıştır (Şekil 8.24). İlgili uygulamanın kodu Github platformu Hafta8 klasöründe yer alan H8_derinogrenme_beyintumoru.py dosyası ile paylaşılmış durumdadır (Uygulamanın veri seti ilgili Kaggle linki altından indirilmelidir).

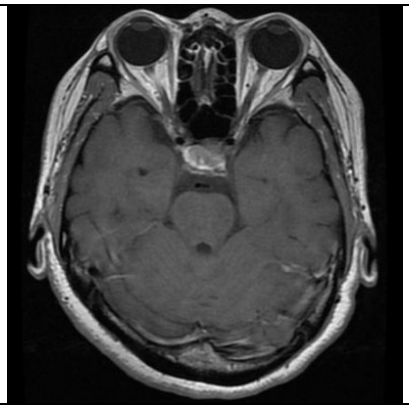


Şekil 8.24. MRI görüntüsü

Öğrenciler Çizelge 8.1'de verilen ResNet152_V2 derin öğrenme tekniği ile beyin tümörü tespiti için giriş çıkış parametresini belirler.

Çizelge 8.1. ResNet152_V2 derin öğrenme tekniği için giriş ve çıkış parametreleri

Veri Sayısı	GİRİŞ PARAMETRESİ	ÇIKIŞ PARAMETRESİ
	MRI Görüntüsü	Tümör Sınıfı
1		Glioma_tumor
150		Meningioma_tumor
…	…	…
1440		No_tumor
…	…	…

3264			Pituitary_tumor

Bu bölümde hastalara ait 3264 adet veri setinde (Web Kaynağı 8.8) MRI görüntüsü giriş parametrelerine göre beyin tümörünün olup olmadığını ve türünü belirlemede ResNet152_V2 derin öğrenme tekniği ile sınıflandırılma yapılması amaçlanmıştır.

VERİ SETİ İÇİN İNDİRME LİNKİ

https://www.kaggle.com/sartajbhuvaji/brain-tumor-classification-mri

PYTHON KODLARI:

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
import cv2
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tqdm import tqdm
import os
from sklearn.utils import shuffle
from sklearn.model_selection import train_test_split
from tensorflow.keras.applications import EfficientNetB0
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau, TensorBoard,
ModelCheckpoint
from sklearn.metrics import classification_report,confusion_matrix
import ipywidgets as widgets
import io
from PIL import Image
from IPython.display import display,clear_output
from warnings import filterwarnings
from PIL import Image

labels = ['glioma_tumor','no_tumor','meningioma_tumor','pituitary_tumor']
```

```
X_train = []
y_train = []
image_size = 128
for i in labels:
  folderPath = os.path.join('D:/Yapay Zekâ/Yapay Zekâ Lise/Hafta
8/archive/','Training/',i)
   klasor=folderPath+"/"
   for j in tqdm(os.listdir(folderPath)):
      yol=os.path.join(klasor,j)
      try:
         img = Image.open(yol)
         img = img.resize((image_size, image_size))
         X_train.append(np.array(img))
         y_train.append(i)
      except:
         continue

for i in labels:
  folderPath = os.path.join('D:/Yapay Zekâ/Yapay Zekâ Lise/Hafta
8/archive/','Testing/',i)
   klasor=folderPath+"/"
   for j in tqdm(os.listdir(folderPath)):
      yol=os.path.join(klasor,j)
      try:
         img =  Image.open(yol)
         img = img.resize((image_size, image_size))
         X_train.append(np.array(img))
         y_train.append(i)
      except:
         continue

X_train = np.array(X_train)
y_train = np.array(y_train)

X_train, y_train = shuffle(X_train,y_train, random_state=101)

datagen = ImageDataGenerator(
   rotation_range=30,
   width_shift_range=0.1,
   height_shift_range=0.1,
   zoom_range=0.2,
   horizontal_flip=True)

datagen.fit(X_train)
print(X_train.shape)
```

```
X_train,X_test,y_train,y_test = train_test_split(X_train,y_train,
test_size=0.1,random_state=101)

y_train_new = []
for i in y_train:
    y_train_new.append(labels.index(i))
y_train = y_train_new
y_train = tf.keras.utils.to_categorical(y_train)

y_test_new = []
for i in y_test:
    y_test_new.append(labels.index(i))
y_test = y_test_new
y_test = tf.keras.utils.to_categorical(y_test)

from tensorflow.keras.applications import ResNet152V2
resnet = ResNet152V2(weights='imagenet',
include_top=False,input_shape=(image_size,image_size,3))

model = resnet.output
model = tf.keras.layers.GlobalAveragePooling2D()(model)
model = tf.keras.layers.Dropout(rate=0.5)(model)
model = tf.keras.layers.Dense(4,activation='softmax')(model)
model = tf.keras.models.Model(inputs=resnet.input, outputs = model)
model.summary()
model.compile(loss='categorical_crossentropy',optimizer = 'Adam', metrics= ['accuracy'])

tensorboard = TensorBoard(log_dir = 'logs')
checkpoint =
ModelCheckpoint("resnet152v1.h5",monitor="val_accuracy",save_best_only=True,mode="auto
",verbose=1)
reduce_lr = ReduceLROnPlateau(monitor = 'val_accuracy', factor = 0.4, patience = 2,
                        min_delta = 0.001,mode='auto',verbose=1)

history = model.fit(X_train,y_train,validation_split=0.1, epochs =5, verbose=1, batch_size=64,
                callbacks=[tensorboard,checkpoint,reduce_lr])

filterwarnings('ignore')
epochs = [i for i in range(25)]
fig, ax = plt.subplots(1,2,figsize=(14,7))
train_acc = history.history['accuracy']
train_loss = history.history['loss']
val_acc = history.history['val_accuracy']
val_loss = history.history['val_loss']

colors_dark = ["#1F1F1F", "#313131", '#636363', '#AEAEAE', '#DADADA']
colors_red = ["#331313", "#582626", '#9E1717', '#D35151', '#E9B4B4']
```

```
colors_green = ['#01411C','#4B6F44','#4F7942','#74C365','#D0F0C0']

sns.palplot(colors_dark)
sns.palplot(colors_green)
sns.palplot(colors_red)

fig.text(s='Epochs vs. Training and Validation Accuracy/Loss',size=18,fontweight='bold',
         fontname='monospace',color=colors_dark[1],y=1,x=0.28,alpha=0.8)

sns.despine()
ax[0].plot(epochs, train_acc,
marker='o',markerfacecolor=colors_green[2],color=colors_green[3],
        label = 'Training Accuracy')
ax[0].plot(epochs, val_acc, marker='o',markerfacecolor=colors_red[2],color=colors_red[3],
        label = 'Validation Accuracy')
ax[0].legend(frameon=False)
ax[0].set_xlabel('Epochs')
ax[0].set_ylabel('Accuracy')

sns.despine()
ax[1].plot(epochs, train_loss,
marker='o',markerfacecolor=colors_green[2],color=colors_green[3],
        label ='Training Loss')
ax[1].plot(epochs, val_loss, marker='o',markerfacecolor=colors_red[2],color=colors_red[3],
        label = 'Validation Loss')
ax[1].legend(frameon=False)
ax[1].set_xlabel('Epochs')
ax[1].set_ylabel('Training & Validation Loss')

fig.show()
pred = model.predict(X_test)
pred = np.argmax(pred,axis=1)
y_test_new = np.argmax(y_test,axis=1)

print(classification_report(y_test_new,pred))

fig,ax=plt.subplots(1,1,figsize=(14,7))
sns.heatmap(confusion_matrix(y_test_new,pred),ax=ax,xticklabels=labels,yticklabels=labels,a
nnot=True,
        cmap=colors_green[::-1],alpha=0.7,linewidths=2,linecolor=colors_dark[3])
fig.text(s='Heatmap of the Confusion Matrix',size=18,fontweight='bold',
         fontname='monospace',color=colors_dark[1],y=0.92,x=0.28,alpha=0.8)
plt.show()
```

Eğitmene Not

Eğitmen, bu haftaki eğitime yönelik öğrencilerin ilgisini arttırmak ve yeni öğrenilen bilgileri önceden öğrenilen bilgiler ile karşılaştırmalarını sağlamak için aşağıda maddeler halinde verilen soruları tartışır.

- İlgili haftada ele alınan yapay zekâ uygulamalarına uygun örnek veri setleri neler olabilir?
- İlgili haftada ele alınan veri setlerinin yapay zekâ teknik ve kütüphanelerinde ne derecede başarım etkisi olmuştur?
- İlgili haftada öğretilen yapay zekâ tekniği ile uyumlu ve uyumsuz olabilecek açık erişimli veri seti çeşitleri sunan (kaagle, github vs.) internet sitelerinden elde edilecek farklı veri çeşitleri ile yapay zekâ tekniği uyumluluğunu sorar.

Eğitmene Not

Eğitmen, öğrencilerin 7. ve 8. Hafta içerisinde elde ettikleri bilgi-becerileri değerlendirmek, aktif katılımlarını sağlamak için 'Kahoot' uygulamasını ya da benzeri bir Web 2.0 uygulamasını kullanarak 'bilgi yarışması' düzenleyebilir.

7. ve 8. Hafta bağlamında sorulabilecek sorular; zeki optimizasyon kavramının ve çözümlerinin temelleri, farklı zeki optimizasyon teknikleri ile uygulamalar, derin öğrenme kavramının temelleri, farklı derin öğrenme tekniklerinin temelleri ve ilgili çözüm yaklaşımları yönünde olabilir.

Kaynakça

Doğan, F., & Türkoğlu, İ. (2019). Derin öğrenme modelleri ve uygulama alanlarına ilişkin bir derleme. Dicle Üniversitesi Mühendislik Fakültesi Mühendislik Dergisi, 10(2), 409-445.

Dai, X., Cheng, J., Gao, Y., Guo, S., Yang, X., Xu, X., & Cen, Y. (2020). Deep belief network for feature extraction of urban artificial targets. Mathematical Problems in Engineering, 2020, Article ID 2387823.

Web Kaynağı 8.1: https://kim.hfg-karlsruhe.de/neural-network-chart/

Web Kaynağı 8.2: https://medium.com/@tuncerergin/convolutional-neural-network-convnet-yada-cnn-nedir-nasil-calisir-97a0f5d34cad

Web Kaynağı 8.3: https://www.bezelyedergi.net/post/biyoinformatik-derin-%C3%B6%C4%9Frenme-deep-learning

Web Kaynağı 8.4: https://paperswithcode.com/method/u-net-gan

Web Kaynağı 8.5: https://medium.com/@hamzaerguder/recurrent-neural-network-nedir-bdd3d0839120

Web Kaynağı 8.6: https://www.bezelyedergi.net/post/biyoinformatik-derin-%C3%B6%C4%9Frenme-deep-learning

Web Kaynağı 8.7: https://blog.adresgezgini.com/faceapp-uygulamasiyla-yaslandik

Web Kaynağı 8.8: https://www.kaggle.com/sartajbhuvaji/brain-tumor-classification-mri

Proje Yarışması

Yapay Zeka

Final Proje Yarışması/
Emisyon Sınıflarının Tahmini

Eğitmene Not

Eğitmen öğrencilere önceki haftalarda genel hatlarıyla anlattığı; eğitim sonu proje yarışması hakkında detaylı bilgilendirmelerde bulunur. Eğitim süreci sonuna doğru, yarışmadaki probleme ilişkin olarak aşağıda sunulan bilgileri öğrenciler ile birlikte inceler ve irdeler.

Eğitmen aynı zamanda ilerleyen sayfalarda sunulan kuralları ve genel süreci dikkate alarak yarışma sürecinin gerçekleştirilip değerlendirilmesini sağlar. Eğitmenin dikkate alacağı kurallar ve değerlendirme dokümanları ayrıca sunulacaktır.

Sevgili Deneyap Öğrencimiz,

İklim değişikliği bütün dünyayı derinden etkileyen ve 20. yüzyıldan bu yana farklı sonuçlarla kendini hissettiren önemli bir küresel sorundur. İklim değişikliği sebepli sorunlar arasında anormal sıcaklıkların ve mevsim geçişlerinin görülmesi, doğal afetlerin artması, buzulların erimesi, ekosistemlerde dengelerin bozulması ve insanlarda kitlesel sağlık problemlerinin baş göstermesi yer almaktadır.

İklim değişikliği sorununa farklı insan faaliyetleri sebep olmakla birlikte özellikle küresel ısınma üzerindeki etkisi büyük olan sera gazı emisyonu, söz konusu sorunlar arasında oldukça kritik bir noktada yer almaktadır. Tıpkı diğer iklim değişikliği sorunları gibi sera gazı emisyonu sorununa da ülkemizdeki etkin çalışmalarla çözümler üretilmeye çalışılmaktadır.

Yapay Zekâ ile Tahmin Sistemi Tasarlayalım!

Deneyap Teknoloji Atölyeleri – Yapay Zekâ Dersi Final Projesi kapsamında, Birleşmiş Milletler tarafından sunulan, farklı ülkelerin 1990-2017 yılları arasındaki emisyon sınıflarını içeren ***'International Greenhouse Gas Emissions'* veri seti** üzerinde analizler yapan en başarılı yapay zekâ programını, derste öğrendikleriniz eşliğinde **Python kodlama dili** ile geliştirmeniz istenmektedir.

Veri setini buraya **tıklayarak** ulaşacağınız klasörden indirebilirsiniz

---

Yarışma Kuralları

- Yarışma sadece Python kodu ile ders süreçlerinde anlatılan yapay zekâ tekniklerinin/yöntemlerinin kullanımını içermektedir. Farklı kodlama ortamları ve tekniklerle yazılan kodlar diskalifiye sebebidir.
- Yazılacak kod içerisinde sadece bir yapay zekâ tekniği kullanılması beklenmektedir. Birden fazla teknik kodu olması durumunda yukarıdan aşağıya doğru yazılan ilk teknik dikkate alınacaktır ancak başka teknikleri kullanmak da puan kazandıracaktır.
- Yarışma süreci 7. hafta sonundan 8. hafta dersi **2 gün öncesine kadar** sürmektedir. Kodlarınızı istediğiniz gibi düzenleyebilirsiniz. Ancak **Değerlendirme** başlığı altındaki kriterlere dikkat etmeniz yüksek puan almanızı sağlayacaktır.
- Hazırlamış olduğunuz programın **py** dosyasını **ekip_adı_il.py şeklinde isimlendirerek** (veri setine ayrıca gerek yoktur) **8. hafta dersinden 2 gün öncesine kadar** eğitmene e-posta yoluyla gönderebilirsiniz).
- **Veri seti *sınıflandırma problemine* tekabül etmektedir. Veri seti detaylarının anlaşılması ve kullanımı yarışmanın bir parçasıdır.**
- **Kodlayacağınız yapay zekâ algoritması veri setinin %50'si eğitim, %50'si test için kullanılmalıdır.**

Değerlendirme

Gönderilen her program (kod bütünü), önce aşağıdaki kıstaslara göre elde edilen **toplam puan** üzerinden değerlendirilecektir. **Her kriter 10 puandır.** Değerlendirmeler her ilin kendi içerisinde olacaktır:

Değerlendirme Kriteri	**Kriter Onayı**
Yapay zekâ kütüphanesinin çağırılması	**Evet/Hayır**
Veri setinin %50 eğitim %50 test şeklinde ayrılması	**Evet/Hayır**
Eğitim ve test veri setlerinin rastgele verilerle oluşturulması	**Evet/Hayır**
Yapay zekâ tekniğinin eğitilmesi	**Evet/Hayır**

Yapay zekâ tekniğinin (eğitimin) test edilmesi	**Evet/Hayır**
Bulguların ekrana doğruluk değeri ile yansıtılması	**Evet/Hayır**
Bulguların ekrana karmaşıklık matrisi ile yansıtılması	**Evet/Hayır**
Beş farklı çalıştırma sonunda ortalama %70'den yüksek doğruluk elde edilmesi	**Evet/Hayır**
Beş farklı çalıştırma sonunda ortalama %80'den yüksek doğruluk elde edilmesi	**Evet/Hayır**
Beş farklı çalıştırma sonunda ortalama %90'dan yüksek doğruluk elde edilmesi	**Evet/Hayır**
Aynı problem için başka yapay zekâ tekniklerinin (en az 2 farklı teknik) uygulanması	**Evet/Hayır**
Bütün yapay zekâ teknikleri ile elde edilen sonuçların karşılaştırılması	**Evet/Hayır**

Tüm öğrencilerimize başarılar dileriz.

DENEYAP TÜRKİYE

Yapay Zeka

YAPAY ZEKÂ

LİSE

DENEYAP TÜRKİYE