

BİLİM TÜRKİYE
YAPAY ZEKA YAZ OKULU
DERS PLANLARI KİTABI

Cilt – 1
(1. ve 2. Haftalar)

Hazırlayan
Doç. Dr. Utku KÖSE
Süleyman Demirel Üniversitesi, Türkiye
Kuzey Dakota Üniversitesi, ABD.

Haziran 2024

İçindekiler

İçindekiler	i
Önsöz	ii
Kitap Kullanım Kılavuzu	iii
1. HAFTA – 1. GÜN – 1. DERS: YAPAY ZEKA NEDİR?	1
1. HAFTA – 1. GÜN – 2. DERS: YAPAY ZEKA UYGULAMA ALANLARI VE YAKLAŞIMLARI	7
1. HAFTA – 1. GÜN – 3. DERS: YAPAY ZEKA’NIN ROBOTİK TEK. KULLANIMI.....	14
1. HAFTA – 2. GÜN – 1. DERS: YAPAY ZEKA İÇİN PYTHON GELİŞTİRME ORTAMLARI	21
1. HAFTA – 2. GÜN – 2. DERS: MAKİNE ÖĞR. YAKLAŞIMI VE MAKİNE ÖĞR. YÖNTEMLERİ ..	31
1. HAFTA – 2. GÜN – 3. DERS: VERİ, VERİ SETİ KAVRAMLARI NELERDİR?	38
1. HAFTA – 3. GÜN – 1. DERS: PANDAS PYTHON KÜTÜPHANESİ	45
1. HAFTA – 3. GÜN – 2. DERS: MATPLOTLIB PYTHON KÜTÜPHANESİ	58
1. HAFTA – 3. GÜN – 3. DERS: NUMPY PYTHON KÜTÜPHANESİ.....	70
1. HAFTA – 4. GÜN – 1. DERS: HUGGING FACE / KAGGLE TANITIMI	88
1. HAFTA – 4. GÜN – 2., 3. DERSLER: PYTHON ORTAMINDA VERİ SETİ OKUMA, MODEL KURMA VE EĞİTİM-TEST İŞLEMLERİ.....	104
1. HAFTA – 5. GÜN – 1., 2., 3. DERSLER: İRİS ÇİÇEĞİ VERİ SETİ İLE ÇİÇEK SINIFI TESPİT UYGULAMASI MODEL KURMA.....	115
2. HAFTA – 1. GÜN – 1. DERS: OUTLIER (AYKIRI DEĞER) TESPİTİ VE FİLTRELEME.....	126
2. HAFTA – 1. GÜN – 2., 3. DERSLER: OUTLIER TESPİTİ VE FİLTRELEME - PYTHON İLE ÖRNEK UYGULAMALAR	133
2. HAFTA – 2. GÜN – 1., 2., 3. DERSLER: BAYES ÖĞR. İLE ARAÇ SATIŞ TAHMİN ETME	149
2. HAFTA – 3. GÜN – 1., 2., 3. DERSLER: KARAR AĞACLARI İLE COVID-19 TAHMİN ETME...161	
2. HAFTA – 4. GÜN – 1., 2., 3. DERSLER: YAPAY SİNİR AĞLARI İLE BİNA ENERJİ VERİMLİLİĞİNİN TAHMİNİ	182
2. HAFTA – 5. GÜN – 1., 2., 3. DERSLER: ENSEMBLE MODELLER İLE ŞEKER HASTALIĞI TAHMİNİ.....	203

Önsöz

Yapay Zeka ortaya çıkışından bu yana teknolojik gelişmelerin odak noktasında yer alan yenilikçi bir problem çözme yolu olarak ilerleyişini sürdürmüştür. Öyle ki, 20. Yüzyıl'da olgunlaşma seyrini sürdürden zeki sistemler 21. Yüzyıl itibariyle günlük hayatın vazgeçilmez parçası olma yolunda ilerlemeler gerçekleştirmeye başlamıştır. Özellikle 2010'dan sonra Yapay Zeka alanında ortaya çıkan hızlı gelişimler, insanlığın sonunu getirebilecek zeki sistemlerin mümkün olup olmayacağı sorularını akla getirmiştir, her ne kadar halen bir distopya düzeyinde olan bu düşünce şu sıralar kendini insan ve makine (zeki sistemler) arasında mesleklerin yok oluşu, ekonomik dengelerin değişmesi ve teknolojinin toplumsal seviyede farklı negatif etkileri kapsamındaki tartışmalarda hayat bulmaya devam etmiştir. 2020'li yıllarda ChatGPT gibi araçlarla tanıdığımız Üretici Yapay Zeka teknikleri, insandan bağımsız kararlar oluşturabilecek Makine Öğrenmesi ve Derin Öğrenme yöntemlerinin ulaşığı üstün yetenekler itibarıyle ilgili tartışmaları iyiden iyiye şekillendirmektedir. Bununla birlikte bütün bu gelişmeler hızla ilerleyen ve değişen teknoloji dünyasında yetişmiş insan gücüne olan ihtiyacı da gözler önüne sermektedir.

Ülkemiz, 21. Yüzyıl gereksinimleriyle beraber milli ve yerli teknoloji üretiminde etkin atılımlar içerisinde bulunmakta, bu çerçevede Yapay Zeka dünyasındaki gelişmelere de önderlik edecek çalışmalar içerisinde bulunmaktadır. Bu bağlamda Bilim Türkiye Yapay Zeka Yaz Okulu programı, yeni nesil Yapay Zeka mühendislerinin ve uzmanlarının gelişmesinde anahtar rol oynayacak teorik ve uygulamalı eğitimlerin sunulması adına tasarlanmış, bu doğrultuda dört haftalık bir ders planı yapılması elde edilmiştir. Ders planlarının ilk iki haftasını içeren 1. Cilt, Yapay Zeka kavramının temellerinden Makine Öğrenmesi yaklaşımına dair teorik detaylara değinmekte, ayrıca Python tabanlı onde gelen kütüphanelerden destek almak suretiyle çeşitli Makine Öğrenmesi algoritmaları ve yöntemleri üzerinden öğrencilerin pratik uygulama geliştirme süreçlerine hızlı bir biçimde dahil olmasını sağlayacak uygulamalar içermektedir. Seçilen uygulamalar ve izlenen problem çözüm yaklaşımları güncel Yapay Zeka uygulamalarında ihtiyaç duyulan bilgi ve becerinin en iyi şekilde elde edilmesi için titizlikle tasarlanmış ve planlı bir öğretim-öğrenme yaklaşımıyla öğretmenlerin kullanımına sunulmuştur.

Bilim Türkiye Yapay Zeka Yaz Okulu'nun ülkemiz yerli ve milli Yapay Zeka teknolojilerinin geliştirilmesi noktasında gayretleriyle geleceği şekillendirecek öğrencilerimize ve öğretmenlerimize faydalı olması dileklerimizle.

Türkiye Teknoloji Takımı (T3) Vakfı ve Yazar

Kitap Kullanım Kılavuzu

Bilim Türkiye Yapay Zeka Yaz Okulu ders planlarını içeren bu kitap, öğretmenler ve öğrencilerin dersler bazında işlenecek konuları ve gerçekleştirilecek uygulamalar karşısındaki rollerini belirlemek adına yönlendirici açıklamalarla donatılmış durumdadır. Konuların haftalar ve dersler içerisindeki sırası ve düzeni, Yapay Zeka uygulamalarının öğrenciler tarafından bilgi ve beceri boyutunda kolay özümsermesi adına tasarlanmıştır. Dersler içerisinde sunulan açıklamalar öğretmenlere ve öğrencilere yol gösterici olmakla beraber aşamalı bir öğretim-öğrenim sürecini de desteklemektedir. Bununla birlikte öğretmenler dersler esnasında öğrencilerden gelen dönütler ya da hakim oldukları konularda öğrencilere aktarmak istedikleri teorik ya da uygulamalı olgular konusunda inisiyatif kullanabileceklerdir.

Sunulan her bir ders planı öncelikli olarak ders ya da derslerin açılış planı ile başlamaktadır. Bu planlar genel olarak **Ders Etiketleri** altında işlenecek *Konu*, hedef öğrenci kitlesinin *Sınıf* düzeyi, toplam *Süre*, öğrencilerden beklenen *Hazır Bulunuşluk* durumları ve işlenecek konuları temsil eden *Anahtar Kelimeler* ile desteklenmiştir. Ders Etiketleri ardından **Beceriler** başlığı altında öğrencilerden ders(ler) esnasında beklenen beceri durumları listelenmiştir. Öğretmenler ilgili beceri durumlarını dikkate alarak öğrencileri yönlendirebilecek, gözlemleyebileceklerdir. Yine ders(ler) sonunda öğrencilerin kazanması beklenen bilgi ve beceri durumları **Kazanımlar** başlığı altında, sınıf ortamında kullanılması beklenen materyaller ise **Ders Materyalleri** tablosu içerisinde listelenmiştir. **Öğretmen Hazırlık Çalışmaları** başlığı altında öğretmenlerin ders öncesi okuyabilecekleri ve izleyebilecekleri kaynaklar listelenmekte, **Atölye Hazırlık Çalışmaları** başlığı altında ise yine derslere hazırlık adına tavsiye edilen ön-hazırlık yaklaşımları ifade edilmektedir. Sunulan ders planı içerisinde kullanılan kaynaklar ise **Kaynakça** başlığı altında listelenmektedir.

Etkin bir öğretim-öğrenim süreci için ders planlarının beş aşamalı; artırımlı ve ilerleyici bir düzende temsil edilen adımlar altında sunumu gerçekleştirilmiştir. Ders açılış planlarında **Genel Bakış** başlığı altında kısaca ifade edilen ve gerekli noktalarda ders planı içeriklerinde detaylandırılan adımlar ve işlevleri genel hatlarıyla şu şekildedir:



1. ADIM: HAREKETE GEÇ: İşlenecek olan konulara yönelik öğrenci ilgisini çekecek aktivitelerin gerçekleştirilmesi, gerekli teorik konuların aktarılması.



2. ADIM: KEŞFET: Bir önceki adımda gerçekleşen ilgi çekme ve teorik girizgahlar ardından ilerleyen konuların ve uygulama adımlarının işlenmeye başlanması, gerekli noktalarda yeni bilgi ve becerilerin elde edilmesi yönünde aksiyonların alınması.



3. ADIM: ÜRET: Bir önceki adımdaki aktivitelerin üzerine yeni aksiyonların alınarak işlenen konu kapsamının derinleştirilmesi ve gerekirse nihayete erdirilmesi.



4. ADIM: İLERLET: Öğrencilerin işlenen konular ve gerçekleştirilen uygulamalar üzerine kendi inisiyatifleri ile düzenlemeler gerçekleştirilmesi, değerlendirmelerde bulunması.



5. ADIM: DEĞERLENDİRME: İşlenen konuların genel değerlendirmesinin yapılması ve önerilen sorular kapsamında ve alternatif soru-cevap süreçleriyle pekiştirmelerin sağlanması.

1. HAFTA – 1. GÜN – 1. DERS: YAPAY ZEKA NEDİR?

DERS PLANI

DERS ETİKETLERİ



KONU: Yapay Zeka Kavramı



SINIF: 12-14 Yaş



SÜRE: 60 dk.



HAZIR BULUNUŞLUK: Temel düzeyde teknoloji farkındalığı, temel İngilizce eğitimi



ANAHTAR KELİMELER: Yapay Zeka, Algoritma, Bilgisayar Programlama, Kodlama



BECERİLER:



Tahmin



Sorgulama



Günlük hayat bağlantıları kurma



KAZANIMLAR:

Bu derste öğrenciler;

- Yapay Zeka kavramının ne anlama geldiğini kavrar.
- Yapay Zeka ile algoritma, bilgisayar programlama ve kodlama kavramları arası bağlantıları kavrar.
- Yapılan tanımlardan yola çıkarak günlük hayatı Yapay Zeka'nın kullanıldığı uygulamalara örnekler verir.
- Yapay Zeka'nın uygulanabildiği problemleri tanımlar.



DERS MATERİYALLERİ:

Malzeme	Miktar
Yazı Tahtası	Öğrenci sayısı ve sınıf büyüklüğüne göre en az bir adet
Projeksiyon	Öğrenci sayısı ve sınıf büyüklüğüne göre en az bir adet
Defter, Kalem	Öğrenci sayısı kadar
Bilgisayar	Öğrenci sayısının yarısı kadar
Tablet	Öğrenci sayısının yarısı kadar



ÖĞRETMEN HAZIRLIK ÇALIŞMALARI:

Okunabilecek Kaynaklar:

Microsoft. (2024). Yapay Zeka Nedir?. Çevrimiçi: <https://azure.microsoft.com/tr-tr/resources/cloud-computing-dictionary/what-is-artificial-intelligence> (Erişim 22 Mayıs 2024).

Türkiye Cumhuriyeti Cumhurbaşkanlığı Dijital Dönüşüm Ofisi. (2024). Yapay Zeka. Çevrimiçi: <https://cbddo.gov.tr/ssss/yapay-zeka/> (Erişim 22 Mayıs 2024).

İzlenebilecek Kaynaklar:

Simplilearn. (2023). What is Artificial Intelligence? | Artificial Intelligence In 5 Minutes | AI Explained. Çevrimiçi: <https://www.youtube.com/watch?v=uMzUB89uSxU> (Erişim 22 Mayıs 2024).

TÜBİTAK Bilim Genç. (2022). Yapay Zeka Nedir? | Dr. Mehmet Haklıdır. Çevrimiçi: <https://www.youtube.com/watch?v=cQjSV7ZCVTQ> (Erişim 22 Mayıs 2024).

ATÖLYE HAZIRLIK ÇALIŞMALARI:

Öğretmen;

- Dersten önce ders materyallerini hazırlamalıdır.
- Yapay Zeka kavramına ilişkin ön bilgi edinmeli, Yapay Zeka ve dijital teknolojiler ekseninde farkındalık ve bilgi düzeyini güncel tutmalıdır.



KAYNAKÇA:

Ertel, W. (2018). *Introduction to Artificial Intelligence*. Springer.

Köse, U., Özsoy, K., & Aksoy, B. (2023). *Yapay Zeka – Ortaokul ve Lise Kitapları*. Deneyap Teknoloji Atölyeleri, TÜBİTAK Yayıncılık.

Nabiyev, V. (2021). *Yapay Zeka: Derin Öğrenme – Stratejili Oyunlar Örüntü Tanıma – Doğal Dil İşleme*. Seçkin Yayıncılık.

GENEL BAKIŞ:

- 1) Harekete Geç:** Öğrencilere Yapay Zeka kavramından ne anladıkları sorulur ve farklı görüşler toplanır.
- 2) Keşfet:** Yapay Zeka, algoritma, bilgisayar programlama, kodlama, program, yazılım gibi kavamlar tanımlanarak, öğrenciler ile birlikte tartışılır.
- 3) Üret:** Öğrencilerden Web üzerinden Yapay Zeka uygulamalarına yönelik araştırma gerçekleştirip, tespit ettiğleri bir uygulamaya dair algoritmik adımların ve hangi problemleri çözdüğüne dair tespitlerini yazmaları istenir.
- 4) İlerlet:** Öğrencilerin kaleme aldığı notlar tartışılır ve bu defa öğrencilerin gruplar halinde çalışarak, başka herhangi bir kaynaktan destek almadan, günlük hayatıta Yapay Zeka'nın kullanıldığı uygulamalara dair örnekler yazmaları istenir.
- 5) Değerlendir:** Yazılan örnekler ve derste öğrenilen bilgiler ışığında soru-cevap etkinliği gerçekleştirilerek Yapay Zeka'nın kavramsal çerçevesi pekiştirilir.

UYGULAMA



1. ADIM: HAREKETE GEÇ

Bu adımda öğrencilere Yapay Zeka kavramına ilişkin düşüncelerini ve güncel teknolojik gelişmelerden yola çıkarak, Yapay Zeka denildiğinde akıllarına neler geldiğini ifade etmeleri istenir. İhtiyaç duyulduğu aşamalarda yapılacak yönlendirmelerle, Yapay Zeka'nın hayatımızdaki etkilerine dair öğrenci düşüncelerinin toplanması sağlanır. Farklı öğrencilerden Yapay Zeka kavramının tanımına ve etkilerine yönelik olarak ifade edilen dikkat çekici hususlar yazı tahtasında not alınarak ortak bir kavram çerçevesi oluşturulur.

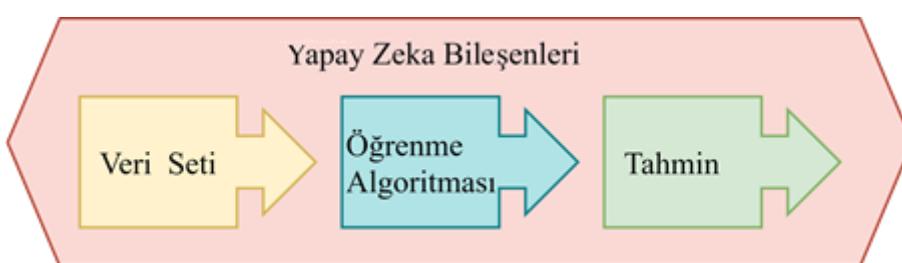


2. ADIM: KEŞFET

Bu adımda Yapay Zeka ile birlikte, algoritma, bilgisayar programlama, kodlama, program, yazılım gibi kavamlar tanımlanır ve kavamlar arası bağlantılar öğrenciler ile birlikte tartışılır.

Yapay Zeka ve İlgili Kavramlar

Yapay Zeka, insan zekasını modelleyebilmek adına insan gibi akıl yürütme, anlam çıkartma, genellikle yapabilme, geçmiş deneyimleri ile öğrenebilme gibi yetkileri bir bilgisayara ya da makineye kazandırmaktır. Oxford İngilizce Sözlük'te ise Yapay Zeka görsel algılama, konuşma tanıma, karar verme ve diller arasında çeviri gibi normalde insan zekasını gerektiren görevleri yerine getirebilen bilgisayar sistemleridir. Her iki tanımda da vurgulanan nokta, Yapay Zeka'nın bir görevi insan beyni gibi öğrenme fonksyonunu gerçekleştiren bilgisayar sistemleridir. Yapay Zeka, Şekil 1'de gösterildiği gibi veri seti, öğrenme algoritmaları ve karar verme süreci olmak üzere üç ana bileşenden oluşur. Yapay Zeka'da metin, görüntü verileri, zaman, uzunluk ölçümleri, video kayıtlarının düzenlenmesi ile veri seti oluşmaktadır. Öğrenme algoritmaları Makine Öğrenmesi ve Derin Öğrenme gibi alt-dallar kapsamında incelenmektedir. Yapay Zeka öğrenme algoritmaları karmaşık bir yapıya sahip gibi gözükse de temelde algoritma ve programlamadan oluşmaktadır (Köse vd., 2023).



Şekil 1. Yapay Zeka bileşenleri.

Yapay Zeka esasında matematiksel ve mantıksal çeşitli eklentilerle geliştirilmiş algoritmalar bütünü olarak ifade edilebilmektedir. Bu nedenle algoritma ve hatta diğer bağlı kavramları tanımlamak Yapay Zeka'nın anlaşılmasını da sağlayabilecektir (Köse vd., 2023).

Algoritma: Bilgisayarlarda bir problemin çözümünde izlenecek yollar bütünü genel tanımlı algoritma olarak adlandırılır. Günlük hayatı planladığımız birçok faaliyette aslında algoritma tabanlı düşünerek planlamalar yapar ve problemleri algoritmik düşünce mantığı ekseninde çözümleriz. Bu noktada 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 işlemi gerçekleştirirken algoritmalarдан faydalananız. Böylece, problemleri bölmelere ayırma ve çözüme ulaşabilmek için uygulanan işlem basamakları algoritmik düşünceye örnek olarak verilebilir.

Program: Yapay zeka uygulamaları için algoritma adımları bilgisayar tarafından gerçekleştirilen program kodlarının bütündür. Programlar Yapay Zeka uygulamalarının bilgisayar tarafından yapılması için gereken adımların bir programlama dili aracılığıyla aktarılmasıdır.

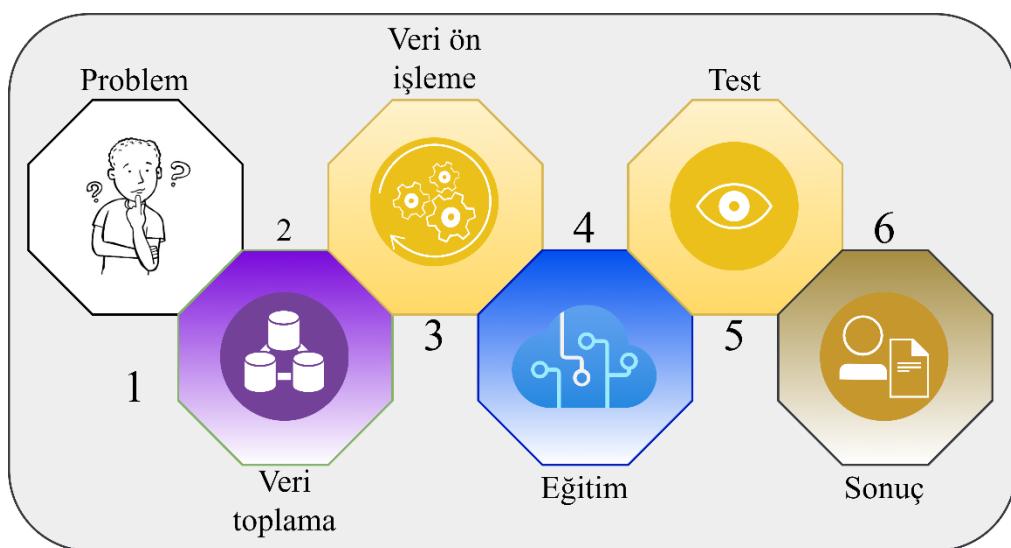
Bilgisayar Programlama: Bilgisayar tabanlı sistemlerin program adı verilen yapılar içerisinde özel kodlarla ifade edilen algoritma adımları içerisinde problem çözümü yönünde yönlendirilmesi bilgisayar programlama anlamına gelmektedir.

Kodlama: Özellikle günümüzde sıkılıkla dile getirilen kodlama kavramı, aslında bilgisayar programmanın günlük dilde ifade ediliş şekli olmakta; özünde programlama dili kodlarıyla program yazmayı işaret etmektedir.

Yapay Zeka'da Problem Modelleme

Yapay Zeka'da problemin modellemesinde kullanılan birçok otomatik karar verme mekanizması vardır. Bu mekanizmalardan en önde geleni “verilerden problemi öğrenebilme” ile ilişkilidir. Problemi öğrenebilen Yapay Zeka modelleri kullanıldığında, uzman bir kişinin vereceği bir kararı “eğitimmiş” matematiksel algoritmalar oluşturur (Ertel, 2018). Böylelikle Yapay Zeka modelleri karar sürecinin yorumlanması yardımcı araçlar haline gelir. Yapay Zeka modelleri, sistemin en doğru kararını vermesi veya maliyeti en aza indirmesi için büyük miktarda veriyi işlemektedir. Farklı ortamlarda gelen verileri analiz eden Yapay Zeka modeli tüm verileri gözden geçirerek uzmanlardan oluşan bir ekibin yapacağı belirli bir karar sürecini tek başına gerçekleştirir. Yapay Zeka modellemesi veri akışı Şekil 2'de gösterildiği gibi şu şekildedir (Köse vd., 2023; Nabihev, 2021):

- **Problem:** Yapay Zeka çözümleri için problemin sınıflandırma, regresyon ya da kümleme yöntemlerinden hangisi üzerine kurgulanacağı belirlenir. Örneğin kedi ve köpek verilerini ayırturan veri seti sınıflandırma problemi, yaya sayısına göre trafik ışıklarının süresinin belirlenmesi bir regresyon problemi olarak kabul edilir.
- **Veri Toplama:** Problemin çözümünü sağlayacak örnek sebep ve sonuç verilerini toplanarak bir araya getirilir.
- **Veri Önisleme:** Bu aşamada, toplanan veriler üzerinde eksik verileri tamamlama, anlamsız verileri çıkarma gibi veri önisleme aşamaları gerçekleştirilerek Yapay Zeka'da olusabilecek problemlerin önlenmesi sağlanır. Veri önisleme aşaması Yapay Zeka'nın eğitim ve test adımlarında sağlıklı çalışmasını sağlayacak verilerin organize edilmesi bakımından oldukça önemlidir.
- **Eğitim:** Veri önisleme sonrasında anlaşılan verilerden eğitim için ayrılan veriler ile Yapay Zeka modellerinin eğitimi gerçekleştirilir.
- **Test:** Eğitilen modellerin doğruluğu test verileri ile değerlendirilerek problem çözümü açısından anlamlı sonuçlar elde edilmeye çalışılır.
- **Sonuç:** Test edilen veriler ile gerçekleştirilen uygulamalar sonucunda üzerinde en doğru bulguları veren Yapay Zeka modeli gerçek uygulamalarda kullanılmak üzere seçilir.



Şekil 2. Yapay Zeka problem modelleme.



3. ADIM: ÜRET

Üret adımda öncelikli olarak öğrencilerin Web üzerinden Yapay Zeka uygulamalarına yönelik araştırma gerçekleştirip, tespit ettikleri bir uygulamaya dair algoritmik adımların ve hangi problemleri çözdüğüne dair tespitlerini yazmaları istenir. Bu süreç için 15 dakikalık bir süre tanınır.



4. ADIM: İLERLET

Öğrencilerin önceki adımda kaleme aldığı notlar tartışıılır ve bu defa öğrencilerin bireysel olarak, başka herhangi bir kaynaktan destek almadan, günlük hayatı Yapay Zeka'nın kullanıldığı uygulamalara dair örnekler yazmaları istenir.



5. ADIM: DEĞERLENDİRME

Öğrencilerin yazdıkları örnekleri kısaca anlatmaları sağlanır. Yazılan örnekler ve derste öğrenilen bilgiler ışığında soru-cevap süreci uygulanarak Yapay Zeka'nın kavramsal çerçevesi pekiştirilir.

Öğrencilere şu sorular sorulabilir:

- Öğrendiklerinizden hareketle Yapay Zeka'nın net bir tanımını nasıl yapabilirsiniz?
- Yapay Zeka'yı oluşturan temel bileşenler nelerdir?
- Yapay Zeka ile nerelerde karşılaşabiliriz?
- Yapay Zeka hayatımıza nasıl şekillendirecek?
- Yapay Zeka günlük hayatımızda ne gibi avantajlar sağlıyor?

1. HAFTA – 1. GÜN – 2. DERS: YAPAY ZEKA UYGULAMA ALANLARI VE YAKLAŞIMLARI

DERS PLANI

DERS ETİKETLERİ



KONU: Yapay Zeka Uygulama Alanları ve Yaklaşımları



SINIF: 12-14 Yaş



SÜRE: 60 dk.



HAZIR BULUNUŞLUK: Temel düzeyde teknoloji farkındalığı, temel İngilizce eğitimi, Yapay Zeka kavramı hakkında temel bilgi



ANAHTAR KELİMELER: Yapay Zeka, Algoritma, Yapay Zeka Uygulamaları, Yapay Zeka Çözüm Yaklaşımları



BECERİLER:



Tahmin



Sorgulama



Grup çalışması



Günlük hayat bağlantısı kurma



KAZANIMLAR:

Bu derste öğrenciler;

- Yapay Zeka'nın farklı alanlardaki uygulamaları hakkında fikir sahibi olur.
- Farklı alanlar içerisindeki Yapay Zeka tabanlı uygulamaları karşılaştırır.
- Yapay Zeka ailesinin oluşturan temel çözüm yaklaşımlarını öğrenir.
- Farklı Yapay Zeka çözüm yaklaşımlarının kullanılabileceği uygulamalara örnekler verir.



DERS MATERİYALLERİ:

Malzeme	Miktar
Yazı Tahtası	Öğrenci sayısı ve sınıf büyüklüğüne göre en az bir adet
Projeksiyon	Öğrenci sayısı ve sınıf büyüklüğüne göre en az bir adet
Defter, Kalem	Öğrenci sayısı kadar
Bilgisayar	Öğrenci sayısının yarısı kadar
Tablet	Öğrenci sayısının yarısı kadar



ÖĞRETMEN HAZIRLIK ÇALIŞMALARI:

Okunabilecek Kaynaklar:

Google. (2024). What are AI applications?. Çevrimiçi: <https://cloud.google.com/discover/ai-applications> (Erişim 22 Mayıs 2024).

İzlenebilecek Kaynaklar:

Bebar Bilim. (2019). Makine Öğrenmesi Nedir? - Makineler Nasıl Öğrenir?. Çevrimiçi: <https://www.youtube.com/watch?v=YuhOCJ6FjC4> (Erişim 22 Mayıs 2024).

Edureka. (2019). What Is Artificial Intelligence? | Artificial Intelligence (AI) In 10 Minutes. Çevrimiçi: <https://www.youtube.com/watch?v=oV74Najm6Nc> (Erişim 23 Mayıs 2024).

ATÖLYE HAZIRLIK ÇALIŞMALARI:

Öğretmen;

- Dersten önce ders materyallerini hazırlamalıdır.
- Yapay Zeka'nın farklı alanlardaki uygulamalarına ilişkin güncel bilgiler edinmeli, Yapay Zeka içerisindeki farklı temel çözüm yaklaşımlarını kavramalıdır.



KAYNAKÇA:

Karaboğa, D. (2017). *Yapay Zeka Optimizasyon Algoritmaları*. Nobel Akademik Yayıncılık.

Keskenler, M. F., & Keskenler, E. F. (2017). Bulanık mantığın tarihi gelişimi. *Takvim-i Vekayi*, 5(1), 1-10.

Köse, U. (2021). *Yapay Zeka Etiği*. Nobel Yayıncılık.

Nabiyev, V. (2021). *Yapay Zeka: Derin Öğrenme – Stratejili Oyunlar Örüntü Tanıma – Doğal Dil İşleme*. Seçkin Yayıncılık.

Shalev-Shwartz, S., & Ben-David, S. (2014). *Understanding machine learning: From theory to algorithms*. Cambridge university press.

Vanneschi, L., & Silva, S. (2023). *Lectures on Intelligent Systems* (pp. 271-281). Springer.

GENEL BAKIŞ:

1) Harekete Geç: Öğrencilerden bir uygulama alanı (Örneğin, tıp, işletme, savunma sanayi, eğitim, edebiyat, sanat) seçimleri ve seçilen uygulama alanı içerisinde Yapay Zeka'nın bilinen uygulamalarına dair yorumlar yapmaları istenir.

2) Keşfet: Yapay Zeka'nın farklı uygulama alanlarındaki bilinen ve potansiyel uygulamalarına yönelik açıklamalar yapılır.

3) Üret: Yapay Zeka uygulamalarına temel teşkil eden farklı çözüm yaklaşımları anlatılır, öğrencilerin günlük hayatı Yapay Zeka uygulamaları içerisindeki çözüm yaklaşımlarına dair fikirlerini yazmaları istenir ve öğrencilerin fikirleri karşılaştırılarak ilgili çözüm yaklaşımlarının pekiştirilmesi sağlanır.

4) İlerlet: Öğrencilerin anlatılan çözüm yaklaşımlarından destek alarak, gruplar halinde çalışarak Yapay Zeka uygulamaları düşünmeleri ve düşündükleri uygulamalarla birlikte hedef faaliyet alanı ve kullanılması tercih edilen çözüm yaklaşımlarına dair açıklamalar yazmaları istenir.

5) Değerlendir: Yazılan açıklamalar ve derste öğrenilen bilgiler ışığında soru-cevap süreci üzerinden Yapay Zeka'nın farklı alanlardaki etkin uygulamaları ve Yapay Zeka ailesini oluşturan çözüm yaklaşımları genel hatlarıyla pekiştirilir.

UYGULAMA



1. ADIM: HAREKETE GEÇ

Bu adımda öğrencilerden bir uygulama alanı (Örneğin, tıp, işletme, savunma sanayi, eğitim, edebiyat, sanat) seçimleri ve seçilen uygulama alanı içerisinde Yapay Zeka'nın bilinen uygulamalarına dair yorumlar yapmaları istenir.



2. ADIM: KEŞFET

Bu adımda, Yapay Zeka'nın farklı uygulama alanlarındaki bilinen ve potansiyel uygulamalarına yönelik açıklamalar yapılır.

Yapay Zeka Uygulamaları

Yapay Zeka modelleri algoritma yapıları gereği farklı problemlere kolaylıkla uyarlanabilmekte ve etkin çözümler üretebilmektedir. Bu aşamada bir Yapay Zeka modeli, hedef problem ile alakalı veriler üzerinden Yapay Zeka algoritma değişkenlerinin döngüler içerisinde düzenlenmesini sağlayan matematiksel ve mantıksal işlevler ile desteklenir. Bu matematiksel ve mantıksal işlevler bilimsel araştırmalar neticesinde ortaya çıkmaktadır (Vanneschi, & Silva, 2023). Bu sayede Yapay Zeka alanı içerisinde yeni atılımlar yapılmış ve Yapay Zeka her alanda uygulanır hale gelmiştir. Yapay Zeka'nın uygulamaları genel olarak şöyle öneklenebilir (Köse, 2021):

- **Tıp:** Yapay Zeka ilk ortaya çıkışından bu yana en yakın ilişkisini Tıp alanı ile sağlamıştır. Bir hasta karar destek sistemi olarak geliştirilen Eliza, ilk önemli Yapay Zeka uygulamaları arasında yer alır. Tıp alanındaki Yapay Zeka uygulamaları genellikle semptom durumları, tahlil sonuçları, medikal görüntüler ya da kişiye özgü diğer tıbbi niteliklerden yola çıkarak hastalık teşhisinde / tahmininde bulunan sistemler ya da tedavi planlaması yapan karar destek yazılımları yönünde ilerlemektedir. Bununla beraber Yapay Zeka'nın ilaç-hastalık etkileşimi tahmini ve ilaç geliştirme uygulamalarında çığır açtığı bilinmektedir. Yapay Zeka'nın devasa veri bütünüleri içerisindeki başarısı, geleneksel çözümlerin yetersiz kaldığı ileri düzey ilaç-protein etkileşimlerinin anlaşılması ve genetik alanında çözümü zor analizlerin gerçekleştirilebilmesinde anahtar rol oynar. Bununla beraber, Nesnelerin İnterneti, giyilebilir teknolojiler ve robotik alanındaki gelişmeler Yapay Zeka tabanlı cerrahi robotların, otomatik tahlil araçlarının ve kişisel sağlık verilerini takip eden mobil cihazların ortaya konulmasını sağlamaktadır.
- **İşletme:** Yapay Zeka uygulamaları karmaşık işletme verilerinin otomatik bir biçimde analiz edilmesini ve karar destek çözümlerinin hızlı bir şekilde oluşturulmasını mümkün hale getirir. Yapay Zeka'nın bu sayede işletmelerde yönetim pozisyonlarında yer alabilecek bir araç olup olamayacağı yönünde tartışmalar her geçen zaman daha fazla artış göstermektedir. İş Zekası olarak da adlandırılan çözümler Yapay Zeka uygulamalarının işletme operasyon adımlarının tahmini, mevcut performans durumlarının analizi, gelecek işletme kararları için tavsiyeler elde edilmesi ve karmaşık işletme verilerinin anlamlı hale getirilmesi gibi uygulamalar yönünde genişlemektedir. Benzer şekilde e-ticaret ve hizmet sektörlerine dair uygulamalar, Yapay Zeka modelleri üzerinden kullanıcı-yazılım odaklı akıllı araçlar şeklinde gelişimini sürdürmektedir.
- **Otonom Araçlar:** Yapay Zeka'nın farklı veri türleri üzerinden kararlar oluşturabilme yeteneği, Yapay Zeka uygulamalarının otonom eylemlerde bulunabilen, insansız / sürücüsüz araçlar içerisinde yoğun bir biçimde kullanılmasını sağlamıştır. Otonom araçlar tipik olarak belli seviyede eylemleri insan onayı ya da kontrolü olmaksızın gerçekleştirilebilen sistemler olarak dikkat çeker. Sürücüsüz otomobiller, dronlar ve farklı ortamlarda çalışabilen robotik sistemler otonom araçlar olarak bilinir. Yapay Zeka otonom araçlarda çevreden toplanan görüntülerin, seslerin, algılanan unsurlara dair verilerin yüksek başarımı işlenerek farklı alanlarda otonom çözümler elde edilmesini mümkün kılmıştır. Otonom araçlar başta taşımacılık olmak üzere, savunma sanayi ve üretim sektörleri özelinde önem taşımakta; Endüstri 4.0 ve gelecek endüstri evrelerinin şekillenmesinde rol oynamaktadır.
- **Eğitim:** Yapay Zeka uygulamaları eğitim verilerinin etkin işlenmesi sayesinde öğrenci başarılarının tahmini, öğrenme karakteristiklerine uygun ders materyallerinin tespit edilmesi, öğretim planlarının otomatik bir biçimde oluşturulması, eğitim yönetiminin sağlanması, eğitim kaynaklarının optimize edilmesi gibi problemlerde ön plana

çökmektedir. Yapay Zeka içeriği esnek ve dinamik algoritmalar sayesinde ders konularına ve kişiye özgü ihtiyaçlara uyarlanmış modeller oluşturulmasını sağlar, böylelikle akıllı öğretim araçları geliştirilebilir. Yine Üretken Yapay Zeka adı verilen gelişmiş modeller bilgi yiğinlarından anlamlı özetler çıkarılabilmesini, yeni bilgiler türetilmesini ve kişisel öğretim ihtiyaçlarının akıllı asistanlar aracılığıyla karşılanması adına da etkili sonuçlar ortaya koyabilir. Bu nedenle Yapay Zeka'nın eğitim alanında mevcut gelişmeler dışında geleceğe dair önemli potansiyeller taşıdığı da düşünülmektedir.

- **Edebiyat, Sanat ve Diğer Alanlar:** Yapay Zeka özellikle Üretken Yapay Zeka modelleri sayesinde edebi metinlerin analizinin yapılması ve hatta edebi unsurlar barından yeni metinlerin oluşturulmasını kolaylaştırmıştır. Üretken Yapay Zeka aynı zamanda benzeri olmayan dijital görsellerin ya da medya materyallerinin hızlı bir biçimde oluşturulmasını sağlayan uygulamalarda ön plana çıkmaktadır. Bu durum insan üretkenliği konusunda tartışmaları alevlendirse de Yapay Zeka uygulamalarının estetik kaygı taşıyan alanlardaki rolü son yıllarda sansasyonel seviyelere ulaşmıştır. Yapay Zeka'nın etkisi hayatın her alanında görülmektedir. Bilinen bilgi bütünüleri üzerinde etkin analizler gerçekleştirmeye ve yeni bilgi keşifleri sağlamaya işlevi, Yapay Zeka'nın Tarih, Antropoloji ya da Sosyoloji gibi alanlarda da uygulanabilmesini sağlamaktadır. Bu yönyle Yapay Zeka bilgi oluşturma ve keşfetmede önemli bir araçtır. Yapay Zeka gezegenlerin keşfedilmesi açısından Astronomi'de, alternatif bileşiklere dair deneylerin ilerletilmesi noktasında Kimya'da, doğal dinamiklerin etkin değerlendirilmesinde Fizik'te, problem modellerinin anlaşılması ve teknik araçların tasarılanıp geliştirilmesinde ise mühendislik alanlarında eşsiz çözümler üretmektedir.



3. ADIM: ÜRET

Üret adımında Yapay Zeka uygulamalarına temel teşkil eden farklı çözüm yaklaşımları anlatılır, öğrencilerin günlük hayatımdaki Yapay Zeka uygulamaları içerisindeki çözüm yaklaşımlarına dair fikirlerini yazmaları istenir. Öğrencilerin çözüm yaklaşımlarına dair fikirlerini organize edip yazmaları için 10 dakika süre tanınır. Süre sonunda öğrencilerin fikirleri karşılaştırılarak ilgili çözüm yaklaşımlarının pekiştirilmesi sağlanır.



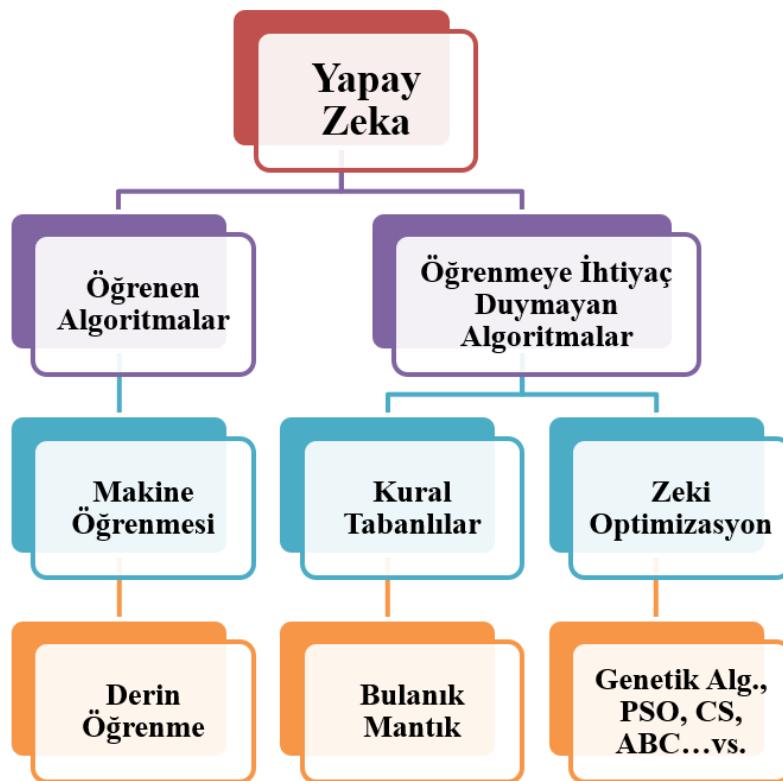
4. ADIM: İLERLET

Öğrencilerin bu defa anlatılan çözüm yaklaşımlarından destek alarak, gruplar halinde çalışarak Yapay Zeka uygulamaları düşünmeleri ve düşündükleri uygulamalarla birlikte hedef faaliyet alanı ve kullanılması tercih edilen çözüm yaklaşımlarına dair açıklamalar yazmaları istenir. Bu etkinlik için toplamda 10 dakikalık bir süre verilir.

Yapay Zeka İçerisinde Çözüm Yaklaşımları

Yapay Zeka uygulamaları genellikle problemleri öğrenen algoritmalar ve bu algoritmalar üzerinde oluşturulan modeller ile ilişkilendirilmektedir. Ancak aralarına her gün yenileri eklenen Yapay Zeka algoritmaları esasında farklı çözüm yaklaşımları ile bağlantılı olabilir. Günümüz Yapay Zeka ailesi, bu çözüm yaklaşımlarına göre öğrenen algoritmalar; yani Makine

Öğrenmesi ve öğrenmeye ihtiyaç duymayan algoritmalar şeklinde iki temel sınıfa ayrılmaktadır. 21. Yüzyıl'daki gelişmeler neticesinde Makine Öğrenmesi altında Derin Öğrenme adı verilen çözüm yaklaşımları doğmuştur. Diğer yandan öğrenmeye ihtiyaç duymayan algoritmalar ise kural tabanlılar ve zeki optimizasyon algoritmalarını içermektedir. İlgili sınıflandırma Şekil 3'te gösterilmektedir.



Şekil 3. Yapay Zeka'da temel çözüm yaklaşımları özelinde algoritmalar.

- **Öğrenen Algoritmalar:** Daha önce Yapay Zeka'da Problem Modelleme başlığı altında da anlatılan (Bkz. 1. Hafta – 1. Gün – 1. Ders) süreçler Yapay Zeka'nın öğrenen algoritmaları ile ilgilidir. Makine Öğrenmesi (Machine Learning) ismiyle de anılan çözüm yaklaşıma göre bu algoritmalar hedef problem ile ilgili toplanan veri setlerinden (toplu haldeki verilerden) problemi öğrenerek yeni durumlara karşı çözüm üretebilecek seviyeye ulaşırlar. Bu amaçla hedef problem ile ilgili veri seti üzerinden eğitim ve test aşamalarının gerçekleştirilmesi önem arz etmektedir. Makine Öğrenmesi yaklaşımı altındaki algoritmalar Yapay Zeka'nın en çok dikkat çeken ve gelişiminde rol oynayan araçlardır. Artan veri yığınları, zorlaşan problemler, daha fazla veri içeren daha zorlu problemler, Makine Öğrenmesi altında Derin Öğrenme yaklaşımının da ortaya çıkışını sağlamıştır. Derin Öğrenme aslında problem ile ilgili verileri "daha derinlemesine" analiz edip yüksek başarılı çözümler üreten Makine Öğrenmesi yaklaşımıdır (Nabihev, 2021; Shalev-Shwartz, & Ben-David, 2014). Makine Öğrenmesi ve Derin Öğrenme Yapay Zeka'nın günümüz başarılarını ve geleceğe yönelik potansiyellerini destekleyen en önemli çözüm yaklaşımıdır.
- **Öğrenmeye İhtiyaç Duymayan Algoritmalar:** Yapay Zeka ailesi altındaki bazı algoritmalar, veri seti üzerinden öğrenmeye ihtiyaç duymadan; çeşitli kurallar ya da matematiksel modeller üzerinden esnek çözümler oluşturabilir. Öğrenmeye ihtiyaç

duymadan çözüm üretebilme yaklaşımı, hızlı ve anlık çözüm üretilmesi gereken problemler için etkili Yapay Zeka uygulamalarının geliştirilmesini sağlar. Buna göre, kural tabanlı algoritmalar çözümlerini uzman bilgisi ile belirlenmiş kurallardan yola çıkararak oluşturmaktadır. Örneğin Bulanık Mantık, insan düşünce mantığını 1 ve 0'lar ile çalışan klasik bilgisayar sistemlerine matematiksel olarak uyarlanması sağlanmaktadır; bilgisayarlar 1 ve 0 arasındaki değerlerin de dikkate alındığı problem durumları üzerinden, tanımlanan kurallarla esnek çözümler üretebilmektedir (Keskenler, & Keskenler, 2017). Böylelikle hassas sulama yapan akıllı sulama sistemleri, verimli çalışan bulaşık makineleri, akıllı sinyalizasyon uygulamaları geliştirilebilir.

Kural tabanlılar dışında Yapay Zeka çözüm yaklaşımları içerisinde önemli yer tutan diğer çözüm yolu ise Zeki Optimizasyon olarak adlandırılır. Doğadaki canlıların koordinasyon ve iş birliklerini içeren davranışlarından ya da farklı doğal dinamikler içerisindeki rutin süreçlerden esinlenen Zeki Optimizasyon algoritmaları, matematiksel modeli tanımlanmış problemler için en uygun değişken değerlerinin tespit edilebilmesini sağlamaktadır (Karaboğa, 2017). Zeki Optimizasyon çözüm yaklaşımı, GPS tabanlı yol tariflerinin oluşturulması, en uygun vardiya çizelgelerinin elde edilmesi, optimum makine / cihaz tasarımlarının elde edilmesi, karı maksimize eden ya da zararı minimize eden parametrelerin tespiti gibi problemler için uygulanabilmektedir.



5. ADIM: DEĞERLENDİRME

Öğrencilerin yazdıkları açıklamalar ve derste öğrenilen bilgiler ışığında soru-cevap etkinliği düzenlenerek Yapay Zeka'nın farklı alanlarda ortaya çıkan ancak ortak benzerlikler taşıyan çözüm yaklaşımlarına dair bilgiler soru-cevap süreçleri üzerinden pekiştirilir.

Öğrencilere şu sorular sorulabilir:

- Farklı alanlardaki Yapay Zeka uygulamalarında ne gibi benzerlikler görüyorsunuz?
- Düşündüğünüz Yapay Zeka uygulamalarının benzerleri farklı alanlarda da uygulanabilir mi?
- Yapay Zeka'nın farklı çözüm yaklaşımları bizlere ne gibi avantajlar sağlar?
- Öğrenen Yapay Zeka modellerinin günümüz problemlerinde ne gibi avantajlar sağlıyor?
- Öğrenen Yapay Zeka modelleri insanlığın geleceği açısından ne gibi avantajlar taşıyor?
- Üretken Yapay Zeka insanlığın geleceği açısından tehdit oluşturabilir mi?

1. HAFTA – 1. GÜN – 3. DERS: YAPAY ZEKA’NIN ROBOTİK TEKNOLOJİLERDE KULLANIMI

DERS PLANI

DERS ETİKETLERİ



KONU: Yapay Zeka’nın Robotik Teknolojilerde Kullanımı



SINIF: 12-14 Yaş



SÜRE: 60 dk.



HAZIR BULUNUŞLUK: Temel düzeyde teknoloji farkındalığı, temel İngilizce eğitimi, Yapay Zeka kavramı ve uygulamaları hakkında temel bilgi



ANAHTAR KELİMELER: Yapay Zeka, Algoritma, Robotik, Teknoloji



BECERİLER:



Tahmin



Sorgulama



Grup çalışması



Günlük hayat bağlantısı kurma



KAZANIMLAR:

Bu derste öğrenciler;

- Yapay Zeka’nın robotik teknolojiler ile bağlantısı hakkında fikir sahibi olur.
- Robotik teknolojilerde Yapay Zeka’nın sağladığı farklı çözümler konusunda farkındalık sahibi olur.
- Yapay Zeka’nın robotik teknolojiler üzerinden destek sağlayabileceği problemlere örnekler verir.



DERS MATERİYALLERİ:

Malzeme	Miktar
Yazı Tahtası	Öğrenci sayısı ve sınıf büyüklüğüne göre en az bir adet
Projeksiyon	Öğrenci sayısı ve sınıf büyüklüğüne göre en az bir adet
Defter, Kalem	Öğrenci sayısı kadar
Bilgisayar	Öğrenci sayısının yarısı kadar
Tablet	Öğrenci sayısının yarısı kadar



ÖĞRETMEN HAZIRLIK ÇALIŞMALARI:

Okunabilecek Kaynaklar:

Bentley, P. J. (2022). *Yapay Zeka ve Robotik*. (Çev. Emre Can Ercan). Kronik Kitap.

İzlenebilecek Kaynaklar:

TheAIGRID. (2024). Top 10 NEW Humanoid Robots For 2024 (Tesla, Figure 01, Agility, Boston Dynamics and More). Çevrimiçi: <https://www.youtube.com/watch?v=ubVoZik-Q2w> (Erişim 24 Mayıs 2024).

World Data Center. (2023). Boston Dynamics: 40 years of development (1983-2023) Atlas. Çevrimiçi: https://www.youtube.com/watch?v=_EZQx87DyzM (Erişim 24 Mayıs 2024).

ATÖLYE HAZIRLIK ÇALIŞMALARI:

Öğretmen;

- Dersten önce ders materyallerini hazırlamalıdır.
- Yapay Zeka'nın robotik teknolojiler odağındaki uygulamalarına ilişkin güncel bilgiler edinmelidir.



KAYNAKÇA:

Bentley, P. J. (2022). *Yapay Zeka ve Robotik*. (Çev. Emre Can Ercan). Kronik Kitap.

Lu, H., & Xu, X. (Eds.). (2018). *Artificial Intelligence and Robotics* (p. 326). Berlin: Springer.

- Mallick, A. N., Chander, A., Choudhari, A. P., Chattar, H. K., & Sahani, A. (2023). A Review on the Role of Soft Robotics in Medical Assistive Devices. *International Journal of Automation and Smart Technology*, 13(1), 2416-2416.
- Probst, P. (2023). A Review of the Role of Robotics in Surgery: To DaVinci and Beyond!. *Missouri Medicine*, 120(5), 389.
- Raj, M., & Seamans, R. (2019). Primer on artificial intelligence and robotics. *Journal of Organization Design*, 8(1), 11.
- Troiano, L., Vaccaro, A., Tagliaferri, R., Kesswani, N., Rodriguez, I. D., Brigui, I., & Parente, D. (2022). *Advances in Deep Learning, Artificial Intelligence and Robotics*. Springer International Publishing.
- Wiriyathammabhum, P., Summers-Stay, D., Fermüller, C., & Aloimonos, Y. (2016). Computer vision and natural language processing: recent approaches in multimedia and robotics. *ACM Computing Surveys (CSUR)*, 49(4), 1-44.
- Yasa, O., Toshimitsu, Y., Michelis, M. Y., Jones, L. S., Filippi, M., Buchner, T., & Katzschatmann, R. K. (2023). An overview of soft robotics. *Annual Review of Control, Robotics, and Autonomous Systems*, 6, 1-29.
- Yıldız, A. (2018). Endüstri 4.0 ve akıllı fabrikalar. *Sakarya University Journal of Science*, 22(2), 546-556.

GENEL BAKIŞ:

- 1) Harekete Geç:** Öğrencilerden robotik teknolojiler ve uygulamaları hakkında bildiklerini ifade etmeleri istenir.
- 2) Keşfet:** Günümüz robotik sistemleri ve bu sistemlerin günümüz kullanım senaryolarına ilişkin açıklamalar yapılır.
- 3) Üret:** Yapılan açıklamalardan da destek alarak öğrencilerden Yapay Zeka'nın söz konusu robotik sistemler içerisindeki potansiyellerine dair düşünmeleri istenir ve öğrencilerin fikirlerini diğer öğrencilerle karşılaştırılarak Yapay Zeka ve robotik uygulamalar arası bağlantılar açıklanır.
- 4) İlerlet:** Öğrencilerin gruplar halinde çalışarak zorluğu bilinen bir problem için Yapay Zeka tabanlı bir robotik uygulama çözümü düşünmeleri ve düşündükleri bu çözüme dair detayları yazmaları istenir.
- 5) Değerlendir:** Yazılan açıklamalar ve derste öğrenilen bilgiler ışığında soru-cevap aşaması gerçekleştirilmek suretiyle Yapay Zeka'nın robotik teknolojilerde kullanımı genel hatlarıyla pekiştirilir.

UYGULAMA



1. ADIM: HAREKETE GEÇ

Bu adımda öğrencilerden robotik teknolojiler ve uygulamaları hakkında bildiklerini ifade etmeleri istenir. Öğrencilerden gelen dönütlere göre robotik uygulamalarda meydana gelen değişimlere vurgu yapılarak robotik sistemlerdeki teknolojik ilerlemeye dikkat çekilir.



2. ADIM: KEŞFET

Keşfet adımda, farklı robotik sistemler ve bu sistemlerin günümüz kullanım senaryolarına ilişkin açıklamalar yapılır.

Günümüz Robotik Sistemleri ve Kullanım Senaryoları

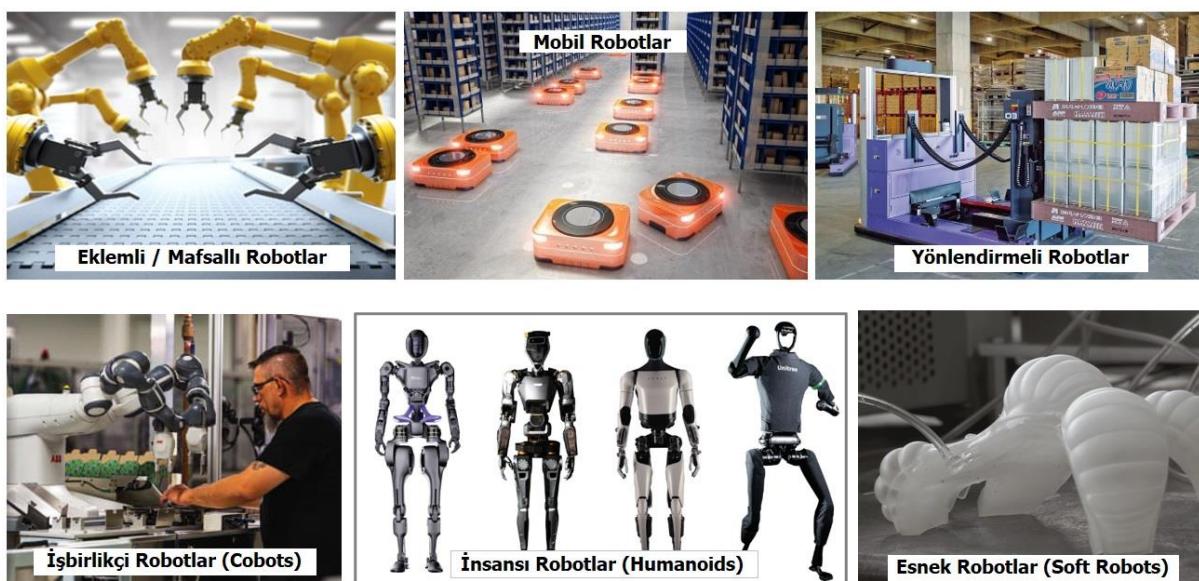
Önceleri daha çok üretim ve taşımacılık gibi alanlarda sundukları kısıtlı çözümlerle bilinen robotik teknolojiler, Yapay Zeka'nın artan desteğiyle birlikte hayatın farklı alanlarına nüfus eder hale gelmiştir. Robotik teknolojilerin özyü, yazılım kodlarıyla yönlendirilen donanımsal bileşenlerin spesifik uygulamalar özeline organize edilmesiyle bağlantılıdır. Bu nedenle robotik sistemler genellikle çevresel faktörleri algılamayı sağlayan sensörler, çevreye uygun dönütlerin verilmesini sağlayan eyleyiciler (aktuatörler), fiziksel manevraları sağlayan motor bileşenleri ve diğer donanımsal bileşenlerden meydana gelmekte, bu fiziksel bileşenlerin organizasyonu yazılımlara bağlanmaktadır. Robotik sistemlerdeki yazılım faktörü daha önce klasik algoritmalar dayanan yazılımlarla bağlantılıken, günümüzde Yapay Zeka ile yakından bağlantılı durumdadır (Raj, & Seamans, 2019). Bu noktada Yapay Zeka faktörünü anlamlandırmak için öncelikli olarak günümüz robotik sistemlerini genel hatlarıyla anlamak gerekmektedir.

Günümüz robotik uygulamaları çeşitli robotik sistemler üzerine kurulu durumdadır (Bentley, 2022; Lu, & Xu, 2018; Troiano vd., 2022; Yasa vd., 2023):

- **Eklemlı / Mafsallı Robotlar:** Eklemlı / mafsallı robotlar, üretim süreçlerinde sıkılıkla kullanılan ve çeşitli eklemlerin bir araya getirilmesiyle insan kol hareketlerine benzer eylemlerde bulunabilecek şekilde geliştirilen robot sistemleridir. Yapıları gereği üretim hatları ve karmaşık geliştirme süreçlerinde etkili araçlar olarak kullanılırlar. Robotik sistemlerin tarihsel gelişimi içerisinde uzun bir geçmişleri vardır.
- **Mobil Robotlar:** Geniş hareket yeteneklerine sahip mobil robotlar, hareket kabiliyetlerinin izin verdiği her ölçüdeki problemi çözebilmektedir. Yapay Zeka desteği ile otonom karar oluşturma ve eyleme geçme yetenekleri kazanan mobil robotlar, sensörleri üzerinden birbirleriyle de haberleşebilmekte, böylelikle üretim ve hizmet sektörlerinde etkili araçlar haline gelmektedir. Bu tür robotlar kompakt yapıları, hareket ve etkileşim kabiliyetleri nedeniyle insanlara yardımcı olan asistanlar rolüne de bürünmektedir.
- **Yönlendirmeli Robotlar:** Yönlendirmeli araçlar olarak da ifade edilen bu tür robotların en önemli işlevi, kılavuz olarak kabul ettikleri işaretler ya da sinyaller üzerinden çalışabilmeleridir. Bu yönleriyle üretim sektörlerinde ve taşımacılıkta önemli görevler alan yönlendirmeli robotlar, belirlenmiş kurallar çerçevesinde hareket edebilmeleri nedeniyle yüksek başarımlı sonuçlar elde etmek için idealdir.
- **İşbirlikçi Robotlar (Cobots):** Uygulama ortamlarında insanlarla birlikte çalışarak, “işbirliği içerisinde” çözümler üretebilmeleri adına geliştirilen robotlar işbirlikçi robotlar olarak bilinir. Diğer robot sistemlere göre insan komut ve eylemlerine daha duyarlı olan ve genellikle fiziksel olarak eklemlı / mafsallı robotlara benzer şekilde geliştirilen bu robotik sistemlere, yine üretim süreçleri başta olmak üzere, farklı görev ortamlarında rastlamak mümkündür.

- **İnsansı Robotlar (Humanoids):** Yapıları ve hareket mekanizmaları gereği insanlara benzetilerek geliştirilen insansı robotlar, özellikle son yıllarda ön plana çıkan ve günlük hayatı farklı görevler içerisinde yer almaları amacıyla geliştirilen robotik sistemler olarak bilinmektedir.
- **Esnek Robotlar (Soft Robots):** Diğer robot sistemlerinden farklı olarak esnek materyallerden (Örneğin, silikon) faydalananarak geliştirilen ve böylelikle erişilmesi ve çalışılması zor ya da adaptasyon gerektiren problemler içerisinde çalışan robotlar esnek robotlar olarak bilinmektedir. Bu tür robotların mimari yapıları daha serbest formlarda olabilmektedir.

Genel olarak açıklanan robotik sistemlere ilişkin örnek görseller Şekil 4 altında sunulmuştur.



Şekil 4. Günümüz robotik uygulamalarında görülen robotik sistemler.

Söz konusu robotik sistemler düşünüldüğünde, robotik teknolojilerin birçok uygulama alanında tercih edildiği görülmektedir. Buna göre, karanlık fabrika adı verilen ve insanların yer almadığı ortamlar, mobil robotların birbirleriyle etkileşimli bir biçimde çalıştığı; üretim başta olmak üzere taşıma / düzenleme ve kargolama gibi eylemler içeresine de girdiği platformlar olarak dikkat çekmektedir. Karanlık fabrikalar, bu bakımdan Endüstri 4.0 ve ötesinin en önemli getirileri arasında yer alır (Yıldız, 2018). Endüstri 4.0 kapsamında yaklaşıldığından, aynı robotik sistem yapılanmaları ağır sanayide, tarımda ve savunma sanayisinde de sıkılıkla görülmektedir. Mobil robotların yine hizmet sektörü odağında, resepsiyon görevlisi, garson ya da farklı destek personeli rollerinde kullanımı da zaman geçtikçe yaygınlaşmaktadır. Eklemli / mafsallı robotlar, yönlendirmeli robotlar ve işbirlikçi robotlar, insanların da yer aldığı üretim ortamlarında ya da deney ortamlarında sıkılıkla kullanılmaktadır. Hatta bu tür robotların tip alanında ameliyat robotları olarak rol aldığı görülmektedir (Probst, 2023). İnsansı robotlar henüz gelişim süreci içerisinde yer alsa da kısa sürede önemli gelişmeler ortaya koymuştur. Bu nedenle insansı robotların günlük hayatı insanlara destek sağlayacak pozisyonlarda yer alması ve hızla yaygınlaşması an meselesidir. Esnek robotlar ise problem dinamikleri çok değişken olan, coğrafi koşulların karmaşık olduğu ortamlarda yaygınlaşmakta, hatta sağlık uygulamalarında, iç organlara ulaşabilecek esneklikte çözümlerle teşhis problemlerinde ön plana çıkmaktadır (Mallick vd., 2023; Yasa vd., 2023).



3. ADIM: ÜRET

Üret adımda öğrencilerden öncelikli olarak bir önceki adımda yapılan açıklamalardan da destek alarak Yapay Zeka'nın anlatılan robotik sistemler içerisindeki potansiyellerine dair düşünmeleri istenir. Bu amaçla 10 dakikalık bir süre tanınır. Ardından, öğrencilerin oluşturdukları fikirler diğer öğrencilerle karşılaştırılarak Yapay Zeka'nın rollerine dair açıklamalar yapılır ve Yapay Zeka ve robotik uygulamalar arası bağlantıların pekiştirilmesi sağlanır.

Yapay Zeka'nın Robotik Sistemler İçerisinde Kullanımı

Yapay Zeka robotik sistemlerde ileri düzey ilerlemelerin elde edilmesinde önemli rol oynar. Bir robotik sistemi oluşturan donanımsal bileşenlerin algı ve eylem düzeyinde istenilen seviyede çalışmasını sağlayan Yapay Zeka altyapısı üzerine kurulan yazılım bileşenleridir. Bu açıdan Yapay Zeka'nın robotik sistemler içerisindeki kullanımı genellikle şu mekanizmalar çerçevesinde ortaya çıkmaktadır (Lu, & Xu, 2018; Raj, & Seamans, 2019; Wiriathammabhum vd., 2016):

- **Koordinasyon:** Robotik sistemde yer alan sensörler üzerinden algılanan çevresel faktörlerin robotik sistemi eyleme geçirecek şekilde analiz edilmesi ve hatta bu yolla tam otomatik sistemlerin elde edilmesi Yapay Zeka algoritmaları sayesinde gerçekleşmektedir. Robotik sistemin altyapısına modüler şekilde dahil edilen Makine Öğrenmesi tabanlı Yapay Zeka modelleri robotik sistemin öğrenen ve gelişen bir davranış yapısı elde etmesini sağlamakta, yine sistemin fiziksel eylemlerinin dengeli ve amaçlar doğrultusunda işletilmesi Yapay Zeka algoritmalarının meydana getirdiği sinerji ile gerçeğe dönüşmektedir.
- **Muhakeme:** Bir robotik sistemin eylem öncesinde çeşitli muhakemeler yapması ve en uygun eylem kararlarına ulaşması arkapanda yer alan Yapay Zeka modelleri ile gerçekleşebilmektedir. Bu aşamada özellikle Makine Öğrenmesi ya da Derin Öğrenme tabanlı modeller günümüz robotik sistemlerin muhakeme yeteneklerinde önemli bir yer tutar.
- **Bilgisayarlı Görü:** Görüntü tabanlı verilerin Yapay Zeka süreçleriyle harmanlandığı bir araştırma alanı olan Bilgisayarlı Görü (Computer Vision), günümüz robotik sistemlerinin görsel algılama ve görsel algılamalar üzerinden eyleme geçme aşamalarını şekillendirmektedir. Bilgisayarlı Görü genellikle Yapay Zeka tabanlı görsel algılamanın muhakeme ve koordinasyon ile iletişim kurduğu noktalarda yer almaktadır.
- **Etkileşim:** Bir robotik sistemin dış ortamda yer alan değişken unsurlarla, insanlarla ve hatta farklı robotik sistemlerle etkileşime girmesi, aktif Yapay Zeka altyapısının sürekli gerçeklestireceği algılama-muhakeme-eylem döngüleri ile bağlantılıdır. Söz konusu etkileşim süreci robotik sistemin fiziksel eylemleriyle ilgili olduğu kadar, sesli iletişim ya da dijital ortamda farklı siber-fiziksel unsurlarla meydana getireceği sinerji ile de bağlantılıdır.



4. ADIM: İLERLET

Öğrencilerin gruplar halinde çalışarak zorluğu bilinen bir problem için Yapay Zeka tabanlı bir robotik uygulama çözümü düşünmeleri ve bu çözüme dair detayları yazmaları istenir. Bu etkinlik için toplamda 10 dakikalık bir süre verilir.



5. ADIM: DEĞERLENDİRME

Yazılan açıklamalar ve derste öğrenilen bilgiler ışığında soru-cevap aşaması gerçekleştirilmek suretiyle Yapay Zeka'nın robotik teknolojilerde kullanımı genel hatlarıyla pekiştirilir.

Öğrencilere şu sorular sorulabilir:

- Robotik teknolojilerin meslekler üzerindeki etkileri hakkında neler düşünüyorsunuz?
- Sizce Yapay Zeka insansı robotlar (humanoids) yönündeki gelişmeleri hangi seviyelere taşıyacaktır?
- Robotik teknolojiler insanlık için ne gibi avantajlar ve dezavantajlar sağlamaktadır?
- İnsanlar ve robotik sistemler arasında en uygun işbirliği hangi şekillerde sağlanabilir?

1. HAFTA – 2. GÜN – 1. DERS: YAPAY ZEKA İÇİN PYTHON GELİŞTİRME ORTAMLARI

DERS PLANI

DERS ETİKETLERİ



KONU: Yapay Zeka İçin Python Geliştirme Ortamları



SINIF: 12-14 Yaş



SÜRE: 60 dk.



HAZIR BULUNUŞLUK: Temel düzeyde teknoloji farkındalığı, temel İngilizce eğitimi, Yapay Zeka farkındalığı, Programlamaya yönelik temel bilgi



ANAHTAR KELİMELER: Yapay Zeka, Algoritma, Kodlama, Python



BECERİLER:



Tahmin



Sorgulama



Grup çalışması



KAZANIMLAR:

Bu derste öğrenciler;

- Yapay Zeka geliştirme süreçlerinde kullanılabilen Python programlama dili hakkında bilgi sahibi olur.
- Python programlama dili ile kodlama gerçekleştirmek için kullanılabilecek geliştirme ortamları hakkında fikir sahibi olur.



DERS MATERİYALLERİ:

Malzeme	Miktar
Yazı Tahtası	Öğrenci sayısı ve sınıf büyüklüğüne göre en az bir adet
Projeksiyon	Öğrenci sayısı ve sınıf büyüklüğüne göre en az bir adet
Defter, Kalem	Öğrenci sayısı kadar
Bilgisayar	Öğrenci sayısının yarısı kadar
Tablet	Öğrenci sayısının yarısı kadar
Python Anaconda Framework	Her bilgisayarda kurulu olacak şekilde



ÖĞRETMEN HAZIRLIK ÇALIŞMALARI:

Okunabilecek Kaynaklar:

Python Wiki. (2024). *Python Programming – Beginners Guide*. Çevrimiçi: <https://wiki.python.org/moin/BEGINNERSGUIDE> (Erişim 24 Mayıs 2024).

İzlenebilecek Kaynaklar:

Yakın Kampüs. (2021). Python Dersleri #0 | Anaconda Kurulum. Çevrimiçi: <https://www.youtube.com/watch?v=k-zK7ltCXCA> (Erişim 24 Mayıs 2024).

TechCodeRealm. (2024). Top 5 Best Python IDE's in 2024. Çevrimiçi: <https://www.youtube.com/watch?v=Mex7XdYS350> (Erişim 24 Mayıs 2024).

ATÖLYE HAZIRLIK ÇALIŞMALARI:

Öğretmen;

- Dersten önce ders materyallerini hazırlamalıdır.
- Python Anaconda geliştirme platformunu yükleyip, genel hatlarıyla incelemelidir.
- Python kodlama için kullanılan Spyder editörünün genel kullanımına aşina olmalıdır.



KAYNAKÇA:

Deperlioğlu, Ö., & Köse, U. (2023). *Python ile Yapay Zekaya Giriş: Kavramsal Çerçeve – Temeller – Kodlama*. Seçkin Yayıncılık.

Köse, U., Özsoy, K., & Aksøy, B. (2023). *Yapay Zeka – Ortaokul ve Lise Kitapları*. Deneyap Teknoloji Atölyeleri, TÜBİTAK Yayıncılık.

GENEL BAKIŞ:

1) Harekete Geç: Öğrencilere Yapay Zeka uygulamaları geliştirme noktasında herhangi bir programlama dili ve Python programlama dili hakkında bilgi sahibi olup olmadığı sorulur ve karşılıklı tartışılır.

2) Keşfet: Öğrencilere Python programlama dilinin genel hatlarıyla tanıtımı yapılır.

3) Üret: Öğrencilerle birlikte Python Anaconda platformu ve Yapay Zeka uygulamaları geliştirmede gerekli olacak kütüphanelerin yüklemeleri gerçekleştirilir.

4) İlerlet: Öğrencilerin gruplar halinde çalışarak, yüklemiş oldukları Python geliştirme ortamını incelemeleri ve temel Python uygulamalarını çalıştırılmaları istenir.

5) Değerlendir: Derste edinilen bilgiler soru-cevap aşaması gerçekleştirilmek suretiyle pekiştirilir.

UYGULAMA



1. ADIM: HAREKETE GEÇ

Öğrencilere Yapay Zeka uygulamaları geliştirme noktasında herhangi bir programlama dili ve Python programlama dili hakkında bilgi sahibi olup olmadığı sorulur. Gelen cevaplara göre tartışmalar Yapay Zeka uygulamaları geliştirmede programlama dillerinin önemi ve hatta pratik kodlamaya imkan veren, hızlı programlama dillerine olan ihtiyaç bağlamında yönlendirilir.



2. ADIM: KEŞFET

Onceki adımı takiben bu adımda öğrencilere Python programlama dilinin genel hatlarıyla tanıtımı yapılır.

Python Programlama Dili

Python programlama dili 1980'li yılların sonlarına doğru, Guido van Rossum tarafından geliştirilen bir programlama dilidir. Yapısı itibarıyle basit bir kod yapısına sahip olan ve dolayısıyla daha az kodlarla etkin sonuçlar üreten Python, bu nedenlerle öğrenilmesi oldukça kolay bir programlama dili olarak dikkat çekmektedir. 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 biridir. Yapay Zeka denince akla ilk olarak Python dili gelmektedir. Bu durum da Python'ın dünya çapında büyük bir kitlesinin olmasına neden olmaktadır.

Python aktif bir geliştirici kitlesine sahip olması neticesinde, oldukça geniş bir kütüphane desteğini de arkasına almaktadır. Python programlama dilinde Yapay Zeka işlemeye genellikle Keras, Numpy, Pytorch, Matplotlib, Scipy, Scikit-learn kütüphaneleri kullanılmaktadır. Python'un diğer programlama dillerinden bir farkı olarak; girintili şekilde kod yazılarak kodların anlamlandırılması oldukça kolaydır. Python'da IDLE, Spyder ve Pycharm gibi farklı kod yazma editörleri kullanabilir. Python'da öğrenciler yazacakları her bir komutu en ince detayına kadar tam olarak bilmek zorundadırlar. Python programlama dili ile; internet sayfaları geliştirme, bilimsel ve sayısal hesaplama, Yapay Zeka uygulamaları, robotik kodlama gibi birçok farklı alanda uygulamalar gerçekleştirilmektedir. Python'un bir diğer önemli özelliği de nesne, özellik ve metod yapıları üzerine kurulu olan Nesne Yönelimli Programlama (OPP: Object Oriented Programming) yaklaşımı izlemesidir. Bu nedenle mantıksal ve organizasyonel olarak etkin uygulamalar kolaylıklar geliştirilebilmekte, karmaşık yapıdaki Yapay Zeka uygulamaları da Python sayesinde pratik ve başarımı yüksek bir biçimde ortaya konulabilmektedir (Deperlioğlu, & Köse, 2023; Köse vd., 2023). Python programlama süreçleri, kodların makine diline dönüşümünde rol oynayan Python yorumlayıcısının tercih edilen geliştirme editörleriyle entegre kullanımı sayesinde mümkün olmaktadır (Şekil 5).

```

Spyder (Python 3.8)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\Koray\Desktop\Yapay Zeka\EI yazılımı rakamların tanınması_CNN.py
temp.py* El yazılımı rakamların tanınması_CNN.py*
1 from __future__ import print_function
2 import keras
3 from keras.datasets import mnist
4 from keras.models import Sequential
5 from keras.layers import Dense, Dropout, Flatten
6 from keras.layers import Conv2D, MaxPooling2D
7 from keras import backend as K
8
9 batch_size = 128 # her bir iterasyonda
10 num_classes = 10 # ayırt etmek istediğimiz "10" ra
11
12 epochs = 12 # eğitim 12 epoch sürsün
13
14 img_rows, img_cols = 28, 28
15
16 # veri önce karıştırılıyor (shuffle) sonra da eğit
17 (x_train, y_train), (x_test, y_test) = mnist.load
18
19
20 if K.image_data_format() == 'channels_first':
21     x_train = x_train.reshape(x_train.shape[0], 1
22     x_test = x_test.reshape(x_test.shape[0], 1,
23     input_shape = (1, img_rows, img_cols)
24 else:
25     x_train = x_train.reshape(x_train.shape[0], i
26     x test = x test.reshape(x test.shape[0], img

```

Şekil 5. Python kodlama ortamı.



3. ADIM: ÜRET

Öğrencilerle birlikte Python Anaconda platformu ve Yapay Zeka uygulamaları geliştirmede gerekli olacak kütüphanelerin yüklemeleri gerçekleştirilir.

Python Geliştirme Ortamının Hazırlanması

Python geliştirme ortamının hazır hale getirilmesi için iki yol kullanılabilir:

1. Python yorumlayıcısı ve seçilen bir editörün entegrasyonu
2. Anaconda Framework yazılımının yüklenmesi

Python Yorumlayıcısı ve Seçilen Bir Editörün Entegrasyonu

Birinci yol tercih edildiğinde öncelikli olarak <https://www.python.org/> adresinden Python yorumlayıcısının indirilmesi gerekmektedir. Python.org Web sitesi, Python geliştiricilerinin ve Python kodlamaya yeni başlayanların birçok yardımcı kaynağı, haberleri ve etkinlikleri bulabildiği bir ortam olarak bilinmektedir (Şekil 6).

The screenshot shows the Python.org homepage. At the top, there's a dark header with the Python logo, navigation links for Python, PSF, Docs, PyPI, Jobs, and Community. Below the header is a search bar with a 'GO' button and a 'Socialize' link. The main content area has a dark background. On the left, there's a code editor window displaying Python code examples related to list comprehensions and enumerate functions. On the right, there's a section titled 'Compound Data Types' with a brief description and a 'More about lists in Python 3' link. At the bottom, there's a footer with a 'Learn More' link.

Şekil 6. Python.org Web sitesi.

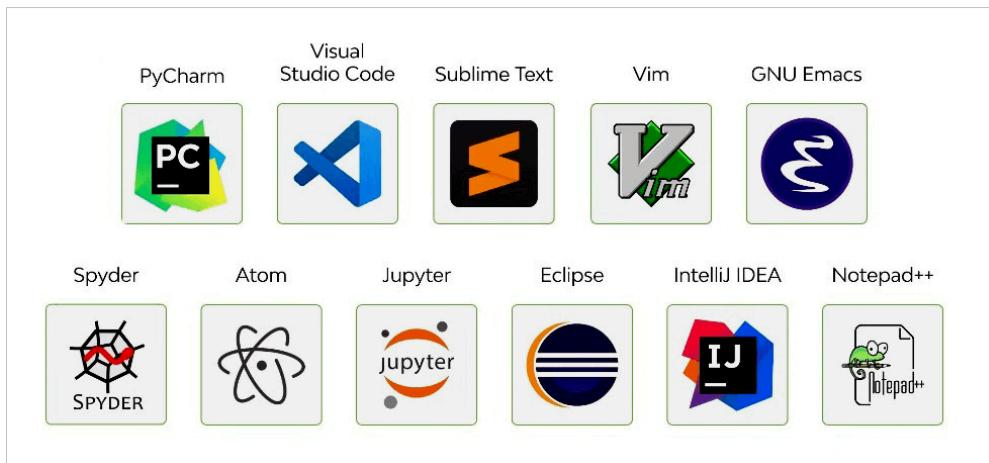
İlgili Web sitesindeki Downloads bölümü, Python güncel sürümünün ya da önceki sürümlerinin indirilmesi amacıyla kullanılır (Şekil 7).

Python version	Maintenance status	First released	End of support	Release schedule
3.13	prerelease	2024-10-01 (planned)	2029-10	PEP 719
3.12	bugfix	2023-10-02	2028-10	PEP 693
3.11	bugfix	2022-10-24	2027-10	PEP 664
3.10	security	2021-10-04	2026-10	PEP 619
3.9	security	2020-10-05	2025-10	PEP 596
3.8	security	2019-10-14	2024-10	PEP 569

Şekil 7. Python.org Downloads sayfası.

Python programlama dili yorumlayıcı (interpreter) adı verilen ve yazılan kodun herhangi bir yazım (syntax) ya da dil hatasıyla karşılaşılana dek yukarıdan aşağıya doğru çalıştırılabilmesini sağlayan bir koddan makine diline (1 ve 0'a) dönüştürücü vazifesi görmektedir. Farklı programlama dillerinde derleyici (compiler) adı verilen bir mekanizma da kullanılmakla beraber, bu dönüştürücü mekanizma ise yazılan kod hatalardan tamamen arındırıldığı zaman sonuca ulaşmaktadır.

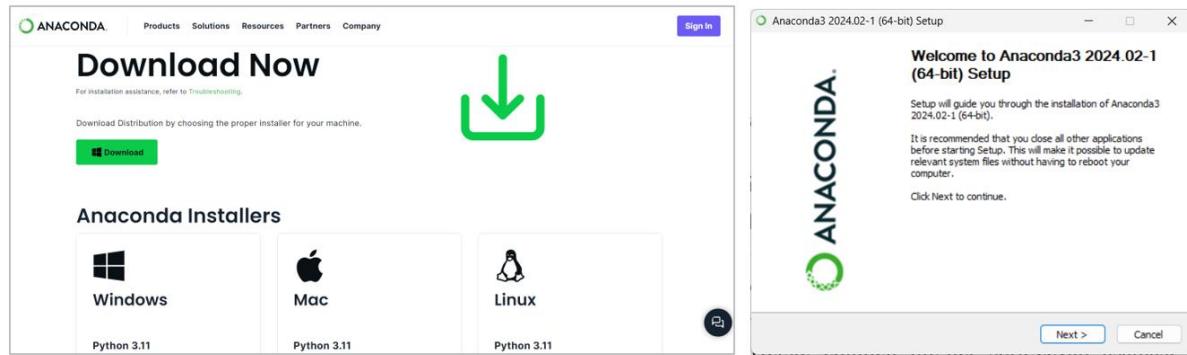
Python yorumlayıcısı indirildikten sonra işletim sisteminin komut sistemi üzerinden “python” komutu kullanılmak suretiyle Python kodlarının yazılıp uygulanması mümkün hale gelmektedir. Ancak geliştiriciler genellikle farklı görsel editörleri yükleyerek Python yorumlayıcı için daha yönetilebilir bir ortamda kod yazabilmektedir (Şekil 8). Bu nedenle yorumlayıcı yüklemesi sonrası tercih edilen bir editör yüklenerek, editör içerisindeki işlevler kullanılmak suretiyle Python yorumlayıcısı ile editör eşleştirilmelidir.



Şekil 8. Python kodlama için kullanılabilen farklı editörler.

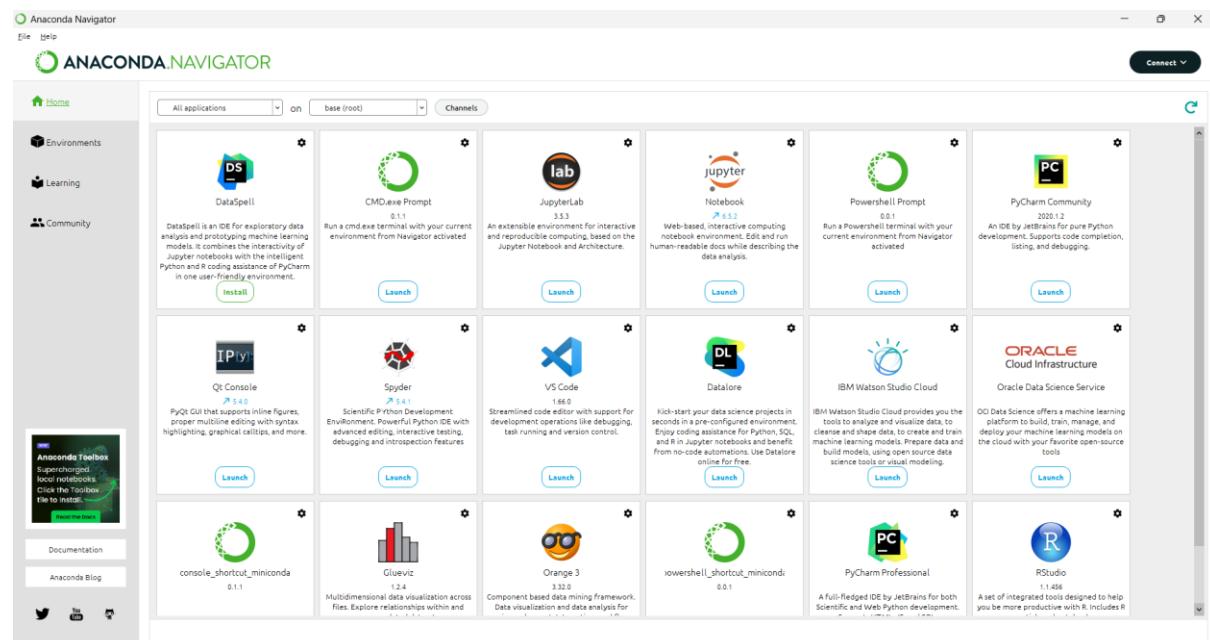
Anaconda Framework Yazılımın Yüklenmesi

Python kodlamaya hızlı ve pratik bir başlangıç yapabilmek için izlenebilecek bir diğer yol ise, kurulan bilgisayarda Python bileşenlerinin tek bir çatı altında yönetimini ve farklı Python editörlerinin kullanımını sağlayan Anaconda Framework yazılımının yüklenmesidir. Anaconda yazılımı <https://www.anaconda.com/download> sayfası üzerinden indirilebilmekte ve yükleme işlemi kolay bir şekilde gerçekleştirilmektedir (Şekil 9).



Şekil 9. Anaconda yazılım indirme sayfası (sol) ve Anaconda yükleme penceresi (sağ).

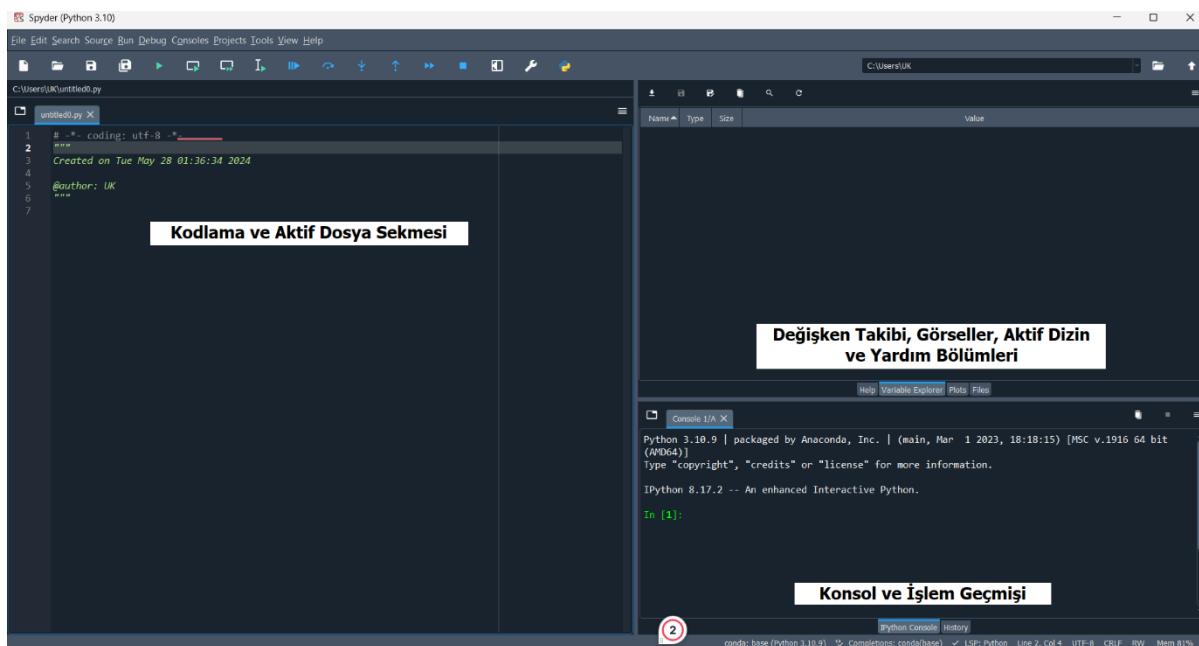
Anaconda platformunun açılış arayüzü Anaconda Navigator olarak isimlendirilir. Bu arayüz farklı Python editörlerinin seçilebildiği bir ortam sunmaktadır (Şekil 10). Yine arayüzdeki diğer seçenekler de işletim sistemine yüklenmiş olan Python bileşenlerinin (özellikle kütüphanelerin) yönetimi ve çeşitli öğrenme materyallerine erişim için kullanılabilirliktedir.



Şekil 10. Anaconda Navigator arayüzü.

Anaconda Navigator üzerinden açılabilen Python editörlerinden biri de "Spyder" olarak bilinmektedir. Spyder editörü, standart Python uygulamaları başta olmak üzere Yapay Zeka

tabanlı uygulamaların geliştirilmesinde etkili bir biçimde kullanılabilirmektedir. Varsayılan Spyder editörü arayüzü, kodların yazıldığı kodlama ve aktif dosya sekmesi, uygulama sonuçlarının yansıtıldığı konsol, işlem geçmişinin görüntülendiği panel ve kodlanan değişkenlerin, oluşturulan görsellerin, aktif kullanılan sistem dizinlerinin takip edildiği, hatta hızlı yardım açıklamalarının görüntülenebildiği bölümlerden oluşmaktadır (Şekil 11). Editör ortamındaki bütün bileşenler ve arayüzde bulunan bölümlerin kullanımına ilişkin daha fazla bilgi <https://docs.spyder-ide.org/current/panes/editor.html> sayfasında sunulmuş durumdadır.



Şekil 11. Spyder editör ortamına genel bakış.



4. ADIM: İLERLET

Öğrencilerin gruplar halinde çalışarak, yüklemiş oldukları Python geliştirme ortamını incelemeleri ve temel Python uygulamalarını çalıştırırları istenir. Bu amaçla 20 dakikalık bir süre tanınır ve bu süre zarfında öğrencilere takıldıkları noktalarda yardım edilir. Editör ortamında öğrencilerin çalıştırıp denemeleri amacıyla öğretmen tarafından çeşitli kodlar yazılabilir ya da şu kaynaklarda yer alan Python örnekleri çalıştırılabilir:

- Python Geliştiricileri Örnek Sayfaları: <https://docs.python.org/3/tutorial/index.html>
- W3 Schools Python Örnekleri: <https://www.w3schools.com/python/>

Spyder Editör Ortamının Kullanımı

Spyder Editör ortamında kodlama sekmesi kullanılmak suretiyle istenilen program kodları yazılabilmekte, klasik dosya işlemleri gerçekleştirilmek suretiyle düzenlenen kodlar kaydedilip, daha önce kaydedilmiş olan kod dosyalarının aynı bölüm üzerinde sekmeler halinde görüntülenmesi sağlanabilmektedir. Yazılan kodlar / uygulamalar Araç Çubuğu üzerinde yer alan “play (oynat)” simgeli düğme ile ya da F5 klavye tuşıyla çalıştırılabilmekte, çalışan kodların sonuçları konsol alanında görüntülenebilmektedir. Spyder konsol alanı Linux

komutları ile çalışır. Bu nedenle Python ortamına ilişkin yönetimsel düzenlemeler, kütüphane yüklemeleri, aktif dizine yönelik konsol tabanlı incelemeler bu alanda yapılabilir.

Spyder editörünü pratik kullanmak için tercih edilen bazı kısayol tuşları ve işlevleri şöyledir:

Tuş Kombinasyonu	İşlevi
F5	Aktif kodu çalıştır
F11	Tam ekran modu
Ctrl + L	İmleci satır numarası belirtilen kod satırına götür
Alt + Yukarı/Aşağı	İmlecin bulunduğu satirdaki kodları yukarı/aşağı taşı
Ctrl + 1	Seçili kod bölümlerini satır halinde yorumlara dönüştür
Ctrl + 4	Seçili kod bölümlerini blok halinde yorumlara dönüştür
Ctrl + 5	Blok halinde yorumlamayı geri al
Ctrl + Z	Yapılan son işlemi geri al
Ctrl + Shift + Z	Yapılan son işlemi tekrarla
Ctrl + X	Kes
Ctrl + C	Kopyala
Ctrl + V	Yapıştır
Ctrl + A	Tümünü seç
Ctrl + F	Bul penceresini aç
Ctrl + R	Değiştir penceresini aç
Ctrl + Page Up/Page Down	Açık durumdaki önceki/sonraki dosyaya geç
Ctrl + Shift + E	İmleci kodlama sekmesine konuştur
Ctrl + Shift + I	İmleci konsol sekmesine konuştur
Ctrl + + (mouse tekerleği ileri)	Zoom in
Ctrl + - (mouse tekerleği geri)	Zoom out
Ctrl + 0	Zoom sıfırla / reset
Ctrl + S	Son değişiklikleri kaydet

Spyder editöründe kullanılabilecek bütün kısayol tuşlarına dair bilgiler <https://docs.spyder-ide.org/current/panes/editor.html> sayfasında sunulmuş durumdadır.



5. ADIM: DEĞERLENDİRME

Derste edinilen bilgiler soru-cevap aşaması gerçekleştirilmek suretiyle pekiştirilir.

Öğrencilere şu sorular sorulabilir:

- Python programlama dili hakkında neler düşünüyorsunuz?

- Python ile Yapay Zeka uygulamaları geliştirme konusunda kendinizi hazır hissediyor musunuz?
- Python ile önceden düşündüğünüz ve tartıştığımız uygulamaları geliştirme konusunda neler düşünüyorsunuz?

1. HAFTA – 2. GÜN – 2. DERS: MAKİNE ÖĞRENMESİ YAKLAŞIMI VE MAKİNE ÖĞRENMESİ YÖNTEMLERİ

DERS PLANI

DERS ETİKETLERİ



KONU: Makine Öğrenmesi Yaklaşımı ve Makine Öğrenmesi Yöntemleri



SINIF: 12-14 Yaş



SÜRE: 60 dk.



HAZIR BULUNUŞLUK: Temel İngilizce eğitimi, Yapay Zeka kavramı hakkında temel bilgi, Makine Öğrenmesi hakkında farkındalık



ANAHTAR KELİMELER: Yapay Zeka, Algoritma, Makine Öğrenmesi, Sınıflandırma, Regresyon, Kümeleme



BECERİLER:



Tahmin



Sorgulama



Grup çalışması



Günlük hayat bağlantısı kurma



KAZANIMLAR:

Bu derste öğrenciler;

- Makine Öğrenmesi yaklaşımı hakkında bilgi sahibi olur.
- Farklı Makine Öğrenmesi yöntemlerinin işlevleri hakkında bilgi sahibi olur.
- Farklı Makine Öğrenmesi yöntemlerinin kullanılabileceği uygulamalara örnekler verir.



DERS MATERİYALLERİ:

Malzeme	Miktar
Yazı Tahtası	Öğrenci sayısı ve sınıf büyüklüğüne göre en az bir adet
Projeksiyon	Öğrenci sayısı ve sınıf büyüklüğüne göre en az bir adet
Defter, Kalem	Öğrenci sayısı kadar
Bilgisayar	Öğrenci sayısının yarısı kadar
Tablet	Öğrenci sayısının yarısı kadar



ÖĞRETMEN HAZIRLIK ÇALIŞMALARI:

Okunabilecek Kaynaklar:

Gürsakal, N. (2021). *Makine Öğrenmesi*. Dora Yayıncıları.

Microsoft. (2024). Makine öğrenmesi nedir?. Çevrimiçi: <https://azure.microsoft.com/tr-tr/resources/cloud-computing-dictionary/what-is-machine-learning-platform> (Erişim 27 Mayıs 2024).

Nabihev, V. (2021). *Yapay Zeka: Derin Öğrenme – Stratejili Oyunlar Örüntü Tanıma – Doğal Dil İşleme*. Seçkin Yayıncılık.

Shalev-Shwartz, S., & Ben-David, S. (2014). *Understanding machine learning: From theory to algorithms*. Cambridge University Press.

İzlenebilecek Kaynaklar:

Albert. (2023). Yapay Zeka ve Makine Öğrenmesi Nedir? Nasıl Robot Yapılır?. Çevrimiçi: <https://www.youtube.com/watch?v=F0OUXSKB4a8> (Erişim 27 Mayıs 2024).

Bebar Bilim. (2019). Makine Öğrenmesi Nedir? - Makineler Nasıl Öğrenir?. Çevrimiçi: <https://www.youtube.com/watch?v=YuhOCJ6FjC4> (Erişim 27 Mayıs 2024).

Fireship. (2022). Machine Learning Explained in 100 Seconds. Çevrimiçi: <https://www.youtube.com/watch?v=PeMlggyqz0Y> (Erişim 27 Mayıs 2024).

ATÖLYE HAZIRLIK ÇALIŞMALARI:

Öğretmen;

- Dersten önce ders materyallerini hazırlamalıdır.
- Yapay Zeka'nın Makine Öğrenmesi yaklaşımına ilişkin bilgi edinmelidir.
- Farklı Makine Öğrenmesi yöntemlerini kavramalıdır.
- Farklı Makine Öğrenmesi yöntemleri arasındaki farklılıklar konusunda yeterli seviyede bilgi edinmelidir.



KAYNAKÇA:

Deperlioğlu, Ö., & Köse, U. (2024). *Python ile Makine Öğrenmesi: Temel Kavramlar – Sınıflandırma – Regresyon – Kümeleme*. Seçkin Yayıncılık.

Köse, U., Özsoy, K., & Aksoy, B. (2023). *Yapay Zeka – Ortaokul ve Lise Kitapları*. Deneyap Teknoloji Atölyeleri, TÜBİTAK Yayıncılık.

Włodarczak, P. (2019). *Machine Learning and Its Applications*. CRC Press.

Bilgin, M. (2017). Gerçek Veri Setlerinde Klasik Makine Öğrenmesi Yöntemlerinin Performans Analizi. *Akademik Bilişim 2017 Konferansı*. Aksaray Üniversitesi, Aksaray.

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

Şahinarslan, F. V. (2019). *Makine Öğrenmesi Algoritmaları ile Nüfus Tahmini: Türkiye Örneği*. Doktora Tezi. İstanbul Teknik Üniversitesi, Sosyal Bilimler Enstitüsü.

GENEL BAKIŞ:

1) Harekete Geç: Öğrencilerden çevrelerinde gördükleri Yapay Zeka uygulamalarında algoritmaların öğrenme süreçlerinin ne şekilde gerçekleştiği hakkında fikirlerini paylaşmaları istenir ve öğrencilerden gelen dönütler, farkındalıkçı Makine Öğrenmesi yöntemlerine yönlendirecek şekilde yorumlanır.

2) Keşfet: Makine Öğrenmesi yaklaşımının kavramsal çerçevesine yönelik açıklamalar yapılır.

3) Üret: Makine Öğrenmesi altındaki yöntemlere ilişkin açıklamalar yapılır ve öğrencilerden Hareket Geç adımında vurgulanan uygulamalarındaki öğrenme süreçlerini tekrar düşünerek, açıklanan Makine Öğrenmesi yöntemleri özelinde tekrar yorumlamaları istenir.

4) İlerlet: Öğrencilerin gruplar halinde çalışarak farklı Makine Öğrenmesi yöntemlerinin kullanılabilirliği uygulamalar düşünerek, söz konusu uygulamalara dair açıklamalar yazmaları istenir.

5) Değerlendir: Yazılan açıklamalar ve derste öğrenilen bilgiler ışığında soru-cevap süreci üzerinden Makine Öğrenmesi yaklaşımı ve Makine Öğrenmesi yöntemleri genel hatlarıyla pekiştirilir.

UYGULAMA



1. ADIM: HAREKETE GEÇ

Bu adımda öğrencilerden çevrelerinde gördükleri Yapay Zeka uygulamalarında algoritmaların öğrenme süreçlerinin ne şekilde gerçekleştiği hakkında fikirlerini paylaşmaları istenir ve öğrencilerden gelen dönütler, farkındalığı Makine Öğrenmesi yöntemlerine yönlendirecek şekilde yorumlanır.

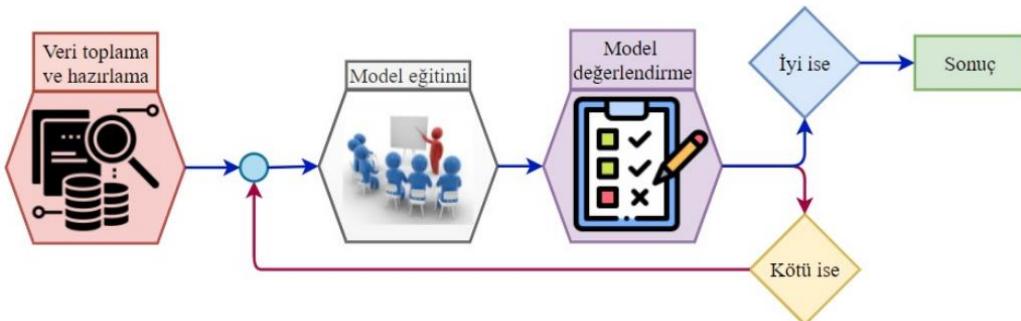


2. ADIM: KEŞFET

Bu adımda, Makine Öğrenmesi yaklaşımının kavramsal çerçevesine yönelik açıklamalar yapılır.

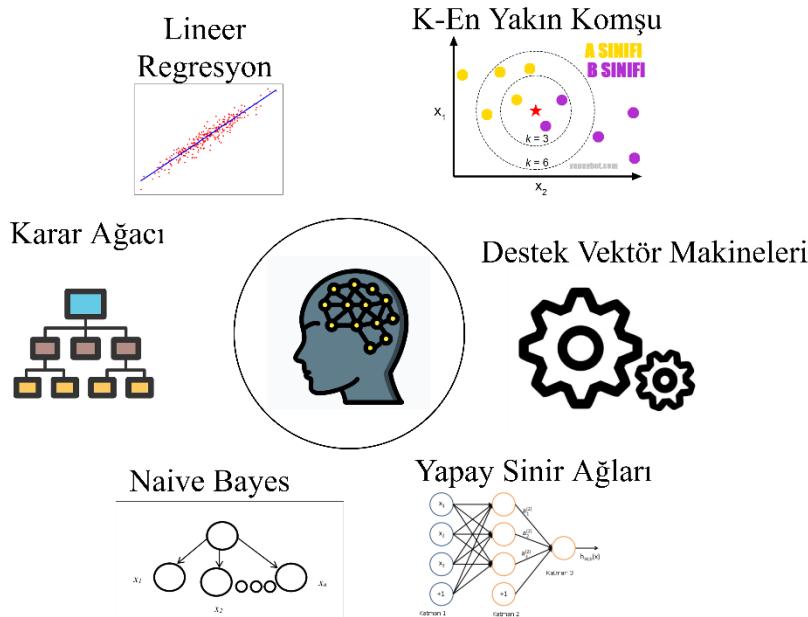
Makine Öğrenmesi Kavramı

Makine Öğrenmesi, önceki dersler kapsamında da ifade edildiği üzere (Bkz. 1. Hafta – 1. Gün – 2. Ders), Yapay Zeka'nın öğrenen algoritmalarına ev sahipliği yapan yaklaşımı verilen isimdir (Bir bakıma Yapay Zeka'nın alt-alanı olarak da ifade edilebilir). Makine Öğrenmesi en basit tabirle, bir bilgisayar tabanlı sistemin 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. 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; Köse vd., 2023). Şekil 12'de gösterildiği gibi Makine Öğrenmesi temel olarak dört aşamadan oluşmaktadır. İlk aşamada, cihazlardan veriler toplanarak ya da halihazırda oluşturulmuş veri setleri kullanılarak çözmek istenen probleme yönelik 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 doğrulu test edilir. Son aşamada ise, test verilerinden elde edilen sonuçlar değerlendirilir (Çevik, 2020; Włodarczak, 2019).



Şekil 12. Makine Öğrenmesi’nde temel aşamalar.

Makine Öğrenmesi'ne dayanan çok sayıda Yapay Zeka algoritması vardır. Bu noktada yapay sinir ağları, karar ağacı, naive-bayes algoritması gibi birçok algoritma Yapay Zeka uygulamalarında sıkılıkla tercih edilmektedir (Şekil 13). Her algoritmanın Makine Öğrenmesi'ni destekleyecek şekilde; kendine has çalışma mekanizmaları mevcuttur. Bu mekanizmalar farklı Makine Öğrenmesi algoritmaları arasında çeşitli avantaj ve dezavantajları beraberinde getirir. Kimi zaman daha basit yapıda Makine Öğrenmesi algoritmalarının basit düzeyde problemler için hızlı çözümler ürettiği gözlemlenirken daha zorlu problemler için daha yavaş çalışan ancak yüksek başarımlı sergileyen algoritmalar kullanılması daha uygundur.



Şekil 13. Yayın olarak kullanılan makine öğrenme algoritmaları.



3. ADIM: ÜRET

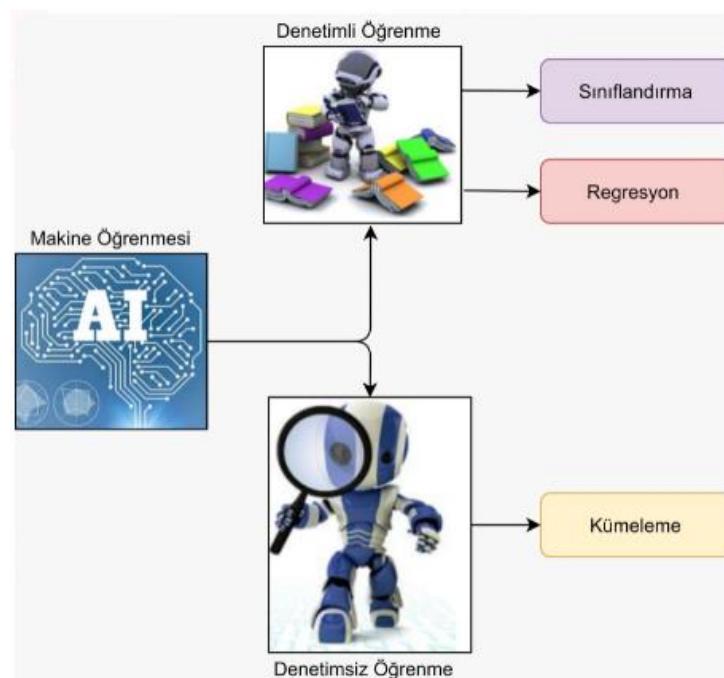
Üret adımında Makine Öğrenmesi altındaki yöntemlere ilişkin açıklamalar yapılır ve öğrencilerden Hareket Geç adımında vurgulanan uygulamalardaki öğrenme süreçlerini tekrar düşünerek, açıklanan Makine Öğrenmesi yöntemleri özelinde tekrar yorumlamaları istenir. Böylelikle öğrencilerin Makine Öğrenmesi yöntemleri arasındaki farklılıklarını düşündükleri uygulamalar bağlamında fark etmeleri sağlanır.

Makine Öğrenmesi Yöntemleri

Yapay Zeka uygulamalarında Makine Öğrenmesi çözümlerini oluşturmak için çeşitli yöntemler kullanılmaktadır. Bu yöntemlerin mantığını anlamak için öncelikli olarak Makine Öğrenmesi algoritmalarının çalışma mantığının tipik sistem olusunu gibi olduğunu kabul etmek gereklidir. Buna göre Makine Öğrenmesi algoritmaları bir problemin sebebinin oluşturan girdi parametrelerini sonuç olarak kabul edebileceğimiz çıktıyı ya da çıktıları oluşturmak için işlemektedir (Deperlioğlu, & Köse, 2024). Örneğin, hava durumu, mesafe ve zaman girdilerine (sebeplerine) göre alışverişe gidip gitmemeye çıktısını (sonucunu) işleyen bir problem tipik Makine Öğrenmesi problem yaklaşımı olarak yorumlanabilir. Bu aşamada, eğitim ve test

aşamalarından geçecek bir Makine Öğrenmesi algoritması için ilgili problemi temsil eden örnek verilerin toplanıp bir veri seti halinde kullanılması gerekmektedir. Problemi öğrenip ileri aşamalarda yeni durumlarda kullanılabilecek bir Makine Öğrenmesi modelinin oluşturulması adımları önceki derslerden bu yana işlenen Problem Modelleme ya da Makine Öğrenmesi’ndeki temel aşamalarla alakalıdır.

Makine Öğrenmesi çözümlerinin elde edilmesi için çeşitli yöntemler geliştirilmiştir. Bu yöntemlerden en bilinen iki tanesi denetimli (danışmanlı / supervised) öğrenme ve denetimsiz (danışmansız / unsupervised) öğrenmedir. Denetimli öğrenmede eğer girdi verileri karşısında elde edilen çıktılar etiketlerle / sınıflarla temsil ediliyorsa, bu tür problem çözümü sınıflandırma (classification) olarak isimlendirilmekte, eğer çıktılar sayılarla temsil ediliyorsa bu tür bir problem çözümü ise regresyon olarak kabul edilmektedir. Eğer problem ile ilgili veriler sadece girdi verilerini içeriyor ancak çıktılar yok ise bu durumda denetimsiz öğrenme kullanılmaktadır. Denetimsiz öğrenme, girdi verilerinin birbirlerine göre yakınlıklarını üzerinden çeşitli kümelere ayrılmasını sağlayan kümeleme (clustering) yöntemini içermektedir (Şahinarslan, 2019) (Şekil 14).



Şekil 14. Makine öğrenme yöntemleri.

Makine Öğrenmesi yöntemleri içerisinde denetimli ve denetimsiz öğrenme dışındaki bir diğer yöntem ise pekiştirmeli (reinforcement) öğrenme olarak bilinmektedir. Pekiştirmeli öğrenmede amaç, ödül ya da ceza dönütlerine göre öğrenen bir Makine Öğrenmesi modeli elde etmektir. Bu nedenle hedeflenen problem daha çok Makine Öğrenmesi sisteminin davranışlarına göre ödül ya da ceza değerleri veren bir yapıda modellenmektedir. Bu öğrenme şekli bir tür “yaşayarak öğrenme” akışına tekabül etmektedir (Deperlioğlu, & Köse, 2014; Włodarczak, 2019) (Şekil 15).



Şekil 15. Pekiştirmeli öğrenme.



4. ADIM: İLERLET

Öğrencilerin gruplar halinde çalışarak farklı Makine Öğrenmesi yöntemlerinin kullanılabildiği uygulamalar düşünerek, söz konusu uygulamalara dair açıklamalar yazmaları istenir. Söz konusu etkinlik için toplamda 10 dakikalık bir süre verilir.



5. ADIM: DEĞERLENDİRME

Yazılan açıklamalar ve derste öğrenilen bilgiler ışığında soru-cevap süreci üzerinden Makine Öğrenmesi yaklaşımı ve Makine Öğrenmesi yöntemleri genel hatlarıyla pekiştirilir.

Öğrencilere şu sorular sorulabilir:

- Makine Öğrenmesi'nin Yapay Zeka'nın ilerleyisi konusundaki rolü hakkında neler düşünüyorsunuz?
- Farklı Makine Öğrenmesi yöntemleri Yapay Zeka uygulamalarında bizlere ne gibi avantajlar sağlar?
- Makine Öğrenmesi Yapay Zeka ve insan işbirliği konusunda ne gibi etkilere sahiptir?

1. HAFTA – 2. GÜN – 3. DERS: VERİ, VERİ SETİ KAVRAMLARI NELERDİR?

DERS PLANI

DERS ETİKETLERİ



KONU: Veri, Veri Seti Kavramları Nelerdir?



SINIF: 12-14 Yaş



SÜRE: 60 dk.



HAZIR BULUNUŞLUK: Temel İngilizce eğitimi, Yapay Zeka kavramı hakkında temel bilgi, Makine Öğrenmesi hakkında farkındalık



ANAHTAR KELİMELER: Yapay Zeka, Algoritma, Makine Öğrenmesi, Veri, Veri Seti



BECERİLER:



Tahmin



Sorgulama



Günlük hayat bağlantısı kurma



KAZANIMLAR:

Bu derste öğrenciler;

- Veri kavramı hakkında bilgi sahibi olur.
- Veri saklama yaklaşımları hakkında bilgi sahibi olur.
- Veri seti kavramı hakkında bilgi sahibi olur.
- Veri seti kullanımının Yapay Zeka ve Makine Öğrenmesi uygulamalarındaki rolü hakkında bilgi sahibi olur.
- Yapay Zeka ile çözmek istenen problemlere uyumlu veri seti hazırlaması gerekiği konusunda farkındalık sahibi olur.



DERS MATERİYALLERİ:

Malzeme	Miktar
Yazı Tahtası	Öğrenci sayısı ve sınıf büyüklüğüne göre en az bir adet
Projeksiyon	Öğrenci sayısı ve sınıf büyüklüğüne göre en az bir adet
Defter, Kalem	Öğrenci sayısı kadar
Bilgisayar	Öğrenci sayısının yarısı kadar
Tablet	Öğrenci sayısının yarısı kadar



ÖĞRETMEN HAZIRLIK ÇALIŞMALARI:

Okunabilecek Kaynaklar:

Alan, A., & Karabatak, M. (2020). Veri seti-sınıflandırma ilişkisinde performansa etki eden faktörlerin değerlendirilmesi. *Fırat Üniversitesi Mühendislik Bilimleri Dergisi*, 32(2), 531-540.

Loukides, M. (2011). *What is data science?*. O'Reilly Media, Inc.

Quiñonero-Candela, J., Sugiyama, M., Schwaighofer, A., & Lawrence, N. D. (Eds.). (2022). *Dataset shift in machine learning*. MIT Press.

İzlenebilecek Kaynaklar:

Data Science Earth. (2021). Veri seti Nedir?. Çevirmiçi:

https://www.youtube.com/watch?v=_exGfkr4iSI (Erişim 27 Mayıs 2024).

Eye on Tech. (2022). What is Data Collection? How Data is Collected. Çevirmiçi: <https://www.youtube.com/watch?v=Lb6Gi6IR-Kc> (Erişim 27 Mayıs 2024).

Knowledge Base. (2021). Data Sets (datasets). Çevirmiçi:

<https://www.youtube.com/watch?v=PvdfGiUXokU> (Erişim 27 Mayıs 2024).

ATÖLYE HAZIRLIK ÇALIŞMALARI:

Öğretmen;

- Dersten önce ders materyallerini hazırlamalıdır.
- Veri, veri seti ve benzeri kavramlar hakkında bilgi edinmelidir.

- Veri setlerinin genel anlamda Yapay Zeka, spesifik olarak ise Makine Öğrenmesi uygulamalarındaki rolü hakkında farkındalık ve bilgi sahibi olmalıdır.



KAYNAKÇA:

- Arslan, İ. (2019). *Python ile Veri Bilimi*. Pusula Yayıncılık.
- Banachewicz, K., & Massaron, L. (2022). *The Kaggle Book: Data analysis and machine learning for competitive data science*. Packt Publishing Ltd.
- Larose, D. T., & Larose, C. D. (2014). *Discovering knowledge in data: An introduction to data mining* (Vol. 4). John Wiley & Sons.
- Provost, F., & Fawcett, T. (2013). Data science and its relationship to big data and data-driven decision making. *Big Data*, 1(1), 51-59.
- Silahtaroğlu, G. (2008). *Veri madenciliği*. Papatya Yayıncılık.

GENEL BAKIŞ:

- 1) Harekete Geç:** Öğrencilerden çevrelerinde gördükleri Yapay Zeka uygulamalarında hangi uygulamaların ve hangi sebep-sonuç (girdi-çıktı) parametrelerinin kullanılıyor olabileceği hakkında düşünüp görüşlerini paylaşmaları istenir.
- 2) Keşfet:** Veri, dosya, veri tabanı gibi kavramlar hakkında açıklamalar yapılır ve öğrencilerin Hareket Geç adımında verdikleri dönütler hatırlanarak kavamlara dair pekiştirmeler yapılır.
- 3) Üret:** Veri seti kavramı açıklanır ve veri setlerinin Makine Öğrenmesi ile olan bağlantısı irdelenir.
- 4) İlerlet:** Öğrencilerin bireysel olarak düşündükleri bir problem için veri seti tasarımları oluşturmaları ve bu veri setine ilişkin detayları yazmaları istenir.
- 5) Değerlendir:** Veri seti bağlamında yazılan detaylar ve derste öğrenilen bilgilerden hareketle soru-cevap süreci gerçekleştirilir ve veri, veri seti kavamları genel hatlarıyla pekiştirilir.

UYGULAMA



1. ADIM: HAREKETE GEÇ

Harekete Geç adımında öğrencilerden çevrelerinde gördükleri Yapay Zeka uygulamalarında hangi uygulamaların ve hangi sebep-sonuç (girdi-çıktı) parametrelerinin kullanılıyor olabileceği hakkında düşünüp görüşlerini paylaşmaları istenir.



2. ADIM: KEŞFET

Veri, dosya, veri tabanı gibi kavamlar hakkında açıklamalar yapılır ve öğrencilerin Hareket Geç adımımda verdikleri dönütler hatırlanarak kavamlara dair pekiştirmeler yapılır.

Veri Kavramı

Veri kavramı genellikle durumlar ya da olaylara dair çıkarımlar oluşturmak adına kullanılan bilgi olarak ifade edilebilir. Bununla birlikte bir başka alternatif bakış açısıyla; bilginin dijital sistemlerde temsil edilen hali olarak da kabul edilebilmektedir. Veri özellikle günümüz teknolojisinde önemli bir yer tutmakla beraber, algoritmalarla ve Yapay Zeka gibi gelişmiş problem çözümleyici yapılarda olmazsa olmaz bir unsur olarak düşünülmektedir. Gününüz şartları altında teknolojinin ilerleyişi verinin önemini artırmakta ve yine teknolojiyle birlikte ortaya çıkan yeni problemlere yönelik çözüm araçlarında sebep-sonuç ilişkilerini kurmakta kullanılmaktadır. Bu nedenle veriler çeşitli araçlarla toplanmakta, geleneksel araçlarla toplanmış bilgiler verilere dönüştürülmemekte ve toplanan ya da işlenen veriler dijital ortamlarda saklanmaktadır.

Bilgisayar ve yazılım teknolojilerinde meydana gelen ilerlemeler, verilerin dijital ortamda tutulma şekillerini de etkilemiştir. Bilgisayar sistemlerinin çözmeye çalıştığı problemlerin karmaşıklaması, yeni çıkan dijital teknolojilerinin veri üretimin artışında rol oynaması, bilginin veri düzeyinde nasıl depolanacağı ve nasıl işleneceği gibi soruların güncellliğini korumasına sebep olmuştur. Buna göre verinin temsil ediliş şekli genel hatlarıyla şu yaklaşımlar üzerinden ilerlemiştir (Silahtaroğlu, 2008; Provost, & Fawcett, 2013; Larose, & Larose, 2014):

- **Dosya:** Dosyalar ortak nitelikte ve işlevdeki verilerin bir arada tutulup işlenmesini sağlayan veri araçları olarak ortaya çıkmıştır. Dosya yapılanması halen yaygınlığını korumakta, farklı türden verilerin kullanıcı nezdinde organize edilmesi ve yine farklı yazılımlar tarafından kullanılacak şekilde depolanması, dosya adı verilen yapılar aracılığıyla gerçekleşmektedir. Dosyalar işletim sistemlerinin merkezi veri depolama unsurları olarak da kabul edilmektedir.
- **Veritabanı:** Fazla mikardaki veriye erişim ve bu verilerin işlenmesi çok sayıda dosyanın kullanımını sayesinde verimli olmamış; bu nedenle tutulacak verilerin parçalı yapılarda ve hatta ilişkisel olarak saklanıp kullanılabilceği veritabanı yapıları geliştirilmiştir. Veritabanları, arkapanda organize bir biçimde tutulması gereken ve aktif bir biçimde veri oluşturulması ya da işlenmesi süreçleriyle çalışan yazılımların ve çevrimiçi platformların olmazsa olmaz unsurları arasında yer almaktadır.
- **Veri ambarı:** Veri ambarları, temel işlevi sadece veri depolama yönünde ağırlık kazanan veritabanlarının bir bakıma veri analizi yetenekleri kazanmış hali olarak tasarlanmıştır.
- **Veri Madenciliği Süreçleri ve Büyük Veri:** Günümüzde elde tutulan veriden yeni bilgi elde edebilmek adına Makine Öğrenmesi ile de ortak paydada buluşan Veri Madenciliği (Data Mining) çözümleri yaygın bir biçimde kullanılmaktadır. Bazı Makine Öğrenmesi algoritmaları aynı zamanda Veri Madenciliği algoritması olarak da kabul edilmektedir. Bu tür araçlar yeni bilgi keşfi için etkin çözümler ortaya koymakta ve hatta günümüzde gerçek zamanlı olarak, farklı kaynaklardan, yüksek miktarda, değerli verilerin oluşturduğu ve ileri düzey veri işleme adımlarına ihtiyaç duyan Büyük Veri (Big Data) için kullanılmaktadır. Günümüzde Yapay Zeka (özellikle Makine

Öğrenmesi / Derin Öğrenme) ve Veri Madenciliği Büyük Veri uygulamalarının odak noktası konumundadır.

Verilerin Yapay Zeka gibi ileri düzey uygulamalara hazır hale getirilmesi günümüzde Veri Bilimi (Data Science) alanının çalışmaları içerisine girmektedir. Veri Bilimi çalışmaları verilerin etkili bir biçimde işlenebilmesi için mümkün olan en temel veri depolama yaklaşımlarına başvurmaktadır. Bu amaçla verilerin parametreler eşliğinde tutulabilmesini sağlayan dosya türleri, “veri seti” kavramının ortaya çıkışında etkili olmuştur. Yapay Zeka uygulamalarında sıkılıkla dile getirilen veri seti, aslında Makine Öğrenmesi tekniklerinin eğitim ve test gibi aşamalarda kullanabileceğİ veri bütünlere karşılık gelmektedir (Arslan, 2019; Provost, & Fawcett, 2013).



3. ADIM: ÜRET

Üret adımda Veri seti kavramı açıklanır ve veri setlerinin Makine Öğrenmesi ile olan bağlantısı irdelenir.

Veri Seti Kavramı

Veri seti özellikle Makine Öğrenmesi algoritmaları üzerinden oluşturulan modellerin hedef problem için eğitilmesi adına organize bir biçimde bir araya getirilen örnekler bütünü olarak ifade edilebilmektedir. Buna göre tipik bir veri seti, hedef problemin sebep ve sonuçlarını (girdiler ve çıktılarını) dikkate alarak (Bkz. 1. Hafta – 1. Gün – 1. Ders ve 1. Hafta – 2. Gün – 2. Ders) eğitilmiş Makine Öğrenmesi modellerini elde etmek için kullanılabilecek veri bütünlere tekabül eder (Banachewicz, & Massaron, 2022; Larose, & Larose, 2014). Çok daha pratik bir bakış açısıyla veri seti, satırlar yönünde örnekler ve sütunlar yönünde parametrelerin yer aldığı bir Excel tablosuna benzetilebilir (Şekil 16).

	A	B	C	D	E	F	G	H	I
1	duration: continuous.	protocol_type: symbolic.	service: symbolic.	flag: symbolic.	src_bytes: continuous.	dst_bytes: continuous.	land: symbolic.	wrong_fragment: continuous.	label
2	0 tcp	http	SF		181	5450	0	0	normal.
3	0 tcp	http	SF		239	486	0	0	normal.
4	0 tcp	http	SF		235	1337	0	0	normal.
5	0 tcp	http	SF		219	1337	0	0	normal.
6	0 tcp	http	SF		217	2032	0	0	normal.
7	0 tcp	http	SF		217	2032	0	0	normal.
8	0 tcp	http	SF		212	1940	0	0	normal.
9	0 tcp	http	SF		159	4087	0	0	normal.
10	0 tcp	http	SF		210	151	0	0	normal.
11	0 tcp	http	SF		212	786	0	0	normal.
12	0 tcp	http	SF		210	624	0	0	normal.
13	0 tcp	http	SF		177	1985	0	0	normal.
14	0 tcp	http	SF		222	773	0	0	normal.
15	0 tcp	http	SF		256	1169	0	0	normal.
16	0 tcp	http	SF		241	259	0	0	normal.
17	0 tcp	http	SF		260	1837	0	0	normal.
18	0 tcp	http	SF		241	261	0	0	normal.
19	0 tcp	http	SF		257	818	0	0	normal.
20	0 tcp	http	SF		233	255	0	0	normal.
21	0 tcp	http	SF		223	504	0	0	normal.

Şekil 16. Excel dosyası ortamında bir veri seti.

Veri setlerinin saklanmasında Excel dosya türleri (xls,xlsx,csv) kullanılabildiği gibi, benzeri organizasyonu sağlayan alternatif dosya türleri de tercih edilir. Problem türüne göre görüntü, ses gibi veri türleri içeren dosyalar da veri seti olarak kabul edilmekte; geliştirilen Makine Öğrenmesi uygulamalarında algoritmaların kullanabileceği doğrultuda işlemelere tabi tutulmaktadır.

Veri Setinin Organizasyonu ve Kullanımı

Veri setlerinin organizasyonu hedef probleme, teknolojik altyapıya ve kullanılacak programlama diline göre gerçekleştirilmektedir. Yazılan Makine Öğrenmesi kodları, denetimli öğrenmeye dayalı problemlerde girdi verileri ayrı bir veri seti dosyasında, çıktı verileri ayrı bir veri seti dosyasından okuyabilmekte ya da geliştirilen uygulama veri setini Makine Öğrenmesi algoritması için bu yönde parçalayabilmektedir (Benzer şekilde Makine Öğrenmesi süreçleri de veri seti dosyası üzerinde değişiklikler yapabilmekte ya da yeni bir dosya oluşturarak problem çözüm aşamalarını destekleyebilmektedir). Denetimsiz öğrenmede ise sadece girdi verilerini içeren veri setleri kullanılmakta; Makine Öğrenmesi aşamaları sonucunda elde edilen kümeler (çıktı verileri) veri setlerinde çıktı parametrelerini temsil edecek şekilde yazılmaktadır (Böylelikle denetimsiz öğrenmenin bir bakıma sadece girdileri bilinen bir problem için küme / sınıf bazlı çıktı verileri oluşturduğu anlaşılmaktadır).

Makine Öğrenmesi sistemleri kurulurken, probleme özgü olarak tek bir veri seti kullanabilecegi gibi, çeşitli veri setlerinin sistemi besleyen bir yaklaşım ile daha gelişmiş Makine Öğrenmesi modelleri de kurulabilmektedir. Örneğin, bir hastalık teşhisini probleminde semptomlarla hastalık durumunu belirten spesifik bir veri seti kullanılabılırken, hastaların farklı tahlil sonuçlarını kullanarak hastalık teşhisinde bulunan, ardından teşhise göre tedavi planlaması da önerilebilir bir sistemde, kurulacak yapı birden fazla veri setinin yine birden fazla Makine Öğrenmesi algoritmasıyla karşılandığı gelişmiş bir karar destek sistemi olarak tasarlanabilmektedir. Kimi zaman bir veri setinin tekabül ettiği çıktı verileri (sonuçlar) başka bir veri setindeki herhangi bir girdi olarak da kullanılabilmektedir. Bu tür kullanım şekilleri veri setinin “çevrimiçi” dosyalardan ya da “veritabanlarından” okunarak işletilmesi yönünde desteklenebileceği gibi, kurulan Yapay Zeka sistemi “gerçek zamanlı” yani anlık veriyi kullanarak arkaplanda veri seti halinde değerlendiren bir mimariye dayandırılarak da geliştirilebilmektedir. Bu tür yaklaşımlar tamamen problem modelleme ve problem zorluğu ile veri toplama ve işleme aşamalarında ihtiyaç duyulan mimari yapılanmalarına göre belirlenmek zorundadır.



4. ADIM: İLERLET

Öğrencilerin bireysel olarak düşündükleri bir problem için veri seti tasarımları ve bu veri setine ilişkin detayları yazmaları istenir. Bu etkinlik için 10 dakikalık bir süre ayrılr.



5. ADIM: DEĞERLENDİRME

Veri seti bağlamında yazılan detaylar ve derste öğrenilen bilgilerden hareketle soru-cevap süreci gerçekleştiriliyor ve veri, veri seti kavramları genel hatlarıyla pekiştiriliyor.

Öğrencilere şu sorular sorulabilir:

- Verilerin Yapay Zeka uygulamalarındaki başarılara olan etkileri hakkında neler düşünüyorsunuz?
- Veri setlerinin oluşturulması ve problemlere göre farklı şekilde organize edilmesi Makine Öğrenmesi uygulamaları açısından ne gibi faydalar sağlar?
- Büyük Veri ve gelecekte Yapay Zeka ile verilerin kullanımı konusundaki gelişmeler hakkında fikirleriniz nelerdir?

1. HAFTA – 3. GÜN – 1. DERS: PANDAS PYTHON KÜTÜPHANESİ

DERS PLANI

DERS ETİKETLERİ



KONU: Pandas Python Kütüphanesi



SINIF: 12-14 Yaş



SÜRE: 60 dk.



HAZIR BULUNUŞLUK: Temel İngilizce eğitimi, Yapay Zeka farkındalığı, Programlamaya yönelik temel bilgi



ANAHTAR KELİMELER: Yapay Zeka, Kodlama, Python, Pandas



BECERİLER:



Tahmin



Sorgulama



Günlük hayat bağlantısı kurma



KAZANIMLAR:

Bu derste öğrenciler;

- Pandas Python kütüphanesi hakkında bilgi sahibi olur.
- Pandas ile gerçekleştirilebilecek temel işlemler hakkında bilgi ve beceri sahibi olur.
- Python ile Yapay Zeka uygulamaları geliştirme aşamasında kullanılabilen Pandas fonksiyonları hakkında bilgi ve beceri sahibi olur.
- Yapay Zeka uygulamaları için verilerin Pandas kullanılarak nasıl incelenileceği ve hazırlanacağı hakkında farkındalık, bilgi ve beceri sahibi olur.



DERS MATERİYALLERİ:

Malzeme	Miktar
Yazı Tahtası	Öğrenci sayısı ve sınıf büyüklüğüne göre en az bir adet
Projeksiyon	Öğrenci sayısı ve sınıf büyüklüğüne göre en az bir adet
Defter, Kalem	Öğrenci sayısı kadar
Bilgisayar	Öğrenci sayısının yarısı kadar
Tablet	Öğrenci sayısının yarısı kadar
Python Anaconda Framework	Her bilgisayarda kurulu olacak şekilde



ÖĞRETMEN HAZIRLIK ÇALIŞMALARI:

Okunabilecek Kaynaklar:

W3 Schools. (2024). *Pandas Tutorial*. W3 Schools Tutorials. Çevrimiçi: <https://www.w3schools.com/python/pandas/default.asp> (Erişim 5 Haziran 2024).

Kaggle. (2024). Pandas Lessons. Çevrimiçi: <https://www.kaggle.com/learn/pandas> (Erişim 5 Haziran 2024).

İzlenebilecek Kaynaklar:

Python Programmer. (2019). LEARN PANDAS in about 10 minutes! A great python module for Data Science!. Çevrimiçi: https://www.youtube.com/watch?v=iGFdh6_FePU (Erişim 6 Haziran 2024).

Çelik, A. (2024). Sıfırdan Pandas Dersleri (Python Pandas Veri Analizi). Çevrimiçi: <https://www.youtube.com/watch?v=l0dHuJX9M-Y&list=PL1i2Llx7XoAU63M36shug4E1mHDzdFUX1> (Erişim 6 Haziran 2024).

ATÖLYE HAZIRLIK ÇALIŞMALARI:

Öğretmen;

- Dersten önce ders materyallerini hazırlamalıdır.
- Python Pandas kütüphanesi hakkında bilgi edinmelidir.

- Pandas ile veri analizi ve hazırlığı süreçleri konusunda bilgi-beceri sahibi olmalıdır.



KAYNAKÇA:

Chen, D. Y. (2017). *Pandas for everyone: Python data analysis*. Addison-Wesley Professional.

Molin, S. (2019). *Hands-On Data Analysis with Pandas: Efficiently perform data collection, wrangling, analysis, and visualization using Python*. Packt Publishing Ltd.

GENEL BAKIŞ:

- 1) Harekete Geç:** Öğrencilere Python Pandas kütüphanesi hakkında bilgi sahibi olup olmadıkları sorulur ve Pandas kütüphanesi hakkında temel bilgiler verilir.
- 2) Keşfet:** Pandas kütüphanesi ile series ve dataframe gibi veri modelleri üzerinden temel veri işlemleri hakkında bilgiler verilir.
- 3) Üret:** Pandas kütüphanesi ile dosya tabanlı işlemler ve diğer alternatif veri işleme süreçleri konusunda bilgiler verilir.
- 4) İlerlet:** Öğrencilerin bireysel olarak Pandas kütüphanesi ile uygulamalar gerçekleştirmeleri istenir.
- 5) Değerlendir:** Derste öğrenilen bilgilerden hareketle soru-cevap süreci gerçekleştirilir ve Python Pandas kütüphanesi kullanımı genel hatlarıyla pekiştirilir.

UYGULAMA



1. ADIM: HAREKETE GEÇ

Harekete Geç adımında öğrencilere Python Pandas kütüphanesi hakkında bilgi sahibi olup olmadıkları sorulur ve Pandas kütüphanesi hakkında temel bilgiler verilir.

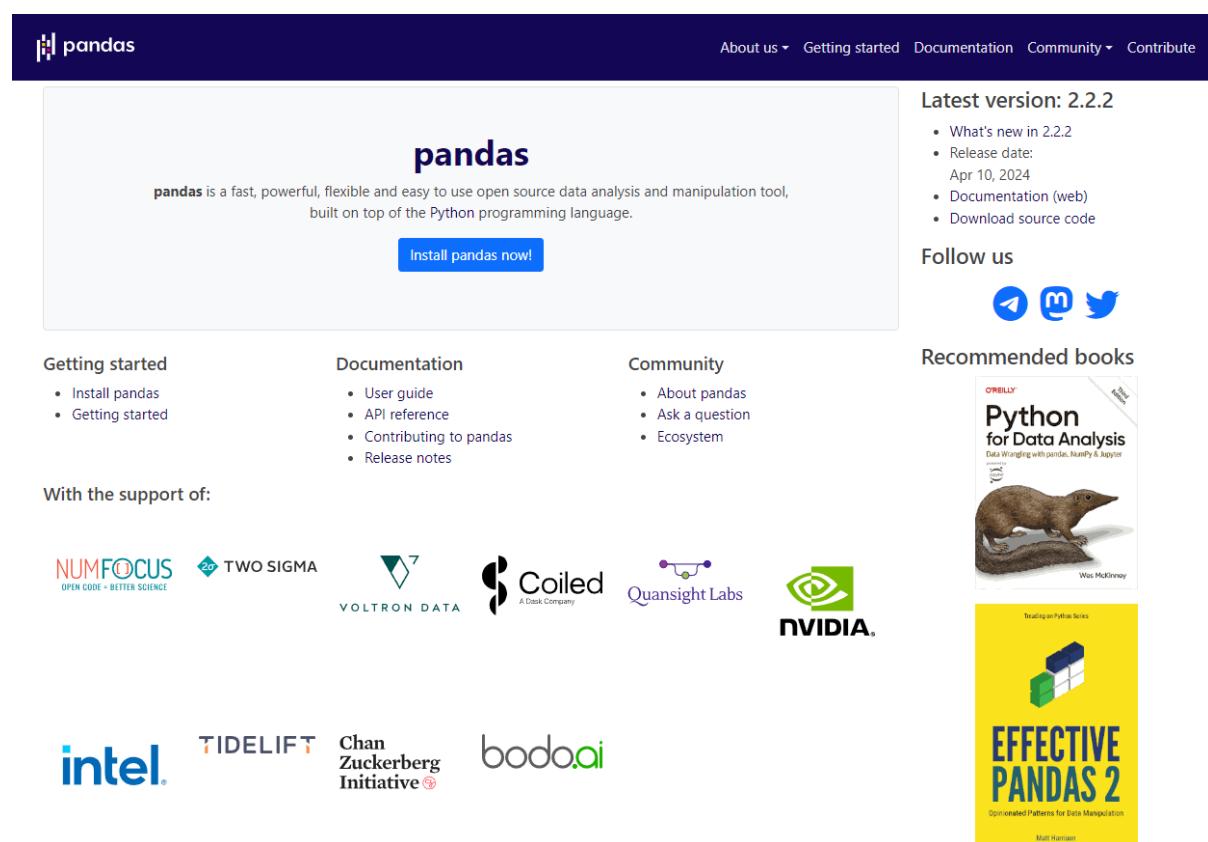
Öğrencilerin temel Python kodlarına ilişkin bildiklerini pekiştirmek ya da ders süreçlerinde öğrendiklerini desteklemek adına bilmeleri gereken temel Python programlama dili altyapısı için şu kaynakların incelenmesi önerilebilir:

- W3 Schools Dersleri: <https://www.w3schools.com/python/>
- Geeks for Geeks Dersleri: <https://www.geeksforgeeks.org/python-basics/>
- Python'a Giriş – Sıfırdan Python Dersleri:
https://www.youtube.com/watch?v=t176iXgG5PI&list=PL3kMAPso9YQ1Ls-5uTTIWWMkJoF_vyj5J

Pandas Kütüphanesi

Pandas ilk olarak 2008 yılında yayınlanmış olan popüler bir Python kütüphanesidir. Halen günümüz güncellemeleri ile beraber gelişimini sürdürden Pandas, veri analizi ve işleme süreçleri için sıkılıkla tercih edilen ve bu nedenle Veri Bilimi, Yapay Zeka gibi ileri düzey uygulamalarda kullanılan bir kütüphane yapısıdır (Chen, 2017). Pandas'ın sunduğu olanaklar, kendine has çeşitli veri modelleri ve kapsayıcı işlevler üzerinden derinlemesine veri analizlerinin ve manipülasyonlarının kolaylıklar gerçekleştirmesini mümkün kılmaktadır. Pandas, diğer popüler Python kütüphaneleriyle bir arada kullanılmak suretiyle uygulamalarda işleme tabi tutulan verilerin ve veri setlerinin etkili bir biçimde kullanılmasını, dönüştürülmesini, görselleştirilmesini ve hedef uygulamaların veri odaklı güçlü altyapılar üzerinden inşa edilmesini oldukça kolaylaştırmaktadır (Molin, 2019).

Pandas sunmuş olduğu olanaklar ve ileri düzey uygulamalarda gösterdiği esneklikler sayesinde önemli yazılım sistemlerine referans olmuş ve geniş kitlelerce destek görmüştür. Pandas kütüphanesine ilişkin dokümantasyonlar, haberler ve topluluk etkileşimleri <https://pandas.pydata.org/> Web sitesi üzerinden sunulmaktadır (Şekil 17).



Şekil 17. Python Pandas kütüphanesi Web sitesi.

Pandas kütüphanesi Python ortamında “**import pandas**” komutu ile çağrılır ve hatta kullanım kolaylığı sağlamak adına “**pd**” kısaltması ile desteklenir (Şekil 18).

```

1 # -*- coding: utf-8 -*-
2 """
3 Created on Tue Jun 6 03:24:58 2024
4
5 @author: UK
6 """
7 import pandas as pd
8

```

Şekil 18. Python Pandas kütüphanesinin çağrılması.

Pandas kütüphanesi Python Anaconda Framework içerisinde hazır bir şekilde gelmektedir. Dolayısıyla Pandas kütüphanesi doğrudan Python kod çağrıları ile kullanılabilir. Bunun dışında kullanıcı taraflı manuel Python kurulumlarında Pandas kütüphanesi kurulumu için https://pandas.pydata.org/getting_started.html adresinde sunulan kurulum (installation) açıklamaları takip edilebilir.



2. ADIM: KEŞFET

Pandas kütüphanesi ile series ve dataframe gibi veri modelleri üzerinden temel veri işlemleri hakkında bilgiler verilir.

Series ve Dataframe Veri Modellerinin Kullanımı

Pandas ile veri analizi ve işleme aşamalarına temel teşkil eden series ve dataframe veri modelleri kullanılmaktadır. Series ve dataframe veri modelleri, standart Python veri yapıları ve modellerinden farklı olarak verilerin tutulduğu satır ve sütun bileşenleri üzerinde daha esnek düzenlemeler sağlamaktadır.

Öğrenciler Şekil 19'da gösterilen kodları yazarak series ve dataframe değişkenleri oluşturur. Ekrana yansıtılan çıktılar yorumlanır, indeks ve sütun yapılarının rolü, seri veri modelinin dataframe içerisinde kullanımı yorumlanır. Öğrencilerin kodlar üzerinde değişiklikler yapmaları istenir ve elde edilen farklı çıktıların gözlemlenmesi sağlanır.

<pre> 9 s = pd.Series([10, 34, 300, "Python"]) 10 df = pd.DataFrame(11 { 12 "Column1": 10.0, 13 "Column2": pd.Timestamp("20130102"), 14 "Column3": pd.Series(1, index=list(range(4)), dtype="float32"), 15 "Column4": s, 16 "Column5": pd.Categorical(["class1", "class2", "class3", "diğer"]), 17 "Deneme Sütunu": "T3", 18 }) 19 20 21 print(s) 22 print(df) </pre>	<pre> 10 1 34 2 300 3 Python dtype: object Column1 Column2 Column3 Column4 Column5 Deneme Sütunu 0 10.0 2013-01-02 1.0 10 class1 T3 1 10.0 2013-01-02 1.0 34 class2 T3 2 10.0 2013-01-02 1.0 300 class3 T3 3 10.0 2013-01-02 1.0 Python diğer T3 </pre>
--	--

(a)

(b)

Şekil 19. (a) Series ve dataframe oluşturma kodu (b) kodların çalıştırılması sonucu konsol görüntüsü.

Öğrenciler Şekil 20'de gösterilen kodları yazarak **df** değişkenin ilk 2 ve son 2 verileriyle birlikte “indeks” ve “sütun” değerlerinin ekrana görüntülenmesini sağlar.

```

26 print(df.head(2))
27 print(df.tail(2))
28
29 print(df.index)
30 print(df.columns)

          Column1    Column2    Column3    Column4    Column5 Deneme Sütunu
0      10.0  2013-01-02      1.0      10  class1      T3
1      10.0  2013-01-02      1.0      34  class2      T3
          Column1    Column2    Column3    Column4    Column5 Deneme Sütunu
2      10.0  2013-01-02      1.0     300  class3      T3
3      10.0  2013-01-02      1.0   Python  diğer      T3
Int64Index([0, 1, 2, 3], dtype='int64')
Index(['Column1', 'Column2', 'Column3', 'Column4', 'Column5',
       'Deneme Sütunu'], dtype='object')

```

(a)

(b)

Şekil 20. (a) df değişkeni verilerinin ve indeks-sütun değerlerinin görüntülenmesine ilişkin kodlar (b) kodların çalıştırılması sonucu konsol görüntüsü.

Öğrencilerle Şekil 21'deki gibi sayısal içerikli yeni bir dataframe değişkeni oluşturulur ve **“describe”** fonksiyonu yardımıyla dataframe değişkenin sayısal detaylarına ilişkin genel bulgular elde edilir.

```

34 df2 = pd.DataFrame(
35     {
36         "Column1": pd.Series([6.565, 3.343, 44.32, -13.233, 78.55, 324, 34]),
37         "Column2": pd.Series([22.55, 41.23, -4.233, -93.21, 7348.5, 43.4, -30]),
38         "Column3": pd.Series([61.03, 77.43, 43.112, 90.1323, 5, 4, 900]),
39         "Column4": pd.Series([-6.323, 32.99, 111, 0.87, -121.4, 30.3, 1134]),
40         "Column5": pd.Series([6.111, 0.555, 4.3, -4.25, 8.35, 51.1, -904]),
41     }
42
43
44 print(df2.describe())

```

(a)

	Column1	Column2	Column3	Column4	Column5
count	7.000000	7.000000	7.000000	7.000000	7.000000
mean	68.220714	1046.891000	168.672043	168.776714	-119.690571
std	116.845547	2779.157093	324.189177	431.212065	346.335653
min	-13.233000	-93.210000	4.000000	-121.400000	-904.000000
25%	4.954000	-17.116500	24.056000	-2.726500	-1.847500
50%	34.000000	22.550000	61.030000	30.300000	4.300000
75%	61.435000	42.315000	83.781150	71.995000	7.230500
max	324.000000	7348.500000	900.000000	1134.000000	51.100000

(b)

Şekil 21. (a) df2 değişkeni tanımı ve describe fonksiyonu kullanımına ilişkin kodlar (b) kodların çalıştırılması sonucu konsol görüntüsü.

Şekil 21'de görüldüğü üzere, describe fonksiyonu ile dataframe değişkeninde yer alan sayısal değerlerin frekansı, ortalaması, en küçük, en büyük değerler ve standart sapma gibi tanımlayıcı bilgiler elde edilebilmektedir.

Dataframe değişkenleri oluştururken sütun ve satır indeks etiketleri de manuel bir biçimde tanımlanabilir. Bu amaçla öğrenciler aşağıdaki kodları yazar ve kodun çalıştırılması ardından Spyder editöründeki “Variable Explorer” bölümünü kullanarak df_player dataframe değişkeninin içeriğini görüntüler. Bu şekilde Variable Explorer kullanımı da tecrübe edilir (Şekil 22):

```
player_list = [['M. George', 29, 77, 1911000],
               ['T. Jackson', 34, 72, 1456000],
               ['H. Williams', 32, 80, 1899000]]

df_player = pd.DataFrame(player_list, columns=['Name', 'Age', 'Weight', 'Salary'])

df_player.index = ['player1', 'player2', 'player3']
```

Index	Name	Age	Weight	Salary
player1	M. George	29	77	1911000
player2	T. Jackson	34	72	1456000
player3	H. Williams	32	80	1899000

Şekil 22. df_player değişkeninin Variable Explorer ortamında görüntülenmesi.

df2 değişkenin transpozesi ve “Column3” sütununa göre sıralanması sağlanır (Şekil 23).

```
45 print("df2 dataframe değişkeninin transpozesi:")
46 print(df2.T)
47 print("df2 dataframe değişkeninin Column3 sütunu ile küçükten büyüğe sıralanması:")
48 print(df2.sort_values(by="Column3"))
```

(a)

	0	1	2	3	4	5	6
Column1	6.565	3.343	44.320	-13.2330	78.55	324.0	34.0
Column2	22.550	41.230	-4.233	-93.2100	7348.50	43.4	-30.0
Column3	61.030	77.430	43.112	90.1323	5.00	4.0	900.0
Column4	-6.323	32.990	111.000	0.8700	-121.40	30.3	1134.0
Column5	6.111	0.555	4.300	-4.2500	8.35	51.1	-904.0
	Column1	Column2	Column3	Column4	Column5		
5	324.000	43.400	4.0000	30.300	51.100		
4	78.550	7348.500	5.0000	-121.400	8.350		
2	44.320	-4.233	43.1120	111.000	4.300		
0	6.565	22.550	61.0300	-6.323	6.111		
1	3.343	41.230	77.4300	32.990	0.555		
3	-13.233	-93.210	90.1323	0.870	-4.250		
6	34.000	-30.000	900.0000	1134.000	-904.000		

(b)

Şekil 23. (a) df2 değişkeni transpoze ve sütun odaklı sıralanmasına ilişkin kodlar
(b) kodların çalıştırılması sonucu konsol görüntüsü.

Dataframe Bölümlerine Ulaşmak

Dataframe değişkenlerinin belirli bölümlerine ulaşmak için köşeli parantezler ([]) arasında çeşitli belirteçler (etiket, iki nokta üst üste karakteri) kullanılabilir. Buna göre:

- **df["sutun ismi"]:** İlgili dataframe değişkenin sadece belirtilen sütunun gösterdiği satırların elde edilmesini sağlar.
- **df[0:3]:** İlgili dataframe değişkeninin 1. satırından başlayarak 4. satıra kadar olan (4. satır dahil değil) bölümünün elde edilmesini sağlar.
- **df[:4]:** İlgili dataframe değişkeninin 1. satırından başlayarak 5. satıra kadar olan (5. satır dahil değil) bölümünün elde edilmesini sağlar.
- **df[2:]:** İlgili dataframe değişkeninin 3. satırından başlayarak son satırına kadar olan (son satır dahil) bölümünün elde edilmesini sağlar.

İlgili kullanım şekilleri df2 değişkeni üzerinde denenir ve konsol çıktıları gözlemlenir (Şekil 24). Öğrencilerin farklı değerler ile denemeler yapmaları sağlanır. : karakterinin rolünün pekiştirilmesi / anlaşılması için farklı denemeler gerçekleştirilebilir.

```
52 print("Sadece Column4:")
53 print(df2["Column4"])
54 print("1-4. satır arası bölüm:")
55 print(df2[0:3])
56 print("1-5. satır arası bölüm:")
57 print(df2[:4])
58 print("3. satırdan son satıra kadar:")
59 print(df2[2:])
```

(a)

1-4. satır arası bölüm:					
	Column1	Column2	Column3	Column4	Column5
0	6.565	22.550	61.030	-6.323	6.111
1	3.343	41.230	77.430	32.990	0.555
2	44.320	-4.233	43.112	111.000	4.300

1-5. satır arası bölüm:					
	Column1	Column2	Column3	Column4	Column5
0	6.565	22.550	61.0300	-6.323	6.111
1	3.343	41.230	77.4300	32.990	0.555
2	44.320	-4.233	43.1120	111.000	4.300
3	-13.233	-93.210	90.1323	0.870	-4.250

3. satırdan son satıra kadar:					
	Column1	Column2	Column3	Column4	Column5
2	44.320	-4.233	43.1120	111.00	4.30
3	-13.233	-93.210	90.1323	0.87	-4.25
4	78.550	7348.500	5.0000	-121.40	8.35
5	324.000	43.400	4.0000	30.30	51.10
6	34.000	-30.000	900.0000	1134.00	-904.00

(b)

Şekil 24. (a) df2 değişkeninin farklı bölgelere ulaşılmasını sağlayan kodlar
(b) kodların çalıştırılması sonucu konsol görüntüsü.

Dataframe bölgelere ulaşmak için “loc” ve “iloc” fonksiyonları da kullanılabilir. loc fonksiyonu etiket tabanlı ulaşım sağlarken iloc fonksiyonu indeks tabanlı ulaşım sağlamaktadır. Öğrenciler Şekil 25’te gösterilen komutları yazarak loc ve iloc fonksiyonlarının kullanımını gözlemler. Öğrencilerden ilgili etiket / indeks değerlerinde değişiklikler yaparak ulaşılan çıktıların değerlendirilmesi istenir.

```

63 print("İlk 3 satır için Column2 ve Column5 sütunları:")
64 print(df2.loc["0":"3", ["Column2", "Column5"]])
65 print("Bastan indeksi 2 olan satıra kadar olan bölüm:")
66 print(df2.loc[:2"])
67
68 print("İndeksi 3 olan satır:")
69 print(df2.iloc[3])
70 print("İndeksi 1,2,4 olan satırlar için 0 ve 2 indeksli sütunlar:")
71 print(df2.iloc[[1, 2, 4], [0, 2]])

```

```

İlk 3 satır için Column2 ve Column5 sütunları:
   Column2  Column5
0    22.550   6.111
1    41.230   0.555
2   -4.233   4.300
3   -93.210  -4.250

Bastan indeksi 2 olan satıra kadar olan bölüm:
   Column1  Column2  Column3  Column4  Column5
0     6.565   22.550   61.030   -6.323   6.111
1     3.343   41.230   77.430   32.990   0.555
2    44.320   -4.233   43.112  111.000   4.300

İndeksi 3 olan satır:
   Column1  Column2  Column3  Column4  Column5
0    -13.233   -93.210   90.1323   0.8700  -4.2500
Name: 3, dtype: float64

İndeksi 1,2,4 olan satırlar için 0 ve 2 indeksli sütunlar:
   Column1  Column3
1     3.343   77.430
2    44.320   43.112
4    78.550   5.000

```

(a)

(b)

Şekil 25. (a) df2 değişkeni üzerinde loc ve iloc fonksiyonlarının kullanımına ilişkin kodlar
(b) kodların çalıştırılması sonucu konsol görüntüsü.

Pandas kütüphanesi standart şart kontrol operatörlerinin kullanımını da desteklemekte, böylece belli şartları sağlayan bölgelerin elde edilmesi sağlanabilmektedir (Şekil 26).

```

75 print("Column3 altında 50'den büyük olan satırlara göre bütün df2:")
76 print(df2[df2["Column3"] > 50])
77
78 print("df2 değişkeninde 100'den büyük olan hücrelerin gösterimi:")
79 print(df2[df2 > 100])

```

```

Column3 altında 50'den büyük olan satırlara göre bütün df2:
   Column1  Column2  Column3  Column4  Column5
0     6.565   22.55   61.030   -6.323   6.111
1     3.343   41.23   77.430   32.990   0.555
3    -13.233   -93.21   90.1323   0.8700  -4.2500
6    34.000   -38.00   900.0000  1134.0000 -904.0000

df2 değişkeninde 100'den büyük olan hücrelerin gösterimi:
   Column1  Column2  Column3  Column4  Column5
0      NaN      NaN      NaN      NaN      NaN
1      NaN      NaN      NaN      NaN      NaN
2      NaN      NaN      NaN      111.0     NaN
3      NaN      NaN      NaN      NaN      NaN
4      NaN    7348.5      NaN      NaN      NaN
5    324.0      NaN      NaN      NaN      NaN
6      NaN      NaN    900.0    1134.0      NaN

```

(a)

(b)

Şekil 26. (a) df değişkeni üzerinde şartlı bölgelerin elde edilmesine ilişkin kodlar
(b) kodların çalıştırılması sonucu konsol görüntüsü.



3. ADIM: ÜRET

Üret adımında Pandas kütüphanesi ile dosya tabanlı işlemler ve diğer alternatif veri işleme süreçlerini konusunda bilgiler verilir.

Pandas ile Temel Dosya İşlemleri

Pandas kütüphanesinin sağladığı dosya işlemlerine yönelik fonksiyonlar, veri analizi ve işleme süreçlerinin Python ortamındaki değişkenlere taşınabilmesi için kullanılabilmektedir. Özellikle Yapay Zeka uygulamalarında tercih edilen csv ve Excel dosyalarının Pandas içerisindeki fonksiyonlarla okunması mümkündür. Bu bağlamda “`read_csv`” ve “`read_excel`” fonksiyonları dosyadan okuma için kullanılabilmektedir.

Öğrenciler `df = pd.read_csv("CSV_veriseti.csv")` şeklinde bir kod yazarak (fonksiyon içerisindeki dosya ismi örnek dosyaya göre değişimlere) dosya okuma işleminin gerçekleştirilebilmesini sağlar. Ardından Spyder editörü ortamındaki “Variable Explorer” içerisinde oluşan dataframe değişkenini görüntüleyerek csv dosyası içeriğinin Python ortamına taşınmış olduğunu gözlemler (Şekil 27). Pandas’ın sağladığı `read_csv` fonksiyonu, sütunlar altında ayrılmamış csv verilerini dataframe değişkenlerine sütunlar altında organize edilmiş bir biçimde aktarabilmektedir. Ayrıca csv dosyası içerisinde en üstte yer alan satırda yer alan değerler dataframe değişkenindeki “columns” değerlerine otomatik bir şekilde okunabilmektedir.

Benzeri dosya okuma işlemi xls vexlsx uzantılı dosyalar için de uygulanabilmektedir. Bu durumda satır ve sütunlar dosyada birbirinden ayrı bir biçimde organize edilmeleri dolayısıyla dataframe değişkenine pratik bir biçimde aktarılmakta, yine en üstte yer alan ve hatta **bold** yazı stili ile belirlenen değerler de dataframe değişkeni “columns” değerleri içerisinde otomatik bir biçimde temsil edilmektedir. İlgili dosya türlerinden okuma şu şekillerde yapılabilmektedir:

- **df = pd.read_csv("dosya.xlsx")**: Doğrudan ilk Excel sayfasındaki verilerin okunmasını sağlar.
 - **df = pd.read_csv("dosya.xlsx", "sayfa_ismi")**: İfade edilen Excel sayfasındaki verilerin okunmasını sağlar.
 - **df = pd.read_csv("dosya.xlsx", "sayfa_ismi", index_col="sütun_ismi")**: İfade edilen Excel sayfasındaki verilerin okunmasını ve index_col altında ifade edilen sütundaki verilerin satır indeksleri olarak kullanılmasını sağlar.

(a)

Index	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	szg_fragr	urgent	hot	%failed_log	logged_in	comprom	root_shell	attempte	num_root	_file_creati	num_shells	n_access	outbound	_host_logi	guest_logi	count	srv_count	error
0	0	tcp	http	SF	181	5450	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	8	8	0
1	0	tcp	http	SF	239	486	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	8	8	0
2	0	tcp	http	SF	235	1337	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	8	8	0
3	0	tcp	http	SF	219	1337	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	6	6	0
4	0	tcp	http	SF	217	2832	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	6	6	0
5	0	tcp	http	SF	217	2832	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	6	6	0
6	0	tcp	http	SF	212	1940	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	2	0
7	0	tcp	http	SF	159	4087	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	5	5	0
8	0	tcp	http	SF	210	151	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	8	8	0
9	0	tcp	http	SF	212	786	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	8	8	0
10	0	tcp	http	SF	210	624	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	18	18	0
11	0	tcp	http	SF	177	1985	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	1	0
12	0	tcp	http	SF	222	773	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	11	11	0
13	0	tcp	http	SF	256	1169	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	4	4	0
14	0	tcp	http	SF	241	259	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	1	0
15	0	tcp	http	SF	260	1837	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	11	11	0
16	0	tcp	http	SF	241	261	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	2	2	0
17	0	tcp	http	SF	257	818	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	12	12	0
18	0	tcp	http	SF	233	255	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	2	8	0
19	0	tcp	http	SF	233	584	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	7	7	0
20	0	tcp	http	SF	256	1273	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	17	17	0
21	0	tcp	http	SF	234	255	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	5	5	0
22	0	tcp	http	SF	241	259	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	12	12	0

(b)

Şekil 27. (a) read_csv fonksiyonu ile okunan bir dosya içeriği (b) kodların çalıştırılması sonucu elde edilen değişkenin içeriği.

Üzerinde işlemler yapılmış bir dataframe değişkeni, “**to_csv**” ve “**to_excel**” fonksiyonları yardımıyla csv ve Excel dosyalarına yazdırılabilmektektir. Bu doğrultuda daha önce dosyadan okunan ve Python ortamında düzenlenmiş veri setleri tekrar başka bir dosyaya kaydedilebilmekte ya da okunan dosya üzerine yazdırma yapılabilmektedir.

Alternatif Veri İşleme Süreçleri

Dataframe olarak temsil edilen veriler, Pandas ortamındaki fonksiyonlarla alternatif birçok veri işleme sürecinden geçirilebilmektedir.

Öğrencilerle daha önce oluşturulan df2 değişkeninde her satır için “Column2” ve “Column4” sütunlarının toplamlarının tutulduğu “toplam” adlı yeni bir sütunun oluşturulması sağlanır (Şekil 28). Bu amaçla **df2["toplam"] = df2["Column2"] + df2["Column4"]** komutu yazılır.

Index	Column1	Column2	Column3	Column4	Column5	toplam
0	6.565	22.55	61.03	-6.323	6.111	16.227
1	3.343	41.23	77.43	32.99	0.555	74.22
2	44.32	-4.233	43.112	111	4.3	106.767
3	-13.233	-93.21	90.1323	0.87	-4.25	-92.34
4	78.55	7348.5	5	-121.4	8.35	7227.1
5	324	43.4	4	30.3	51.1	73.7
6	34	-30	900	1134	-984	1104

Şekil 28. df2 değişkeninde toplam sütununun oluşturulması.

Öğrencilere söz konusu kod yapısının dataframe değişkenlerde yeni sütunlar oluşturmak ve sütunlar arası işlemler adına farklı şekillerde kullanılabileceği açıklanır. Bu doğrultuda farklı denemeler gerçekleştirilerek sonuçlar gözlemlenebilir.

Pandas kütüphanesinin “**copy**” fonksiyonu kullanılmak suretiyle mevcut bir dataframe değişkeninin kopyası oluşturulabilmektedir. Öğrenciler aşağıda sıralanan kodları yazmak suretiyle df_yeni adındaki yeni bir dataframe ortamında “**durum**” sütunu oluşturur ve her bir satırda veri için ilgili “durum” sütunu altında; “toplaml” sütunu değeri 100’e eşit ya da 100’den yüksek olanlar için “onaylandı”, diğerleri için “onaylanmadı” etiketlerinin eklenmesini sağlar (Şekil 29):

```
df_yeni = df2.copy()
df_yeni["durum"] = ""
df_yeni.loc[df2['toplaml'] >= 100, 'durum'] = "onaylandı"
df_yeni.loc[df2['toplaml'] < 100, 'durum'] = "onaylanmadı"
```

Şekil 29. Oluşturulan df_yeni adlı dataframe değişkeninin içeriği.

Pandas’ta yer alan “**sort_values**” fonksiyonu dataframe içeriğinin belirli bir sütundaki değerlere göre sıralanmasını sağlayabilmektedir. Diğer yandan “**group_by**” ve “**size**” fonksiyonlarının bir arada kullanımı farklı sütun değerleri içeren satır sayılarının elde edilmesi için kullanılmaktadır. Öğrenciler Şekil 30’da gösterilen kodları yazmak suretiyle “onay durumu” kapsamındaki verilere ilişkin alternatif çıktılar elde eder.

<pre>104 print("Durum sütununa göre sıralama:") 105 print(df_yeni.sort_values(by="durum")) 106 107 print("Durum sütunundaki farklı değerlerdeki satırların sayıları:") 108 print(df_yeni.groupby("durum", observed=False).size())</pre>	<pre>Durum sütununa göre sıralama: Column1 Column2 Column3 Column4 Column5 toplam durum 2 44.320 -4.233 43.1120 111.000 4.300 106.767 onaylandı 4 78.550 7348.500 5.0000 -121.400 8.350 7227.100 onaylandı 6 34.000 -30.000 900.0000 1134.000 904.000 1104.000 onaylandı 0 6.565 22.550 61.0300 -6.323 6.111 16.227 onaylanmadı 1 3.343 41.230 77.4300 32.990 0.555 74.220 onaylanmadı 3 -13.233 -93.210 90.1323 0.870 -4.250 -92.340 onaylanmadı 5 324.000 43.400 4.0000 30.300 51.100 73.700 onaylanmadı Durum sütunundaki farklı değerlerdeki satırların sayıları: durum onaylandı 3 onaylanmadı 4 dtype: int64</pre>
(a)	(b)

Şekil 30. (a) df_yeni değişkeni üzerinde sort_values ve group_by fonksiyonlarının kullanımına ilişkin kodlar (b) kodların çalıştırılması sonucu konsol görüntüsü.



4. ADIM: İLERLET

Öğrencilerin bireysel olarak Pandas kütüphanesi ile uygulamalar gerçekleştirmeleri istenir. Bu etkinlik için 20 dakikalık bir süre ayrılr.



5. ADIM: DEĞERLENDİRME

Derste öğrenilen bilgilerden hareketle soru-cevap süreci gerçekleştirilir ve Python Pandas kütüphanesi kullanımı genel hatlarıyla pekiştirilir.

Öğrencilere şu sorular sorulabilir:

- Pandas kütüphanesi hakkında neler düşünüyorsunuz?
- Pandas kütüphanesinin en çok hangi fonksiyonlarını başarılı buldunuz?
- Pandas kütüphanesi Yapay Zeka uygulamalarında ne gibi avantajlar sağlayabilir?
- Pandas kütüphanesini kullanarak hangi tür uygulamalar gerçekleştirebilirsiniz?

1. HAFTA – 3. GÜN – 2. DERS: MATPLOTLIB PYTHON KÜTÜPHANESİ

DERS PLANI

DERS ETİKETLERİ



KONU: Matplotlib Python Kütüphanesi



SINIF: 12-14 Yaş



SÜRE: 60 dk.



HAZIR BULUNUŞLUK: Temel İngilizce eğitimi, Yapay Zeka farkındalığı, Programlamaya yönelik temel bilgi



ANAHTAR KELİMELER: Yapay Zeka, Kodlama, Python, Matplotlib, Pyplot



BECERİLER:



Tahmin



Sorgulama



Günlük hayat bağlantıları kurma



KAZANIMLAR:

Bu derste öğrenciler;

- Matplotlib Python kütüphanesi hakkında bilgi sahibi olur.
- Matplotlib Pyplot ile gerçekleştirilebilecek temel görsel çizim işlemleri hakkında bilgi ve beceri sahibi olur.
- Yapay Zeka uygulamaları kapsamında Matplotlib Pyplot yardımıyla görsel çıktıların nasıl elde edilebileceği hakkında farkındalık, bilgi ve beceri sahibi olur.



DERS MATERİYALLERİ:

Malzeme	Miktar
Yazı Tahtası	Öğrenci sayısı ve sınıf büyüklüğüne göre en az bir adet
Projeksiyon	Öğrenci sayısı ve sınıf büyüklüğüne göre en az bir adet
Defter, Kalem	Öğrenci sayısı kadar
Bilgisayar	Öğrenci sayısının yarısı kadar
Tablet	Öğrenci sayısının yarısı kadar
Python Anaconda Framework	Her bilgisayarda kurulu olacak şekilde



ÖĞRETMEN HAZIRLIK ÇALIŞMALARI:

Okunabilecek Kaynaklar:

Matplotlib. (2024). *Pyplot tutorial*. Matplotlib Library Tutorials. Çevirmiçi: <https://matplotlib.org/stable/tutorials/pyplot.html> (Erişim 6 Haziran 2024).

Kaggle. (2024). Data Visualization Using Matplotlib. Çevirmiçi: <https://www.kaggle.com/code/sanikamal/data-visualization-using-matplotlib> (Erişim 6 Haziran 2024).

İzlenebilecek Kaynaklar:

Corey Schafer. (2019). Matplotlib Tutorials. Çevirmiçi: <https://www.youtube.com/watch?v=UO98IJQ3QGI&list=PL-osiE80TeTvipOqomVEeZ1HRrcEvtZB> (Erişim 6 Haziran 2024).

Kocaaslan, B. (2022). Matplotlib Nedir, Kurulumu ve Kullanımı. Çevirmiçi: <https://www.youtube.com/watch?v=PU5q24QA1IA> (Erişim 6 Haziran 2024).

ATÖLYE HAZIRLIK ÇALIŞMALARI:

Öğretmen;

- Dersten önce ders materyallerini hazırlamalıdır.
- Python Matplotlib kütüphanesi hakkında bilgi edinmelidir.

- Matplotlib Pyplot ile görsel çıktılar oluşturma süreçleri konusunda bilgi-beceri sahibi olmalıdır.



KAYNAKÇA:

Devert, A. (2014). *Matplotlib Plotting Cookbook*. Packt Publishing Ltd.

Tosi, S. (2009). *Matplotlib for Python developers*. Packt Publishing Ltd.

GENEL BAKIŞ:

- 1) Harekete Geç:** Öğrencilere Python Matplotlib kütüphanesi hakkında bilgi sahibi olup olmadıkları sorulur ve söz konusu kütüphane hakkında temel bilgiler verilir.
- 2) Keşfet:** Matplotlib Pyplot kütüphane bileşenleri ile görsel çıktılar oluşturma hakkında bilgiler verilir.
- 3) Üret:** Matplotlib Pyplot kütüphanesi ile alternatif görselleştirme düzenlemeleri ve Pandas işbirliği içerisinde veri görselleştirme adımları konularında bilgiler verilir.
- 4) İlerlet:** Öğrencilerin bireysel olarak Matplotlib Pyplot kütüphanesi ile uygulamalar gerçekleştirmeleri istenir.
- 5) Değerlendir:** Derste öğrenilen bilgilerden hareketle soru-cevap süreci gerçekleştirilir ve Python Matplotlib Pyplot kütüphanesi kullanımı genel hatlarıyla pekiştirilir.

UYGULAMA

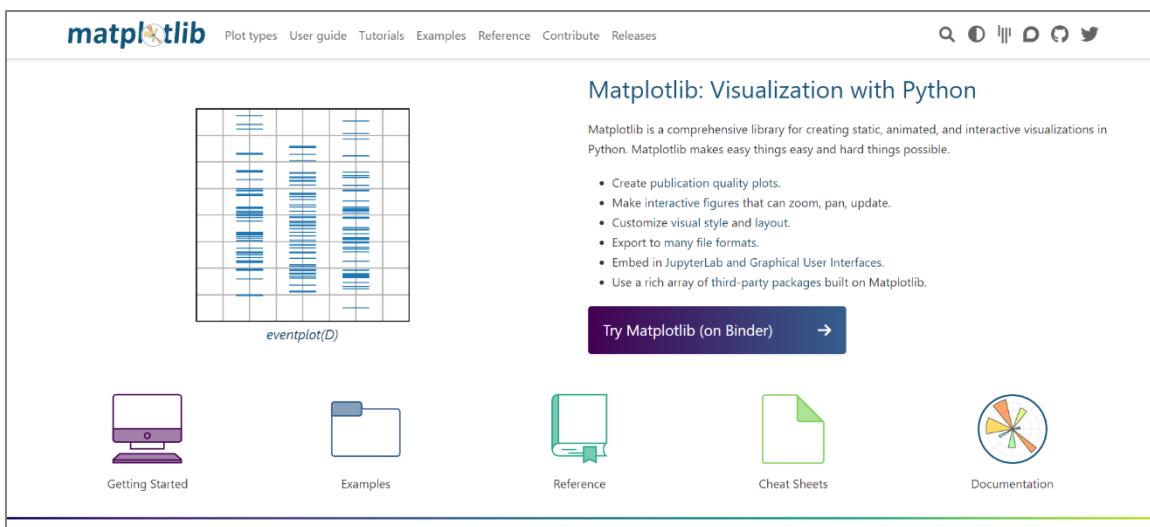


1. ADIM: HAREKETE GEÇ

Öğrencilere Python Matplotlib kütüphanesi hakkında bilgi sahibi olup olmadıkları sorulur ve söz konusu kütüphane hakkında temel bilgiler verilir.

Matplotlib Kütüphanesi

Matplotlib kütüphanesi Python ortamında görsel çıktılar oluşturmak için sıkılıkla tercih edilen bir kütüphane bileşenidir. Matplotlib içerisinde yer alan alt modüller ve ilgili fonksiyonlar, birçok farklı grafik türünün ve verilerin görsel olarak temsil edilmesini sağlayan alternatif gösterim şekillerinin hızlı bir biçimde elde edilebilmesini mümkün kılmaktadır (Tosi, 2009). Matplotlib kütüphanesi özellikle Pyplot bileşeni üzerinden Veri Bilimi ve Yapay Zeka uygulamalarında elde edilen bulguların görsel olarak anlatılmasında büyük rol oynamaktadır (Devert, 2014). Matplotlib kütüphanesine ilişkin temel dokümantasyonlar, eğitimler ve haberler <https://matplotlib.org/> Web sitesi üzerinden sunulmaktadır (Şekil 31).



Şekil 31. Python Matplotlib kütüphanesi Web sitesi.

Matplot kütüphanesi Python ortamında “**import matplotlib**” komutu ile çağrılmakta ve görselleştirme süreçlerinde tercih edilen “**pyplot**” bileşenini çalışma ortamına dahil etmek için ilgili çağrı “**import matplotlib.pyplot**” komutu genişletilmektedir. Pyplot'un kodlama süreçlerinde kolay kullanımını sağlamak için “**plt**” kısaltması da tercih edilir (Şekil 32).

```

1  # -*- coding: utf-8 -*-
2  """
3  Created on Thu Jun 6 04:07:10 2024
4
5  @author: UK
6  """
7  import matplotlib.pyplot as plt
8

```

Şekil 32. Python Matplotlib Pyplot kütüphanesinin çağrılması.

Matplotlib Pyplot kütüphane yapısı Python Anaconda Framework ile birlikte hazır bir biçimde gelmektedir. Anaconda harici manuel kurulumlar için ilgili kütüphanenin Web platformunda <https://matplotlib.org/stable/install/index.html> linkinde sunulan yönergeler takip edilebilir.



2. ADIM: KEŞFET

Matplotlib Pyplot kütüphane bileşenleri ile görsel çıktılar oluşturma hakkında bilgiler verilir.

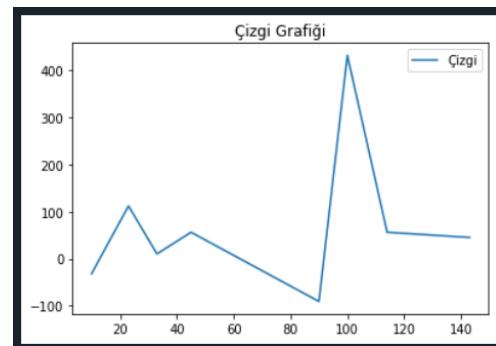
Pyplot ile Farklı Grafik Türlerinin Kullanımı

Matplotlib Pyplot ile farklı grafik türlerini çizdirmek için grafikleri besleyecek verilerin elde edilmesi önemlidir. İlgili veriler basit değişkenlerden farklı veri modellerine kadar yaygın kullanılan Python bileşenleri ile kodlama ortamında dahil edilebilmektedir.

Çizgi ve Çubuk Grafikleri

Öğrenciler öncelikli olarak Şekil 33'te gösterildiği gibi x ve y değişkenleri altında koordinat bilgilerini oluşturarak x ve y değişkenlerinin parametre olarak kullanıldığı **plot** fonksiyonu ile grafik oluşturma sürecini gerçekleştirir. Grafiği daha okunur hale getirmek için **title** fonksiyonu ile grafik başlığını, **legend** fonksiyonu ile çizгиyi temsil eden göstergeyi tanımlar ve **show** fonksiyonu ile grafiğin ekranda gösterim adımını tamamlar. Kodun çalıştırılması sonucu elde edilen grafik "Plots" sekmesi altında görüntülenir.

```
7 import matplotlib.pyplot as plt
8 x = [10, 23, 33, 45, 90, 100, 114, 143]
9 y = [-32, 112, 10, 56, -91, 432, 56, 45]
10
11 plt.plot(x, y)
12 plt.title("Çizgi Grafiği")
13
14 plt.legend(["Çizgi"])
15 plt.show()
```



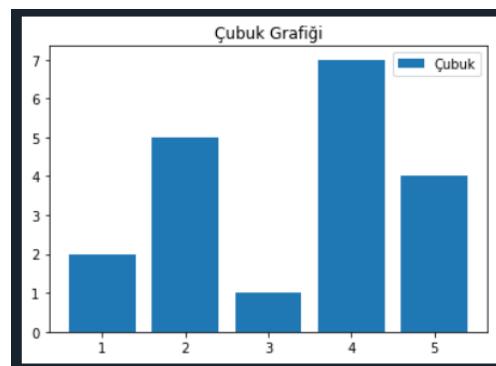
(a)

(b)

Şekil 33. (a) Çizgi grafiği oluşturma kodları (b) kodların çalıştırılması sonucu Plot sekmesi altında oluşan görüntü.

Öğrenciler mevcut kodlar altında Şekil 34'de gösterilen yeni x-y değişkenlerini tanımlayarak **bar** fonksiyonu üzerinden çubuk grafiği türünü tercih eder ve **show** fonksiyonu ile sonuç grafiğini elde eder. Her iki grafik türünde x ve y değişkenlerinde temsil edilen sayıların farklı grafik türlerinde nasıl sonuçlar ürettiği yorumlanır.

```
17 x = [5, 1, 3, 5, 2, 4, 4]
18 y = [3, 2, 1, 4, 5, 6, 7]
19 plt.bar(x, y)
20 plt.title("Çubuk Grafiği")
21
22 plt.legend(["Çubuk"])
23 plt.show()
```



(a)

(b)

Şekil 34. (a) Çubuk grafiği oluşturma kodları (b) kodların çalıştırılması sonucu Plot sekmesi altında oluşan görüntü.

Histogram ve Dağılım Grafikleri

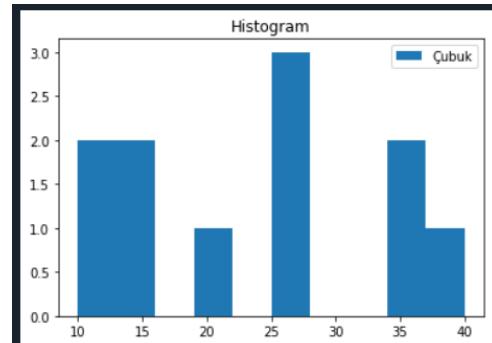
Öğrenciler kod ortamında "yeni_veri" adlı bir değişken oluşturarak Şekil 35'te gösterildiği gibi çeşitli değerler tanımlar. Ardından **hist** ve **show** fonksiyonları ile histogram grafiği görüntülenir. Grafikte yeni_veri altındaki değerlerin x ekseninde tanımlandığı, y ekseninde ise

her bir verinin tekrar sayısının belirtildiği vurgulanır. Öğrencilerden alternatif verilerle farklı histogram sonuçlarını gözlemlemeleri sağlanır.

```

25 yeni_veri = [10, 15, 20, 25, 15, 35, 40, 25, 25, 35, 10]
26 plt.hist(yeni_veri)
27 plt.title("Histogram")
28 plt.legend(["Çubuk"])
29 plt.show()

```



(a)

(b)

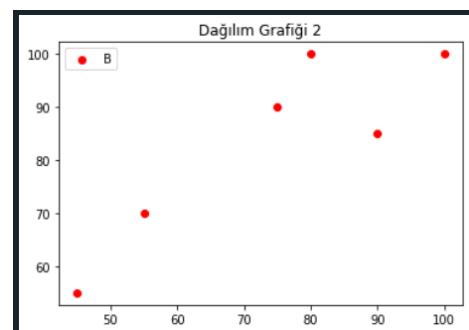
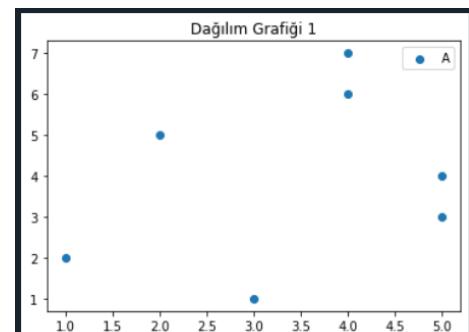
Şekil 35. (a) Histogram grafiği oluşturma kodları (b) kodların çalıştırılması sonucu Plot sekmesi altında oluşan görüntü.

Öğrencilere farklı değerlerin noktalar ile gösterileceği dağılım (scatter) grafiği açıklanır. Matplotlib Pyplot ortamında dağılım grafiği çağrısı için “scatter” fonksiyonu kullanılmaktadır. Daha önce tanımlanan x, y değişkenleri ile birlikte “yazılı” ve “sözlü” adlı değişkenler altında tanımlanan yeni değerler ile iki farklı dağılım grafiği çizilir. İkinci dağılım grafiğinde plot fonksiyonu içerisinde color parametresi kullanılarak farklı renklerle görseller elde edilebileceği gözlemlenir (Şekil 36).

```

31 plt.scatter(x, y)
32 plt.legend('A')
33 plt.title("Dağılım Grafiği 1")
34 plt.show()
35
36 yazılı = [80, 75, 90, 100, 55, 45]
37 sözlü = [100, 90, 85, 100, 70, 55]
38 plt.scatter(yazılı, sözlü, color="red")
39 plt.legend("B")
40 plt.title("Dağılım Grafiği 2")
41 plt.show()

```



(a)

(b)

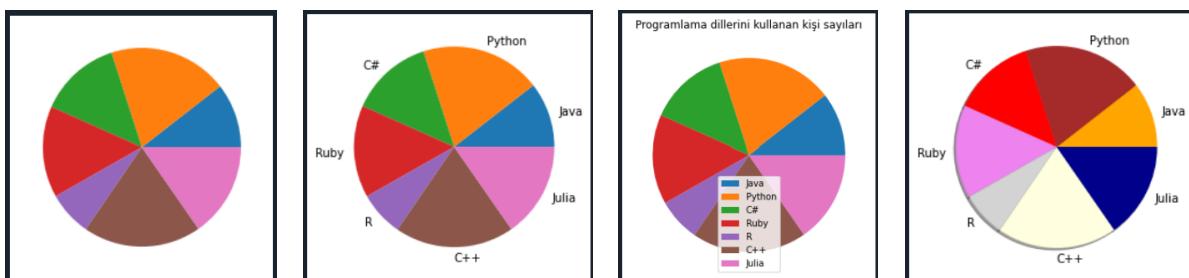
Şekil 36. (a) Dağılım grafikleri oluşturma kodları (b) kodların çalıştırılması sonucu Plot sekmesi altında oluşan görüntüler.

Pasta Grafiği

Matplotlib Pyplot ile pasta grafikleri oluşturmak için Şekil 37'de gösterilen kodlar yazılır. Kütüphanenin sağladığı ek özellikler kullanılmak suretiyle pasta grafikleri etiketler, farklı renkler ve göstergeler ile desteklenebilmektedir. Bu amaçla öğrencilerin alternatif kodlarla farklı pasta grafiği sonuçlarını görmeleri sağlanır (Şekil 37).

```
45 veri = [48,89,61,68,33,88,70]
46 plt.pie(veri)
47 plt.show()
48
49 etiketler=[ 'Java', 'Python', 'C#', 'Ruby', 'R', 'C++', 'Julia']
50 plt.pie(veri,labels=etiketler)
51 plt.show()
52
53 plt.figure(figsize=(8, 5))
54 plt.pie(veri)
55 plt.title('Programlama dillerini kullanan kişi sayıları', fontsize=12)
56 plt.legend(etiketler)
57 plt.show()
58
59 renkler = ['orange', 'brown', 'red', 'violet', 'lightgrey','lightyellow','darkblue']
60 plt.pie(veri, labels=etiketler, colors=renkler, shadow=True)
61 plt.show()
```

(a)



(b)

Şekil 37. (a) Pasta grafikleri oluşturma kodları (b) kodların çalıştırılması sonucu Plot sekmesi altında oluşan görüntüler.

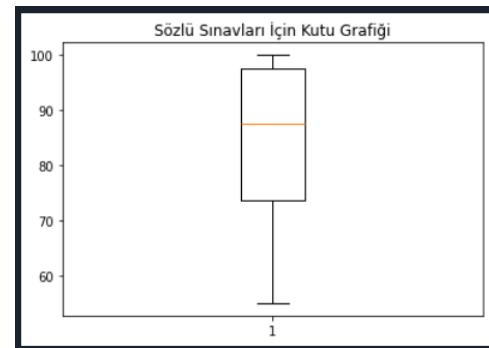
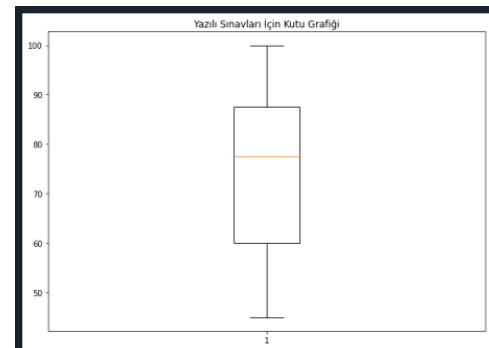
Kutu ve Stack Grafikleri

Öğrenciler Şekil 38'de gösterildiği gibi “**boxplot**” fonksiyonunu kullanmak suretiyle, daha önce tanımlanan “yazılı” ve “sözlü” değişkenlerindeki değerler için kutu grafikleri oluşturur. Kodlarda “**figure**” fonksiyonu altında “**figsize**” parametrelerinin görselin inç boyutunda genişlik ve yüksekliğini belirlemekte kullanıldığı bilgisi verilir. Kutu grafiklerinin genel özellikleri, kutu yapısı içerisinde ortalama, minimum/maksimum değerlerin gösterimi gibi farklı bileşenlerin okunması konusunda açıklamalar yapılır. Öğrencilerin farklı değişken değerleri ve figure boyutları deneyerek elde ettikleri görselleri yorumlamaları istenir.

```

67 fig = plt.figure(figsize =(10, 7))
68
69 plt.boxplot(yazili)
70 plt.title("Yazılı Sınavları İçin Kutu Grafiği")
71 plt.show()
72 plt.title("Sözlü Sınavları İçin Kutu Grafiği")
73 plt.boxplot(sozlu)
74 plt.show()

```



(a)

(b)

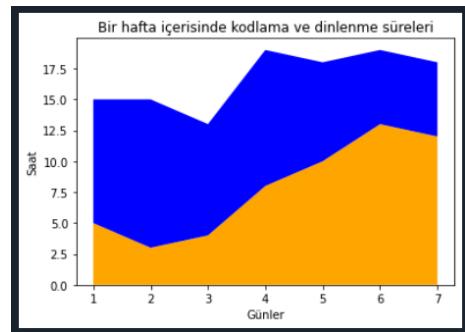
Şekil 38. (a) Kutu grafikleri oluşturma kodları (b) kodların çalıştırılması sonucu Plot sekmesi altında oluşan görüntüler.

Öğrencilerden kodlama ve dinlenme sürelerini içeren iki değişken tanımlamaları ve haftanın yedi gününü temsil eden üçüncü bir değişken eşliğinde, Şekil 39'da görüldüğü gibi “**stackplot**” fonksiyonu ile stack grafiği oluşturmaları sağlanır. Yazılan kodlarda “colors” parametresi ile iki farklı veri bütüne farklı renklerin atıldığı, “xlabel” ve “ylabel” ile x-y koordinat başlıklarının belirlendiği vurgulanır.

```

83 kodlama_suresi = [5, 3, 4, 8, 10, 13, 12]
84 dinlenme_suresi = [10, 12, 9, 11, 8, 6, 6]
85 gunler = [1, 2, 3, 4, 5, 6, 7]
86
87 plt.stackplot(gunler, kodlama_suresi, dinlenme_suresi,
88 colors =['orange', 'blue'])
89
90 plt.xlabel('Günler')
91 plt.ylabel('Saat')
92 plt.title('Bir hafta içerisinde kodlama ve dinlenme süreleri')
93 plt.show()

```



(a)

(b)

Şekil 39. (a) Stack grafiği oluşturma kodları (b) kodların çalıştırılması sonucu Plot sekmesi altında oluşan görüntü.



3. ADIM: ÜRET

Matplotlib Pyplot kütüphanesi ile alternatif görselleştirme düzenlemeleri ve Pandas işbirliği içerisinde veri görselleştirme adımları konularında bilgiler verilir.

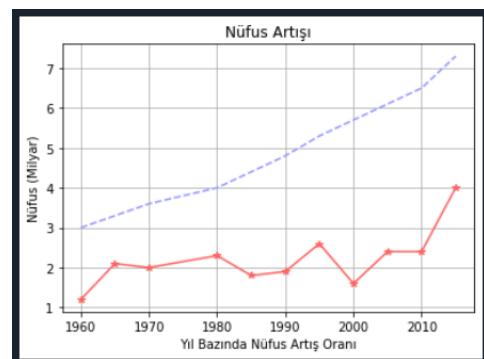
Grafik Görüntüleme Düzenlemeleri

Öğrencilere aynı grafik ortamında farklı verilere ilişkin gösterimler yapılabileceği anlatılır. Bu amaçla Şekil 40'da verilen çizgi grafiği kodları yazılır. Kodlarda çizgi renkleri ve çizgi stillerine ilişkin parametreler vurgulanır, “grid” fonksiyonu ile grafik ızgara gösteriminin aktif hale getirildiği açıklanır. Öğrencilerin veriler ve grafik parametrelerinde düzenlemeler yaparak değişiklikleri gözlemlemeleri istenir.

```

96  yillar=[1960,1965,1970,1980,1985,1990,1995,2000,2005,2010,2015]
97  nufus=[3.0,3.3,3.6,4.0,4.4,4.8,5.3,5.7,6.1,6.5,7.3]
98  olsun_orani=[1.2,2.1,2.0,2.3,1.8,1.9,2.6,1.6,2.4,2.4,4.0]
99
100 plt.plot(yillar,nufus,'--',color=(.6,.6,1))
101 plt.plot(yillar,olsun_orani,'*-',color=(255/255,100/255,100/255))
102
103 plt.ylabel("Nüfus (Milyar)")
104 plt.xlabel("Yıl Bazında Nüfus Artış Oranı")
105 plt.title("Nüfus Artışı")
106
107 plt.grid(True)
108 plt.show()

```



(a)

(b)

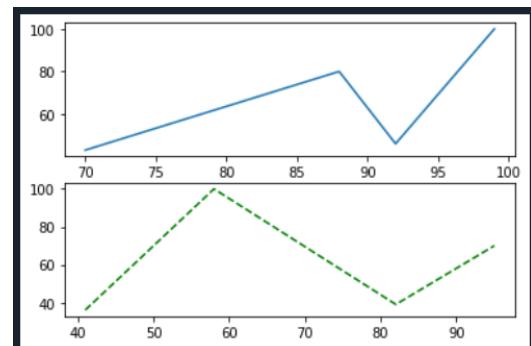
Şekil 40. (a) Birden fazla çizgi grafiği çizimini sağlayan kodlar (b) kodların çalıştırılması sonucu Plot sekmesi altında oluşan görüntü.

Öğrencilere oluşturulan grafiklerin “subplot” fonksiyonu yardımıyla tek bir görüntü içerisinde bir arada sunulabildiği anlatılır. Bu amaçla Şekil 41'de gösterilen kodlar yazılarak elde edilen grafik gözlemlenir. Kodlar içerisinde subplot fonksiyonu için kullanılan parametreler sırasıyla birden fazla grafiğin yerleştirilmesi amacıyla ayrılan; **satır sayısı**, **sütun sayısı** ve **yerleştirilecek grafiği sıra değeri** parametrelerine karşılık gelmektedir.

```

112 #this is plot 1
113 x = [70, 88, 92, 99]
114 y = [43, 80, 46, 100]
115
116 plt.subplot(2, 1, 1)
117 plt.plot(x,y)
118
119 #this is plot 2
120 x = [41, 58, 82, 95]
121 y = [36, 100, 39, 70]
122
123 plt.subplot(2, 1, 2)
124 plt.plot(x,y,'--',color='green')
125
126 plt.show()

```



(a)

(b)

Şekil 41. (a) Birden fazla çizgi grafiği çiziminde subplot kullanımına dair kodlar (b) kodların çalıştırılması sonucu Plot sekmesi altında oluşan görüntü.

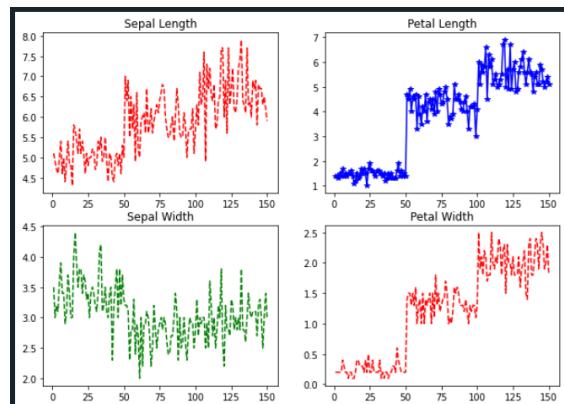
Pandas ve Pyplot Kullanımı

Öğrencilerden <https://www.kaggle.com/datasets/uciml/iris> linkini ziyaret ederek iris veri seti dosyasını indirmeleri istenir. Öğrencilere söz konusu Kaggle platformu hakkında bir sonraki ders günü (1. Hafta – 4. Gün) bilgi verileceği ve veri setini indirmeleri için üye olmalarının yeterli olacağı vurgulanır. Benzer şekilde iris veri setinin çeşitli yaprak ölçüm değerlerine göre iris çiçek sınıfının belirlenmesini sağlayan bir veri seti olduğu, ilgili veri seti üzerinde örnek uygulamaların yine 4. Gün (1. Hafta) haftanın son ders günü (1. Hafta – 5. Gün) yapılacağı ayrıca açıklanır.

Öğrenciler **import pandas as pd** kodu ile Pandas kütüphanesi çağrısını yaparlar. Ardından **read_csv** fonksiyonu ile indirilen iris veri setinin dataframe değişkenine okunması sağlanır.

İris veri setini içeren dataframe değişkeninin elde edilmesi ile birlikte **subplot** fonksiyonu kullanılarak, farklı grafik parametreleri eşliğinde veri setinin dört farklı girdi değerleri ekrana görüntülenenir (Şekil 42).

```
128 import pandas as pd
129
130 iris = pd.read_csv("Iris.csv")
131
132 fig = plt.figure(figsize = (10, 7))
133
134 plt.subplot(2, 2, 1)
135 plt.title("Sepal Length")
136 plt.plot(iris.Id, iris["SepalLengthCm"], "r--")
137 plt.subplot(2, 2, 2)
138 plt.title("Petal Length")
139 plt.plot(iris.Id, iris["PetalLengthCm"], "b*-")
140 plt.subplot(2, 2, 3)
141 plt.title("Sepal Width")
142 plt.plot(iris.Id, iris["SepalWidthCm"], "g--")
143 plt.subplot(2, 2, 4)
144 plt.title("Petal Width")
145 plt.plot(iris.Id, iris["PetalWidthCm"], "r--")
146 plt.show()
```



(a)

(b)

Şekil 42. (a) Iris veri seti girdi değerlerinin subplot ile görüntülenmesine dair kodlar
(b) kodların çalıştırılması sonucu Plot sekmesi altında oluşan görüntü.

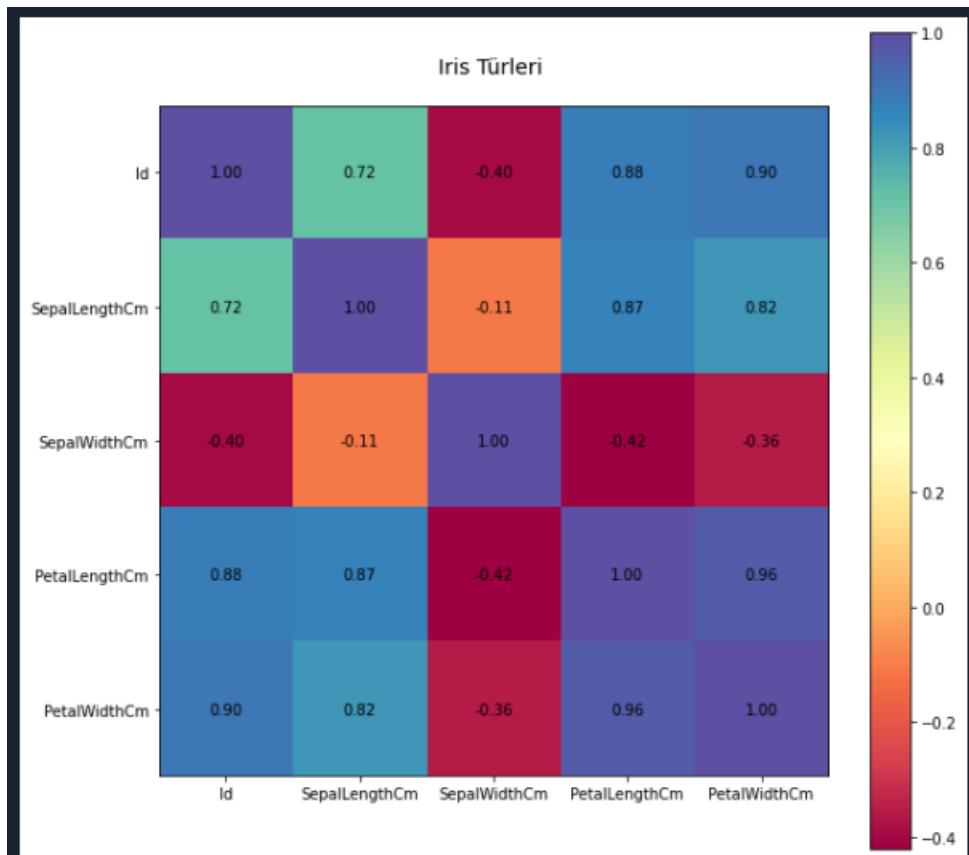
Pyplot bileşeni içerisinde yer alan “**imshow**” fonksiyonu veriler üzerinden ısı haritaları (heat maps) oluşturulması için kullanılabilirliktedir. Bu doğrultuda veri setleri de görsel olarak alternatif bir yaklaşım ile yorumlanabilemektedir. İlgili imshow ile kullanılan “**corr**” fonksiyonu veri setindeki farklı değerlerin bütün veri setindeki dağılımlarını sayısal olarak tespit edilmesi için kullanılmaktadır. Öğrencilerden Şekil 43’te verilen kod bütününe yazmaları ve çalıştırarak elde edilen ısı haritası görüntüsünü incelemeleri istenir.

```

148 plt.imshow(iris.corr(), cmap="Spectral")
149 plt.colorbar()
150 plt.gcf().set_size_inches(10, 10)
151 plt.xticks(range(len(iris.corr().columns)), iris.corr().columns)
152 plt.yticks(range(len(iris.corr().columns)), iris.corr().columns)
153 labels = iris.corr().values
154 for a in range(labels.shape[0]):
155     for b in range(labels.shape[1]):
156         plt.text(a, b, '{:.2f}'.format(labels[b, a]),
157                  ha='center', va='center', color='black')
158 plt.title('Iris Türleri \n', fontsize=14)
159 plt.show()

```

(a)



(b)

Şekil 43. (a) İris veri seti değerlerinin ısı haritası içerisinde gösterimine dair kodlar
(b) kodların çalıştırılması sonucu Plot sekmesi altında oluşan görüntü.



4. ADIM: İLERLET

Öğrencilerin bireysel olarak Matplotlib Pyplot kütüphaneleri ile uygulamalar gerçekleştirmeleri istenir. Öğrencilerin özellikle Pandas kütüphanesi ile ilgili öğrendikleri fonksiyonları Pyplot ile birleştirerek uygulamalar yapmaları ve oluşan görüntülerini yorumlamaları istenir. Bu etkinlik için 15 dakikalık bir süre ayrılrı.



5. ADIM: DEĞERLENDİRME

Derste öğrenilen bilgilerden hareketle soru-cevap süreci gerçekleştirilir ve Python Matplotlib Pyplot kütüphanesi kullanımı genel hatlarıyla pekiştirilir.

Öğrencilere şu sorular sorulabilir:

- Matplotlib Pyplot kütüphanesi hakkında neler düşünüyorsunuz?
- Matplotlib Pyplot kütüphanesinin en çok hangi fonksiyonlarını başarılı buldunuz?
- Matplotlib Pyplot kütüphanesi Yapay Zeka uygulamalarında ne gibi avantajlar sağlayabilir?
- Matplotlib Pyplot kütüphanesini kullanarak Yapay Zeka uygulamalarında hangi tür görselleştirmeler gerçekleştirmeyi düşünürsünüz?

1. HAFTA – 3. GÜN – 3. DERS: NUMPY PYTHON KÜTÜPHANESİ

DERS PLANI

DERS ETİKETLERİ



KONU: Numpy Python Kütüphanesi



SINIF: 12-14 Yaş



SÜRE: 60 dk.



HAZIR BULUNUŞLUK: Temel İngilizce eğitimi, Yapay Zeka farkındalığı, Programlamaya yönelik temel bilgi



ANAHTAR KELİMELER: Yapay Zeka, Kodlama, Python, Numpy, Diziler



BECERİLER:



Tahmin



Sorgulama



Günlük hayat bağlantısı kurma



KAZANIMLAR:

Bu derste öğrenciler;

- Numpy Python kütüphanesi hakkında bilgi sahibi olur.
- Numpy ile gerçekleştirilebilecek dizi işlemleri hakkında bilgi ve beceri sahibi olur.
- Yapay Zeka uygulamalarında Numpy ile nasıl etkin veri düzenleme ve yönetme işlemlerinin yapılabildiği hakkında farkındalık, bilgi ve beceri sahibi olur.



DERS MATERİYALLERİ:

Malzeme	Miktar
Yazı Tahtası	Öğrenci sayısı ve sınıf büyüklüğüne göre en az bir adet
Projeksiyon	Öğrenci sayısı ve sınıf büyüklüğüne göre en az bir adet
Defter, Kalem	Öğrenci sayısı kadar
Bilgisayar	Öğrenci sayısının yarısı kadar
Tablet	Öğrenci sayısının yarısı kadar
Python Anaconda Framework	Her bilgisayarda kurulu olacak şekilde



ÖĞRETMEN HAZIRLIK ÇALIŞMALARI:

Okunabilecek Kaynaklar:

NumPy. (2024). *NumPy: The absolute basics for beginners*. Numpy Web Platform. Çevrimiçi: https://numpy.org/doc/stable/user/absolute_beginners.html (Erişim 7 Haziran 2024).

Kaggle. (2024). Numpy Tutorial For Beginners [Data Science]. Çevrimiçi: <https://www.kaggle.com/code/legendadnan/numpy-tutorial-for-beginners-data-science> (Erişim 7 Haziran 2024).

İzlenebilecek Kaynaklar:

Free Code Camp.org. (2020). Python NumPy Tutorial for Beginners. Çevrimiçi: <https://www.youtube.com/watch?v=QUT1VHiLmmI> (Erişim 8 Haziran 2024).

Mühendisin Blogu. (2021). Yarım Saatte Python Numpy Kütüphanesi Dersi. Çevrimiçi: <https://www.youtube.com/watch?v=63EHOcsYd5w> (Erişim 8 Haziran 2024).

ATÖLYE HAZIRLIK ÇALIŞMALARI:

Öğretmen;

- Dersten önce ders materyallerini hazırlamalıdır.
- Python Numpy kütüphanesi hakkında bilgi edinmelidir.

- Numpy ile dizilerin kullanımı aracılığıyla verilerin düzenlenmesi yönündeki işlemler konusunda bilgi-beceri sahibi olmalıdır.



KAYNAKÇA:

Idris, I. (2015). *NumPy: Beginner's Guide*. Packt Publishing Ltd.

McKinney, W. (2013). *Python for data analysis*. O'Reilly Media, Inc.

GENEL BAKIŞ:

- 1) Harekete Geç:** Öğrencilere Python Numpy kütüphanesi hakkında bilgi sahibi olup olmadıkları sorulur ve söz konusu kütüphane hakkında temel bilgiler verilir.
- 2) Keşfet:** Numpy kütüphane bileşenleri ile dizi tanımlama, inceleme süreçleri hakkında bilgiler verilir.
- 3) Üret:** Numpy ile alternatif dizi işlemleri ve Pandas ile Matplotlib işbirliği içerisinde gerçekleştirilebilecek çözümler kapsamında bilgiler verilir.
- 4) İlerlet:** Öğrencilerin bireysel olarak Numpy kütüphanesi ile uygulamalar gerçekleştirmeleri istenir.
- 5) Değerlendir:** Derste öğrenilen bilgilerden hareketle soru-cevap süreci gerçekleştirilir ve Python Numpy kütüphanesi kullanımı genel hatlarıyla pekiştirilir.

UYGULAMA

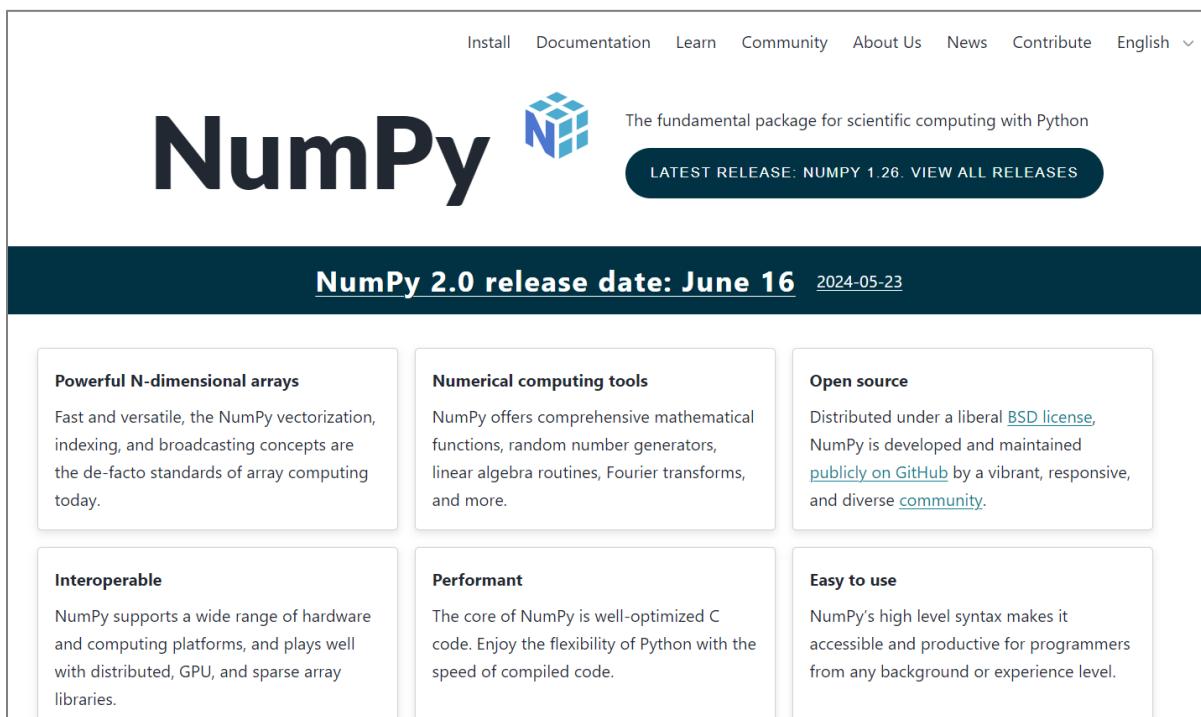


1. ADIM: HAREKETE GEÇ

Öğrencilere Python Numpy kütüphanesi hakkında bilgi sahibi olup olmadıkları sorulur ve söz konusu kütüphane hakkında temel bilgiler verilir.

Numpy Kütüphanesi

Numpy kütüphanesi, Pandas ve Matplotlib gibi Python kütüphaneleri yanında, Veri Bilimi ve Yapay Zeka uygulama geliştirme aşamalarında yaygın olarak tercih edilen bir diğer kütüphane olarak bilinmektedir. İlk olarak 2005 yılında ortaya çıkan ve “Numerical Python” ifadelerinin kısaltmasını temsil eden Numpy, dizi adı verilen veri yapılarından faydalananmadır (McKinney, 2013). Python ortamında yer alan liste, demet, sözlük gibi veri modellerinden farklı olarak, bir tür veri yapısı olan dizilerden faydalanan Numpy, bu yönyle daha kompakt ve daha hızlı veri işlemlerine olanak sağlama; Python ortamında Lineer Cebir işlemlerinin yapılmasını kolaylaştırmaktadır (Idris, 2015; McKinney, 2013). Numpy kütüphanesine ilişkin temel dokümantasyonlar, eğitimler ve haberler <https://numpy.org/> Web sitesi üzerinde yer almaktadır (Şekil 44).



Şekil 44. Python Numpy kütüphanesi Web sitesi.

Numpy kütüphanesi Python uygulamalarına “**import numpy**” komutu ile eklenebilmektedir. Kütüphane bileşenlerinin kodlama aşamalarında kolay kullanımını sağlamak adına “**np**” kısaltması da sıkılıkla kullanılmaktadır (Şekil 45).

```
1 # -*- coding: utf-8 -*-
2 """
3 Created on Fri Jun 8 06:15:08 2024
4
5 @author: UK
6 """
7 import numpy as np
8
```

Şekil 45. Python Numpy kütüphanesinin çağrılması.

Tıpkı diğer yaygın kullanılan kütüphaneler gibi; Numpy kütüphanesi de Python Anaconda Framework içerisinde hazır bir biçimde gelmektedir. Anaconda haricindeki ortamlarda Numpy kütüphanesinin kurulumu için <https://numpy.org/install/> sayfasındaki açıklamalar incelenebilir.



2. ADIM: KEŞFET

Numpy kütüphane bileşenleri ile dizi tanımlama, inceleme süreçleri hakkında bilgiler verilir.

Numpy ile Dizi İşlemleri

Numpy kütüphanesi gücünü dizi işlemlerinden almaktadır. Farklı programlama dillerinde varsayılan veri tanımlama şekilleri içerisinde var olan diziler, Python programlama dilinde Numpy kütüphanesi ile birlikte anılır hale gelmiştir. Standart olarak Numpy içerisinde yer alan “array” fonksiyonu kullanmak suretiyle diziler tanımlanabilmektedir.

Öğrenciler Şekil 46’da olduğu gibi np.array söz dizimini kullanarak iki farklı dizi tanımlaması yapar ve ekran çıktılarını inceler. Numpy ortamında diziler tanımlanırken köşeli parantez kullanılmakta ancak liste veri modelinden farklı olarak array fonksiyonu dizi veri yapısının oluşturulmasını sağlar. Ayrıca birden fazla boyuta sahip diziler (matrisler) tanımlanırken parantez sayısı ve virgüler gözetilerek tanımlamalar yapılır.

```
9  dizi = np.array([10, 20, 30, 40, 50])
10 print("1 boyutlu bir dizi (vektör):",dizi)
11
12 dizi2 = np.array([[10, 20], [30, 40], [50, 60]])
13 print("3x2'lik bir dizi (matris):")
14 print(dizi2)
```

(a)

```
1 boyutlu bir dizi (vektör): [10 20 30 40 50]
3x2'lik bir dizi (matris):
[[10 20]
 [30 40]
 [50 60]]
```

(b)

Şekil 46. (a) Basit dizi tanımlama kodları (b) kodların çalıştırılması sonucu konsol çıktıları.

Numpy kütüphanesi başlangıçta liste olarak tanımlanmış değişkenleri diziye dönüştürebilmektedir. Bu fonksiyon özellikle farklı veri modelleri ile çalışan kütüphaneler (Örneğin, Pandas) ile Numpy arasında iletişim kolaylaşımaktadır. Şekil 47’de liste olarak tanımlanan bir değişkenin diziye dönüşümü gösterilmektedir.

```
18  liste = [3, 5, 7, 9, 11]
19  print("Liste:",liste)
20
21  dizi = np.array(liste)
22  print("Dizi:",dizi)
```

(a)

```
Liste: [3, 5, 7, 9, 11]
Dizi: [ 3  5  7  9 11]
```

(b)

Şekil 47. (a) Listeden diziye dönüşüm kodları (b) kodların çalıştırılması sonucu konsol çıktıları.

Numpy ile Farklı Dizi Tanımlama Şekilleri

Numpy kütüphanesi içerisindeki farklı fonksiyonlarla dizi tanımlama işlemlerini oldukça kolaylaştırmaktadır. Buna göre farklı dizi tanımlama fonksiyonlarının ve kullanım şekilleri genel olarak şöyledir:

- **arange:** Verilen aralıklardaki sayıları kullanarak diziler oluşturulmasını sağlamaktadır. İki parametrelî kullanım şekli birinci parametreden başlayıp, ikinci parametrenin bir eksigine kadar olan değerleri taşıyan bir dizinin oluşturulmasını sağlarken, üç parametrelî kullanımında üçüncü parametre artırım miktarını belirlemektedir. Örneğin **np.arange(0, 20)** kodu 0 ile 19 arasındaki değerlerin olduğu 20 elemanlı bir diziyi oluştururken, **np.arange(0, 20, 2)** kodu 0 ile 19 değerleri arasındaki değerleri 2'şer artırarak belirlemekte; toplamda 10 elemanlı bir dizi oluşturmaktadır.
- **zeros:** Parametre olarak verilen değer kadar satır ve sütun içeren; yalnızca 0'lardan oluşan bir “sıfır dizisi” oluşturmaktadır.
- **ones:** Parametre olarak verilen değer kadar satır ve sütun içeren; yalnızca 1'lerden oluşan bir “birler dizisi” oluşturmaktadır.
- **eye:** Parametre olarak verilen değer kadar satır ve sütun içeren bir “birim dizisi” (köşegenleri 1, diğer hücreleri 0 olan dizi) oluşturmaktadır.
- **linspace:** Üç parametre alan linspace fonksiyonu, birinci parametre ile ikinci parametrede belirtilen değerler arasında, üçüncü parametrede belirtilen değer kadar eleman içeren bir dizi oluşturmaktadır. Örneğin, **linspace(0, 6, 3)** kullanımı içeriği [0., 3., 6.] olan bir diziyi oluşturur.

Öğrencilerden her farklı tanımlama şekli için Şekil 48'de gösterilen kodları yazmaları sağlanır. Elde edilen çıktılar gözlemlendikten sonra öğrencilerin parametrelerde değişiklikler yaparak elde edilen farklı çıktıları yorumlamaları istenir.

```
26  dizi1 = np.arange(0, 20)
27  print("0 ile 20 arası değerler içeren dizi")
28  print(dizi1)
29
30  dizi2 = np.arange(2, 10, 2)
31  print("2 ile 10 arası 2 artırımılı değerleri içeren dizi")
32  print(dizi2)
33
34  dizi3 = np.zeros(5)
35  print("5x5 sıfır dizisi:")
36  print(dizi3)
37
38  dizi4 = np.ones(3)
39  print("3x3 birler dizisi:")
40  print(dizi4)
41
42  dizi5 = np.eye(6)
43  print("6x6 birim dizisi:")
44  print(dizi5)
45
46  dizi6 = np.linspace(0, 6, 3)
47  print("0 ile 6 arası toplamda 3 elaman içeren dizi:")
48  print(dizi3)
49
50  dizi6 = np.linspace(4, 10, 8)
51  print("4 ile 10 arası toplamda 8 elaman içeren dizi:")
52  print(dizi3)
```

(a)

```

0 ile 20 arası değerler içeren dizi
[ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19]
2 ile 10 arası 2 artırımlı değerleri içeren dizi
[2 4 6 8]
5x5 sıfır dizisi:
[0. 0. 0. 0. 0.]
3x3 birler dizisi:
[1. 1. 1.]
6x6 birim dizisi:
[[1. 0. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0. 0.]
 [0. 0. 1. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0.]
 [0. 0. 0. 0. 1. 0.]
 [0. 0. 0. 0. 0. 1.]]
0 ile 6 arası toplamda 3 elaman içeren dizi:
[0. 0. 0. 0. 0.]
4 ile 10 arası toplamda 8 elaman içeren dizi:
[0. 0. 0. 0. 0.]

```

(b)

Şekil 48. (a) Numpy ile farklı dizi tanımlama şekillerine dair kodları (b) kodların çalıştırılması sonucu elde edilen konsol çıktıları.

Rastgele Sayılar

Numpy kütüphanesinin sık kullanılan özellikler biri de rastgele sayılardır. Bu amaçla Numpy altında yer alan “**random**” bileşeni kullanılmaktadır. Genellikle Veri Bilimi ve Yapay Zeka süreçlerinde de ihtiyaç duyulan rastgele sayı üretimi, Numpy kütüphanesi random bileşeni altında sunulan farklı fonksiyonlarla yapılmaktadır. Söz konusu fonksiyonlar genel olarak şu şekildedir:

- **rand:** 0 ile 1 arasında rastgele reel sayılar üretmesini sağlamaktadır. Bir adet değer verildiğinde ilgili değer kadar sayı içeren bir dizi oluşturulurken, birden fazla değer satır ve sütun sayılarını (çok boyutlu dizi / matris oluşumunu) temsil etmektedir.
- **randn:** Rastgele sayı üretimini “normal dağılım” kuralına uyarak gerçekleştirilmektedir. Parametre kullanımı rand ile aynıdır.
- **randint:** Verilen parametreler aralığında yer alan “rastgele tamsayı” üretmesini sağlamaktadır. İki parametre değeri olması durumunda bir adet tamsayı, üç parametre değeri olması durumunda üçüncü parametredeki değer kadar tamsayı oluşturulur.

Öğrenciler farklı rastgele sayı oluşturma fonksiyonlarını denemek için Şekil 49’da gösterilen kodları yazar ve çalıştırır. Elde edilen çıktılar incelendikten sonra öğrencilerden parametreleri değiştirerek farklı rastgele sayı dizilerini oluşturmaları ve elde edilen dizileri yorumlamaları istenir.

```

56 r_dizi1 = np.random.rand(5)
57 print("0 ile 1 arası 5 adet rastgele reel sayı içeren dizi:")
58 print(r_dizi1)
59
60 r_dizi2 = np.random.rand(4, 4)
61 print("0 ile 1 arası rastgele reel sayılar içeren 4x4'lük dizi:")
62 print(r_dizi2)
63
64 r_dizi2 = np.random.rand(4, 4)
65 print("0 ile 1 arası rastgele reel sayılar içeren 4x4'lük dizi:")
66 print(r_dizi2)
67
68 r_dizi3 = np.random.randn(3, 3)
69 print("Normal dağılımdan rastgele reel sayılar içeren 3x3'lük dizi:")
70 print(r_dizi3)
71
72 print("5 ile 100 arası rastgele bir tam sayı:",np.random.randint(5,100))
73
74 r_dizi4 = np.random.randint(3, 300, 5)
75 print("3 ile 300 arası rastgele 5 adet tam sayı içeren dizi:")
76 print(r_dizi4)

```

(a)

```

0 ile 1 arası 5 adet rastgele reel sayı içeren dizi:
[0.05086508 0.42333382 0.74001877 0.3264336 0.42431861]
0 ile 1 arası rastgele reel sayılar içeren 4x4'lük dizi:
[[0.85501355 0.05674734 0.85952244 0.69896417]
 [0.38025585 0.67365434 0.28864102 0.8439194]
 [0.11196527 0.1281257 0.49983446 0.3840754]
 [0.78624122 0.16902713 0.18522105 0.43510515]]
0 ile 1 arası rastgele reel sayılar içeren 4x4'lük dizi:
[[0.72994968 0.84035607 0.50828165 0.89028123]
 [0.44836264 0.05693501 0.12230751 0.64470651]
 [0.77613757 0.48960943 0.24068175 0.39609448]
 [0.12824646 0.1746707 0.3947208 0.9286724]]
Normal dağılımdan rastgele reel sayılar içeren 3x3'lük dizi:
[[ 1.12501077 -0.73176346 -1.50427113]
 [ 0.16120241 0.2678975 -0.00290729]
 [ 1.48867597 0.12874028 -1.42084758]]
5 ile 100 arası rastgele bir tam sayı: 67
3 ile 300 arası rastgele 5 adet tam sayı içeren dizi:
[242 43 37 121 64]

```

(b)

Şekil 49. (a) Numpy ile farklı rastgele sayı oluşturma fonksiyonlarına dair kodları
(b) kodların çalıştırılması sonucu elde edilen konsol çıktıları.

Numpy ile Dizi İnceleme İşlemleri

Numpy kütüphanesi ile tanımlı durumdaki diziler hakkında çeşitli bilgiler elde etmek mümkündür. Bu tür işlemler, gerçekleştirilecek uygulama içerisinde yer alan dizi tabanlı verilerin gerekli noktalarda anlaşılması ve ilerleyen adımların tasarlanması için önem arz edebilmektedir. Yaygın şekilde kullanılan dizi inceleme fonksiyonları şöyledir:

- **shape:** İlgili dizinin satır, sütun ve varsa diğer boyutları içerisindeki hücre sayıları hakkında bilgi verir.
- **ndim:** Dizinin toplam kaç boyutu olduğu hakkında bilgi verir.

- **itemsize:** Tanımlanan dizinin içerisinde tutulan elemanların bellek boyutlarına dair bilgi verir.

Öğrenciler Şekil 50'de gösterilen komutları yazmak suretiyle tanımlanan diziler için elde edilen bulguları gözlemler ve dizilerin içeriğini değiştirmek suretiyle farklı diziler için ilgili fonksiyonların verdikleri çıktıları inceler.

```

80     x = np.array([[10, 15, 20, 30], [40, 50, 60, 70], [80, 90, 100, 110]])
81     print("Dizinin farklı boyut hücre değerleri: ",x.shape)
82     print("Dizinin toplam boyut sayısı: ",x.ndim)
83     print("Dizi elemanlarının bellek boyutu: ",x.itemsize)

```

(a)

```

Dizinin farklı boyut hücre değerleri: (3, 4)
Dizinin toplam boyut sayısı: 2
Dizi elemanlarının bellek boyutu: 4

```

(b)

Şekil 50. (a) Numpy dizi inceleme fonksiyonlarına dair kodlar (b) kodların çalıştırılması sonucu konsol çıktıları.



3. ADIM: ÜRET

Üret adımda öğrencilere Numpy ile alternatif dizi işlemleri ve Pandas ile Matplotlib işbirliği içerisinde gerçekleştirilebilecek çözümler kapsamında bilgiler verilir.

Alternatif Dizi İşlemleri

Python ortamında tanımlanan diziler üzerinde Numpy kütüphanesini kullanmak suretiyle birçok farklı işlem uygulanabilmektedir. İlgili işlemler genelde matematiksel işlemler, parça ulaşma, dilimleme, indeksleme ve manipülasyon yönünde olmaktadır.

Dizilerde Matematiksel İşlemler

Numpy içerisinde yer alan fonksiyonlar kullanılmak suretiyle diziler üzerinde farklı matematiksel işlemler gerçekleştirilebilmektedir. Diziler üzerinde gerçekleştirilebilecek temel matematiksel işlemler şu şekilde ifade edilebilir:

- **add:** Parametre olarak verilen dizilerin toplama işlemine tabi tutulması.
- **subtract:** Parametre olarak verilen dizilerin çıkarma işlemine tabi tutulması.
- **multiply:** Parametre olarak verilen dizilerin çarpma işlemine tabi tutulması.
- **sum:** Dizi elemanlarının toplamının bulunması.
- **min ve max:** Dizinin minimum ya da maksimum elemanının tespit edilmesi.
- **mean:** Dizi ortalama değerinin bulunması.

- **std:** Dizi standart sapma değerinin bulunması.

Öğrenciler ilgili fonksiyonları Şekil 51'de olduğu gibi iki adet dizi tanımlayarak gerçekleştirirler.

```

88 a = np.array([[1, 2, 3], [3, 4, 5], [6, 7, 8]])
89 b = np.array([9, 10, 11])
90
91 print("a ve b dizilerinin toplamı:\n",np.add(a,b))
92
93 print("a dizisinden b dizisinin çıkartılması:\n",np.subtract(a,b))
94
95 print("a ve b dizilerinin çarpımı:\n",np.multiply(a,b))
96
97 print("a dizisi elemanları toplamı:",np.sum(a),"ve","b dizisi elemanları toplamı:",np.sum(b))
98
99 print("a dizisi en küçük elemanı:",np.min(a),"ve","a dizisi en büyük elemanı:",np.max(a))
100
101 print("b dizisi en küçük elemanı:",np.min(b),"ve","b dizisi en büyük elemanı:",np.max(b))
102
103 print("a dizisi ortalaması:",np.mean(a),"ve","b dizisi ortalaması:",np.mean(b))
104
105 print("a dizisi standart sapması:",np.std(a),"ve","b dizisi standart sapması:",np.std(b))

```

(a)

```

a ve b dizilerinin toplamı:
[[10 12 14]
 [12 14 16]
 [15 17 19]]
a dizisinden b dizisinin çıkartılması:
[[-8 -8 -8]
 [-6 -6 -6]
 [-3 -3 -3]]
a ve b dizilerinin çarpımı:
[[ 9 20 33]
 [27 40 55]
 [54 70 88]]
a dizisi elemanları toplamı: 39 ve b dizisi elemanları toplamı: 30
a dizisi en küçük elemanı: 1 ve a dizisi en büyük elemanı: 8
b dizisi en küçük elemanı: 9 ve b dizisi en büyük elemanı: 11
a dizisi ortalaması: 4.33333333333333 ve b dizisi ortalaması: 10.0
a dizisi standart sapması: 2.211083193570267 ve b dizisi standart sapması: 0.816496580927726

```

(b)

Şekil 51. (a) Numpy ile diziler üzerinde matematiksel işlemlere yönelik kodlar
 (b) kodların çalıştırılması sonucu konsol çıktıları.

Dizi Parçalarına Ulaşma, Dilimleme ve İndeksleme

Numpy sayesinde diziler içerisindeki belli parçalara kolaylıkla ulaşılır. Bir dizinin belli parçalarına ulaşmak için köşeli parantezler içerisinde indeks değerlerinin , (virgül) karakterleri ile tanımlanması tercih edildiği gibi, : (iki nokta üst üste) karakterini kullanarak da dilimleme olarak adlandırılan daha esnek bir yaklaşım da kullanılabilmektedir.

Öğrenciler köşeli parantezlerle dizi parçalarına ulaşmayı gözlemlemek için Şekil 52'de verilen örnek kodları yazar ve çalıştırır. Öğrencilere `dtype=np.int64` kodu ile dizi içerisindeki verilerin değişken türü ve boyutlarının belirlendiği açıklanır. Ayrıca dizinin tanımlanmasında kullanılan köşeli parantezler irdelenmek suretiyle konsolda elde edilen çıktılara nasıl ulaşıldığı

pekiştirilir. Söz konusu kodlar içerisinde yer alan indeks değerleri değiştirilmek suretiyle elde edilen farklı çıktılar yorumlanır.

```
110 cok_boyutlu_dizi = np.array([[1,2,3,4], [5,6,7,8]],  
111 [[1,2,3,4], [9,10,11,12]]], dtype=np.int64)  
112 print(cok_boyutlu_dizi[1,1,1])  
113 print(cok_boyutlu_dizi[0],[1])  
114 print(cok_boyutlu_dizi[0,1,1])
```

10
[[1 2 3 4]
[5 6 7 8]] [1]
6

(a)

(b)

Şekil 52. (a) Dizi parçalarına ulaşım kapsamında kodlar (b) kodların çalıştırılması sonucu konsol çıktıları.

Dilimleme ile dizi parçalarına erişim, doğrudan indeksli parçaya erişimden ziyade diziye bölgесel erişim olarak bilinmektedir. Bu amaçla tipki dataframe veri modellerindeki erişim yaklaşımlarında olduğu gibi (Bkz. 1. Hafta – 3. Gün – 1. Ders) iki nokta üst üste karakteri kullanılır.

Öğrenciler Şekil 53'te olduğu gibi, **cok_boyutlu_dizi** adıyla tanımlanan aynı diziyi kullanarak farklı dilimlemeler ile ulaşılan çıktıları gözlemler, kendi indeks değerlerini deneyerek çıktılardaki farklılıklarını yorumlar.

```
118 print("Dilimleme-1:\n",cok_boyutlu_dizi[0:3])  
119 print("Dilimleme-2:\n",cok_boyutlu_dizi[0:5,1])  
120 print("Dilimleme-3:\n",cok_boyutlu_dizi[1,0:])
```

(a)

```
Dilimleme-1:  
[[[ 1 2 3 4]  
[ 5 6 7 8]]  
  
[[ 1 2 3 4]  
[ 9 10 11 12]]]  
Dilimleme-2:  
[[[ 5 6 7 8]  
[ 9 10 11 12]]]  
Dilimleme-3:  
[[[ 1 2 3 4]  
[ 9 10 11 12]]]
```

(b)

Şekil 53. (a) Dizi dilimleme yönünde kodlar (b) kodların çalıştırılması sonucu konsol çıktıları.

Dizi parçalarına ulaşım ya da farklı bir bakış açısıyla mevcut dizilerden yeni parçaların elde edilmesi için **indeksleme** adı verilen; mantıksal kontrollere dayanan kodlar da tanımlanabilmektedir. Kullanımı oldukça pratik olan bu yaklaşım ile öncelikli olarak şartlara uyan hücreler tespit edilmekte, ardından bu diziler ile şartlara uyan yeni dizi oluşturulabilmektedir.

Öğrencilerden Şekil 54'te gösterilen yeni dizi değişkenini tanımlamaları istenir. Ardından öğrenciler devam eden satırlarda verilen farklı indeksleme kodlarını yazarak, elde edilen sonuçları yorumlar.

```
123 cok_boyutlu_dizi2 = np.array([[[11,200,-3,-45], [105,126,71,-88]],
124                                [[1,20,-33,44], [900,0,121,619]],
125                                [[67,-34,-677,0], [0, 0, 34, -89]]], dtype=np.int64)
126
127 pozitifler = (cok_boyutlu_dizi2 > 0)
128 negatifler = (cok_boyutlu_dizi2 < 0)
129 sifirlar = (cok_boyutlu_dizi2 == 0)
130 cift_sayilar = (cok_boyutlu_dizi2 % 2 == 0)
131
132 print("Pozitifler:\n",cok_boyutlu_dizi2[pozitifler])
133 print("Negatifler:\n",cok_boyutlu_dizi2[negatifler])
134 print("Sifirlar:\n",cok_boyutlu_dizi2[sifirlar])
135 print("Cift Sayilar:\n",cok_boyutlu_dizi2[cift_sayilar])
```

(a)

```
Pozitifler:
 [ 11 200 105 126  71   1  20   44 900 121 619  67  34]
Negatifler:
 [-3 -45 -88 -33 -34 -677 -89]
Sifirlar:
 [0 0 0 0]
Cift Sayilar:
 [200 126 -88  20  44 900   0 -34   0   0   0  34]
```

(b)

Şekil 54. (a) Farklı dizi indeksleme adımlarına yönelik kodlar (b) kodların çalıştırılması sonucu konsol çıktıları.

Diziler Üzerinde Manipülasyonlar

Tanımlanan diziler, geliştirilen uygulama ihtiyaç duyulan çözüm adımlarına göre çeşitli manipülasyonlara tabi tutulabilmektedir. Genellikle Yapay Zeka uygulamalarında verilerin diziler / matrisler içerisinde temsil edilmesi, bu verilerin Makine Öğrenmesi ve Derin Öğrenme adımlarında transpoze edilmesi (hücrelerin satır ve sütun ekseninde yer değiştirmesi), boyut çerçevesinde güncellenmesi ya da dizi içerisinde güncellenmeleri gibi işlem basamaklarını kolaylaştırmaktadır.

Python ortamında tanımlanan diziler, Numpy kütüphanesinin sağladığı “**transpose**” fonksiyonu ya da “**T**” bileşeni ile pratik bir biçimde transpoze işleminden geçirilebilir. Öğrenciler Şekil 55'de gösterilen kodları yazmak suretiyle tanımlı durumdaki **cok_boyutlu_dizi2** değişkeninin transpoze edildiğini ve boyut değerlerinin değişmiş olduğunu tespit eder.

```

139 t_cok_boyutlu_dizi2 = cok_boyutlu_dizi2.transpose()
140 print("Transpoze değişken içeriği:\n"
141     ,t_cok_boyutlu_dizi2)
142 print("T ile dönüşüm:\n"
143     ,cok_boyutlu_dizi2.T)
144 print("Transpoze öncesindeki dizinin boyutları:\n"
145     ,cok_boyutlu_dizi2.shape)
146 print("Transpoze sonucu dizinin boyutları:\n"
147     ,t_cok_boyutlu_dizi2.shape)

Transpoze değişken içeriği:
[[[ 11   1   67]
 [ 105  900   0]]]

[[ 200   20  -34]
 [ 126    0    0]]

[[ -3  -33 -677]
 [ 71  121   34]]

[[ -45   44    0]
 [ -88  619  -89]]]

T ile dönüşüm:
[[[ 11   1   67]
 [ 105  900   0]]]

[[ 200   20  -34]
 [ 126    0    0]]

[[ -3  -33 -677]
 [ 71  121   34]]

[[ -45   44    0]
 [ -88  619  -89]]]

Transpoze öncesindeki dizinin boyutları:
(3, 2, 4)
Transpoze sonucu dizinin boyutları:
(4, 2, 3)

```

(a)

(b)

Şekil 55. (a) Dizi transpoze işlemlerine dair kodlar (b) kodların çalıştırılması sonucu konsol çıktıları.

Numpy ile diziler üzerinde yaygın bir biçimde uygulanan bir diğer işlem dizi boyutlarının istenilen değerlere göre yeniden revize edilmesidir. Bu durumda dizi içeriğindeki verilerde herhangi bir değişim olmadan satır, sütun ve tanımlanmışsa diğer boyutların değerleri mevcut dizi üzerinde uygulanarak dizinin yeniden şekillenmesi sağlanabilmektedir. Söz konusu amaç doğrultusunda Numpy içerisinde kullanılabilen fonksiyon “**resize**” fonksiyonudur.

Öğrencilerden Şekil 56’da olduğu gibi **cok_boyutlu_dizi3** adındaki yeni dizi değişkenini tanımlamaları ve bu değişken üzerinde yeniden şekillenme sağlayacak örnek kodu yazmaları istenir. Değişkenin kod çalıştırılmadan önce ve sonrasında Variable Explorer içerisinde gözlemlenmesi sağlanır.

```

153 cok_boyutlu_dizi3 = np.array([[-11,  20,  78,  79,  90],
154                               [103,  416, 503, -45,  77],
155                               [-19, -288,  80, 111, 233]])
156
157 cok_boyutlu_dizi3.resize(5,3)

```

(a)

Variable Explorer window showing a 3x5 NumPy array:

	0	1	2	3	4
0	-11	20	78	79	90
1	103	416	503	-45	77
2	-19	-288	80	111	233

Buttons at the bottom: Format, Resize, Background color, Save and Close, Close.

(b)

Variable Explorer window showing the same 3x5 NumPy array after modification:

	0	1	2
0	-11	20	78
1	79	90	103
2	416	503	-45
3	77	-19	-288
4	80	111	233

Buttons at the bottom: Format, Resize, Background color, Save and Close, Close.

(c)

Şekil 56. (a) Dizi yeniden şekillendirmeye dair kod (b) kodun çalıştırılması öncesinde Variable Explorer ortamındaki değişken içeriği (c) kodların çalıştırılması sonrasında Variable Explorer ortamındaki değişken içeriği.

Python ortamında yer alan herhangi bir dizi değişkenine, “**append**” fonksiyonu yardımıyla bir başka dizi içeriği eklenebilmektedir. Fonksiyonun çalışma şekli parametre olarak verilen dizinin sonuna yeni dizi içeriğinin eklenmesine dayanmaktadır. Şayet yeni dizi içeriğinin mevcut dizideki belli bir noktasından itibaren eklenmesi istenirse bu durumda “**insert**” fonksiyonu tercih edilmektedir. Benzer şekilde dizideki belli bir indeks konumunda yer alan bir verinin silinmesi için “**delete**” fonksiyonu kullanılmaktadır.

Öğrencilerin append, insert ve delete fonksiyonlarını anlamaları adına, Şekil 57'de gösterilen **veri** adlı dizi değişkeni tanımlamaları istenir ve ardından takip eden kodların yazılması suretiyle dizi üzerinde gerçekleşen değişimlerin gözlemlenmesi sağlanır. İlgili kodlarda sayısal değerlerle birlikte metin tabanlı (string) değerlerin de kullanıldığı vurgulanır; Python ortamının bu tür farklı veri türlerinin bir arada tutulmasını kolaylaştırdığı açıklanır. Ayrıca append, insert ve delete fonksiyonlarının yazım şekilleri sonucunda hangi değişkenlerin etkilendiği konusunda gözlemler yapılarak yazılacak kod içeriğinin her zaman dikkatli tasarlanması gerektiği vurgulanır.

```

160  veri = np.array(["metin1", 20, "metin2", 100])
161  print("Ekleme öncesi dizi:", veri)
162  yeni_veri = np.append(veri, ["metin3", 70, "metin4", 50])
163  print("Append sonrası yeni dizi:", yeni_veri)
164
165  veri = np.insert(veri, 3, "metin3")
166  print("Insert sonrası dizi:", veri)
167
168  yeni_veri2 = np.delete(veri,[1])
169  print("Delete sonrası yeni dizi içeriği:", yeni_veri2)

```

(a)

```

Ekleme öncesi dizi: ['metin1' '20' 'metin2' '100']
Append sonrası yeni dizi: ['metin1' '20' 'metin2' '100' 'metin3' '70' 'metin4' '50']
Insert sonrası dizi: ['metin1' '20' 'metin2' 'metin3' '100']
Delete sonrası yeni dizi içeriği: ['metin1' 'metin2' 'metin3' '100']

```

(b)

Şekil 57. (a) Append, insert ve delete fonksiyonlarının kullanımına dair kodlar
(b) kodların çalıştırılması sonucu konsol çıktıları.

Pandas ve Matplotlib ile Kullanım

Anlatılan alternatif dizi işlemlerinden de yola çıkmak suretiyle, Numpy ile Pandas ve Matplotlib'in bir arada kullanıldığı aşamalar Veri Bilimi ve Yapay Zeka uygulamalarında veri işleme ve kullanıcı etkileşimi mekanizmalarının zenginleştirilmesine katkıda bulunmaktadır.

Pandas ortamında oluşturulan series ve dataframe modelleri, taşıdıkları index ve column alanları neticesinde verilere erişimi ve düzenleme yollarını daha pratik ve çeşitli hale getirir. Ancak bunun aksine Numpy ile oluşturulan diziler daha çok verilerin üzerinde gerçekleştirilebilecek dizi odaklı işlemler konusunda güçlündür. Bu nedenle Numpy ile oluşturulan diziler, Pandas ortamındaki series ve dataframe modelleri içeresine entegre edilerek daha esnek hale getirilebilmektedir.

Öğrencilerin geçen süre içerisinde yazmış oldukları kodlarda henüz Pandas çağrıları yapılmadığı için öncelikli olarak Pandas kütüphanesini çağrımaları istenir (Şekil 58). Ardından yine Şekil 58'de gösterilen kodları yazmak suretiyle standart bir Numpy dizisi Pandas series ve dataframe veri modelleri ile genişletilir. Dizi, series ve dataframe değişken içeriklerinin konsoldaki yansımaları gözlemlenerek farklar irdelenir. Gerçekleştirilen uygulama üzerinden Numpy ve

Pandas işbirliği ile Veri Bilimi ve Yapay Zeka uygulamalarındaki veri setlerinin uygun veri içeriği ve index-column unsurları ile yönetilebileceği vurgulanır.

```

173 import pandas as pd
174
175 boy = np.array([171, 183, 190, 176])
176 series = pd.Series(boy,index=['Ahmet', 'Ayşe', 'Hasan', 'Fatma'])
177 dataframe = pd.DataFrame(boy,index=['Ahmet', 'Ayşe', 'Hasan', 'Fatma'],
178                           columns=['Boy'])
179 print("Dizi:\n",boy)
180 print("Series:\n",series)
181 print("Dataframe:\n",dataframe)

```

```

Dizi:
[171 183 190 176]
Series:
Ahmet    171
Ayşe     183
Hasan    190
Fatma    176
dtype: int32
Dataframe:
          Boy
Ahmet   171
Ayşe    183
Hasan   190
Fatma   176

```

(a)

(b)

Şekil 58. (a) Diziden series ve dataframe veri modellerine genişletme kodları
(b) kodların çalıştırılması sonucu konsol çıktıları.

Python uygulama ortamında dataframe olarak tutulan bir veriden diziye geçiş de yapılabilmektedir. Öğrencilerden önceki kodlar ile oluşturmuş oldukları dataframe değişkenine “Boy (inç)” adında yeni bir alan ekleyerek mevcut boy değerlerini 0,394 ile çarpmak suretiyle her bir boy değerinin inch karşılığını elde etmeleri istenir. Ardından dataframe değişkeninde yer alan bu yeni sütundan Numpy yardımıyla boy_inch adı altında bir dizi oluşturulması sağlanır (Şekil 59).

```

185 dataframe["Boy (inç)"] = dataframe["Boy"] * 0.394
186 boy_inc = np.array(dataframe["Boy (inç)"])
187
188 print("Dataframe:\n",dataframe)
189 print("Boy (inç) dizisi:\n",boy_inc)

```

```

Dataframe:
          Boy  Boy (inç)
Ahmet   171   67.374
Ayşe    183   72.102
Hasan   190   74.860
Fatma   176   69.344
Boy (inç) dizisi:
[67.374 72.102 74.86 69.344]

```

(a)

(b)

Şekil 59. (a) Dataframe veri modelinden diziye geçiş (b) kodların çalıştırılması sonucu konsol çıktıları.

Numpy kütüphanesi ile oluşturulan diziler Matplotlib kütüphanesi ile oluşturulan görselleri de besleyebilmektedir. Matplotlib Pyplot ile elde edilmesi planlanan herhangi bir grafikte veri parametrelerine dizilerin aktarılması, ilgili dizilerin kolaylıkla görsel çıktılar halinde elde edilmesini sağlamaktadır.

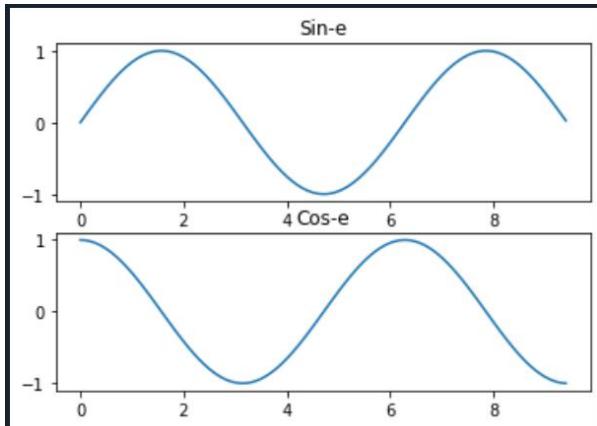
Öğrenciler Matplotlib Pyplot çağrıları yaparak ilgili kütüphanenin kodlama ortamına dahil edilmesini sağlar (Numpy kütüphanesinin sürekli kodlamalardan hareketle çağrılmış durumda olduğu varsayılar). Ardından Numpy’ın “arange” fonksiyonunu kullanmak suretiyle sin ve cos sayı dizileri oluşturur ve bu dizilerin subplot’lar altından görüntülenmesini sağlar (Şekil 60). Bu kod yazımında Numpy kütüphanesinden “pi” değerinin de çağırıldığı

vurgulanır. İstendiği takdirde söz konusu çağrı bölümünde 3.14 değeri de kullanılabileceği açıklanır.

```

193 import matplotlib.pyplot as plt
194
195 x = np.arange(0, 3 * np.pi, 0.1)
196 sin_y = np.sin(x)
197 cos_y = np.cos(x)
198
199 plt.subplot(2, 1, 1)
200 plt.plot(x, sin_y)
201 plt.title('Sin-e')
202
203 plt.subplot(2, 1, 2)
204 plt.plot(x, cos_y)
205 plt.title('Cos-e')
206
207 plt.show()

```



(a)

(b)

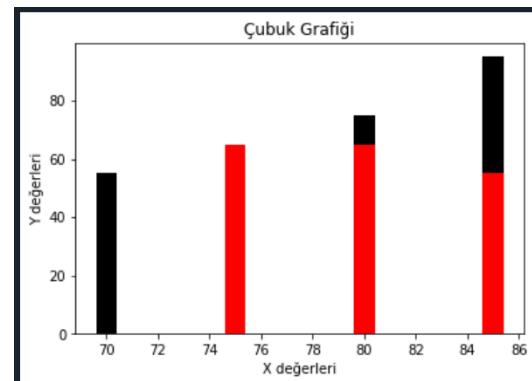
Şekil 60. (a) Numpy ve Matplot Pyplot kullanımı ile sin-cos eğrilerinin çizimi
 (b) kodların çalıştırılması sonucu elde edilen görsel.

Dizi değişkenlerinin x ve y eksenlerini beslemesi yoluyla farklı grafik türleri de elde edilebilmektedir. Bu doğrultuda öğrencilerden Şekil 61'de olduğu gibi x ve y dizi değişkenlerini tanımlayıp Matplot Pyplot üzerinden çubuk grafiği çizmeleri sağlanır.

```

211 x_1 = [70, 80, 85]
212 y_1 = [55, 75, 95]
213 x_2 = [75, 85, 80]
214 y_2 = [65, 55, 65]
215
216 plt.bar(x_1, y_1, color = 'black', align = 'center')
217 plt.bar(x_2, y_2, color = 'red', align = 'center')
218 plt.title('Çubuk Grafiği')
219 plt.ylabel('Y değerleri')
220 plt.xlabel('X değerleri')
221
222 plt.show()

```



(a)

(b)

Şekil 61. (a) Numpy ve Matplot Pyplot kullanımı ile çubuk grafiği çizimi
 (b) kodların çalıştırılması sonucu elde edilen görsel.



4. ADIM: İLERLET

Öğrencilerin bireysel olarak Numpy kütüphanesi ile uygulamalar gerçekleştirmeleri istenir. Bu etkinliğe 20 dakikalık bir süre ayrılr.



5. ADIM: DEĞERLENDİRME

Derste öğrenilen bilgilerden hareketle soru-cevap süreci gerçekleştirilir ve Python Numpy kütüphanesi kullanımı genel hatlarıyla pekiştirilir.

Öğrencilere şu sorular sorulabilir:

- Numpy kütüphanesi hakkında neler düşünüyorsunuz?
- Numpy kütüphanesinde en çok hangi fonksiyonları başarılı buldunuz?
- Numpy kütüphanesi Yapay Zeka uygulamalarında ne gibi avantajlar sağlayabilir?

1. HAFTA – 4. GÜN – 1. DERS: HUGGING FACE / KAGGLE TANITIMI

DERS PLANI

DERS ETİKETLERİ



KONU: Hugging Face / Kaggle Tanıtımı



SINIF: 12-14 Yaş



SÜRE: 60 dk.



HAZIR BULUNUŞLUK: Temel İngilizce eğitimi, Yapay Zeka farkındalığı, Programlamaya yönelik temel bilgi



ANAHTAR KELİMELER: Yapay Zeka, Kodlama, Python, Hugging Face, Kaggle



BECERİLER:



Tahmin



Sorgulama



Günlük hayat bağlantıları kurma



KAZANIMLAR:

Bu derste öğrenciler;

- Hugging Face ve Kaggle platformları hakkında genel bilgi sahibi olur.
- Hugging Face ve Kaggle platformlarında sunulan farklı araçlar ve arayüzler hakkında bilgi ve beceri sahibi olur.
- Yapay Zeka uygulamaları geliştirmede Hugging Face ve Kaggle platformlarından nasıl yararlanabileceği konusunda farkındalık elde eder.



DERS MATERİYALLERİ:

Malzeme	Miktar
Yazı Tahtası	Öğrenci sayısı ve sınıf büyüklüğüne göre en az bir adet
Projeksiyon	Öğrenci sayısı ve sınıf büyüklüğüne göre en az bir adet
Defter, Kalem	Öğrenci sayısı kadar
Bilgisayar	Öğrenci sayısının yarısı kadar
Tablet	Öğrenci sayısının yarısı kadar



ÖĞRETMEN HAZIRLIK ÇALIŞMALARI:

Okunabilecek Kaynaklar:

Coderspace. (2023). *Kaggle*. Coderspace Sözlük. Çevrimiçi: <https://coderspace.io/sozluk/kaggle> (Erişim 9 Haziran 2024).

Lutkevich. B. (2023). *Hugging Face*. Tech Target Web Site. Çevrimiçi: <https://www.techtarget.com/whatis/definition/Hugging-Face> (Erişim 9 Haziran 2024).

İzlenebilecek Kaynaklar:

Geeks for Geeks. (2021). How to Use Kaggle For Learning Data Science?. Çevrimiçi: <https://www.youtube.com/watch?v=u9MIwoFWXVg> (Erişim 10 Haziran 2024).

Hugging Face. (2022). Welcome to the Hugging Face course. Çevrimiçi: https://www.youtube.com/watch?v=00GKzGyWFEs&list=PLo2EIpI_JMQvWfQndUesu0nP_BAtZ9gP1o (Erişim 10 Haziran 2024).

ATÖLYE HAZIRLIK ÇALIŞMALARI:

Öğretmen;

- Dersten önce ders materyallerini hazırlamalıdır.
- Hugging Face ve Kaggle platformları hakkında ön bilgi edinmeli, söz konusu platformların kullanımı konusunda bilgi ve beceri sahibi olmalıdır.



KAYNAKÇA:

Banachewicz, K., & Massaron, L. (2022). *The Kaggle Book: Data analysis and machine learning for competitive data science*. Packt Publishing Ltd.

Rebelo, M. (2023). *What is Hugging Face?*. Zapier. Çevrimiçi: <https://zapier.com/blog/hugging-face/> (Erişim 11 Haziran 2024).

Rothman, D. (2022). *Transformers for Natural Language Processing: Build, train, and fine-tune deep neural network architectures for NLP with Python, Hugging Face, and OpenAI's GPT-3, ChatGPT, and GPT-4*. Packt Publishing Ltd.

GENEL BAKIŞ:

- 1) Harekete Geç:** Öğrencilere Yapay Zeka ve Veri Bilimi uygulamalarına yönelik platformlardan hangileri hakkında bilgi sahibi oldukları sorulur ve görüşler tartışılar.
- 2) Keşfet:** Öğrencilere Hugging Face ve Kaggle platformları hakkında genel bilgi verilir.
- 3) Üret:** Öğrencilere Hugging Face ve Kaggle platformlarında yer alan araçlar ve arayüzler açıklanır.
- 4) İlerlet:** Öğrencilerin Hugging Face ve Kaggle ortamlarını genel hatlarıyla deneyimlemeleri ve elde ettikleri bilgileri / deneyimleri not almaları istenir.
- 5) Değerlendir:** Alınan notlar ve derste öğrenilenler ışığında soru-cevap etkinliği gerçekleştirilecek Hugging Face ve Kaggle platformlarına dair bilgiler pekiştirilir.

UYGULAMA



1. ADIM: HAREKETE GEÇ

Bu adımda öğrencilere Yapay Zeka ve Veri Bilimi uygulamalarına yönelik platformlardan hangileri hakkında bilgi sahibi oldukları sorulur ve görüşler tartışılar. Söz konusu tartışma süreci için 10 dakikalık bir zaman ayrıılır.



2. ADIM: KEŞFET

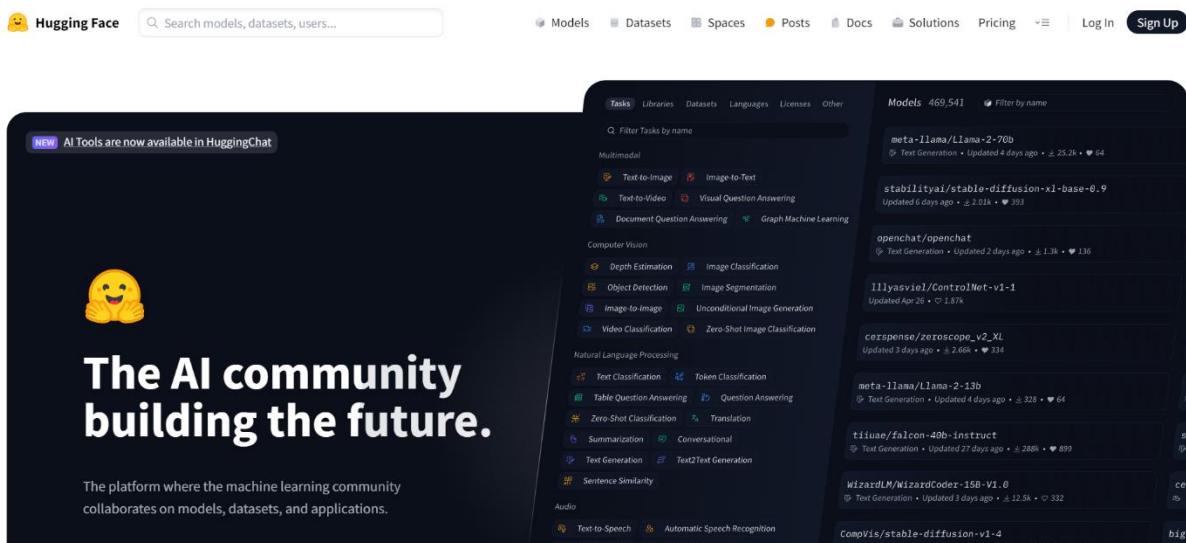
Keşfet adımda öğrencilere Hugging Face ve Kaggle platformları hakkında genel bilgi verilir.

Hugging Face Platformu

Hugging Face, ilk olarak 2016 yılında New York ABD'de bir chat bot uygulaması fikriyle ortaya çıkan, ancak zamanla bir Yapay Zeka ve Veri Bilimi platformuna dönüşerek yazılım

geliştiricilerin ilgi odağına dönüsen kitlesel bir araçtır. Üç Fransız girişimcinin çabalarıyla hayatı geçen Hugging Face, özellikle transformer ağ yapılarına yönelik kütüphaneleri Yapay Zeka dünyasına kazandıran ve açık kaynak Makine Öğrenmesi çözümlerine ivme kazandıran bir platform olarak bilinmektedir. Hatta son yıllarda dil modellerinin gelişimiyle birlikte Hugging Face platformunun özellikle doğal dil işleme modelleri geliştirme ve paylaşım yönünde ön plana çıkan bir platform olduğu da kabul edilmektedir (Rebelo, 2023; Rothman, 2022).

Hugging Face platformuna <https://huggingface.co/> adresinden erişim sağlanarak ücretsiz bir şekilde üye kaydı oluşturulabilir (Şekil 62). Yine platform üzerinde gerçekleştirilecek ileri düzey uygulama geliştirme süreçleri için çeşitli ücretle tabi planlara da geçiş yapılmaktadır. Öğrencilerin Hugging Face üzerinden kendilerine kayıt oluşturmaları ve sisteme giriş yapmaları sağlanır.



Şekil 62. Hugging Face platformu giriş sayfası.

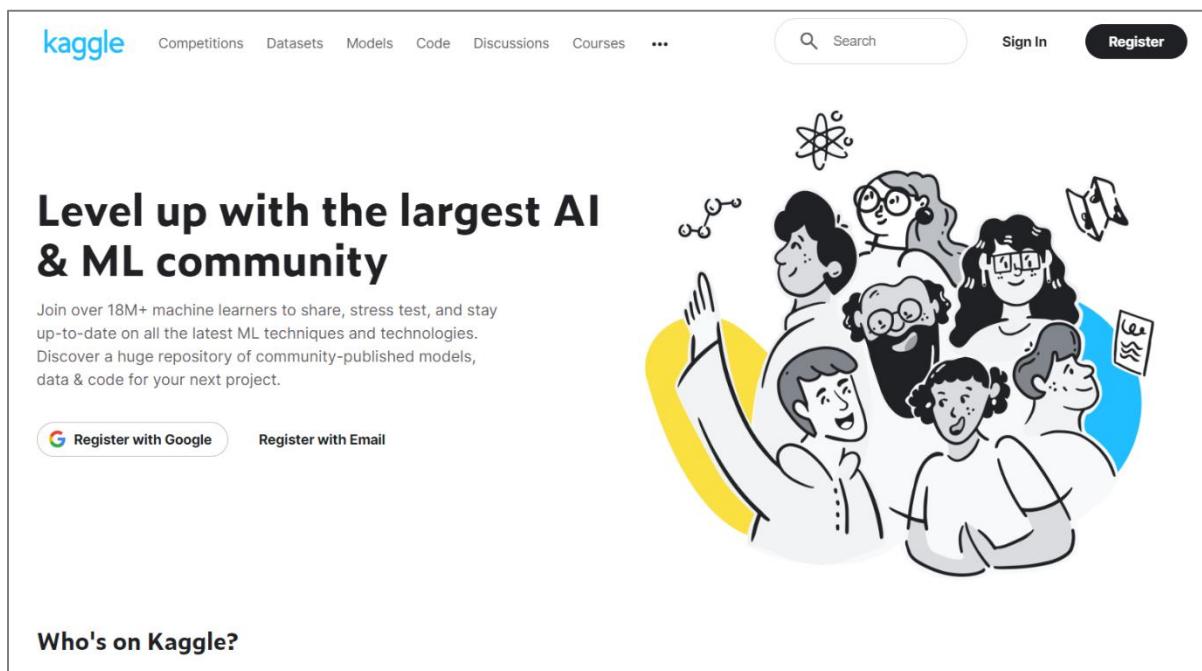
Hugging Face platformunun kullanıcılarına sağladığı hizmetler genel hatlarıyla şöyledir:

- Kullanıcılar kendi Yapay Zeka modellerini kodlayıp platforma yükleyebilirler ya da platformdaki hazır modellerden ve kullanıcıların tanımladıkları kodlardan faydalananabilmektedir.
- Hugging Face platformundan Yapay Zeka uygulamalarında kullanılabilecek veri setlerine erişim sağlanabilmekte, kullanıcılar kendi veri setlerini yine platform üzerinden paylaşabilmektedir.
- Hugging Face platformu kullanıcılar kendi profillerini ve portfolyo ortamlarını oluşturmalarını, farklı kullanıcılarla işbirlikleri içerisine girmelerini oldukça kolay hale getirmektedir.
- Hugging Face platformunun sunduğu transformer yapılar, NLP ve görüntü işleme odaklı güncel modeller platformun sunduğu API kanalı üzerinden kullanılabilen, eğitim, test, fine-tuning (eğitilmiş modele ince ayar uygulanması) işlemleri etkin bir biçimde gerçekleştirilebilmektedir.

- Kullanıcılar platformda uygulamalarına yönelik demolar gerçekleştirebilmekte, araştırma projelerine dahil olabilmekte ve hatta iş süreçlerine yönelik geliştirmelerde platformun sunduğu araçlardan faydalananabilmektedir.
- Hugging Face platformu genel ölçekte açık kaynak paylaşım prensiplerinden hareketle Yapay Zeka'ya ilgi duyan herkese aktif bir paylaşım ekosistemi ortaya koymaktadır.

Kaggle Platformu

Kaggle, tıpkı Hugging Face gibi; Yapay Zeka ve Veri Bilimi uygulamalarına yönelik geliştirilmiş kitlesel bir platformdur. <https://www.kaggle.com/> adresinden erişilebilen platform (Şekil 63) ilk olarak 2010 yılında kurulmuş, zamanla popüleritesi artmış ve özellikle kullanıcılar arası yarışmalar (competitions) özelliği ile ön plana çıkmıştır. Google şirketi 2017 yılında Kaggle'ı satın almıştır ve halen platformu bünyesinde tutmaktadır. Platformda yarışmaların haricinde Yapay Zeka modelleri, veri setleri ve problemlere özgü kullanıcılar tarafından oluşturulmuş kodlar paylaşılmaktadır (Banachewicz, & Massaron, 2022).



Şekil 63. Kaggle platformu giriş sayfası.

Kaggle platformu ile kullanıcılara sunulan hizmetler kısaca şu şekildedir:

- Kullanıcılar platformda yer alan özel yarışmalarda problemlerin çözümüne yönelik Yapay Zeka uygulamalarını geliştirebilmekte, elde ettikleri sonuçlara göre Kaggle ortamındaki sıralamalara (rankings) dahil olabilmektedir.
- Kaggle platformundan Yapay Zeka uygulamalarına dahil edilebilecek veri setlerine erişilebilmekte, yine özgün veri setleri de platform ortamında paylaşılabilmektedir.
- Platformda sunulan Yapay Zeka uygulama kodlarına erişilebilmekte, alternatif çözümlere, kullanılan veri setlerine dair bilgiler pratik bir şekilde edinilmektedir.
- Tıpkı Hugging Face platformu gibi Kaggle platformu da kullanıcılara yaygın kullanılan modelleri halihazırda sunmaktadır.

- Yine Hugging Face platformuyla benzer bir biçimde Kaggle'in sunduğu kendine has ekosistem, Yapay Zeka ve Veri Bilimi'ne ilgi duyan ve bu alanlarda faaliyetlerde bulunan bütün kullanıcıları aktif bir iletişim-paylaşım ortamına dahil etmektedir.

Öğrenciler esasında daha önce Kaggle platformuna üye olmuş olduğu varsayılmaktadır (Bkz. 1. Hafta – 3. Gün – 2. Ders). Ancak üye olmayanların olması durumunda yine platformu ziyaret ederek hesap oluşturmaları ve sisteme giriş yapmaları istenir. Ardından öğrencilerin Hugging Face ve Kaggle arayüzleri hakkında edindikleri ilk izlenimler tartışılar.

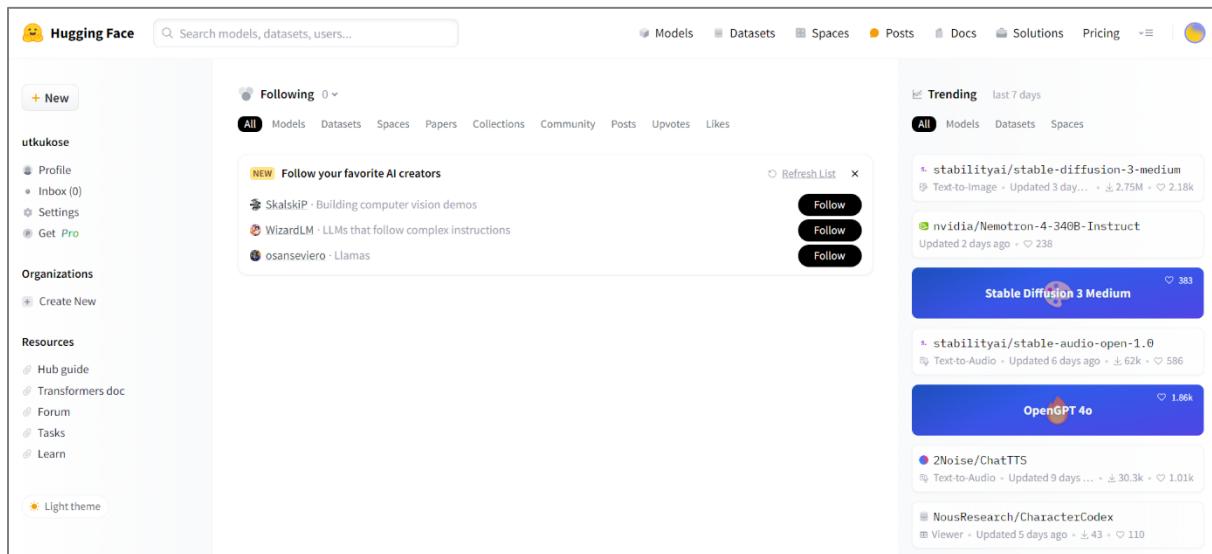


3. ADIM: ÜRET

Öğrencilere Hugging Face ve Kaggle platformlarında yer alan araçlar ve arayüzler açıklanır.

Hugging Face Platformunda Araçlar / Arayüzler

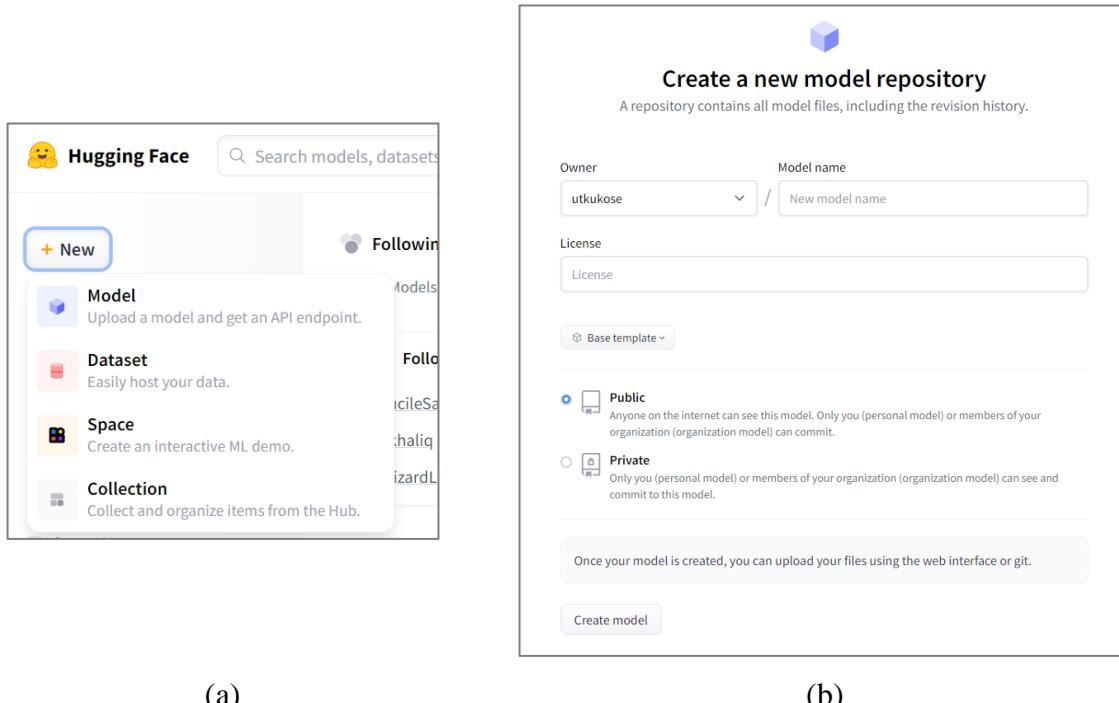
Hugging Face ortamına giriş yapıldığında kullanıcıyı karşılayan arayüzde farklı araçlara erişim sağlayan düğmeler ve trend durumda olan modellerin, veri setlerinin ya da demoların gerçekleştirildiği space alanlarının görüldüğü paneller yer almaktadır. Arayüzün orta alanı ise kullanıcının takibe aldığı diğer kullanıcıları ya da model, veri seti gibi bileşenleri listelemektedir (Şekil 64).



Şekil 64. Kaggle platformu kullanıcı arayüzü.

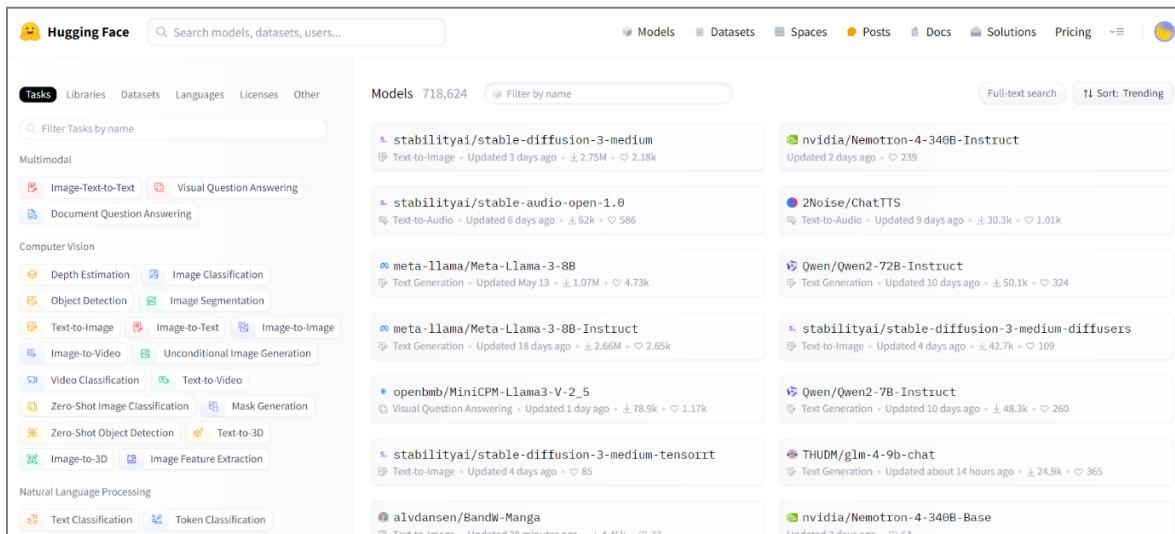
Öğrencilerin kullanıcı arayüzüni incelemeleri, popüler profillerden ve model, veri seti bileşenlerinden istediklerini takip etmeleri istenir. Ardından öğrencilerin dikkati sol üst köşedeki New düğmesine çekilir. Kullanıcılar New düğmesine tıklayarak kendi model, veri seti, demo ya da koleksiyonlarını Hugging Face ortamına dahil edebilmektedir. Yüklenen bileşenler herkese açık (public) ya da özel (private) olarak paylaşılabilme, belli lisans modelleri ile kullanımına açılabilmektedir (Şekil 65). Burada oluşturulan bileşenler kullanıcıları bir tür

GitHub benzeri kaynak kod paylaşım arayüzüne götürmektedir (Bu aşamada öğrencilere GitHub ortamı hakkında görüşleri sorularak Deneyap Yapay Zeka atölyelerine ait <https://github.com/deneyapyz> profili örnek olarak gösterilir).



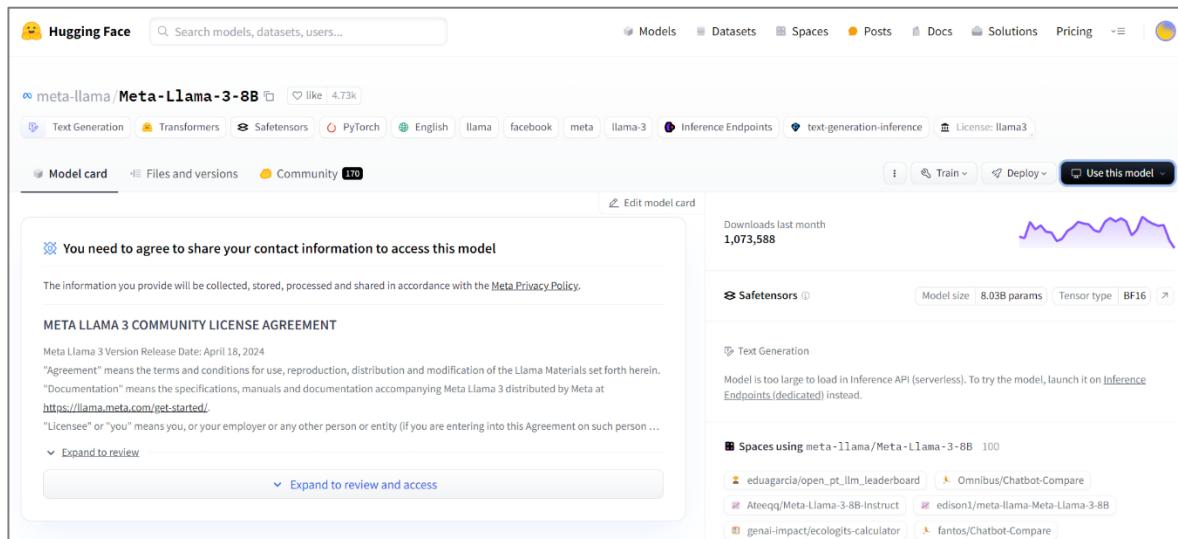
Şekil 65. (a) Hugging Face New düğmesi seçenekleri (b) Yeni model ekleme arayüzü.

Hugging Face'in en önemli avantajlarından biri kullanıcıların tanınmış modelleri ya da başka kullanıcıların paylaştığı modelleri kendi çalışmalarına Python kütüphane yapıları gibi dahil edebilmeleridir. Öğrencilerin kullanıcı arayüzündeki Models düğmesine tıklayarak kullanıma açık durumda modelleri listelemeleri ve herhangi bir modeli seçmeleri istenir (Şekil 66). Söz konusu arayüzde sol alandaki anahtar kelimeler yardımıyla amaca uygun modellerin filtrelenmesi mümkündür.

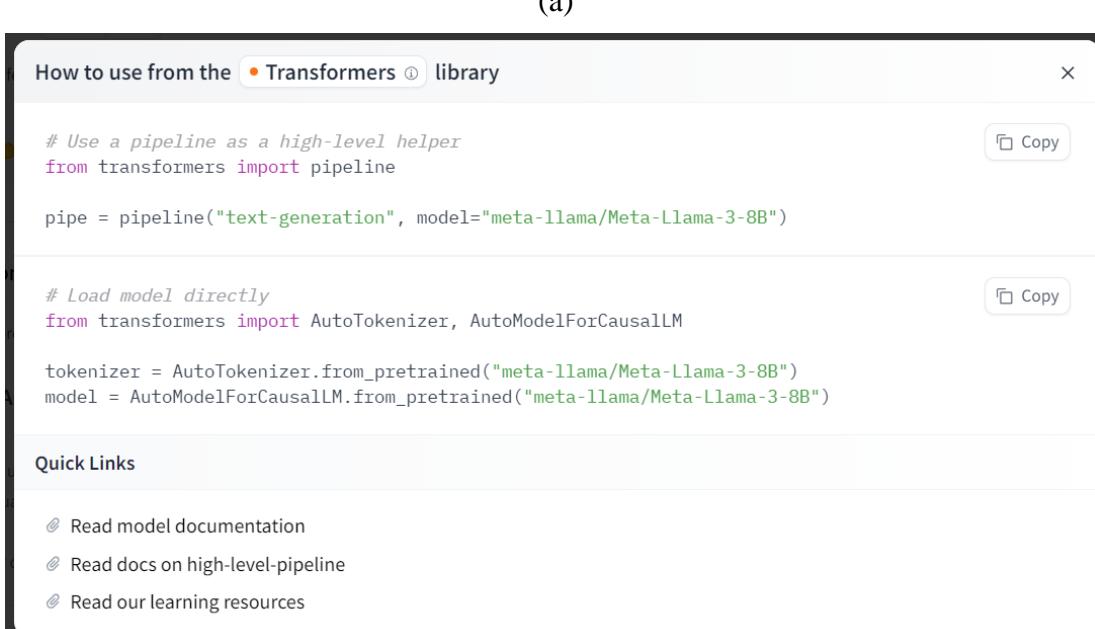


Şekil 66. Hugging Face Models arayüzü.

Örneğin **meta-llama/Meta-Llama-3-8B** modeli seçildiğinde bu modele dair kullanım istatistikleri, tartışma postları ve farklı dosya / dokümanların görüntülendiği arayüze ulaşılmaktadır. Sağ üst alanda yer alan “**Use this model**” düğmesi ile modeli transformer kütüphanesi ile Python ortamına dahil edecek kodlara ulaşılabilir almaktadır (Şekil 67). Öğrencilerin ilgili adımları izleyerek Python kodlarını gözlemlemeleri sağlanır.



(a)

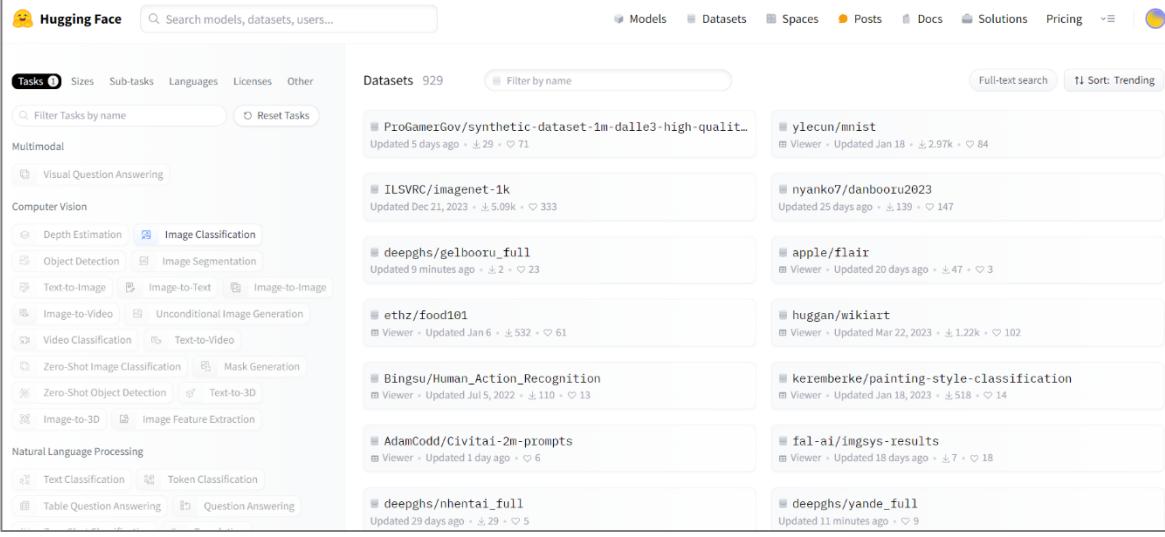


(b)

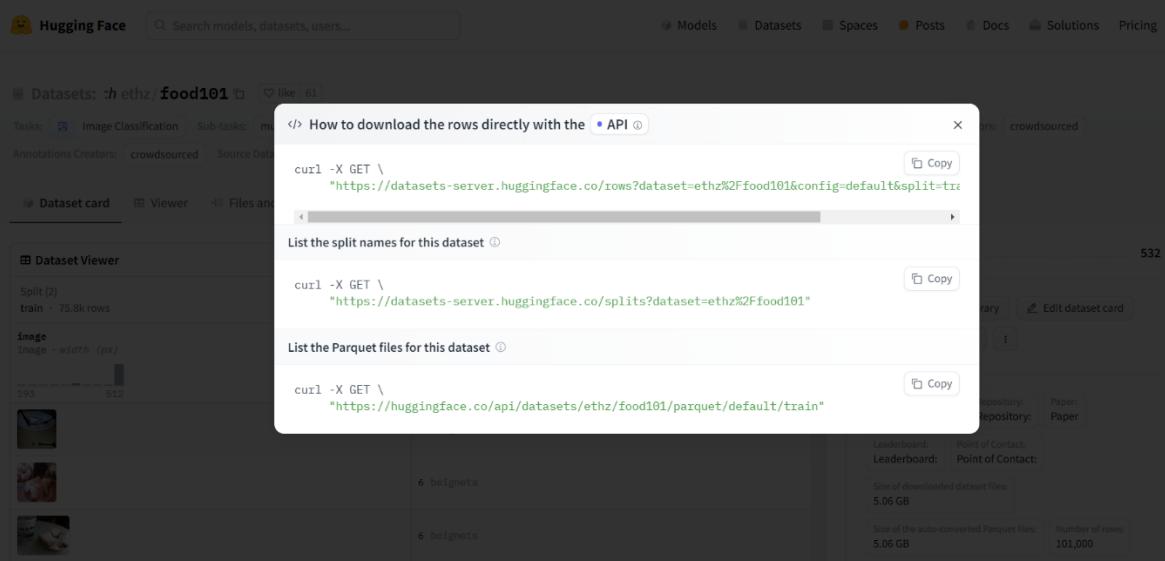
Şekil 67. (a) Seçili bir modele ait detay arayüzü (b) Modelin Python ortamına dahil edilmesine yönelik Python kodları.

Hugging Face platformunda veri setlerinin kullanımı da model kullanımına benzer arayüzler üzerinden gerçekleştiriliyor. Öğrencilerin bu defa Datasets düğmesine tıklamaları, ardından yeni arayüzde sol alanda Image Classification filtresini uygulamaları ve sağ alanda listelenen

görüntü veri setlerinden herhangi birini seçmeleri istenir. Seçilen veri setiyle ilgili olarak gelen detay arayüzünde veri setiyle ilgili bilgilere, tartışma postlarına ulaşıldığı gibi, veri setinin indirilmesi ya da API üzerinden uygulamalara dahil edilmesine yönelik seçenekler de görüntülenebilmektedir (Şekil 68).



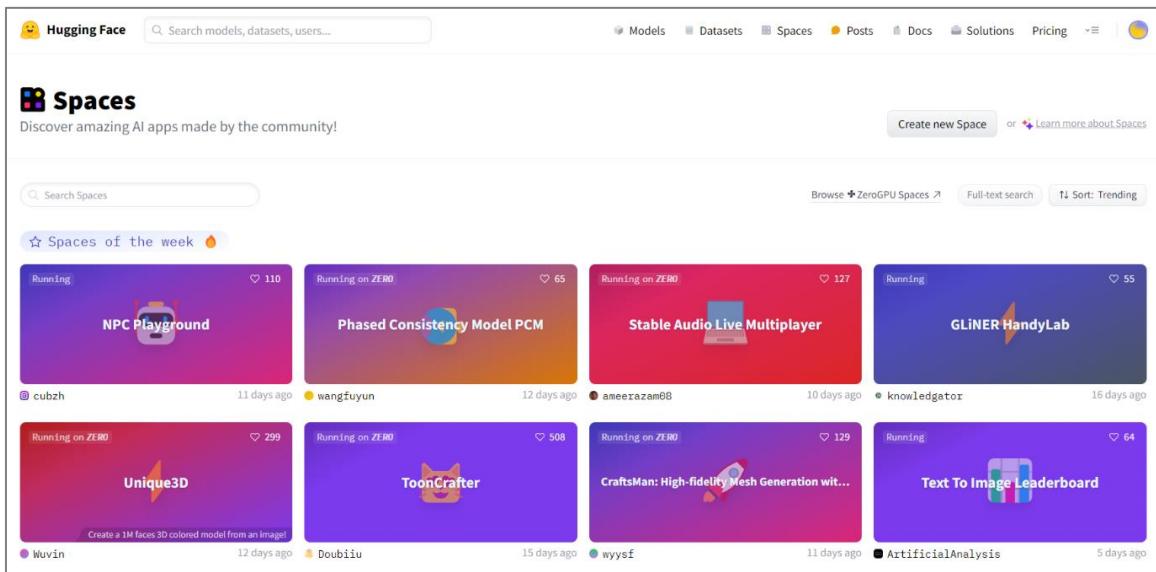
(a)



(b)

Şekil 68. (a) Veri setleri arayüzü (b) Seçili bir veri setine ait API kullanım bilgileri ve arkaplanda veri seti detay arayüzü.

Hugging Face platformundaki kritik arayızlarından biri de Spaces arayüzüdür. Bu arayüz altında kullanıcılar geliştirdikleri uygulamaları demo olarak kullanıma sunmaktadır (Şekil 69). Arayüz altında popüler demolara ulaşılabilir gibi kullanıcılar Create new space düğmesi üzerinden tipki model ekler gibi kendi demolarını platforma dahil edebilmektedir. Spaces arayüzüne deneyimlemeleri için öğrencilerin Spaces düğmesine tıklamaları ve açılan arayüzden herkesin tercih edilen bir demoyu açmaları istenir.



Şekil 69. Hugging Face Spaces arayüzü.

Örneğin, **Phased Consistency Model PCM** olarak adlandırılan bir demo seçildiğinde geliştiricinin hazırlamış olduğu arayüze ulaşılmaktadır (Söz konusu uygulama metin / prompt üzerinden image oluşturma üzerine geliştirilen alternatif bir Yapay Zeka modelidir). Buradaki arayüzde prompt alanında “**robots with humans in Türkiye streets**” metni girildiğinde Şekil 70’de gösterildiği gibi bir görsel üretilmektedir. Öğrencilerden farklı promptları denemeleri ve elde ettikleri görüntüleri yorumlamaları istenir. Bu aktivite için 5 dakikalık bir süre ayrılr.

The screenshot shows the "Phased Consistency Model" app page. At the top, it says "stency-Model-PCM" and "Running on ZERO". There's a "App" button. The main title is "Phased Consistency Model". Below the title, there's a brief description: "Phased Consistency Model (PCM) is an image generation technique that addresses the limitations of the Latent Consistency Model (LCM) in high-resolution and text-conditioned image generation. PCM outperforms LCM across various generation settings and achieves state-of-the-art results in both image and video generation." Below the description are links to "[paper]", "[arXiv]", "[code]", and "[project page]". The main interface has a "Prompt" input field, a "Select inference steps" dropdown set to "4-Step", a "Number of Inference Steps" slider set to 4, and two buttons: "Run on SDXL" and "Run on SD15". Below this is a large input field labeled "PCM Image" with a placeholder "PCM Image" and a "x" button to clear it. At the bottom, there's a "Examples" section with a table:

Prompt	Select inference steps	Number of Inference Steps
astronaut walking on the moon	4-Step	4
Photo of a dramatic cliffside lighthouse in a storm, waves	8-Step	8

(a)



(b)

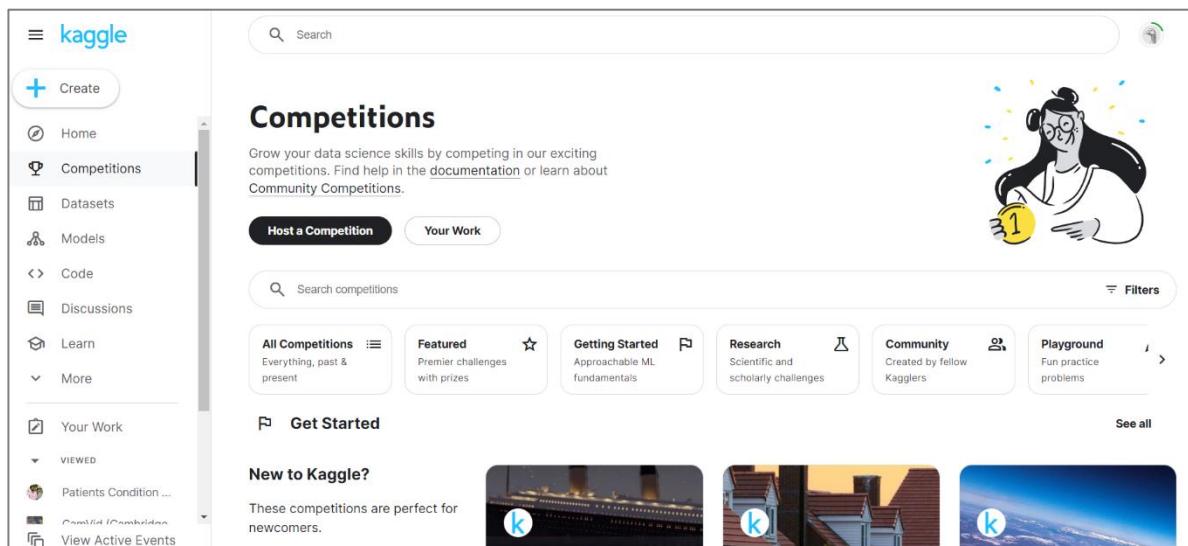
Şekil 70. (a) Seçilen bir demoya (Phased Consistency Model PCM) ait arayüz
 (b) “robots with humans in Türkiye streets” metni için elde edilen bir görsel.

Kaggle Platformunda Araçlar / Arayüzler

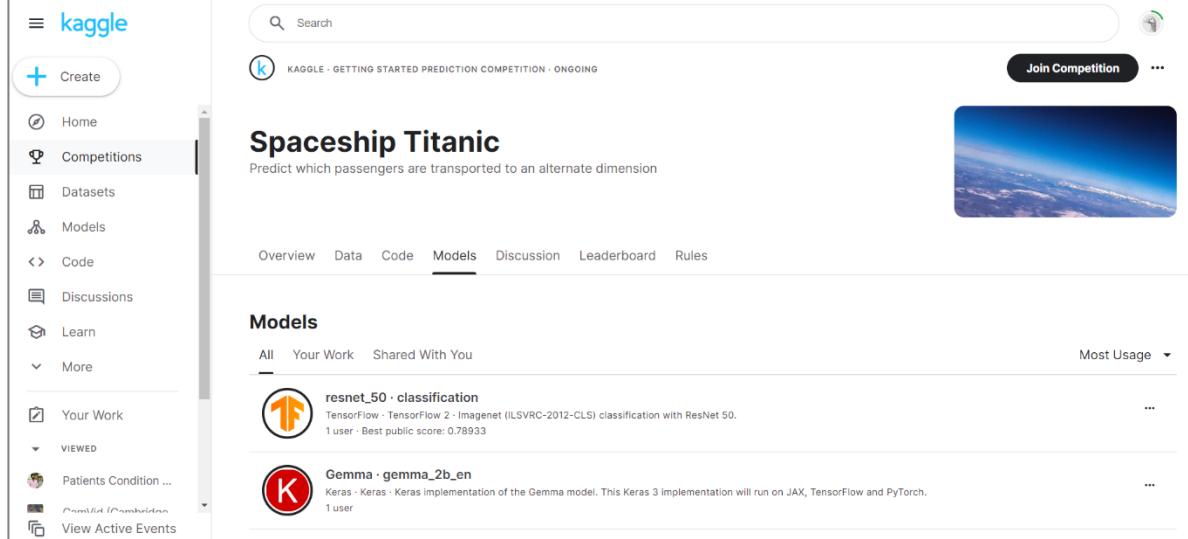
Kaggle ortamına giriş yapıldığında kullanıcıların gerçekleştirdikleri aktivitelere dair istatistiklerin ve platformdaki aktif aktivitelere dair bilgilerin sunulduğu giriş arayüzüne ulaşılmaktadır. Aynı arayüzün sol tarafında yer alan navigation paneli üzerinden yarışmalar, veri setleri, modeller, kodlar ve diğer platform araçlarına ulaşımı sağlayan seçenekler listelenmiştir (Şekil 71).

Şekil 71. Kaggles kullanıcı arayüzü.

Öğrencilerin sol panelden Competitions düğmesine tıklayarak yarışma arayüzüne ulaşmaları istenir. Bu arayüzde kullanıcılar yeni bir yarışma açabildiği gibi mevcut olanlara da dahil olabilmektedir. Herhangi bir yarışma seçildiğinde ilgili yarışmaya dair detay açıklamalar, kurallar, liderlik sıralaması, veri setleri, o ana kadar kullanılan modeller, kodlar ve tartışma postlarına ulaşılmakta, yarışmaya “Join Competition” düğmesi ile dahil olunabilmektedir (Şekil 72).



(a) Screenshot of the Kaggle homepage showing the 'Competitions' section. It features a search bar, a sidebar with navigation links like Home, Competitions, Datasets, Models, etc., and a main area with sections for 'All Competitions', 'Featured', 'Getting Started', 'Research', 'Community', and 'Playground'. A cartoon character is shown holding a large coin.

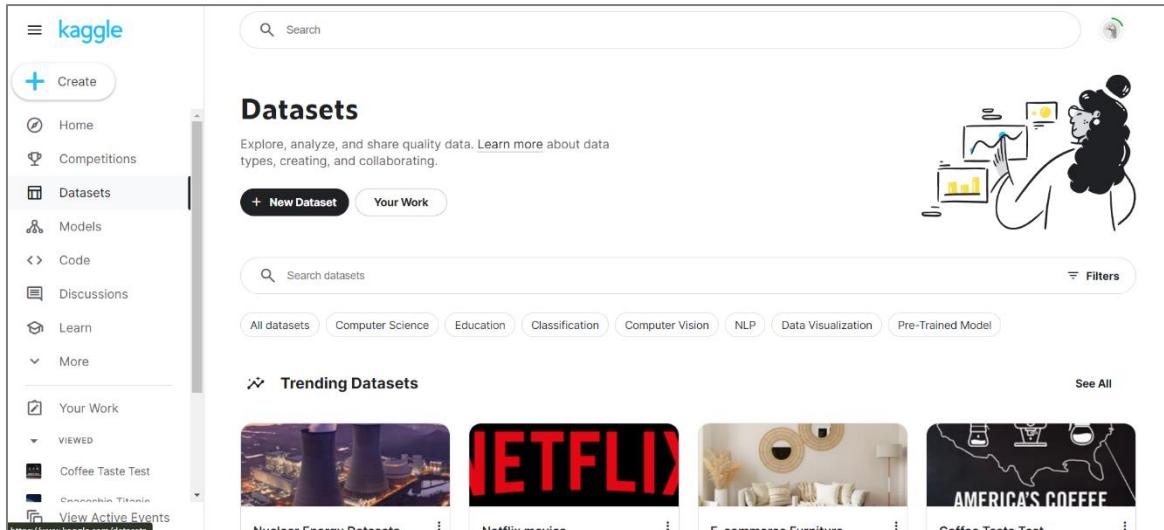


(b) Screenshot of the Kaggle 'Spaceship Titanic' competition page. It shows the competition title, a brief description, and tabs for Overview, Data, Code, Models, Discussion, Leaderboard, and Rules. The 'Models' tab is active, displaying two entries: 'resnet_50 · classification' and 'Gemma · gemma_2b_en'. Each entry includes a thumbnail icon, the model name, description, and usage statistics.

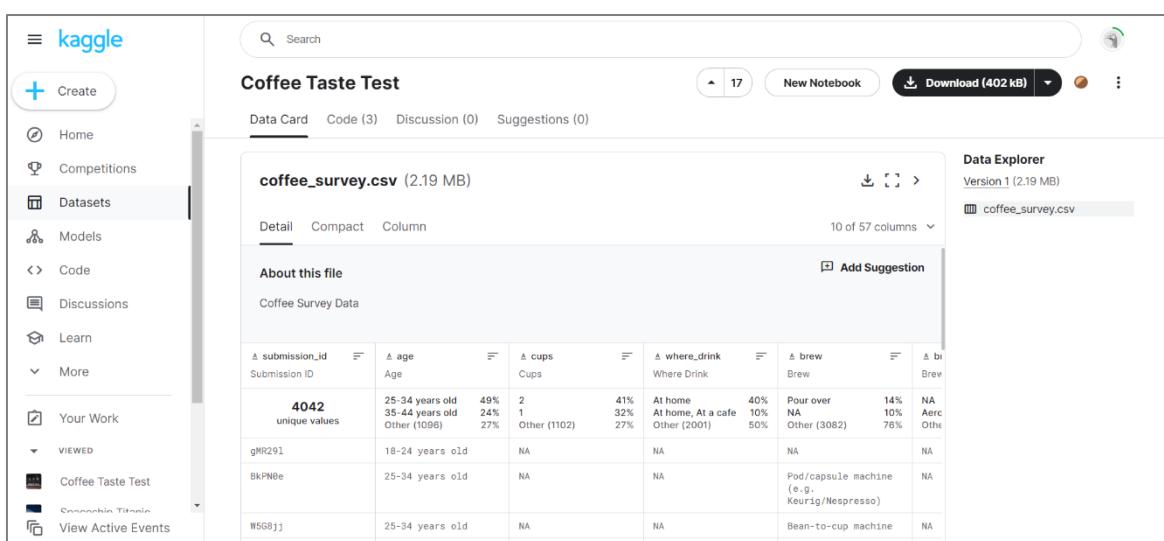
Şekil 72. (a) Kaggle competitions (yarışmalar) arayüzü (b) Seçilen bir yarışmaya dair detay arayüzü.

Hugging Face platformunda olduğu gibi, Kaggle platformunda yer alan farklı arayızler kullanım kolaylığı adına birbirine benzer şekilde organize edilmiş durumdadır. Datasets seçeneği tıklandığında ekrana gelen arayüzde bu defa platformda paylaşılan veri setleri listelenmektedir. Kullanıcılar aynı arayüzde kendi veri setlerini Kaggle ortamıyla

paylaşabilmektedir. Herhangi bir veri seti seçildiğinde, ilgili veri setinin içeriğine dair detaylar, veri setiyle ilgili paylaşılan kodlar ve tartışma postları görüntülenebilmekte, veri seti dosyası indirilebilmektedir (Şekil 73). Öğrencilerin Datasets düğmesine tıklamak suretiyle söz konusu arayüzleri ve veri setlerini incelemeleri sağlanır.



(a)



(b)

Şekil 73. (a) Kaggle datasets (veri setleri) arayüzü (b) Seçilen bir veri setine dair detay arayüzü.

Öğrencilerden Models düğmesine tıklayarak listelenen modelleri görüntülemeleri ve herhangi bir modeli seçerek detay arayüze ulaşmaları sağlanır. Modellere ilişkin detay arayızlar ilgili modellere yönelik açıklamalara, modeller üzerinden yazılmış kodlara ve tartışma postlarına yer vermektedir. Model arayızları özellikle modellerin kullanımına dair örnek Python kodlarını sunması açısından oldukça faydalıdır (Şekil 74). Öğrencilerin söz konusu arayızları ve örnek Python kodları incelemeleri için 5-10 dakikalık bir süre verilir.

Gemma

Model Card Code (579) Discussion (59) Competitions

Example Use

```
!pip install -U keras-nlp
!pip install -U keras
```

```
import keras
import keras_nlp
import numpy as np
```

Use `generate()` to do text generation.

```
gemma_lm = keras_nlp.models.GemmaCausalLM.from_preset("gemma_1.1_instruct_2b_en")
gemma_lm.generate("Keras is a", max_length=30)
```

```
# Generate with batched prompts.
gemma_lm.generate(["Keras is a", "I want to say"], max_length=30)
```

Compile the `generate()` function with a custom sampler.

Şekil 74. Kaggle ortamında seçilen bir modelin Python kod örnekleri.

Kaggle platformunun Codes (kodlar) bölümü kullanıcılar tarafından geliştirilmiş uygulamaların paylaşıldığı ve detay arayüzler içerisindeki uygulamalara dair Python kodlarına, veri setlerine, tartışma postlarına ulaşımının sağlandığı araç konumundadır. Kullanıcılar aynı aracı kullanarak kendi Python kodlarını (Notebook düzenmelerini) ve ilgili bileşenleri bu araç / arayüz üzerinden Kaggle kullanıcıları ile paylaşabilmektedir. Paylaşılan kodlara dair detay arayüzler ziyaret edilerek, uygulama kodlarının indirilip denenmesi ve geliştirilmesi oldukça kolaydır. Kodlar genellikle İçindekiler (Table of Contents) seçenekleri altında adım adım paylaşılması dolayısıyla uygulama geliştirme adımlarının takibi de gözlemlenebilmektedir (Şekil 75). Öğrencilerin bu noktada söz konusu seçenekler altından Kaggle ortamında paylaşmış kodlar üzerinde incelemelerde bulunması istenir (Daha önce üzerinde denemeler yapılmış ve kodlar, veri setleri...vs. bütünü içerisinde bilgisayar ortamına aktararak kolay bir şekilde çalıştırılan bir örnek kod / uygulama öğrencilere gösterilebilir).

Code

Explore and run machine learning code with Kaggle Notebooks.
Find help in the Documentation.

+ New Notebook Your work

Search public notebooks

All notebooks Recently Viewed Python R Beginner NLP Random Forest GPU TPU Competition notebook Scheduled notebook

Trending

See all (394)

Sign Language MNIST House Prices - Advanced Regression Techniques Nuclear Energy Datasets tsemodule7

(a)

```

In [1]:
# Data handling and visualization
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# preprocessing and data transformation
from sklearn.preprocessing import OneHotEncoder
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import MinMaxScaler
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline, make_pipeline

# Model selection and evaluation
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score

```

(b)

Şekil 75. (a) Kaggle codes (kodlar / uygulamalar) arayüzü (b) Seçilen bir uygulamanın Python kodlarına dair detay arayüzü.

Öğrencilere sol alandaki navigasyon panelinde yer alan Discussions düğmesi üzerinden tartışma postalarına doğrudan ulaşabilecekleri, yine Learn düğmesi ile eğitim amaçlı uygulamalara / anlatımlara ulaşabilecekleri ve yine alternatif seçeneklerle Kaggle ortamındaki kullanıcı ya da takım sıralamaları, ek dokümanlar ve araçlara da ulaşabilecekleri açıklanır.



4. ADIM: İLERLET

Öğrencilerin Hugging Face ve Kaggle ortamlarını genel hatlarıyla deneyimlemeleri ve elde ettikleri bilgileri / deneyimleri not almaları istenir. Söz konusu süreç için 20 dakikalık bir süre ayrılr.



5. ADIM: DEĞERLENDİRME

Öğrencilerin bir önceki adımda aldığı notlar ve derste öğrenilenler ışığında soru-cevap etkinliği gerçekleştirilerek Hugging Face ve Kaggle platformlarına dair bilgiler pekiştirilir.

Öğrencilere şu sorular sorulabilir:

- Hugging Face platformu hakkında neler düşünüyorsunuz?
- Kaggle platformu hakkında neler düşünüyorsunuz?
- Hugging Face ve Kaggle platformlarında yer alan hangi araçlar / arayüzler ilginizi daha çok çekti?

- Hugging Face ve Kaggle platformları Yapay Zeka uygulamaları geliştirmede ne gibi avantajlar sağlayabilir?
- Hugging Face ve Kaggle platformlarını Yapay Zeka ve Veri Bilimi’ne ilişkin detayları öğrenmek ve bildiklerinizi ilerletmek konusunda ne kadar sıklıkla kullanmak istersiniz?

1. HAFTA – 4. GÜN – 2., 3. DERSLER: PYTHON ORTAMINDA VERİ SETİ OKUMA, MODEL KURMA VE EĞİTİM-TEST İŞLEMLERİ

DERS PLANI

DERS ETİKETLERİ



KONU: Python Ortamında Veri Seti Okuma, Model Kurma ve Eğitim-Test İşlemleri



SINIF: 12-14 Yaş



SÜRE: 120 dk.



HAZIR BULUNUŞLUK: İngilizce eğitimi, Yapay Zeka kavramı ve uygulamaları hakkında temel bilgi, Makine Öğrenmesi hakkında temel bilgi, Python programlama dili ve onde gelen kütüphaneler hakkında temel bilgi



ANAHTAR KELİMELER: Yapay Zeka, Algoritma, Kodlama, Python, Veri seti, Sklearn



BECERİLER:



Tahmin



Sorgulama



Grup çalışması



Günlük hayat bağlantısı kurma



KAZANIMLAR:

Bu derste öğrenciler;

- Python Scikit-learn (sklearn) kütüphanesi hakkında genel bilgi sahibi olur.
- Python ortamında veri seti okuma, model kurma ve eğitim-test süreçlerine dair genel bilgi ve beceri sahibi olur.
- Yapay Zeka uygulamaları geliştirmede izlenen temel adımlara dair farkındalık edinir.



DERS MATERİYALLERİ:

Malzeme	Miktar
Yazı Tahtası	Öğrenci sayısı ve sınıf büyüklüğüne göre en az bir adet
Projeksiyon	Öğrenci sayısı ve sınıf büyüklüğüne göre en az bir adet
Defter, Kalem	Öğrenci sayısı kadar
Bilgisayar	Öğrenci sayısının yarısı kadar
Tablet	Öğrenci sayısının yarısı kadar
Python Anaconda Framework	Her bilgisayarda kurulu olacak şekilde



ÖĞRETMEN HAZIRLIK ÇALIŞMALARI:

Okunabilecek Kaynaklar:

Geeks for Geeks. (2024). Learning Model Building in Scikit-learn. Çevrimiçi: <https://www.geeksforgeeks.org/learning-model-building-scikit-learn-python-machine-learning-library/> (Erişim 12 Haziran 2024).

Pykes, K. (2023). Python Machine Learning: Scikit-learn Tutorial. Datacamp.com. Çevrimiçi: <https://www.datacamp.com/tutorial/machine-learning-python> (Erişim 12 Haziran 2024).

İzlenebilecek Kaynaklar:

Brain-tr. (2023). Makine Öğrenmesi KNN Algoritması Adım Adım Örnek Uygulama. Çevrimiçi: <https://www.youtube.com/watch?v=gxBhnRTGdZU> (Erişim 12 Haziran 2024).

Data Professor. (2022). Build your first machine learning model in Python. Çevrimiçi: <https://www.youtube.com/watch?v=29ZQ3TDGgRQ> (Erişim 13 Haziran 2024).

ATÖLYE HAZIRLIK ÇALIŞMALARI:

Öğretmen;

- Dersten önce ders materyallerini hazırlamalıdır.
- Python kodlama için kullanılan Spyder editörünün genel kullanımına aşina olmalıdır.
- Derse konu olan Scikit-learn (sklearn) kütüphanesini ve Python kodlarını önceden inceleyerek üzerinde çalışmış olmalıdır.



KAYNAKÇA:

Agarwal, A., & Saxena, A. (2018). Malignant tumor detection using machine learning through scikit-learn. *International Journal of Pure and Applied Mathematics*, 119(15), 2863-2874.

Das, S., & Cakmak, U. M. (2018). *Hands-On Automated Machine Learning: A beginner's guide to building automated machine learning systems using AutoML and Python*. Packt Publishing Ltd.

Deperlioğlu, Ö., & Köse, U. (2024). *Python ile Makine Öğrenmesi: Temel Kavramlar – Sınıflandırma – Regresyon – Kümeleme*. Seçkin Yayıncılık.

Hackeling, G. (2017). *Mastering Machine Learning with scikit-learn*. Packt Publishing Ltd.

GENEL BAKIŞ:

1) Harekete Geç: Öğrencilere önceki haftalardan bu yana öğrendiklerinden hareketle bir Yapay Zeka uygulaması geliştirme sürecinde genel hatlarıyla ne tür işlemler yapmaları gereği konusunda fikirleri sorulur ve elde edilen cevaplar tartışılır.

2) Keşfet: Öğrencilere Scikit-learn (sklearn) kütüphanesi hakkında genel bilgi verilir.

3) Üret: Öğrencilerin Scikit-learn (sklearn) kütüphanesinin de desteğiyle veri seti okuma, model oluşturma ve eğitim-test adımlarına yönelik basit bir uygulama üzerinde ön bilgiler-beceriler edinmeleri sağlanır.

4) İlerlet: Öğrencilerin gruplar halinde çalışarak, gerçekleştirilen uygulama üzerinde kendi düzenlemelerini yapmaları ve elde edilen sonuçlar hakkında notlar almaları istenir.

5) Değerlendir: Derste edinilen bilgiler soru-cevap aşaması gerçekleştirilmek suretiyle pekiştirilir.

UYGULAMA



1. ADIM: HAREKETE GEÇ

Öğrencilere önceki haftalardan bu yana öğrendiklerinden hareketle bir Yapay Zeka uygulaması geliştirme sürecinde genel hatlarıyla ne tür işlemler yapmaları gereği konusunda fikirleri sorulur ve elde edilen cevaplar tartışılır. İlgili süreç için 10 dakikalık bir zaman ayırılır.

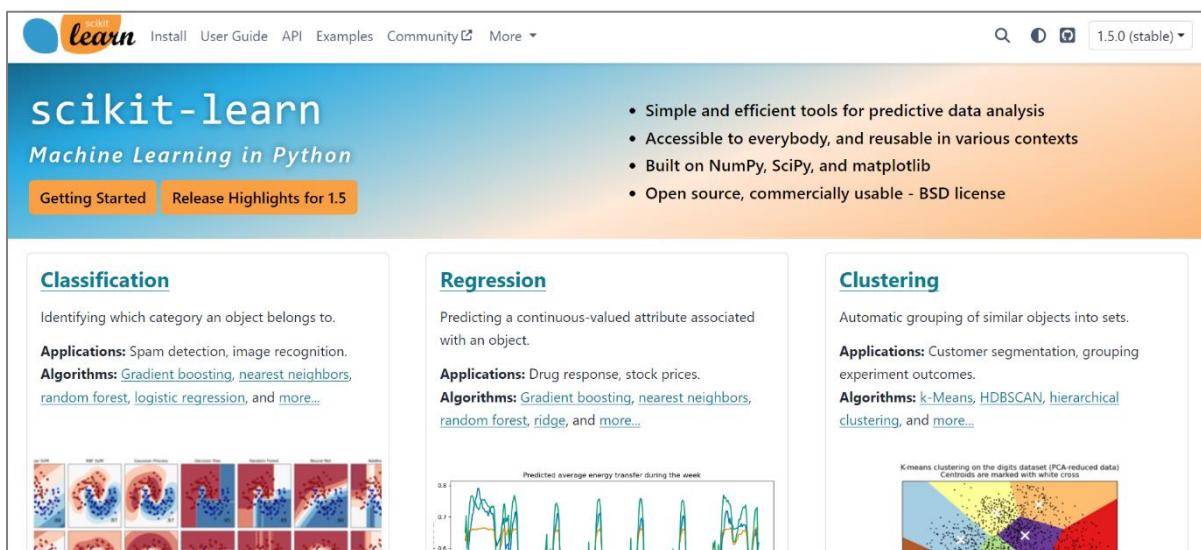


2. ADIM: KEŞFET

Öğrencilere Scikit-learn (sklearn) kütüphanesi hakkında genel bilgi verilir.

Scikit-learn (sklearn) Kütüphanesi

Python programlama dili ile Yapay Zeka uygulamaları geliştirmek için kullanılan farklı kütüphaneler bulunmaktadır. Bu kütüphaneler arasında Makine Öğrenmesi temellerinden başlanarak ileri düzey Derin Öğrenme uygulamalarını da kapsam içerisinde alacak geliştirmeler yapmak için Scikit-learn kullanılabilmektedir. Python kodlama ortamında kullanılan **sklearn** komutu üzerinden de isimlendirilen Scikit-learn kütüphanesi kod yazım kolaylığı ve kapsamı dolayısıyla özellikle Makine Öğrenmesi uygulamaları geliştirme konusunda oldukça idealdir (Das, & Cakmak, 2018; Deperlioğlu, & Köse, 2023). Kütüphaneye ilişkin dokümantasyonlar, örnek uygulamalar, haberler ve topluluk araçlarına <https://scikit-learn.org/> adresi üzerinden ulaşılabilmektedir (Şekil 76).



Şekil 76. Scikit-learn (sklearn) Web sitesi.

Scikit-learn kütüphanesi içerisinde hazır Yapay Zeka modelleri, hazır veri setleri ve eğitim-test fonksiyonları yer almaktadır. Farklı bileşenlere ulaşmak için **from sklearn** komutuna kullanılmak istenen bileşen yazılarak Python ortamına dahil edilmektedir. Kütüphanenin Python ile Makine Öğrenmesi kodlamayı öğrenenler için **toy datasets** (oyuncak veri setleri) adı altında sunduğu örnek veri setleri işlem adımlarını anlamak için idealdir. Uygun kod yazımları ile veri setleri ayrıca dosya okuma işlemine ihtiyaç duyulmadan uygulama ortamına doğrudan dahil edilmektedir. Detaylı bir şekilde https://scikit-learn.org/stable/datasets/toy_dataset.html adresinden incelenebilecek veri setleri Haziran 2024 itibariyle sınıflandırma ve regresyon problemlerini kapsamaktadır (Şekil 77).

The screenshot shows the scikit-learn User Guide page for '7.1. Toy datasets'. The left sidebar has a tree structure with sections like 'Supervised learning', 'Unsupervised learning', etc., and '7.1. Toy datasets' is selected. The main content area shows the title '7.1. Toy datasets' and a paragraph about the datasets. Below it is a table listing six datasets with their corresponding load functions and descriptions:

Function	Description
<code>load_iris (*[, return_X_y, as_frame])</code>	Load and return the iris dataset (classification).
<code>load_diabetes (*[, return_X_y, as_frame, scaled])</code>	Load and return the diabetes dataset (regression).
<code>load_digits (*[, n_class, return_X_y, as_frame])</code>	Load and return the digits dataset (classification).
<code>load_linnerud (*[, return_X_y, as_frame])</code>	Load and return the physical exercise Linnerud dataset.
<code>load_wine (*[, return_X_y, as_frame])</code>	Load and return the wine dataset (classification).
<code>load_breast_cancer (*[, return_X_y, as_frame])</code>	Load and return the breast cancer wisconsin (diagnostic) dataset.

Şekil 77. Scikit-learn (sklearn) Web sitesinde toy datasets (oyuncak veri setleri).



3. ADIM: ÜRET

Öğrencilerin Scikit-learn (sklearn) kütüphanesinin de desteğiyle veri seti okuma, model oluşturma ve eğitim-test adımlarına yönelik basit bir uygulama üzerinde ön bilgiler-beceriler edinmeleri sağlanır.

Veri Seti Okuma

Öğrencilere daha önceki derslerde öğrendikleri Pandas dosya okuma işlemleri aracılığıyla (Bkz. 1. Hafta – 3. Gün – 1 Ders) Excel dosyaları (csv, xls, xlsx) üzerinden ya da farklı dosyalardan veri seti okuma işlemleri gerçekleştirilebilir. Bu doğrultuda örnek olmasından öğrencilerin daha önceki indirdikleri (Bkz. 1. Hafta – 3. Gün – 2. Ders) iris veri seti için Şekil 78’de gösterilen kodlar yazılarak veri seti okumaya ve girdi (X) ile çıktı (Y) değişkenleri içerisinde ayırmaya dair kodlar ve Variable Explorer’daki çıktılar incelenir (Veri setinin tekrar indirilmesi gereklirse Web adresi <https://www.kaggle.com/datasets/uciml/iris> şeklindeki (Ders akışlarının negatif etkilenmemesi adına aynı veri seti üzerinden ilerleme tercih edilmiştir. Tercih edilmesi halinde, sklearn içerisinde de çağrılabilen diğer veri setlerinden herhangi birinin dosya versiyonu temin edilerek uygulama gerçekleştirilebilir).

```

8 import pandas as pd
9
10 iris = pd.read_csv("Iris.csv")
11
12 X = iris.iloc[:,0:4]
13 Y = iris.iloc[:,5]

```

(a)

iris - DataFrame

Index	Id	palLengthC	spalWidthC	stallLengthC	etalWidthC	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5	3.6	1.4	0.2	Iris-setosa
5	6	5.4	3.9	1.7	0.4	Iris-setosa
6	7	4.6	3.4	1.4	0.3	Iris-setosa
7	8	5	3.4	1.5	0.2	Iris-setosa
8	9	4.4	2.9	1.4	0.2	Iris-setosa
9	10	4.9	3.1	1.5	0.1	Iris-setosa
10	11	5.4	3.7	1.5	0.2	Iris-setosa

Format Resize Background color Column min/max Save and Close Close

(b)

X - DataFrame

Index	Id	palLengthC	spalWidthC	stallLengthC
0	1	5.1	3.5	1.4
1	2	4.9	3	1.4
2	3	4.7	3.2	1.3
3	4	4.6	3.1	1.5
4	5	5	3.6	1.4
5	6	5.4	3.9	1.7
6	7	4.6	3.4	1.4
7	8	5	3.4	1.5
8	9	4.4	2.9	1.4
9	10	4.9	3.1	1.5
10	11	5.4	3.7	1.5

Format Resize Background color Column min/max Save and Close Close

Y - Series

Index	Species
0	Iris-setosa
1	Iris-setosa
2	Iris-setosa
3	Iris-setosa
4	Iris-setosa
5	Iris-setosa
6	Iris-setosa
7	Iris-setosa
8	Iris-setosa
9	Iris-setosa
10	Iris-setosa

Format Resize Background color Column min/max Save and Close Close

(c)

Şekil 78. (a) Iris.csv dosyasının Pandas ile okunması (b) okunan dataframe (c) girdi (X) ve çıktı (Y) değişkenlerine ayrılmış veriler.

Öğrencilere aynı veri setinin sklearn kapsamındaki toy_datasets altından da çağrılabileceği anlatılarak mevcut kodları yorum satırları kapsamına alması ve iris veri seti okuma için aşağıdaki kodları yazmaları istenir:

```
from sklearn.datasets import load_iris  
iris = load_iris()  
X = iris.data  
Y = iris.target
```

Bu aşamada daha önce de öğrenciler tarafından kullanılan iris veri seti genel hatlarıyla kısaca hatırlanır veri setinin daha detaylı incelemesinin bir sonraki gün gerçekleştirilecek uygulamada (1. Hafta – 5. Gün) yapılacağı vurgulanır.

Veri seti okuma işlemleri ardından tipik bir Makine Öğrenmesi uygulaması için gerçekleştirilmesi gereken bir sonraki adım veri setinin eğitim ve test setlerine ayrılmasıdır. Bu işlem manuel bir biçimde yapılabildiği gibi sklearn kütüphanesinin sunduğu “**train_test_split**” fonksiyonu yardımıyla otomatikleştirilebilir. Bu fonksiyon her çalıştırıldığında veriler rastgele bir biçimde, verilen parametre değerlerine göre ayrılmaktadır (Agarwal, & Saxena, 2018; Hackeling, 2017).

Öğrenciler Şekil 79’daki kodları yazmak suretiyle verileri %70 eğitim, %30 test setleri halinde ayırır ve elde edilen sonuçları gözlemler. **X_train** değişkeni eğitimde kullanılacak girdi verilerini, **Y_train** bu verilere karşılık gelen çıktıları temsil ederken, **X_test** test aşamasında kullanılacak girdilere ve **Y_test** bu verilerin çıktılarına karşılık gelmektedir.

```
22 from sklearn.model_selection import train_test_split  
23 X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.30)  
24 print("Eğitim veri seti boyutu=",len(X_train))  
25 print("Test veri seti boyutu=",len(X_test))
```

(a)

```
Eğitim veri seti boyutu= 105  
Test veri seti boyutu= 45
```

(b)

Şekil 79. (a) Veri setlerinin eğitim ve test setlerine ayrılmasına dair kodlar
(b) Konsol ortamında yansiyan çıktılar.

Model Kurma

Öğrencilere bir sonraki aşamanın Makine Öğrenmesi için tercih edilen bir algoritma ile model kurma olduğu anlatılır. Bu noktada farklı algoritmalar için model kurma kodlarının farklılık gösterebileceği açıklanır ve mevcut uygulamada Makine Öğrenmesi literatüründe basit yapısı ancak sınıflandırma problemlerindeki etkin sonuçlarıyla bilinen **kNN** (k Nearest Neighbor / k En Yakın Komşu) algoritmasının kullanılacağı ifade edilir.

kNN Algoritması

kNN algoritması, sonucu (çıktısı) tahmin edilmek istenen girdi parametreleri ile eldeki veri setinde yer alan örnek verilerin her biri için girdi parametreleri arasındaki sayısal uzaklığı Öklid, Minkowski ya da Manhattan gibi metriklerle ölçen bir algoritmadır. Ölçüm sonucunda uzaklık değeri en küçük çıkan k adet örnek tespit edilir ve bu örneklerde en çok görülen sınıf (etiket) tahmin sonucu olarak kabul edilir (Hackeling, 2017). Gerek duyulması halinde öğrencilerin <https://www.geeksforgeeks.org/k-nearest-neighbours/> adresini ziyaret ederek algoritmaya dair bilgilerini pekiştirmeleri sağlanır.

Öğrencilerin Şekil 80’de gösterilen kodları yazmak suretiyle sklearn kütüphanesi ortamından kNN algoritmasını çağırırları sağlanır. Örnekte algoritma k değeri 7, uzaklık metriği ise hipotenüs kurallarından destek alarak ölçüm sağlayan ve sıkılıkla tercih edilen Öklid olarak belirlenmiştir.

```
27 from sklearn.neighbors import KNeighborsClassifier  
28 knn = KNeighborsClassifier(n_neighbors=7, metric='euclidean')
```

Şekil 80. kNN algoritmasının Python ortamında çağırılması.

Model Eğitimi ve Test / Değerlendirme İşlemleri

Python Yapay Zeka kütüphanelerinde kullanılan modellerin eğitim aşamalarında genellikle “fit” fonksiyonu kullanılmaktadır. Söz konusu fonksiyon parametre olarak eğitime tabi girdi verileri ve bu verilere karşılık gelen çıktı verilerini kullanmaktadır. Eğitimi sağlanan bir modelin tahminde bulunarak test edilmesi ise “predict” fonksiyonu ile gerçekleştirilmektedir. Bu fonksiyon ise sadece tahmine tabi tutularak test girdi verilerini parametre olarak kullanmakta, elde edilen tahminler daha sonra farklı metriklerle değerlendirme bulguları elde etme yönünde kullanılmaktadır.

Öğrenciler ilgili fonksiyonları kullanmak suretiyle daha önce ayrılmış olan eğitim veri seti üzerinden kNN algoritmasının eğitimi sağlar ve yine test veri seti üzerinden tahminlerde bulunur. Tahminler “Y_pred” adlı değişkene aktarılmıştır. Ayrıca fit fonksiyonunun gerekliliklerine uygun bir biçimde, Numpy kütüphanesi kullanılmak suretiyle “ravel” fonksiyonu üzerinden çıktı veri setinin 1 boyutlu dizi şeklinde kullanımı sağlanmıştır (Şekil 81).

```
30 import numpy as np  
31  
32 knn.fit(X_train, np.ravel(Y_train))  
33 Y_pred = knn.predict(X_test)  
34  
35 from sklearn import metrics
```

Şekil 81. kNN algoritmasının eğitim ve tahmin adımları.

Yapay Zeka uygulamalarında eğitim işlemi ardından tahminlerde bulunan bir Makine Öğrenmesi algoritması bu sayede bir bakıma teste tabi tutulmaktadır. Bu doğrultuda çeşitli

değerlendirme metrikleri ile elde edilen bulgular anlamlı değerlere dönüştürilmekte ve kurulan modellerin başarımları anlaşılmaktadır. Başarımı yeterli olmayan modeller farklı düzenlemelerle yeniden eğitim-test adımları içerisinde dahil edilebilmektedir. sklearn kütüphanesi değerlendirme adına çeşitli metrikleri bünyesinde barındırmaktadır. Buradaki uygulamada öğrencilerin Şekil 82'de gösterildiği gibi **from sklearn import metrics** çağrısını yaptıktan sonra “**accuracy_score**” ve “**confusion_matrix**” fonksiyonları ile sınıflandırma değerlendirmeleri yapmaları sağlanır.

```
35  from sklearn import metrics
36  print('Doğruluk (Accuracy): {}'.format(metrics.accuracy_score(Y_test, Y_pred)))
37  print(metrics.confusion_matrix(Y_test, Y_pred))
```

(a)

```
Eğitim veri seti boyutu= 105
Test veri seti boyutu= 45
Doğruluk (Accuracy): 0.9777777777777777
[[16  0  0]
 [ 0 11  1]
 [ 0  0 17]]
```

(b)

Şekil 82. (a) Tahmin süreçlerinin değerlendirilmesine dair kodlar
(b) Kodların çalıştırılması sonucu elde edilen konsol çıktıları.

Kodlarda 36. satırdaki `accuracy_score` fonksiyonu doğru tahmin sayısının toplam test veri sayısına bölünmesi suretiyle elde etmektedir. Kısaca doğruluk (accuracy) oranı olarak adlandırılan bu hesaplama, sınıflandırma problemlerinde ilk akla gelen metriktir. Uygulamada kNN algoritmasının %98 gibi oldukça yüksek bir başarı elde ettiği görülmektedir.

Yine 37. satırda yer alan kod, sınıflandırma problemlerinde sınıflar bazında alternatif değerlendirme bulgularının elde edilmesi için kullanılan hata matrisinin (karmaşıklık matrisi / confusion matrix) elde edilmesini sağlamaktadır. Konsola yansiyen bulgularda her matris satırı bir sınıfın tahmin bulgularını temsil etmektedir. Iris veri setinde yer alan 1. sınıf (16) ve 3. sınıf (17) tam doğruluk ile tespit edilmiştir. %2'lik hatalı tespit 2. sınıf olarak tespit edilmesi gereken bir örneğin 3. sınıf olarak tahmin edilmesiyle bağlantılıdır. Öğrencilere hata matrisi konusuna gelecek derslerdeki uygulamada daha detaylı değinileceği açıklanır.



4. ADIM: İLERLET

Öğrencilerin gruplar halinde çalışarak, gerçekleştirilen uygulama üzerinde kendi düzenlemelerini yapmaları ve elde edilen sonuçlar hakkında notlar almaları istenir. Söz konusu süreç için 10 dakikalık bir süre ayrılrı.



5. ADIM: DEĞERLENDİRME

Derste edinilen bilgiler soru-cevap aşaması gerçekleştirilmek suretiyle pekiştirilir.

Öğrencilere şu sorular sorulabilir:

- Gerçekleştirdiğiniz uygulama ile Yapay Zeka çözümleri oluşturma konusudan edindiğiniz bilgi ve beceriler nelerdir?
- Scikit-learn (sklearn) kütüphanesi hakkında neler düşünüyorsunuz?
- kNN algoritması hakkında neler düşünüyorsunuz?

UYGULAMANIN PYTHON KODLARI

```
import pandas as pd

# =====
# iris = pd.read_csv("Iris.csv")
#
# X = iris.iloc[:,0:4]
# Y = iris.iloc[:,5]
#
# =====

from sklearn.datasets import load_iris
iris = load_iris()
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 = 0.30)
print("Eğitim veri seti boyutu=",len(X_train))
print("Test veri seti boyutu=",len(X_test))

from sklearn.neighbors import KNeighborsClassifier
```

```
knn = KNeighborsClassifier(n_neighbors=7, metric='euclidean')

import numpy as np

knn.fit(X_train, np.ravel(Y_train))
Y_pred = knn.predict(X_test)

from sklearn import metrics
print('Doğruluk (Accuracy): {}'.format(metrics.accuracy_score(Y_test, Y_pred)))
print(metrics.confusion_matrix(Y_test, Y_pred))
```

1. HAFTA – 5. GÜN – 1., 2., 3. DERSLER: İRİS ÇİÇEĞİ VERİ SETİ İLE ÇİÇEK SINIFI TESPİT UYGULAMASI MODEL KURMA

DERS PLANI

DERS ETİKETLERİ



KONU: Iris Çiçeği Veri Seti ile Çiçek Sınıfı Tespit Uygulaması Model Kurma



SINIF: 12-14 Yaş



SÜRE: 180 dk.



HAZIR BULUNUŞLUK: İngilizce eğitimi, Yapay Zeka kavramı ve uygulamaları hakkında temel bilgi, Makine Öğrenmesi hakkında temel bilgi, Python programlama dili hakkında temel bilgi



ANAHTAR KELİMELER: Yapay Zeka, Algoritma, Kodlama, Python, Sklearn



BECERİLER:



Tahmin



Sorgulama



Grup çalışması



Günlük hayat bağlantısı kurma



KAZANIMLAR:

Bu derste öğrenciler;

- Python ile veri seti okuma, model kurma ve eğitim-test işlemlerinin gerçekleştirildiği genel bir Makine Öğrenmesi uygulamasını gerçekleştirirler.
- Python ortamında iris veri seti üzerinden sınıflandırmaya dayalı bir Makine Öğrenmesi çözümünün nasıl uygulanabileceği hakkında bilgi ve beceri sahibi olurlar.



DERS MATERİYALLERİ:

Malzeme	Miktar
Yazı Tahtası	Öğrenci sayısı ve sınıf büyüklüğüne göre en az bir adet
Projeksiyon	Öğrenci sayısı ve sınıf büyüklüğüne göre en az bir adet
Defter, Kalem	Öğrenci sayısı kadar
Bilgisayar	Öğrenci sayısının yarısı kadar
Tablet	Öğrenci sayısının yarısı kadar
Python Anaconda Framework	Her bilgisayarda kurulu olacak şekilde



ÖĞRETMEN HAZIRLIK ÇALIŞMALARI:

Okunabilecek Kaynaklar:

Kaggle. (2017). *Machine Learning with Iris Dataset*. Çevrimiçi: <https://www.kaggle.com/code/jchen2186/machine-learning-with-iris-dataset> (Erişim 13 Haziran 2024).

İzlenebilecek Kaynaklar:

Data School. (2015). Getting started in scikit-learn with the famous iris dataset. Çevrimiçi: <https://www.youtube.com/watch?v=hd1W4CyPX58> (Erişim 13 Haziran 2024).

ATÖLYE HAZIRLIK ÇALIŞMALARI:

Öğretmen;

- Dersten önce ders materyallerini hazırlamalıdır.
- Python kodlama için kullanılan Spyder editörünün genel kullanımına aşina olmalıdır.
- Derset konu olan veri set(ler)ini ve Python kodlarını önceden inceleyerek üzerinde çalışmış olmalıdır.



KAYNAKÇA:

Géron, A. (2022). *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow*. O'Reilly Media, Inc..

Köse, U., Özsoy, K., & Aksoy, B. (2023). *Yapay Zeka – Ortaokul ve Lise Kitapları*. Deneyap Teknoloji Atölyeleri, TÜBİTAK Yayıncılık.

GENEL BAKIŞ:

- 1) Harekete Geç:** Öğrencilerle birlikte uygulamaya konu veri setinin Python kodları yardımıyla analizi ve Makine Öğrenmesi süreci öncesi hazır hale getirilmesi sağlanır.
- 2) Keşfet:** Öğrencilerle birlikte ilgili Makine Öğrenmesi modelinin çalıştırılmasına dair kodlamaların yapılması sağlanır.
- 3) Üret:** Öğrencilerle birlikte çalıştırılan Makine Öğrenmesi modelinin elde ettiği sonuçların değerlendirilmesine yönelik kodlamaların yapılması sağlanır.
- 4) İlerlet:** Öğrencilerin gruplar halinde çalışarak, gerçekleştirilen uygulama üzerinde kendi düzenlemelerini yapmaları ve elde edilen sonuçlar hakkında notlar almaları istenir.
- 5) Değerlendir:** Derste edinilen bilgiler soru-cevap aşaması gerçekleştirilmek suretiyle pekiştirilir.

UYGULAMA



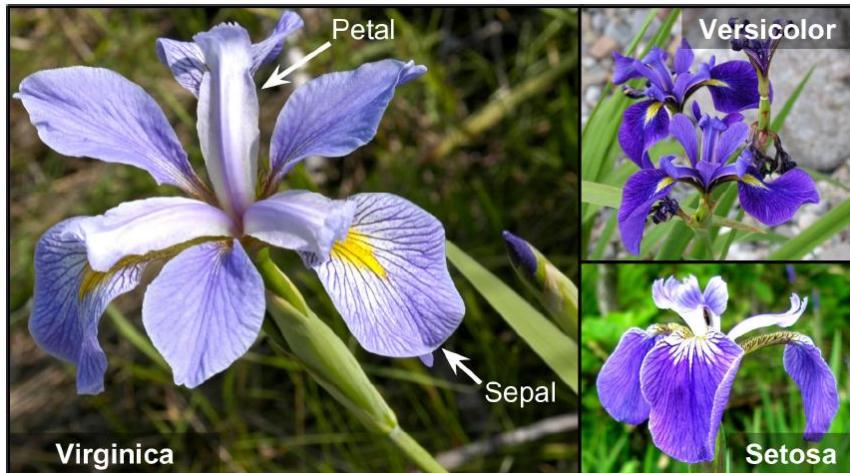
1. ADIM: HAREKETE GEÇ

Harekete Geç adımında öğrencilerle birlikte uygulamaya konu veri setinin Python kodları yardımıyla analizi ve Makine Öğrenmesi süreci öncesi hazır hale getirilmesi sağlanır.

Veri Setini Öğreniyorum

Sınıftaki öğrencilere Şekil 83'te görüldüğü gibi iris bir çiçeğin üç türüne (*setosa*, *versicolor*, *virginica*) ait fotoğraf gösterilir. Daha sonra üç türde 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.

Her bir öğrenci Tablo 1'de verilen iris çiçeğine ait yaprak ölçülerini ve türünü inceler ve karşılaştırır. Öğrenci veri seti tablosundan alt ve üst yaprak uzunluğu-genişliğine giriş parametresi olarak isimlendirir. Ölçümlerin değerlerine göre iris çiçeği yaprak türlerine ise çıkış parametresi olarak isimlendirir. Öğrenci veri setindeki ondalıklı sayılarında virgül yerine noktanın kullanıldığını öğrenir.



Şekil 83. İris çiçeği ve türleri (Géron, 2022).

Tablo 1. Iris 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

Python ile Veri Seti Hazırlıyorum

Öğrenci daha önce de kullanılan iris veri setini (Bkz. 1. Hafta – 3. Gün – 2. Ders ve 1. Hafta – 4. Gün – 2., 3. Dersler) Python scikit-learn kütüphanesi içerisinde çağırarak farklı alternatif bir sınıflandırma uygulamasında kodlar. Tasarım aşamasında iris çiçeğinin alt ve üst yaprak uzunluk- genişlik verilerini kullanarak çiçeğin türünü sınıflandırmaya çalışır. Şekil 84’te gösterilen gerekli kütüphane ve veri setini indirir.

```

Spyder (Python 3.8)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\Koray\Desktop\Yapay Zeka\Yapay Zeka Ortaokul\VeriSetiHazırlıyorum.py
VeriSetiHazırlıyorum.py

1 from sklearn.datasets import load_iris
2 iris = load_iris()

```

Şekil 84. Gerekli kütüphaneleri ve veriyi indirme.

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 zekada 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 sınıfından sayısalça çevrilmiştir. Sklearn kütüphanesinden alınan veriyi daha iyi anlayabilmek için verimizdeki bağımsız değişkenlerin (nitelikler) isimleri Şekil 85a'da verilen kod satırı kullanılarak elde edilir. Şekil 85b'de dört bağımsız değişkenin isim listesi görülmektedir.

(a)

(b)

```

Spyder (Python 3.8)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\Koray\Desktop\Yapay Zeka\Yapay Zeka Ortaokul\VeriSetiHazırlıyorum.py
VeriSetiHazırlıyorum.py

1 from sklearn.datasets import load_iris
2 iris = load_iris()
3 print (iris.feature_names)

In [1]: runfile('C:/Users/Koray/Desktop/Yapay Zeka/Ortaokul/VeriSetiHazırlıyorum.py', wdir='C:/Users/Koray/Desktop/Yapay Zeka/Ortaokul')
['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']

```

Şekil 85. Veri analizi (a) bağımsız değişkenleri görüntüleme kodu (b) bağımsız değişkenler konsol görüntüsü.

Sklearn kütüphanesinden alınan veriyi daha iyi anlayabilmek için verimizdeki bağımlı değişkenlerin isimleri Şekil 86a'da verilen kod satırı kullanılarak elde edilir. Şekil 86b'de bağımlı değişkenlerin isimleri listesi görülmektedir.

(a)

(b)

```

Spyder (Python 3.8)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\Koray\Desktop\Yapay Zeka\Yapay Zeka Ortaokul\VeriSetiHazırlıyorum.py
VeriSetiHazırlıyorum.py

1 from sklearn.datasets import load_iris
2 iris = load_iris()
3 print (iris.feature_names)
4 print (iris.target_names)
5

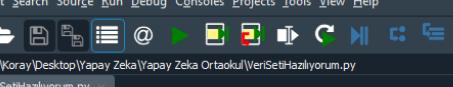
In [1]: runfile('C:/Users/Koray/Desktop/Yapay Zeka/Ortaokul/VeriSetiHazırlıyorum.py', wdir='C:/Users/Koray/Desktop/Yapay Zeka/Ortaokul')
['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']
['setosa' 'versicolor' 'virginica']

In [2]: 

```

Şekil 86. Veri analizi (a) bağımlı değişkenleri görüntüleme kodu (b) bağımlı değişkenler konsol görüntüsü.

Öğrenciler iris çiçek türüne ait üç farklı sınıfı listeler. Şekil 87'de bağımlı değişkenlerimiz olan setosa için "0", versicolor için "1" ve virginica için "2" ile kodlanarak sayısallaştırılmıştır. Şekil 87a ve 87b'de iris veri kodunu, Şekil 87a ve 87b'de bağımsız değişkenlerin sayısal görüntüsü verilmiştir.



The screenshot shows the Spyder Python IDE interface. The title bar reads "Spyder (Python 3.8)". The menu bar includes File, Edit, Search, Source, Run, Debug, Consoles, Projects, Tools, View, and Help. Below the menu is a toolbar with various icons for file operations like Open, Save, Copy, Paste, and Run. The current file path in the status bar is "C:\Users\Koray\Desktop\Yapay Zeka\Yapay Zeka Ortaokulu\VeriSetiHazırlıyorum.py". The code editor window contains the following Python script:

```
1 from sklearn.datasets import load_iris
2 iris = load_iris()
3 print (iris.feature_names)
4 print (iris.target_names)
5 print (iris.target)
```

(a)

(b)

Şekil 87. Veri analizi (a) bağımlı değişkenleri sayısallaştırma kodu (b) konsol görüntüsü.



The screenshot shows the Spyder Python IDE interface. The menu bar includes File, Edit, Search, Source, Run, Debug, Consoles, Projects, Tools, View, and Help. The toolbar features icons for file operations like Open, Save, and Print, along with navigation and execution icons. The current file is 'VeriSetiHazırlıyorum.py' located at 'C:\Users\koray\Desktop\Yapay Zeka\VeriSetiHazırlıyorum.py'. The code editor displays the following Python script:

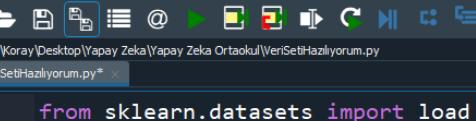
```
from sklearn.datasets import load_iris
iris = load_iris()
print (iris.feature_names)
print (iris.target_names)
print (iris.target)
print (iris.data)
```

(a)

(b)

Sekil 88. Veri analizi (a) bağımsız değişkenleri sayısallaştırma kodu (b) konsol görüntüsü.

Veri seti hazırlıyorum son aşamasında öğrenciler bağımlı ve bağımsız değişkenleri oluşturmak için bağımsız nitelikleri X, bağımlı niteliği Y değişkenine atanır (Şekil 89).



The screenshot shows the Spyder Python IDE interface. The menu bar includes File, Edit, Search, Source, Run, Debug, Consoles, Projects, Tools, View, and Help. Below the menu is a toolbar with icons for file operations like Open, Save, and Print. The current file path is displayed as C:\Users\Koray\Desktop\Vapay Zeka\Vapay Zeka Ortakul\VeriSetiHazırlıyorum.py. The code editor contains the following Python script:

```
1 from sklearn.datasets import load_iris
2 iris = load_iris()
3 print (iris.feature_names)
4 print (iris.target_names)
5 print (iris.target)
6 print (iris.data)
7 X = iris.data
8 Y = iris.target
9
```

Şekil 89. Bağımlı ve bağımsız değişkenleri X ve Y değişkenine atama kod satırı.

Eğitim ve Test Verilerini Ayırma

Öğrenciler iris çiçeği türüne ait verilerin %80’ini eğitim için, %20’sini ise test için ayırır. Şekil 90a’dı gösterildiği gibi veri setlerini ayırmak için Python kodlarını yazar. Şekil 90b’de ekran görüntüsü verilmiştir. Öğretmen, öğrencilere veri setini eğitim ve test verilerini ayırirken, eğitim veri setinin test veri setinden daha büyük olması gerektiğini belirtir. Eğer test ve eğitim veri seti arasında yüzdesel olarak birbirine yakın olursa modelden elde edilen sonuçların yanlış olabileceğini irdeler. Bu nedenle eğitim veri seti genellikle %70’ten başlayarak %80, %85’e kadar ayrılırken test veri seti ise %15 ile %25-%30 arasında değişebilmektedir. Modelde eğitim ve test verileri yüzdeleri kullanılan veri setine bağlı olarak ayarlanabilmektedir.

The screenshot shows the Spyder Python IDE interface. The menu bar includes File, Edit, Search, Source, Run, Debug, Consoles, Projects, Tools, View, Help. The toolbar has icons for file operations like Open, Save, Print, and various execution and navigation tools. The current file is 'VeriSetiHazırlıyorum.py' located at 'C:\Users\Koray\Desktop\Yapay Zeka\Yapay Zeka Ortaokul\VeriSetiHazırlıyorum.py'. The code in the editor is:

```
1 from sklearn.datasets import load_iris
2 iris = load_iris()
3 print (iris.feature_names)
4 print (iris.target_names)
5 print (iris.target)
6 print (iris.data)
7 X = iris.data
8 Y = iris.target
9 from sklearn.model_selection import train_test_split
10 X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.20, random_state = 0)
11 print("Eğitim veri seti boyutu=",len(X_train))
12 print("Test veri seti boyutu=",len(X_test))
```

The 'Console 1/A' tab shows the execution results:

```
[7.7 3. 6.1 2.3]
[6.3 3.4 5.6 2.4]
[6.4 3.1 5.5 1.8]
[6. 3. 4.8 1.8]
[6.9 3.1 5.4 2.1]
[6.7 3.1 5.6 2.4]
[6.9 3.1 5.1 2.3]
[5.8 2.7 5.1 1.9]
[6.8 3.2 5.9 2.3]
[6.7 3.3 5.7 2.5]
[6.7 3. 5.2 2.3]
[6.3 2.5 5. 1.9]
[6.5 3. 5.2 2. ]
[6.2 3.4 5.4 2.3]
[5.9 3. 5.1 1.8]]
Eğitim veri seti boyutu= 120
Test veri seti boyutu= 30
```

(a)

(b)

Şekil 90. Eğitim ve test verileri ayırma (a) kod (b) konsol görüntüsü.

Model Kurma

Öğrenciler model için scikit-learn kütüphanesinden karar ağaçları sınıflandırıcısını çağırır ve model adında bir değişkene aktarır. Şekil 91’de verilen karar ağaçlarına ait algoritmanın Python kodunu editöre yazar. Karar ağıacı sınıflandırıcısı, her bir dalın birleşme noktası (düğüm) bir özelliği (özniteliği) gösterir ve her dal bir kararı işaret eder. Karar ağacında düğümler, sınıflandırılacak gruptaki özellikleri temsil eder ve dallar ise düğümlerin alabileceği değerleri temsil eder (Köse vd., 2023).

The screenshot shows the Spyder Python 3.8 IDE interface. The title bar reads "Spyder (Python 3.8)". The menu bar includes File, Edit, Search, Source, Run, Debug, Consoles, Projects, Tools, View, and Help. Below the menu is a toolbar with various icons for file operations like Open, Save, Copy, Paste, and Run. The current file path is "C:\Users\Koray\Desktop\Yapay Zeka\Yapay Zeka Ortaokul\VeriSetiHazırlıyorum.py". The code editor displays the following Python script:

```
1  from sklearn.datasets import load_iris
2  iris = load_iris()
3  print (iris.feature_names)
4  print (iris.target_names)
5  print (iris.target)
6  print (iris.data)
7  X = iris.data
8  Y = iris.target
9  from sklearn.model_selection import train_test_split
10 X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.20, random_state = 0)
11 print("Eğitim veri seti boyutu=",len(X_train))
12 print("Test veri seti boyutu=",len(X_test))
13 from sklearn.tree import DecisionTreeClassifier
14 model = DecisionTreeClassifier()
```

Sekil 91. Karar ağaçlarına ait algoritmanın kodu.



2. ADIM: KEŞFET

Önceki adımı takiben bu adımda öğrencilerle birlikte ilgili Makine Öğrenmesi modelinin çalıştırılmasına dair kodlamaların yapılması sağlanır.

Modelin Eğitimi

Öğrenci, Şekil 92'de gösterildiği gibi Python programlama editöründe iris çiçeği türlerine ait eğitim verilerini **model.fit** komutu ile eğitimini gerçekleştirir.

```
Spyder (Python 3.8)

File Edit Search Source Run Debug Consoles Projects Tools View Help

C:\Users\Koray\Desktop\Yapay Zeka\Yapay Zeka Ortaokul\VeriSetiHazliyorum.py
VeriSetiHazliyorum.py

1 from sklearn.datasets import load_iris
2 iris = load_iris()
3 print (iris.feature_names)
4 print (iris.target_names)
5 print (iris.target)
6 print (iris.data)
7 X = iris.data
8 Y = iris.target
9 from sklearn.model_selection import train_test_split
10 X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.20, random_state = 0)
11 print("Eğitim veri seti boyutu=",len(X_train))
12 print("Test veri seti boyutu=",len(X_test))
13 from sklearn.tree import DecisionTreeClassifier
14 model = DecisionTreeClassifier()
15 model.fit(X_train,Y_train)
```

Şekil 92. Modelin eğitime.

Model ile Tahminler

Öğrenci, Şekil 93'te gösterildiği gibi Python programlama editöründe iris çiçeği türlerine ait test verilerini **model.predict** komutu ile giriş verilerine göre çiçeğin hangi tür çiçek sınıfına ait olduğunu tahmin eder.

```
Spyder (Python 3.8)

File Edit Search Source Run Debug Consoles Projects Tools View Help

C:\Users\Koray\Desktop\Yapay Zeka\Yapay Zeka Ortaokul\VeriSetiHazliyorum.py
VeriSetiHazliyorum.py

1 from sklearn.datasets import load_iris
2 iris = load_iris()
3 print (iris.feature_names)
4 print (iris.target_names)
5 print (iris.target)
6 print (iris.data)
7 X = iris.data
8 Y = iris.target
9 from sklearn.model_selection import train_test_split
10 X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.20, random_state = 0)
11 print("Eğitim veri seti boyutu=",len(X_train))
12 print("Test veri seti boyutu=",len(X_test))
13 from sklearn.tree import DecisionTreeClassifier
14 model = DecisionTreeClassifier()
15 model.fit(X_train,Y_train)
16 Y_tahmin = model.predict(X_test)
```

Şekil 93. Model ile tahminler.



3. ADIM: ÜRET

Öğrencilerle birlikte çalıştırılan Makine Öğrenmesi modelinin elde ettiği sonuçların değerlendirilmesine yönelik kodlamaların yapılması sağlanır.

Tahmin-Test Sonuçlarını Karşılaştırma

Öğrenciler, karar ağaçları modeli ile eğitilen iris çiçeği sınıfının eğitim ve tahmin süreci ardından genel değerlendirmesini yapmak için hata matrisini (karmaşıklık matrisi / confusion matrix) oluştururlar. Hata matrisi özellikle Yapay Zeka'nın sınıflandırma problemlerinde başarımını ölçmek için kullanılmaktadır. Bu doğrultuda Şekil 94a'daki Python programlama editörüne hata matrisine ait kodlar yazılır. Şekil 94b'de ise hata matrisine ait sonuçlar gözlemlenir.

```
Spyder (Python 3.8)
File Edit Search Source Run Debug Cnsoles Projects Tools View Help
C:\Users\Koray\Desktop\Yapay Zeka\Yapay Zeka Ortaçlu\VeriSehaziyorum.py
[6. 3. 4.8 1.8]
[6.9 3.1 5.4 2.1]
[6.7 3.1 5.6 2.4]
[6.9 3.1 5.1 2.3]
[5.8 2.7 5.1 1.9]
[6.8 3.2 5.9 2.3]
[6.7 3.3 5.7 2.5]
[6.7 3. 5.2 2.3]
[6.3 2.5 5. 1.9]
[6.5 3. 5.2 2. ]
[6.2 3.4 5.4 2.3]
[5.9 3. 5.1 1.8]]
Eğitim veri seti boyutu= 120
Test veri seti boyutu= 30
[[11  0  0]
 [ 0 13  0]
 [ 0  0  6]]
```

(a)

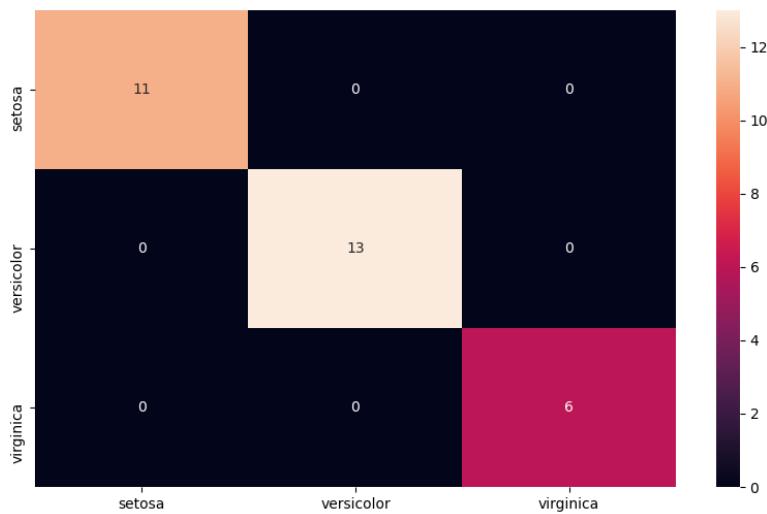
(b)

Şekil 94. Hata matrisi (a) kod (b) sonuç konsol görüntüsü.

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 modeli 120 kayıt ile eğitti. Geriye kalan 30 kayıt (X_{test}) ise modeli tahmin etmek için kullandı (Y_{tahmin}). Öğrenciler, Şekil 95'te gösterildiği gibi 30 test kaydına ait gerçek sonuçlar (Y_{test}) ile tahmin sonuçlarını karşılaştırmış ve hata matrisini elde etmiştir. Böylelikle matristeki sayıların toplamının test verisi olan 30'a eşit olduğu görülmektedir.

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 versicolor varmış ve hepsi doğru tahmin edilmiş ve 9 tane virginica varmış ve hepsi de başarıyla virginica olarak tahmin edildiğini görmüştür.



Şekil 95. Hata matrisi.

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)
```



4. ADIM: İLERLET

İlerlet adımda öğrencilerin gruplar halinde çalışarak, gerçekleştirilen uygulama üzerinde kendi düzenlemelerini yapmaları ve elde edilen sonuçlar hakkında notlar almaları istenir. Bu amaçla 30 dakikalık bir çalışma süresi verilir. Çalışma süresi esnasında öğrencilerin yaptıkları düzenlemeler uzaktan gözlemlenir, düzenlemelere esnasında sorusu olan öğrencilerin soruları cevaplandırılıp, yardım edilir.



5. ADIM: DEĞERLENDİRME

Derste edinilen bilgiler soru-cevap aşaması gerçekleştirilmek suretiyle pekiştirilir.

Öğrencilere şu sorular sorulabilir:

- Gerçekleştirilen Makine Öğrenmesi uygulamasından elde ettiğiniz bilgi ve beceriler neler olmuştur?
- Uygulığınız Python kodlarında veri hazırlığı ve Makine Öğrenmesi modelinin oluşturularak eğitim-test süreçlerinden geçirilmesi aşamalarına dair görüşleriniz nelerdir?
- Python ile Yapay Zeka sınıflandırma problemleri nasıl çözümlenmekte, başarıım nasıl değerlendirilebilmektedir?
- Benzeri bir uygulamayı gerçekleştirebileceğiniz problemler neler olabilir?

2. HAFTA – 1. GÜN – 1. DERS: OUTLIER (AYKIRI DEĞER) TESPİTİ VE FILTRELEME

DERS PLANI

DERS ETİKETLERİ



KONU: Outlier (Aykırı Değer) Tespiti ve Filtreleme



SINIF: 12-14 Yaş



SÜRE: 60 dk.



HAZIR BULUNUŞLUK: Temel düzeyde teknoloji farkındalığı, temel İngilizce eğitimi, Yapay Zeka kavramı ve uygulamaları hakkında temel bilgi, Makine Öğrenmesi hakkında temel bilgi,



ANAHTAR KELİMELER: Yapay Zeka, Algoritma, Bilgisayar Programlama, Kodlama, Outlier, Aykırı Değer



BECERİLER:



Tahmin



Sorgulama



Günlük hayat bağlantısı kurma



KAZANIMLAR:

Bu derste öğrenciler;

- Outlier (aykırı değer) kavramının ne anlama geldiğini kavrar.
- Veri setlerinde outlier olması durumunda Yapay Zeka uygulamalarında ortaya çıkabilecek problemleri kavrar.
- Outlier problemini çözmek için kullanılabilecek tespit ve filtreleme yolları hakkında bilgi sahibi olur.
- Gerçek hayatta outlier durumu söz konusu olabilecek problem türlerini tanımlar.



DERS MATERİYALLERİ:

Malzeme	Miktar
Yazı Tahtası	Öğrenci sayısı ve sınıf büyüklüğüne göre en az bir adet
Projeksiyon	Öğrenci sayısı ve sınıf büyüklüğüne göre en az bir adet
Defter, Kalem	Öğrenci sayısı kadar
Bilgisayar	Öğrenci sayısının yarısı kadar
Tablet	Öğrenci sayısının yarısı kadar



ÖĞRETMEN HAZIRLIK ÇALIŞMALARI:

Okunabilecek Kaynaklar:

Aydın, A. G., & Tunçer, A. A. N. (2022). Aykırı Değer Tespit Yöntemlerinin Karşılaştırılması ve Diğer Ön İşleme Adımlarının Otomatik Makine Öğrenmesi (Automl) ile İlişkisinin İncelenmesi. In *The 14th International Scientific Research Congress* (p. 106).

NIST. (2012). *What are outliers in the data?*. e-Handbook of Statistical Methods. Çevrimiçi: <https://www.itl.nist.gov/div898/handbook/prc/section1/prc16.htm> (Erişim 14 Haziran 2024).

İzlenebilecek Kaynaklar:

CampusX. (2021). What are Outliers | Outliers in Machine Learning. Çevrimiçi: https://www.youtube.com/watch?v=LJn1PKgGr_M (Erişim 14 Haziran 2024).

Khan Academy Türkçe. (2020). Veri Kümesinde Aykırı Değerleri Yorumlama (Matematik) (İstatistik ve Olasılık). Çevrimiçi: <https://www.youtube.com/watch?v=njoI0GB56EI> (Erişim 14 Haziran 2024).

ATÖLYE HAZIRLIK ÇALIŞMALARI:

Öğretmen;

- Dersten önce ders materyallerini hazırlamalıdır.
- Outlier (aykırı değer) ilişkin ön bilgi edinmeli, kullanılabilecek tespit ve filtreleme yolları hakkında farkındalık ve bilgi düzeyini güncel tutmalıdır.



KAYNAKÇA:

- Dangeti, P. (2017). *Statistics for machine learning*. Packt Publishing Ltd.
- Deitz, S. (2023). Outlier bias: AI classification of curb ramps, outliers, and context. *Big Data & Society*, 10(2), 20539517231203669.
- Frost, J. (2024). 5 Ways to Find Outliers in Your Data. Çevrimiçi: <https://statisticsbyjim.com/basics/outliers/> (Erişim 14 Haziran 2024).
- Köse, U. (2021). *Yapay Zeka Etiği*. Nobel Yayıncılık.
- Smiti, A. (2020). A critical overview of outlier detection methods. *Computer Science Review*, 38, 100306.
- Sugiyama, M. (2015). *Introduction to statistical machine learning*. Morgan Kaufmann.
- Suri, N. M. R., Murty, M. N., & Athithan, G. (2019). *Outlier detection: Techniques and applications*. Springer Nature.

GENEL BAKIŞ:

- 1) Harekete Geç:** Öğrencilere outlier (aykırı değer) kavramı ile ne anladıkları sorulur ve farklı görüşler toplanarak tartışılır.
- 2) Keşfet:** Outlier (aykırı değer) kavramı, outlier problemine sebep olan durumlar ve outlier nedeniyle Yapay Zeka uygulamalarında ortaya çıkabilecek sorunlar anlatılır.
- 3) Üret:** Outlier (aykırı değer) problemlerini tespit edip filtrelemek için kullanılabilen yöntemlere ilişkin genel bilgi verilir.
- 4) İlerlet:** Öğrencilerin gerçek hayatı ortaya çıkarabilecek problem türlerini düşünerek bu problemler hakkında notlar almaları sağlanır.
- 5) Değerlendir:** Alınan notlar ve derste öğrenilen bilgiler ışığında soru-cevap etkinliği gerçekleştirilerek outlier (aykırı değer) kavramı pekiştirilir.

UYGULAMA



1. ADIM: HAREKETE GEÇ

Harekete geç adımında öğrencilere outlier (aykırı değer) kavramı ile ne anladıkları sorulur ve farklı görüşler toplanarak tartışılır.



2. ADIM: KEŞFET

Outlier (aykırı değer) kavramı, outlier problemine sebep olan durumlar ve outlier nedeniyle Yapay Zeka uygulamalarında ortaya çıkabilecek sorunlar anlatılır.

Outlier (Aykırı Değer) Kavramı

İstatistik, Veri Bilimi ve Yapay Zeka uygulamaları (özellikle Makine Öğrenmesi / Derin Öğrenme) doğrudan veriye dayalı çözümler içermektedir. Bu nedenle bu tür uygulamalarda kullanılan veri içeriklerinin sağlıklı olması kritiktir. Ancak çeşitli nedenlerden dolayı veri setleri içerisinde verilerin ortak özelliklerini dışında kalan çeşitli veriler yer almaktadır. Bu tür veriler outlier (aykırı değer / veri) olarak adlandırılmaktadır.

Çeşitli kaynaklarda “gürültü (noise)” ya da “anomali (anomaly)” olarak da adlandırılan outlier türündeki veriler özellikle günümüz Yapay Zeka uygulamalarında karşılaşılan önemli problemlerden biri konumundadır (Sugiyama, 2015; Suri vd., 2019). İstatistik gibi klasik matematiksel temellere dayanan araştırma alanlarının çoktan beri odak noktasında yer alan outlier, bir bakıma eldeki verilerden uzak noktalarda yer alan verileri tanımlamak için de kullanılan bir kavram olarak bilinmektedir (Dangeti, 2017; Sugiyama, 2015).

Outlier (Aykırı Değer) Sebepleri ve Yapay Zeka Bağlamında Etkileri

Outlier problemi çeşitli sebeplerden kaynaklanabilmektedir. Bunlardan önde gelenleri şöyledir (Dangeti, 2017; Smiti, 2020):

- Veri toplama ve ölçüm aracında oluşan teknik problemler.
- Verilerin problemi temsil edecek bütün olası durumları kapsayacak şekilde toplanmamış olması.
- İnsan hatası içeren veriler ile veri seti oluşturulması.
- Bilimsel ve teknik olarak hatalı veri seti oluşturma işlemleri.
- Verilerin daha dar bir ortamdan toplandığı halde geniş bir kapsama uygulanmaya çalışılması.

Outlier içeren veri setleri ile Makine Öğrenmesi / Derin Öğrenme sistemleri geliştirildiği zaman bu sistemlerle bağlantılı Yapay Zeka uygulamalarında ortaya çıkabilecek problemler şu şekildedir (Deitz, 2023; Köse, 2021):

- Yapay Zeka uygulamaları ön yargı (bias) içeren kararlar oluşturabilmektedir.
- İnsanlar Yapay Zeka uygulamalarına bağlı olarak hatalı kararlar alabilmektedir.
- Yapay Zeka uygulamaların tetiklediği makineler ya da yazılımlar dış ortamda zarar oluşturabilecek ya da sistemlerde teknik, ekonomik çöküşleri ortaya çıkarabilecek sonuçlara yol açabilmektedir.



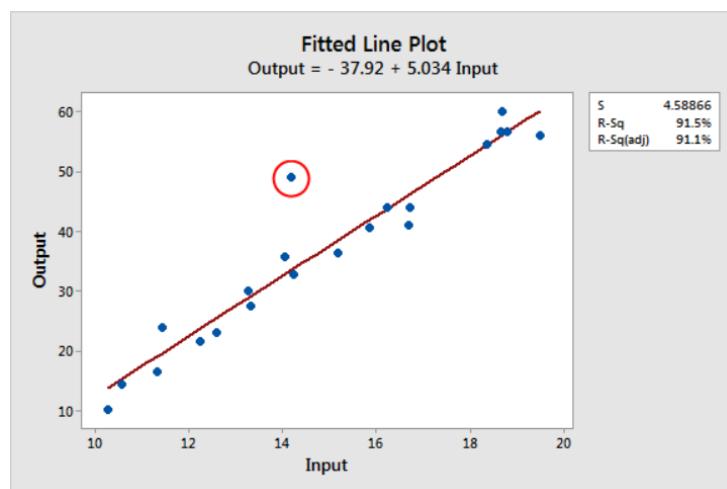
3. ADIM: ÜRET

Outlier (aykırı değer) problemlerini tespit edip filtrelemek için kullanılabilen yöntemlere ilişkin genel bilgi verilir.

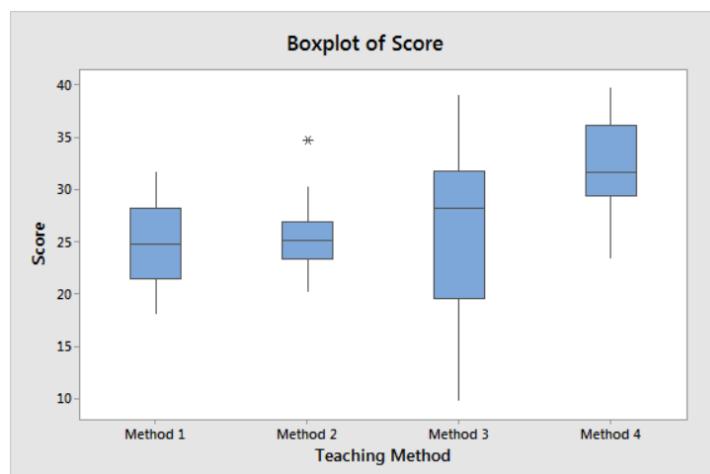
Outlier (Aykırı Değer) Tespiti ve Filtreleme

Outlier tespiti konusunda farklı yaklaşımlar tercih edilebilmektedir. Aslında verilere genel bir bakış atılması sağlayarak aykırı durumda kilerin tespitini kolaylaştırın her türlü yöntem etkili olabilmektedir. Örneğin, verilerin belli bir sırada (küçükten büyüğe ya da büyükten küçüğe) sıralanması bile nispeten az sayıda veri içeren veri setlerinde oldukça basit bir tespit yöntemi haline gelebilmektedir. Böylelikle belli bir sırayı takip eden verilerde artış veya azalış durumu anormal durumda olanlar gözlemlenebilir. Diğer yandan verilerin dağılımını görsel araçlarla incelemek de outlier tespitinde etkilidir. Bu amaçla “çizgi grafiği”, “kutu grafiği” ya da “histogram” outlier konumundaki verileri işaret etmektedir.

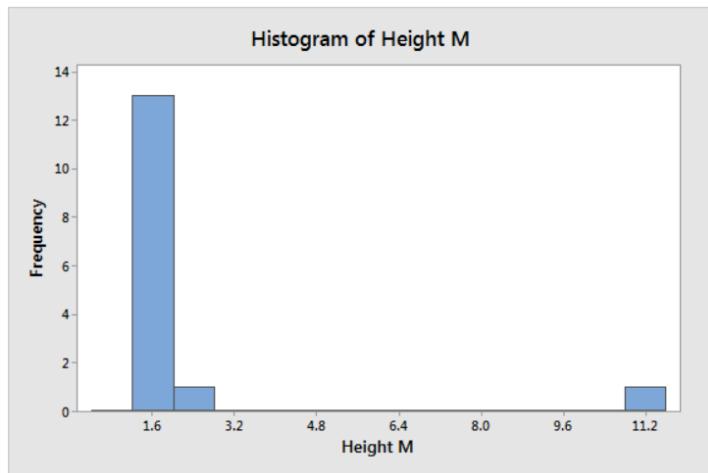
Öğrencilere Şekil 96'da yer alan grafikler gösterilir. Çizgi grafiğinde outlier olarak görülen (kırmızı çember içerisindeki) verinin tespit edilen veri eğrisi dışında olması durumu (Şekil 96a) irdelenir. Benzer şekilde kutu grafiğinde yıldız (*) karakteri ile gösterilen veri Method 2 girdisi / parametresi için bilinen sınırların dışında yer almakla (Şekil 96b), histogram grafiğinde de Height girdisi / parametresi için sağ ucta gözlemlenen dağılım sol tarafta yoğunlaşan dağılım ötesinde olmakla (Şekil 96c) outlier karakteri sergilemektedir (Frost, 2024).



(a)



(b)



(c)

Şekil 96. Outlier (aykırı değer) tespitinde (a) çizgi grafiği (b) kutu grafiği (c) histogram grafiği (Frost, 2024).

Outlier verilerin tespitinde istatistiksel yöntemler ve hatta Makine Öğrenmesi algoritmaları (Örneğin, kNN algoritması ile yakınlıklara göre tespit ya da kümeleme algoritmaları ile kümeler özelinde tespit) kullanılabilmektedir. İstatistiksel olarak “Z-skoru” yöntemi, “Interquartile Range (IQR)” ya da “hipotez testleri” yaygın tercih edilen yöntemler arasındadır (Frost, 2024; Smiti, 2020; Suri vd., 2019). Z-skoru yönteminde her bir verinin ortalamadan uzaklığını tespit edilmekte [$Z_{\text{skoru}} = (\text{veri_noktası} - \text{ortalama}) / \text{standart_sapma}$] ve ± 3 Z-skoru veren veriler outlier olarak tespit edilmektedir. IQR yönteminde ise birinci çeyreklik (Q_1) ve üçüncü çeyreklik (Q_3) değerine göre belirlenen aralığın dışında kalanlar [$IQR = Q_3 - Q_1$ için $Q_1 - (1,5 * IQR)$ 'dan küçük olanlar ve $Q_3 + (1,5 * IQR)$ 'dan büyük olanlar] outlier kabul edilmekte, hipotez testlerinde ise belli güven aralıklarına göre veriler arasında anlamlı farklılıklar oluşturan noktalar outlier içeresine girmektedir.

Outlier düzeyindeki veriler tespit edildiğinde veri setlerinin negatif yönde etkilenmemesi için şu çözüm yolları tercih edilmektedir:

- Outlier düzeyinde veriler filtrelenmek suretiyle outlier verilerin yer almadığı; sağlıklı veri seti elde edilir.
- Outlier düzeyinde veriler filtrelenir ancak bu verilerin yerine farklı yöntemlerle (Örneğin, regresyon tabanlı Makine Öğrenmesi algoritmalarıyla ya da geleneksel istatistik ile) yeni veriler tahmin edilir. Böylelikle orijinal veri setinde yer alan veri miktarı etkilenmez.
- Outlier düzeyinde veriler filtrelenir ancak bu veriler manipüle edilerek outlier seviyesinden çıkışlarını sağlayacak yönde yeni değerlere doğru değişimleri sağlanır. Böylelikle orijinal veri setinde yer alan veri miktarı etkilenmez.



4. ADIM: İLERLET

Öğrencilerin gerçek hayatı outlier problemi ortaya çıkarabilecek problem türlerini düşünerek bu problemler hakkında notlar almaları sağlanır. Bu etkinlik için ortalama 10 dakikalık bir süre ayrılr.



5. ADIM: DEĞERLENDİRME

Önceki adımda alınan notlar ve derste öğrenilen bilgiler ışığında soru-cevap etkinliği gerçekleştirilerek outlier (aykırı değer) kavramı pekiştirilir.

Öğrencilere şu sorular sorulabilir:

- Öğrendiklerinizden hareketle outlier (aykırı değer) kavramının tanımını nasıl yapabilirsiniz?
- Outlier nedeniyle Yapay Zeka uygulamalarında ortaya çıkabilecek problemler bizleri nasıl etkileyebilir?
- Outlier olduğunu düşündüğünüz bir Yapay Zeka uygulaması var mıdır?
- Şu ana kadar öğrendiğiniz Python konuları ile outlier problemini çözebilir misiniz?

2. HAFTA – 1. GÜN – 2., 3. DERSLER: OUTLİER TESPİTİ VE FİLTRELEME - PYTHON İLE ÖRNEK UYGULAMALAR

DERS PLANI

DERS ETİKETLERİ



KONU: Outlier Tespiti ve Filtreleme - Python ile Örnek Uygulamalar



SINIF: 12-14 Yaş



SÜRE: 120 dk.



HAZIR BULUNUŞLUK: Temel düzeyde teknoloji farkındalığı, temel İngilizce eğitimi, Yapay Zeka kavramı ve uygulamaları hakkında temel bilgi, Makine Öğrenmesi hakkında temel bilgi,



ANAHTAR KELİMELER: Yapay Zeka, Algoritma, Bilgisayar Programlama, Kodlama, Outlier, Aykırı Değer



BECERİLER:



Tahmin



Sorgulama



Grup çalışması



Günlük hayat bağlantısı kurma



KAZANIMLAR:

Bu derste öğrenciler;

- Python ortamında outlier (aykırı değer) tespiti için gerçekleştirilebilecek kodlamalar konusunda beceri sahibi olurlar.
- Python ile tespit edilen outlier (aykırı değer) verilerin filtrelenmesi konusunda beceri sahibi olurlar.
- Python ortamında iki farklı veri seti üzerinden outlier (aykırı değer) tespiti ve filtrelemeye dair adımlar hakkında bilgi ve beceri sahibi olurlar.



DERS MATERİYALLERİ:

Malzeme	Miktar
Yazı Tahtası	Öğrenci sayısı ve sınıf büyüklüğüne göre en az bir adet
Projeksiyon	Öğrenci sayısı ve sınıf büyüklüğüne göre en az bir adet
Defter, Kalem	Öğrenci sayısı kadar
Bilgisayar	Öğrenci sayısının yarısı kadar
Tablet	Öğrenci sayısının yarısı kadar
Python Anaconda Framework	Her bilgisayarda kurulu olacak şekilde



ÖĞRETMEN HAZIRLIK ÇALIŞMALARI:

Okunabilecek Kaynaklar:

Boukerche, A., Zheng, L., & Alfandi, O. (2020). Outlier detection: Methods, models, and classification. *ACM Computing Surveys (CSUR)*, 53(3), 1-37.

NIST. (2012). *What are outliers in the data?*. e-Handbook of Statistical Methods. Çevrimiçi: <https://www.itl.nist.gov/div898/handbook/prc/section1/prc16.htm> (Erişim 14 Haziran 2024).

Zhao, Y., Nasrullah, Z., & Li, Z. (2019). Pyod: A python toolbox for scalable outlier detection. *Journal of Machine Learning Research*, 20(96), 1-7.

İzlenebilecek Kaynaklar:

Data Camp. (2020). Python Tutorial: Data outliers and scaling. Çevrimiçi: <https://www.youtube.com/watch?v=eVaZC2VYTIQ> (Erişim 14 Haziran 2024).

Hackers Realm. (2022). How to Detect and Remove Outliers in the Data | Python. Çevrimiçi: <https://www.youtube.com/watch?v=Cw2IvmWRcXs> (Erişim 14 Haziran 2024).

Khan Academy Türkçe. (2020). Veri Kümesinde Aykırı Değerleri Yorumlama (Matematik) (İstatistik ve Olasılık). Çevrimiçi: <https://www.youtube.com/watch?v=njoI0GB56EI> (Erişim 14 Haziran 2024).

ATÖLYE HAZIRLIK ÇALIŞMALARI:

Öğretmen;

- Dersten önce ders materyallerini hazırlamalıdır.
- Outlier (aykırı değer) ilişkin ön bilgi edinmeli, kullanılabilecek tespit ve filtreleme yolları hakkında farkındalık ve bilgi düzeyini güncel tutmalıdır.
- Python kodlama için kullanılan Spyder editörünün genel kullanımına aşina olmalıdır.
- Derse konu olan veri set(ler)ini ve Python kodlarını önceden inceleyerek üzerinde çalışmış olmalıdır.



KAYNAKÇA:

Frost, J. (2024). 5 Ways to Find Outliers in Your Data. Çevrimiçi: <https://statisticsbyjim.com/basics/outliers/> (Erişim 14 Haziran 2024).

Smiti, A. (2020). A critical overview of outlier detection methods. *Computer Science Review*, 38, 100306.

GENEL BAKIŞ:

1) Harekete Geç: Öğrencilere Python ile ilgili bildiklerinden yola çıkarak outlier (aykırı değer) tespiti ve filtrelemesi uygulamaları kodlama konusundaki kişisel yetkinlikleri konusunda görüşleri sorularak tartışılar.

2) Keşfet: Öğrencilerin örnek bir veri seti üzerinde outlier tespiti yaparak standart bir threshold değeri üzerinden ve ayrıca Z-skoru yöntemi ile filtrelemeler gerçeklestirmesi sağlanır.

3) Üret: Öğrenciler kodlanan Python uygulaması üzerinde IQR yöntemi ile filtreleme gerçekleştirir.

4) İlerlet: Öğrencilerin outlier (aykırı değer) tespiti ve filtreleme işlemlerini farklı bir veri seti üzerinde uygulamaları sağlanır ve ardından uygulama üzerinde güncellentirmeler yapmaları istenerek elde ettikleri sonuçlar hakkında notlar almaları istenir.

5) Değerlendir: Güncelleştirmeler sonrası alınan notlar ve derste öğrenilen bilgiler ışığında soru-cevap etkinliği gerçekleştirilerek Python ile outlier (aykırı değer) tespiti ve filtreleme bilgi-becerileri pekiştirilir.

UYGULAMA



1. ADIM: HAREKETE GEÇ

Bu adımda öğrencilere Python ile ilgili bildiklerinden yola çıkarak outlier (aykırı değer) tespiti ve filtrelemesi uygulamaları kodlama konusundaki kişisel yetkinlikleri konusunda görüşleri sorularak tartışılar.



2. ADIM: KEŞFET

Bu adımda öğrencilerin örnek bir veri seti üzerinde outlier tespiti yaparak standart bir threshold değeri üzerinden ve ayrıca Z-skoru yöntemi ile filtrelemeler gerçekleştirmesi sağlanır.

Öğrenciler, ilk olarak Şekil 97'de olduğu gibi sklearn kütüphanesinden şeker hastalığı (diyabet) veri setini, Pandas kütüphanesini ve Matplotlib kütüphanesini çağırır.

Uygulama ortamına çağrıılan şeker hastalığı veri seti içerisindeki veriler ve sütun isimleri yine Şekil 97'de olduğu gibi (11.-14. satır arası) dataframe altında tutulur.

```
7  from sklearn.datasets import load_diabetes  
8  import pandas as pd  
9  import matplotlib.pyplot as plt  
10  
11  diabet = load_diabetes()  
12  sutun_isimleri = diabet.feature_names  
13  df_diabet = pd.DataFrame(diabet.data)  
14  df_diabet.columns = sutun_isimleri
```

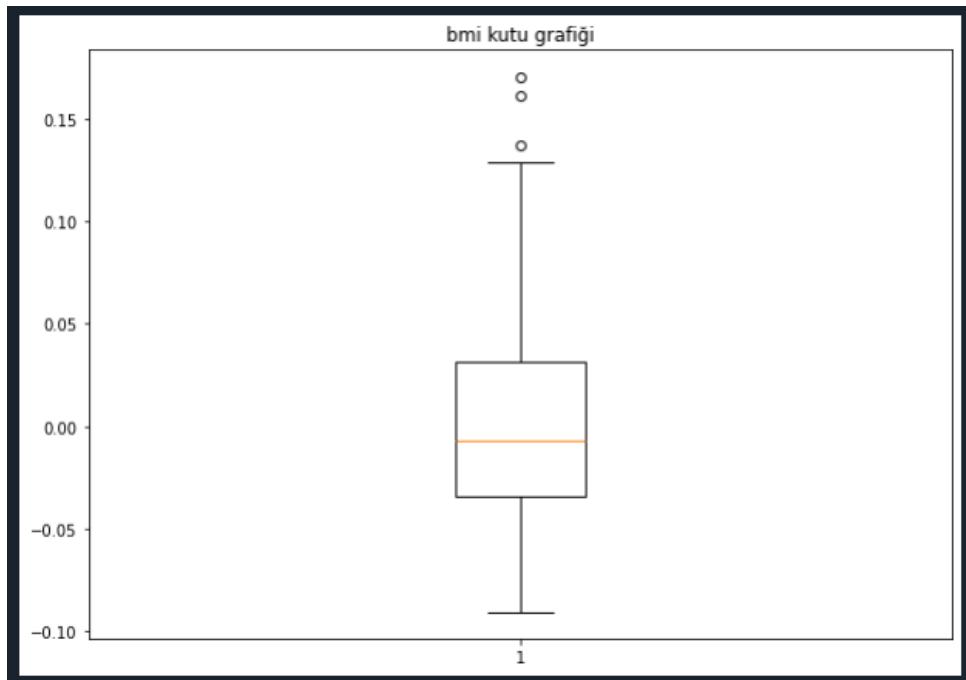
Şekil 97. Outlier (aykırı değer) uygulamasında veri seti ve temel kütüphaneler.

Bilindiği üzere kutu grafikleri outlier gözleme konusunda kullanılabilen araçlardan biridir (Bkz. 2. Hafta – 1. Gün – 1. Ders). Dolayısıyla öğrenciler Şekil 98'de olduğu gibi şeker hastalığı veri setindeki **bmi** parametresini baz almak suretiyle outlier değerlerin görüleceği kutu grafiği çağrısını kodlar.

```
16  fig = plt.figure(figsize =(10, 7))  
17  
18  plt.boxplot(df_diabet['bmi'])  
19  plt.title("bmi kutu grafiği")  
20  plt.show()
```

Şekil 98. Outlier (aykırı değer) gözlemi için kutu grafiği kodları.

Yazılan kodların çalıştırılması sonucunda Şekil 99'da görülen kutu grafiği elde edilir. Grafikte üst alanda yer alan outlier değerlere dikkat çekilir.



Şekil 99. Şeker hastalığı veri setindeki bmi parametresi için kutu grafiğinde outlier gözlemi.

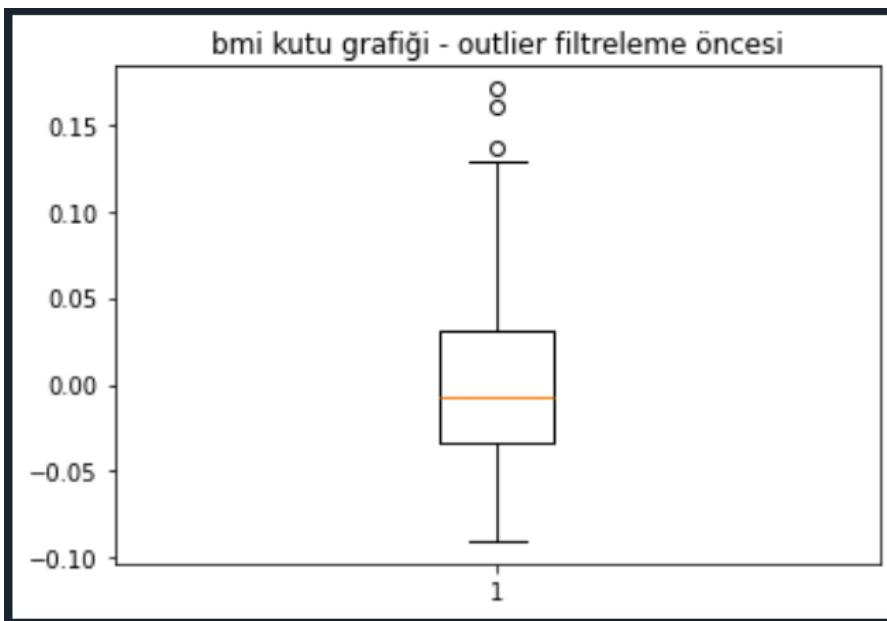
Öğrencilere outlier filtreleme noktasında oldukça basit bir yaklaşım olarak belli bir “threshold” değerinin dikkate alınabileceği anlatılır ve Şekil 100’de görülen fonksiyon kodları ile beraber filtreleme kodlarının uygulanması neticesinde, 0.10’luk bir threshold değeri üzerinden ulaşılan kutu grafiği değişimi incelenir (Şekil 101).

```

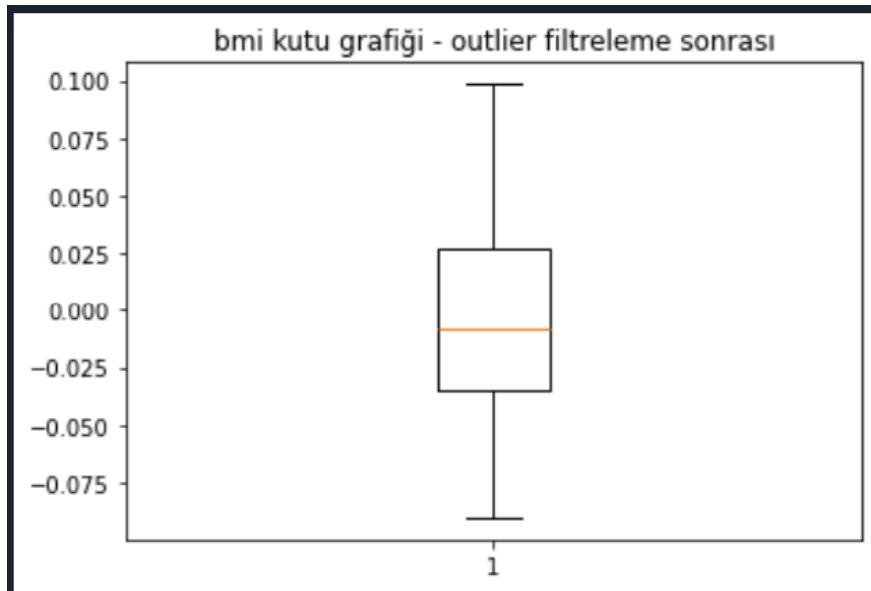
22 def outlier_filtrele(df, column, threshold):
23     plt.boxplot(df[column])
24     plt.title("bmi kutu grafiği - outlier filtreleme öncesi")
25     plt.show()
26
27     tespit_edilen_outlierlar = df[df[column] <= threshold]
28
29     plt.boxplot(tespit_edilen_outlierlar[column])
30     plt.title("bmi kutu grafiği - outlier filtreleme sonrası")
31     plt.show()
32     return tespit_edilen_outlierlar
33
34 threshold = 0.10
35
36 outlier_yok = outlier_filtrele(df_diabet, 'bmi', threshold)

```

Şekil 100. Şeker hastalığı veri setindeki bmi parametresinin threshold değer üzerinden filtrelenmesine dair kodlar.



(a)



(b)

Şekil 101. Threshold değeri uygulanması neticesinde (a) outlier değerler görülen (b) outlier değerlerden temizlenmiş olan bmi parametre değerleri bütünü.

Z-skor ile Filtreleme

Öğrencilerden bu defa bmi parametresi için outlier filtrelemesini Z-skor yöntemi (Frost, 2024; Smiti, 2020) ile gerçekleştirmeleri istenir. Bu amaçla Şekil 102'de görülen kodlar yazılmak suretiyle ekrana z-skorları yazdırılır. Bu noktada Z-skor hesaplaması için pratik olarak “scipy” adlı bilinen bir Python kütüphanesinin **stats** bileşeni kullanılmaktadır.

```

40 from scipy import stats
41 import numpy as np
42
43 z_değerleri = np.abs(stats.zscore(df_diabet['bmi']))
44 print(z_değerleri)

```

(a)

0	1.297088
1	1.082180
2	0.934533
3	0.243771
4	0.764944
	...
437	0.413360
438	0.334410
439	0.334410
440	0.821235
441	1.535374
Name: bmi, Length: 442, dtype: float64	

(b)

Şekil 102. Z-skoru tespiti (a) kod (b) sonuç konsol görüntüsü.

Z-skorlarının tespiti ardından belli bir z-skoru (threshold) değerine göre outlier konumuna düşen verilerin filtrelenmesi sağlanarak yeni veri seti elde edilir. Veri setindeki değişikliği anlamak adına dataframe değişkenlerinde “shape” özelliği çağrırlabilmektedir (Şekil 103).

```

46 threshold_z_skor = 2
47
48 outlier_indisler = np.where(z_değerleri > threshold_z_skor)[0]
49 outlier_filtrelendi = df_diabet.drop(outlier_indisler)
50
51 print("Z-skor öncesi:", df_diabet.shape)
52 print("Z-skor ilefiltreleme sonrası:", outlier_filtrelendi.shape)

```

(a)

Z-skor öncesi: (442, 10)
Z-skor ilefiltreleme sonrası: (427, 10)

(b)

Şekil 103. Z-skoru ile filtreleme (a) kod (b) sonuç konsol görüntüsü.

UYGULAMANIN PYTHON KODLARI

```
from sklearn.datasets import load_diabetes
import pandas as pd
import matplotlib.pyplot as plt

diabet = load_diabetes()
sutun_isimleri = diabet.feature_names
df_diabet = pd.DataFrame(diabet.data)
df_diabet.columns = sutun_isimleri

#
=====

# fig = plt.figure(figsize =(10, 7))
#
# plt.boxplot(df_diabet['bmi'])
# plt.title("bmi kutu grafiği")
# plt.show()

#
# def outlier_filtrele(df, column, threshold):
#     plt.boxplot(df[column])
#     plt.title("bmi kutu grafiği - outlier filtreleme öncesi")
#     plt.show()

#
#     tespit_edilen_outlierlar = df[df[column] <= threshold]
#
#     plt.boxplot(tespit_edilen_outlierlar[column])
#     plt.title("bmi kutu grafiği - outlier filtreleme sonrası")
#     plt.show()

#
#     return tespit_edilen_outlierlar

#
# threshold = 0.10
#
# outlier_yok = outlier_filtrele(df_diabet, 'bmi', threshold)
```

```
#  
=====  
=====  
  
from scipy import stats  
import numpy as np  
  
z_degerleri = np.abs(stats.zscore(df_diabet['bmi']))  
print(z_degerleri)  
  
threshold_z_skor = 2  
  
outlier_indisler = np.where(z_degerleri > threshold_z_skor)[0]  
outlier_filtrelendi = df_diabet.drop(outlier_indisler)  
  
print("Z-skor öncesi:", df_diabet.shape)  
print("Z-skor ile filtreleme sonrası:", outlier_filtrelendi.shape)
```



3. ADIM: ÜRET

Öğrenciler kodlanan Python uygulaması üzerinde IQR yöntemi (Frost, 2024; Smiti, 2020) ile filtreleme gerçekleştirir.

Şeker Hastalığı Veri Setinde IQR Yöntemi Uygulaması

Öğrenciler önceki adımda dikkate aldıkları şeker hastalığı veri setinde IQR yöntemini uygulamak için ilgili kütüphane çağrılarını yapar, veri setinin IQR öncesi shape durumunu ve IQR'ye yönelik değerleri ekranaya yazdırır (Şekil 104).

```

7  from sklearn.datasets import load_diabetes
8  import pandas as pd
9  import numpy as np
10
11 diabet = load_diabetes()
12 sutun_isimleri = diabet.feature_names
13 df_diabet = pd.DataFrame(diabet.data)
14 df_diabet.columns = sutun_isimleri
15
16 print("IQR Öncesi Shape:", df_diabet.shape)
17
18 #Tespit aşaması
19 Q1 = df_diabet[ 'bmi' ].quantile(0.25)
20 Q3 = df_diabet[ 'bmi' ].quantile(0.75)
21
22 IQR = Q3 - Q1
23 print("IQR:",IQR)
24 L = Q1 - 1.5*IQR
25 U = Q3 + 1.5*IQR
26 print("Lower: {0} ve Upper: {1}".format(L,U) )

```

(a)

```

IQR Öncesi Shape: (442, 10)
IQR: 0.06547708348825859
Lower: -0.13244469328909578 ve Upper: 0.1294636406639386

```

(b)

Şekil 104. IQR ile filtreleme için ön hazırlık ve IQR değerlerinin görüntülenmesi (a) kod
(b) sonuç konsol görüntüleri.

IQR değerlerinin tespiti ardından yapılması gereken veri seti içerisinde ilgili değer aralıkları dışında kalan verilere göre filtreleme işleminin yapılmasıdır. Bu amaç doğrultusunda veri seti dataframe değişkeninden outlier durumundaki verilerin silinmesi sağlanır ve sonuç veri setinin shape durumunun değiştiği gözlemlenir (Şekil 105).

```

31 #Filtreleme aşaması
32 df_diabet.drop(index=Upper_dizisi, inplace=True)
33 df_diabet.drop(index=Lower_dizisi, inplace=True)
34
35 print("IQR Sonrası Shape", df_diabet.shape)

```

(a)

```

IQR Sonrası Shape (439, 10)

```

(b)

Şekil 105. IQR ile filtreleme süreçleri (a) kod (b) filtreleme sonrası shape durumu.

UYGULAMANIN PYTHON KODLARI

```
from sklearn.datasets import load_diabetes
import pandas as pd
import numpy as np

diabet = load_diabetes()
sutun_isimleri = diabet.feature_names
df_diabet = pd.DataFrame(diabet.data)
df_diabet.columns = sutun_isimleri

print("IQR Öncesi Shape:", df_diabet.shape)

#Tespit aşaması
Q1 = df_diabet['bmi'].quantile(0.25)
Q3 = df_diabet['bmi'].quantile(0.75)

IQR = Q3 - Q1
print("IQR:", IQR)
L = Q1 - 1.5*IQR
U = Q3 + 1.5*IQR
print("Lower: {0} ve Upper: {1}".format(L,U) )

Upper_dizisi = np.where(df_diabet['bmi'] >= U)[0]
Lower_dizisi = np.where(df_diabet['bmi'] <= L)[0]

#Filtreleme aşaması
df_diabet.drop(index=Upper_dizisi, inplace=True)
df_diabet.drop(index=Lower_dizisi, inplace=True)

print("IQR Sonrası Shape", df_diabet.shape)
```



4. ADIM: İLERLET

Bu adımda öğrencilerin outlier (aykırı değer) tespit ve filtreleme işlemlerini farklı bir veri seti üzerinde uygulamaları sağlanır ve ardından uygulama üzerinde güncelleştirmeler yapmaları istenerek elde ettikleri sonuçlar hakkında notlar almaları istenir.

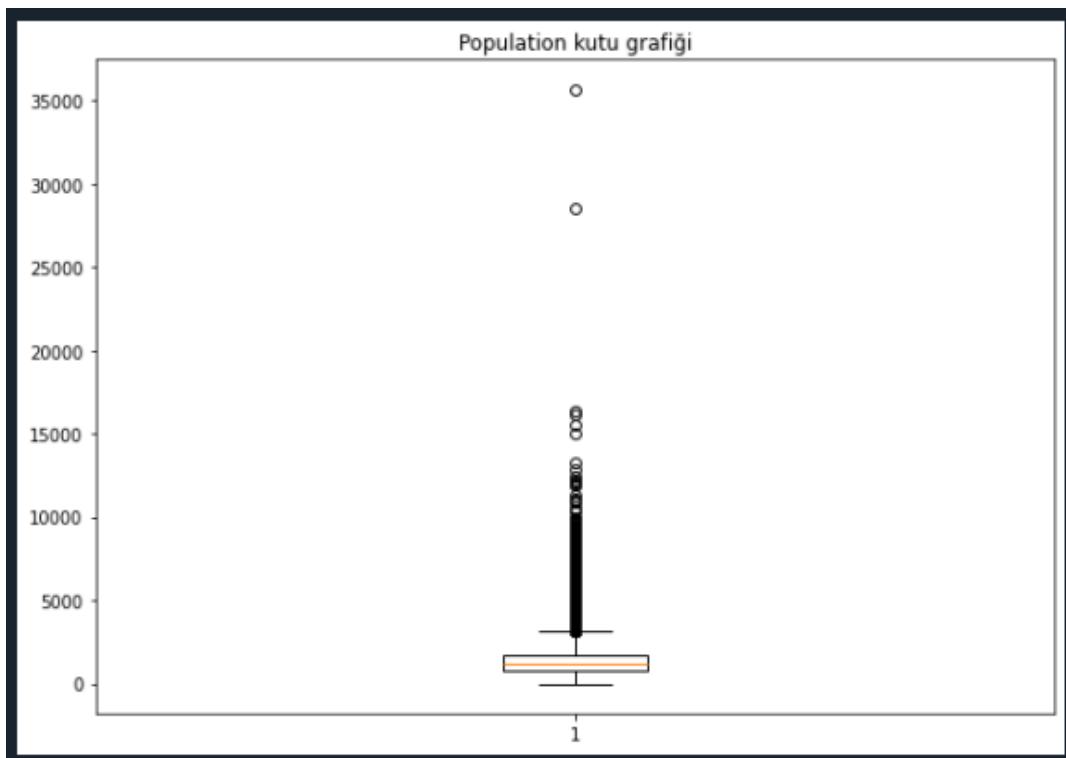
Farklı Bir Veri Seti Üzerinde Z-Skoru ve IQR Yöntemlerinin Uygulanması

Öğrencilere önceki adımlarda kullandıkları şeker hastalığı veri seti dışında, sklearn kütüphanesi ile gelen alternatif veri setleri içerisinde yer alan **california housing** veri seti üzerinde Z-skoru ve IQR yöntemlerini kullanarak outlier süreçlerini işletmeleri sağlanır. California Housing veri seti 8 girdi parametresine göre ev fiyatlarını tahmin etmeye odaklanan bir regresyon veri setidir. Öğrencilere veri setine dair daha fazla bilgi vermek için https://scikit-learn.org/stable/modules/generated/sklearn.datasets.fetch_california_housing.html adresi ziyaret edilir.

Uygulama içerisinde ilgili veri setinde outlier tespiti için “**Population**” parametresi dikkate alınır. Bu amaçla önceki adımlarda kodlanan uygulamalardan da esinlenerek ilgili kütüphanelerin ve veri setinin çağrılması sağlanır. Ardından kutu grafiği ile outlier değerler gözlemlenir (Şekil 106).

```
7   from sklearn.datasets import fetch_california_housing
8   import pandas as pd
9   import matplotlib.pyplot as plt
10  from scipy import stats
11  import numpy as np
12
13  housing = fetch_california_housing()
14  sutun_isimleri = housing.feature_names
15  df_housing = pd.DataFrame(housing.data)
16  df_housing.columns = sutun_isimleri
17
18  fig = plt.figure(figsize =(10, 7))
19  plt.boxplot(df_housing[ 'Population' ])
20  plt.title("Population kutu grafiği")
21  plt.show()
```

(a)



(b)

Şekil 106. California Housing veri seti için ön hazırlıklar ve outlier kutu grafiği (a) kod
(b) elde edilen kutu grafiği.

Kutu grafiği ile görülen outlier yoğunluğuna dikkat çekilir. Filtreleme işlemleri için Z-skoru ve IQR yöntemleri ardı ardına çalıştırılarak elde edilen bulgular tartışılır (Şekil 107).

```

23 print("Z-skoru ile tespit ve filtreleme:")
24 z_degerleri = np.abs(stats.zscore(df_housing[ 'Population' ]))
25 print(z_degerleri)
26 threshold_z_skor = 2
27 outlier_indisler = np.where(z_degerleri > threshold_z_skor)[0]
28 outlier_filtrelendi = df_housing.drop(outlier_indisler)
29 print("Z-skor öncesi:", df_housing.shape)
30 print("Z-skor ile filtreleme sonrası:", outlier_filtrelendi.shape)
31
32 print("IQR ile tespit ve filtreleme:")
33 print("IQR Öncesi Shape:", df_housing.shape)
34 Q1 = df_housing[ 'Population' ].quantile(0.25)
35 Q3 = df_housing[ 'Population' ].quantile(0.75)
36 IQR = Q3 - Q1
37 print("IQR:", IQR)
38 L = Q1 - 1.5*IQR
39 U = Q3 + 1.5*IQR
40 print("Lower: {} ve Upper: {}".format(L,U) )
41 Upper_dizisi = np.where(df_housing[ 'Population' ] >= U)[0]
42 Lower_dizisi = np.where(df_housing[ 'Population' ] <= L)[0]
43 df_housing.drop(index=Upper_dizisi, inplace=True)
44 df_housing.drop(index=Lower_dizisi, inplace=True)
45 print("IQR Sonrası Shape", df_housing.shape)

```

(a)

```

Z-skoru ile tespit ve filtreleme:
0      0.974429
1      0.861439
2      0.820777
3      0.766028
4      0.759847
...
20635   0.512592
20636   0.944405
20637   0.369537
20638   0.604429
20639   0.033977
Name: Population, Length: 20640, dtype: float64
Z-skor öncesi: (20640, 8)
Z-skor ile filtreleme sonrası: (19884, 8)
IQR ile tespit ve filtreleme:
IQR Öncesi Shape: (20640, 8)
IQR: 938.0
Lower: -620.0 ve Upper: 3132.0
IQR Sonrası Shape (19442, 8)

```

(b)

Şekil 107. California Housing veri setine Z-skoru ve IQR yöntemlerinin uygulanması
(a) kodlar (b) elde edilen konsol çıktıları.

Öğrenciler bu noktada farklı girdi parametreleri ve threshold değerleri kullanarak elde edilen çıktıları gözlemlayabilir. Bu amaçla öğrencilerin gruplar halinde çalışması için 10 dakikalık bir süre ayrılr.

UYGULAMANIN PYTHON KODLARI

```
from sklearn.datasets import fetch_california_housing
import pandas as pd
import matplotlib.pyplot as plt
from scipy import stats
import numpy as np

housing = fetch_california_housing()
sutun_isimleri = housing.feature_names
df_housing = pd.DataFrame(housing.data)
df_housing.columns = sutun_isimleri

fig = plt.figure(figsize =(10, 7))
plt.boxplot(df_housing['Population'])
plt.title("Population kutu grafiği")
plt.show()

print("Z-skoru ile tespit ve filtreleme:")
z_degerleri = np.abs(stats.zscore(df_housing['Population']))
print(z_degerleri)
threshold_z_skor = 2
outlier_indisler = np.where(z_degerleri > threshold_z_skor)[0]
outlier_filtrelendi = df_housing.drop(outlier_indisler)
print("Z-skor öncesi:", df_housing.shape)
print("Z-skor ile filtreleme sonrası:", outlier_filtrelendi.shape)

print("IQR ile tespit ve filtreleme:")
print("IQR Öncesi Shape:", df_housing.shape)
Q1 = df_housing['Population'].quantile(0.25)
Q3 = df_housing['Population'].quantile(0.75)
IQR = Q3 - Q1
```

```
print("IQR:",IQR)
L = Q1 - 1.5*IQR
U = Q3 + 1.5*IQR
print("Lower: {0} ve Upper: {1}".format(L,U) )
Upper_dizisi = np.where(df_housing['Population'] >= U)[0]
Lower_dizisi = np.where(df_housing['Population'] <= L)[0]
df_housing.drop(index=Upper_dizisi, inplace=True)
df_housing.drop(index=Lower_dizisi, inplace=True)
print("IQR Sonrası Shape", df_housing.shape)
```



5. ADIM: DEĞERLENDİRME

Önceki adımda güncelleştirmeler sonrası alınan notlar ve derste öğrenilen bilgiler ışığında soru-cevap etkinliği gerçekleştirilerek Python ile outlier (aykırı değer) tespiti ve filtreleme bilgi-becerileri pekiştirilir.

Öğrencilere şu sorular sorulabilir:

- Gerçekleştirilen outlier (aykırı değer) tespit ve filtreleme uygulamalarından elde ettiğiniz bilgi ve beceriler neler olmuştur?
- Kullanılan outlier (aykırı değer) tespit yöntemleri hakkında görüşleriniz nelerdir?
- Son uygulamada yaptığınız güncellemeler ardından ne gibi farklı sonuçlar elde ettiniz?
- Outlier (aykırı değer) tespit ve filtreleme yaklaşımlarının Yapay Zeka uygulamalarında oluşturacağı avantajlar ve dezavantajlar neler olabilir?

2. HAFTA – 2. GÜN – 1., 2., 3. DERSLER: BAYES ÖĞRENMESİ İLE ARAÇ SATIŞ DURUMUNU TAHMİN ETME

DERS PLANI

DERS ETİKETLERİ



KONU: Bayes Öğrenmesi ile Araç Satış Durumunu Tahmin Etme



SINIF: 12-14 Yaş



SÜRE: 180 dk.



HAZIR BULUNUŞLUK: İngilizce eğitimi, Yapay Zeka kavramı ve uygulamaları hakkında temel bilgi, Makine Öğrenmesi hakkında temel bilgi, Python programlama dili hakkında temel bilgi



ANAHTAR KELİMELER: Yapay Zeka, Algoritma, Kodlama, Python, Bayes Öğrenmesi, Naive Bayes Algoritması



BECERİLER:



Tahmin



Sorgulama



Grup çalışması



Günlük hayat bağlantısı kurma



KAZANIMLAR:

Bu derste öğrenciler;

- Bayes Öğrenmesi algoritması hakkında bilgi ve beceri sahibi olurlar.
- Python ile Bayes Öğrenmesi üzerinden sınıflandırmaya dayanan bir Makine Öğrenmesi uygulamasını gerçekleştirirler.
- Python ortamında araç satışı ile bağlantılı bir veri seti üzerinden sınıflandırmaya dayalı bir Makine Öğrenmesi çözümünün nasıl gerçekleştirilebileceği hakkında bilgi ve beceri sahibi olurlar.



DERS MATERİYALLERİ:

Malzeme	Miktar
Yazı Tahtası	Öğrenci sayısı ve sınıf büyüklüğüne göre en az bir adet
Projeksiyon	Öğrenci sayısı ve sınıf büyüklüğüne göre en az bir adet
Defter, Kalem	Öğrenci sayısı kadar
Bilgisayar	Öğrenci sayısının yarısı kadar
Tablet	Öğrenci sayısının yarısı kadar
Python Anaconda Framework	Her bilgisayarda kurulu olacak şekilde



ÖĞRETMEN HAZIRLIK ÇALIŞMALARI:

Okunabilecek Kaynaklar:

Kaggle. (2020). *Naive Bayes Classifier in Python*. Çevrimiçi: <https://www.kaggle.com/code/prashant111/naive-bayes-classifier-in-python> (Erişim 30 Mayıs 2024).

Kaggle. (2022). *Classification With Naive Bayes Classification*. Çevrimiçi: <https://www.kaggle.com/code/ihscnkz/classification-with-naive-bayes-classification> (Erişim 30 Mayıs 2024).

İzlenebilecek Kaynaklar:

Normalized Nerd. (2021). The Math Behind Bayesian Classifiers Clearly Explained!. Çevrimiçi: <https://www.youtube.com/watch?v=lFJbZ6LVxN8> (Erişim 30 Mayıs 2024).

ATÖLYE HAZIRLIK ÇALIŞMALARI:

Öğretmen;

- Dersten önce ders materyallerini hazırlamalıdır.
- Python kodlama için kullanılan Spyder editörünün genel kullanımına aşina olmalıdır.
- Darse konu olan veri set(ler)ini ve Python kodlarını önceden inceleyerek üzerinde çalışmış olmalıdır.



KAYNAKÇA:

Flach, P. A., & Lachiche, N. (2004). Naive Bayesian classification of structured data. *Machine learning*, 57, 233-269.

Köse, U., Özsoy, K., & Aksoy, B. (2023). *Yapay Zeka – Ortaokul ve Lise Kitapları*. Deneyap Teknoloji Atölyeleri, TÜBİTAK Yayıncılık.

Martin, O. (2016). *Bayesian analysis with python*. Packt Publishing Ltd.

GENEL BAKIŞ:

- 1) Harekete Geç:** Öğrencilere uygulamaya konu Bayes öğrenme algoritması anlatılır ve Python kodlama ile Makine Öğrenmesi sürecine yönelik ön hazırlık gerçekleştirilir.
- 2) Keşfet:** Öğrencilerle birlikte ilgili Makine Öğrenmesi modelinin çalıştırılmasına dair kodlamaların yapılması sağlanır.
- 3) Üret:** Öğrencilerle birlikte çalıştırılan Makine Öğrenmesi modelinin elde ettiği sonuçların değerlendirilmesine yönelik kodlamaların yapılması sağlanır.
- 4) İlerlet:** Öğrencilerin gruplar halinde çalışarak, gerçekleştirilen uygulama üzerinde kendi düzenlemelerini yapmaları ve elde edilen sonuçlar hakkında notlar almaları istenir.
- 5) Değerlendir:** Derste edinilen bilgiler soru-cevap aşaması gerçekleştirilmek suretiyle pekiştirilir.

UYGULAMA



1. ADIM: HAREKETE GEÇ

Harekete Geç admında öğrencilere uygulamaya konu Bayes öğrenme algoritması anlatılır ve Python kodlama ile Makine Öğrenmesi sürecine yönelik ön hazırlık gerçekleştirilir.

Naive Bayes Öğrenmesi Algoritması / Tekniği

Naive Bayes öğrenme tekniği temeli Bayes Teoremi’ne dayanır. Bayes Teoremi bir sonucun sebebini bulurken sonucun hangi olasılıkla hangi sebepten kaynaklandığını bulur. Bu amaçla olasılıksal bir yaklaşım izlenir ve farklı girdi parametreleri karşısında çıktıların olasılıksal gerçekleşme değerleri tespit edilir (Flach, & Lachiche, 2004; Köse vd., 2023; Martin, 2016).

Örneğin, ülkemizde en popüler alışveriş internet sitesinde akıllı cep telefonunu 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:

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

Bir gün içinde satılan cep telefonunun A marka olması halinde iade edilme olasılığını hesaplayalım. Toplam 360 adet cep telefonuna göre günlük 18 adet cep telefonu (%5) iade

edilmektedir. İade edilen cep telefonlarının markalarına göre dağılımını eşit kabul edersek iade edilen cep telefonlarının 9 adeti A markasına aittir. O halde A marka cep telefonunun iade edilme olasılığı $9 / 200 = \%4,5$ 'tur. Bu duruma yönelik Bayes öğrenme formülü şöyledir:

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

Formüle sayısal veriler uyarlandığında şu sonuç elde edilir:

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

Öğrenciler, ilk olarak Bayes öğrenme ile araç satış değerlendirilme problemini anlar. Bu bağlamda Tablo 2'de verilen giriş çıkış parametreleri ve örnek veriler irdelenir.

Tablo 2. Bayes öğrenme için giriş ve çıkış parametreleri ve örnek veriler.

Veri Sayısı	GİRİŞ PARAMETRELERİ						ÇIKIŞ PARAMETRESİ
	Fiyat	Onarım	Kapı	Kişi	Bagaj Hacmi	Güvenlik	
1	çok yüksek	çok yüksek	2	2	küçük	düşük	zor
2	çok yüksek	çok yüksek	2	2	küçük	orta	zor
3	çok yüksek	çok yüksek	2	2	küçük	yüksek	zor
4	çok yüksek	çok yüksek	2	2	orta	düşük	zor
5	çok yüksek	çok yüksek	2	2	orta	orta	zor
6	çok yüksek	çok yüksek	2	2	orta	yüksek	zor
7	çok yüksek	çok yüksek	2	2	büyük	düşük	zor
8	çok yüksek	çok yüksek	2	2	büyük	orta	zor
9	çok yüksek	çok yüksek	2	2	büyük	yüksek	zor
10	çok yüksek	çok yüksek	2	4	küçük	düşük	zor
11	çok yüksek	çok yüksek	2	4	küçük	orta	zor
12	çok yüksek	çok yüksek	2	4	küçük	yüksek	zor
13	çok yüksek	çok yüksek	2	4	orta	düşük	zor
14	çok yüksek	çok yüksek	2	4	orta	orta	zor
15	çok yüksek	çok yüksek	2	4	orta	yüksek	zor
16	çok yüksek	çok yüksek	2	4	büyük	düşük	zor
17	çok yüksek	çok yüksek	2	4	büyük	orta	Zor
...
1729	düşük	düşük	5 den fazla	4 den fazla	büyük	yüksek	çok kolay

Uygulama kapsamında araçlara ait 1729 adet veri setinde fiyat, onarım, kapı, kişi sayısı, bagaj hacmi ve güvenlik giriş parametrelerine göre aracın satış durumunu makine öğrenmesi ile tahminlenmesi amaçlanmıştır.

VERİ SETİ İÇİN İNDİRME LİNKİ

<https://github.com/deneyapyz/ortaokul/raw/main/Hafta3/car.csv>



2. ADIM: KEŞFET

Bu adımda öğrencilerle birlikte ilgili Makine Öğrenmesi modelinin çalıştırılmasına dair kodlamaların yapılması sağlanır.

Öğrenci, verilen car.csv isimli veri setini dosyasını kullanarak bir Bayes öğrenme tahminleme işlemi yapar. Tasarım aşamasında aracın fiyat, onarım, kapı, kişi sayısı, bagaj hacmi ve güvenlik parametrelerine göre aracın satış durumunu tahmin etmeye çalışır. Şekil 108'de gösterildiği gibi “car.csv” dosyasını yüklemek için gerekli kütüphane, komutlar ve veri_copy fonksiyonunu yazar.

Öğrenciler Şekil 108'de görüldüğü gibi veri setinde yer alan giriş ve çıkış parametrelerinin metinsel değerlerini sayısallaştırılmak için Sklearn kütüphanesinde yer alan “ön işleme komutunu (preprocessing)” çağrıarak “**preprocessing.LabelEncoder()**” fonksiyonunu çalıştırır. Böylece öğrenciler, “**sayisallastırma**” isimli değişken oluşturmuş olur.

The screenshot shows the Spyder Python 3.8 IDE interface. The code editor window displays the following Python script:

```
1 import numpy as np
2 import pandas as pd
3 veri =pd.read_csv("car.csv",encoding='unicode_escape')
4 veri_copy=veri.copy()
5
6 from sklearn import preprocessing
7 sayisallastirma = preprocessing.LabelEncoder()
```

The line "from sklearn import preprocessing" and "sayisallastirma = preprocessing.LabelEncoder()" are highlighted with a red rectangle.

Şekil 108. Sayısallaştmak için kütüphane ve fonksiyon çağrıımı kod ekranı.

Öğrenciler veri setinde yer alan fiyat, onarım, kapı sayısı, kişi sayısı, bagaj boyutu, güvenlik ve satış parametrelerini değerleri olan iyi, normal, kolay, zor, çok kolay gibi metinsel ifadelerini bayes algoritması ile eğitim yapmadan önce Sklearn kütüphanesinde yer alan “**ön işleme (preprocessing)**” yöntemi kullanarak “0”, “1”, “2”, “3” gibi sayısal değerlere dönüştürür. Öğrenciler Şekil 109'da görüldüğü gibi **sayisallastirma.fit_transform** (**veri_copy[“giriş/çıkış parametresi”]**) komutu kullanarak verileri sayısallaştırır. Örneğin fiyat değeri “düşük, orta, yüksek” ve “çok yüksek” metinsel ifadeleri “0”, “1”, “2” ve “3” şeklinde sayısallaştırır.

```

1 import numpy as np
2 import pandas as pd
3 veri =pd.read_csv("car.csv",encoding='unicode_escape')
4 veri_copy=veri.copy()
5
6 from sklearn import preprocessing
7 sayisallastirma = preprocessing.LabelEncoder()
8
9 veri_copy["fiyat"] =sayisallastirma.fit_transform(veri_copy["fiyat"])
10 veri_copy["onarim"] =sayisallastirma.fit_transform(veri_copy["onarim"])
11 veri_copy["kapi sayisi"] =sayisallastirma.fit_transform(veri_copy["kapi sayisi"])
12 veri_copy["kisi sayisi"] =sayisallastirma.fit_transform(veri_copy["kisi sayisi"])
13 veri_copy["bagaj boyutu"] =sayisallastirma.fit_transform(veri_copy["bagaj boyutu"])
14 veri_copy["Guvenlik"] =sayisallastirma.fit_transform(veri_copy["Guvenlik"])
15 veri_copy["satis "] =sayisallastirma.fit_transform(veri_copy["satis "])
16

```

Şekil 109. Veri setindeki metin değerlerinin sayısallaştırılması kod ekranı.

Öğrenciler Şekil 110'da gösterilen ilk komut satırında numpy kütüphanesindeki dizi (array) özelliği kullanarak veri çerçevesi (data frame) formatındaki “car.csv” dosyasındaki son parametre olan “**satis**” parametresi hariç geriye kalan tüm parametrelerini “**veri_copy.drop**” komutu ile giriş parametresi olarak ayarlar. Burada yer alan “axis=1” ifadesinin anlamı satis parametresindeki verileri sütun silme işlemi yapar.

```

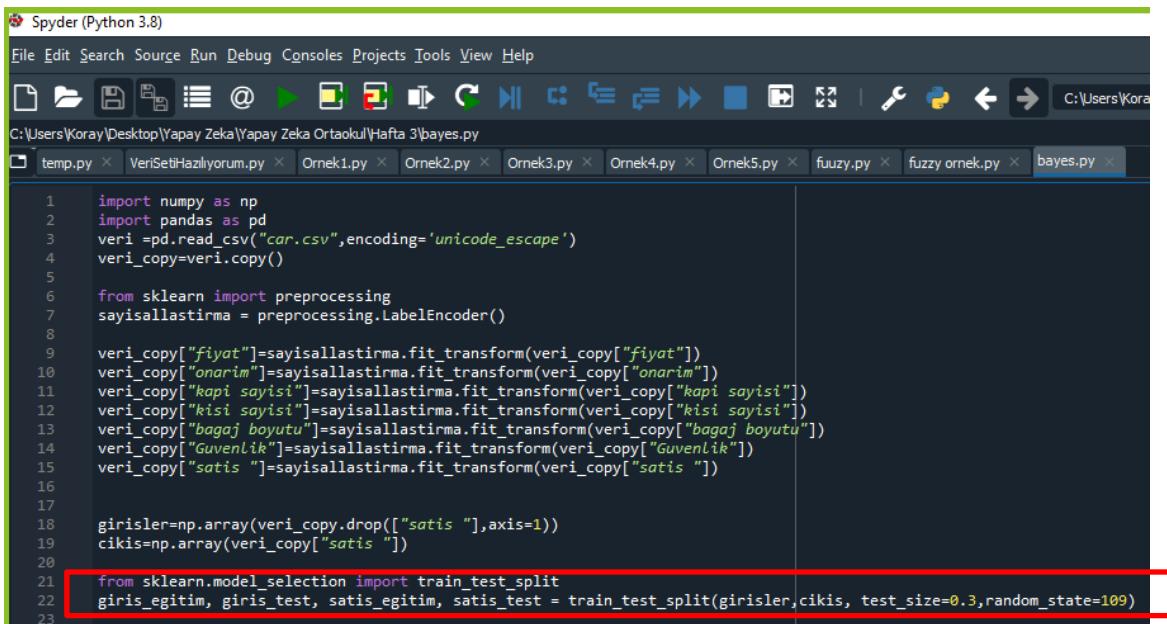
1 import numpy as np
2 import pandas as pd
3 veri =pd.read_csv("car.csv",encoding='unicode_escape')
4 veri_copy=veri.copy()
5
6 from sklearn import preprocessing
7 sayisallastirma = preprocessing.LabelEncoder()
8
9 veri_copy["fiyat"] =sayisallastirma.fit_transform(veri_copy["fiyat"])
10 veri_copy["onarim"] =sayisallastirma.fit_transform(veri_copy["onarim"])
11 veri_copy["kapi sayisi"] =sayisallastirma.fit_transform(veri_copy["kapi sayisi"])
12 veri_copy["kisi sayisi"] =sayisallastirma.fit_transform(veri_copy["kisi sayisi"])
13 veri_copy["bagaj boyutu"] =sayisallastirma.fit_transform(veri_copy["bagaj boyutu"])
14 veri_copy["Guvenlik"] =sayisallastirma.fit_transform(veri_copy["Guvenlik"])
15 veri_copy["satis "] =sayisallastirma.fit_transform(veri_copy["satis "])
16
17
18 girisler=np.array(veri_copy.drop(["satis "],axis=1))
19 cikis=np.array(veri_copy["satis "])

```

Şekil 110. Veri setindeki metin değerlerinin sayısallaştırılması kod ekranı.

Öğrenciler ilk satırda, veri seti üzerindeki düzenlemeleri tamamladıktan sonra veri setini eğitim ve test olarak ayırmak için “**sklearn.model_selection**” kütüphanesinde yer alan “**train_test_split**” kütüphanesini çağırır. İkinci satırda ise, veri setindeki eğitim ve test verilerini %70 eğitim, %30 test olacak şekilde “**test_size=0,3**” komutu kullanarak rastgele

ayırır. Burada “**random_state=109**” ifadesi her eğitim tekrarında alınacak verilerin aynı kalmasını sağlar (Şekil 111).



The screenshot shows the Spyder Python 3.8 IDE interface. The title bar says "Spyder (Python 3.8)". The menu bar includes File, Edit, Search, Source, Run, Debug, Consoles, Projects, Tools, View, Help. The toolbar has various icons for file operations like Open, Save, Copy, Paste, Find, etc. The current working directory is "C:\Users\Koray\Desktop\Yapay Zeka\Yapay Zeka Ortaokul\Hafta 3\bayes.py". Below the toolbar is a tab bar with multiple files: temp.py, VeriSetiHazirlıyorum.py, Ornek1.py, Ornek2.py, Ornek3.py, Ornek4.py, Ornek5.py, fuzy.py, fuzzy ornek.py, and bayes.py. The main code editor area contains Python code for data preprocessing and splitting. A red box highlights the last two lines of code:

```
1 import numpy as np
2 import pandas as pd
3 veri =pd.read_csv("car.csv",encoding='unicode_escape')
4 veri_copy=veri.copy()
5
6 from sklearn import preprocessing
7 sayisallastirma = preprocessing.LabelEncoder()
8
9 veri_copy["fiyat"] =sayisallastirma.fit_transform(veri_copy["fiyat"])
10 veri_copy["onarim"] =sayisallastirma.fit_transform(veri_copy["onarim"])
11 veri_copy["kapi sayisi"] =sayisallastirma.fit_transform(veri_copy["kapi sayisi"])
12 veri_copy["kisi sayisi"] =sayisallastirma.fit_transform(veri_copy["kisi sayisi"])
13 veri_copy["bagaj boyutu"] =sayisallastirma.fit_transform(veri_copy["bagaj boyutu"])
14 veri_copy["Guvenlik"] =sayisallastirma.fit_transform(veri_copy["Guvenlik"])
15 veri_copy["satis "] =sayisallastirma.fit_transform(veri_copy["satis "])
16
17
18 girisler=np.array(veri_copy.drop(["satis "],axis=1))
19 cikis=np.array(veri_copy["satis "])
20
21 from sklearn.model_selection import train_test_split
22 giris_egitim, giris_test, satis_egitim, satis_test = train_test_split(girisler,cikis, test_size=0.3,random_state=109)
```

Şekil 111. Veri setinin eğitim ve test olarak rastgele ayrılması için kod ekranı

Öğrenciler ilk satırda aracın özelliklerine göre satış durumu için oluşturmuş olduğu Bayes öğrenme algoritmasını kullanabilmek için “**sklearn.naive_bayes**” kütüphanesinde yer alan “**CategoricalNB**” özellik fonksiyonunu çağırır. İkinci satırda **CategoricalNB()** fonksiyonu “**model**” isimli bir değişkene aktarılır. Üçüncü satırda, “**model.fit**” komutu yazarak giriş eğitim verilerine göre çıkış eğitim verileri için Bayes algoritması ile eğitim gerçekleştirir. Son kod satırında ise, Bayes algoritması ile eğitim işlemi sonunda giriş test verilerine göre satış tahminini gerçekleştirilir (Şekil 112).

The screenshot shows the Spyder Python 3.8 IDE interface. The menu bar includes File, Edit, Search, Source, Run, Debug, Consoles, Projects, Tools, View, and Help. The toolbar has various icons for file operations like Open, Save, and Run. The current file is 'bayes.py' located at 'C:\Users\Koray\Desktop\Yapay Zeka\Yapay Zeka Ortaokul\Hafta 3\bayes.py'. The code editor contains the following Python code:

```
7     sayisallastirma = preprocessing.LabelEncoder()
8
9     veri_copy["fiyat"] = sayisallastirma.fit_transform(veri_copy["fiyat"])
10    veri_copy["onarim"] = sayisallastirma.fit_transform(veri_copy["onarim"])
11    veri_copy["kapi_sayisi"] = sayisallastirma.fit_transform(veri_copy["kapi sayisi"])
12    veri_copy["kisi_sayisi"] = sayisallastirma.fit_transform(veri_copy["kisi sayisi"])
13    veri_copy["bagaj_boyutu"] = sayisallastirma.fit_transform(veri_copy["bagaj boyutu"])
14    veri_copy["Guvenlik"] = sayisallastirma.fit_transform(veri_copy["Guvenlik"])
15    veri_copy["satis"] = sayisallastirma.fit_transform(veri_copy["satis"])
16
17
18    girisler = np.array(veri_copy.drop(["satis"], axis=1))
19    cikis = np.array(veri_copy["satis"])
20
21    from sklearn.model_selection import train_test_split
22    giris_egitim, giris_test, satis_egitim, satis_test = train_test_split(girisler, cikis,
23
24
25    from sklearn.naive_bayes import CategoricalNB
26    model = CategoricalNB()
27    model.fit(giris_egitim, satis_egitim)
28    satis_tahmin = model.predict(giris_test)
```

Şekil 112. Bayes algoritmasında eğitim ve tahminleme işlemleri için kod ekranı



3. ADIM: ÜRET

Öğrencilerle birlikte çalıştırılan Makine Öğrenmesi modelinin elde ettiği sonuçların değerlendirilmesine yönelik kodlamaların yapılması sağlanır.

Tahmin-Test Sonuçlarını Karşılaştırma

Bu aşamada, önceki aşamadaki uygulamaya eklemeler devam etmektedir. Öğrenciler, Bayes öğrenme modeli ile eğitilen araç satış durumuna tahminlemesine ait hata matrisi (confusion matrix) kullanarak modelin başarısını ölçer. Şekil 113'te gösterildiği gibi 31 numaralı komut satırında, `sklearn.metrics` kütüphanesinden `confusion_matrix` özelliği yüklenir. 32 numaralı satırda ise, hata matrisini görüntüleyebilmek için `seaborn` kütüphanesini yükler. 33 numaralı satırda ise, hata matrisindeki grafikleri çizebilmek için `matplotlib` kütüphanesi içerisinde yer alan `pyplot` fonksiyonunu yükler. 34 numaralı kod satırında ise, `cm` değişkeni ile oluşturulan hata matrisinin satır (`satis_test`) ve sütunları (`satis_tahmin`) oluşturur. 35 ve 36 numaralı kod satırlarında ise `index` ve `columns` değişkenleri kullanarak hata matrisinde yer alacak olan metinleri belirler. 37 numaralı kod satırında `cm_df` değişkeni ile `satis_test` ve `satis_tahmin` değerlerini veri çerçevesine (`DataFrame`) aktarılmasını sağlar. 38 numaralı kod satırında ise `"plt.figure"` komutu ile 10x6 cm çerçeve boyutunda boş bir çizim ekranı açar. 39 numaralı kod satırında ise `"sns.heatmap"` komutu ile oluşturulan veri çerçevesini renkli olarak çizer. Burada `annot=True` ile sayısal değerler gösterilirken, `fmt="d"` ile sayısal değerler tam sayı olarak gösterir.

```

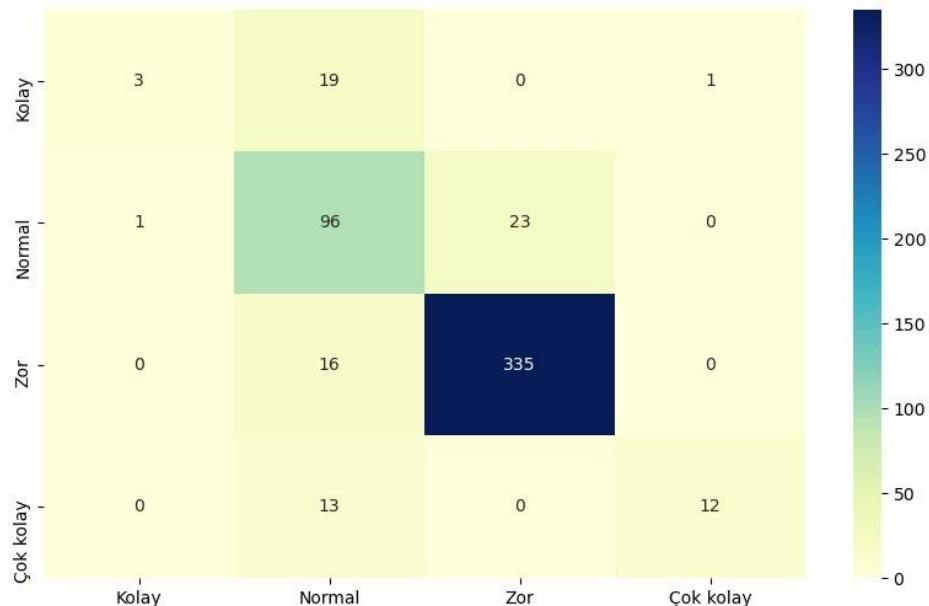
22     giris_egitim, giris_test, satis_egitim, satis_test = train_test_split(giris, satis, test_size=0.3, random_state=42)
23
24
25     from sklearn.naive_bayes import CategoricalNB
26     model = CategoricalNB()
27     model.fit(giris_egitim, satis_egitim)
28     satis_tahmin = model.predict(giris_test)
29
30
31     from sklearn.metrics import confusion_matrix
32     import seaborn as sns
33     import matplotlib.pyplot as plt
34     cm = confusion_matrix(satis_test, satis_tahmin)
35     index = ['Kolay', 'Normal', 'Zor', 'Çok kolay']
36     columns = ['Kolay', 'Normal', 'Zor', 'Çok kolay']
37     cm_df = pd.DataFrame(cm, columns, index)
38     plt.figure(figsize=(10, 6))
39     sns.heatmap(cm_df, annot=True, fmt="d")

```

Şekil 113. Bayes algoritmasında eğitim ve tahminleme işlemleri için kod ekranı.

Hata Matrisini Değerlendirme

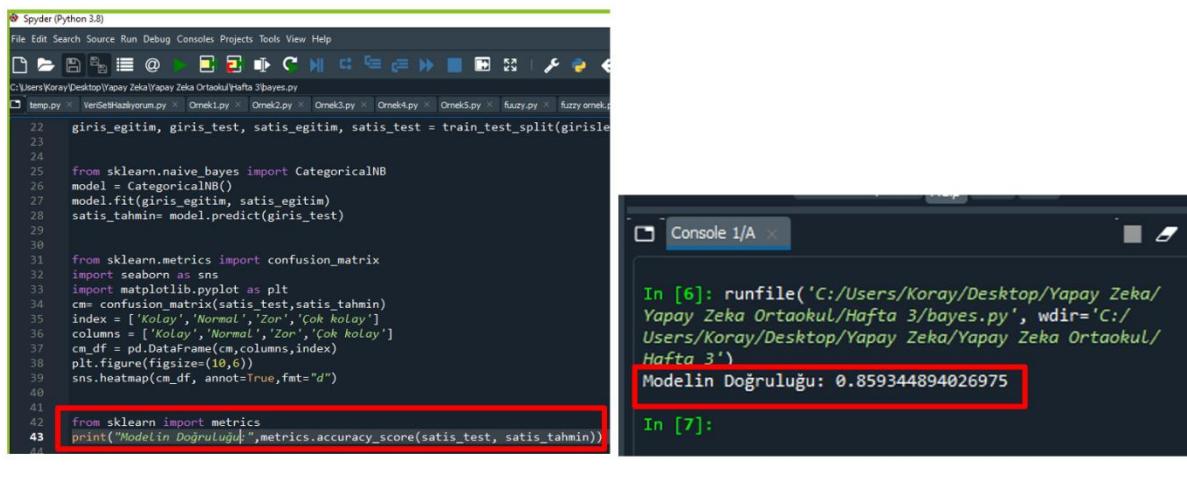
Öğrenciler, araç satış durumuna ait toplam 1729 adet veri setinde kayıt verinin %70'ini (1210) eğitim, %30'ı (519) test eğitim seti olarak ayırmıştı. Bayes modeli 1210 kayıt ile eğitti. Geriye kalan 519 kayıt (giris_test) ise modeli tahmin etmek için kullandı (satis_tahmin). Öğrenciler, Şekil 114'te gösterildiği gibi 519 test kaydına ait gerçek sonuçlar (satis_test) ile tahmin sonuçlarını karşılaştırdı ve hata matrisi elde etti. Böylece öğrenci, matristeki sayılar toplamının test verisi olan 519'a eşit olduğunu görmektedir.



Şekil 114. Hata matrisi.

Öğrenciler, hata matrisi incelendiğinde 519 adet test kaydında araç satış durumuna ait 4 tane kolay test verisinden 3 adetini doğru (kolay), 1 adetini ise yanlış (normal) tahmin edildiğini görmüştür. 144 adet normal test verisinden 96 adetini doğru (normal), 19 adeti yanlış (kolay), 16 adet yanlış (zor), 13 adet yanlış (çok kolay) olmak üzere toplam 48 adetini yanlış tahminlendiğini görür. 358 adet zor test verisinden 335 adetini doğru (zor) ve 23 adeti yanlış (normal) tahminlendiğini görür. Son olarak 13 adet çok kolay test verisinden 12 adeti doğru (çok kolay) ve 1 adeti yanlış (kolay) olarak tahmin ettiğini görür.

Öğrenciler Şekil 115'te gösterildiği gibi 42 numaralı kod satırında bayes öğrenme modeli ile eğitilen modelin doğruluğunu değerlendirmek için “sklearn” kütüphanesinden “metrics” fonksiyon özelliğini yükler. 43 numaralı kod satırında “print” komutu kullanarak “satis_test” veri setindeki veriler ile “satis_tahmin” işlemi ile modelin doğruluğu konsol ekranında %85,9 başarı oranında tahminlendiğini görür.



The image shows two parts of a Spyder Python IDE session. Part (a) displays the code in a script named 'bayes.py'. The code imports necessary libraries, performs train-test split, fits a CategoricalNB model, and prints the accuracy score. Part (b) shows the 'Console 1/A' window with the command 'runfile' executed, followed by the output 'Modelin Doğruluğu: 0.859344894026975'.

```

22 giris_egitim, giris_test, satis_egitim, satis_test = train_test_split(girisle
23
24
25 from sklearn.naive_bayes import CategoricalNB
26 model = CategoricalNB()
27 model.fit(giris_egitim, satis_egitim)
28 satis_tahmin = model.predict(giris_test)
29
30
31 from sklearn.metrics import confusion_matrix
32 import seaborn as sns
33 import matplotlib.pyplot as plt
34 cm= confusion_matrix(satis_test,satis_tahmin)
35 index = ['Kolay','Normal','Zor','Çok kolay']
36 columns = ['Kolay','Normal','Zor','Çok kolay']
37 cm_df = pd.DataFrame(cm,columns=index)
38 cm_df = pd.DataFrame(cm,columns=index)
39 plt.figure(figsize=(10,6))
40 sns.heatmap(cm_df, annot=True,fmt="d")
41
42 from sklearn import metrics
43 print("Modelin Doğruluğu:",metrics.accuracy_score(satis_test, satis_tahmin))

```

(a)

(b)

Şekil 115. Bayes öğrenme ile araç satış durumunu için (a) model doğruluk belirleme kod ekranı (b) model doğruluk sonucu

UYGULAMANIN PYTHON KODLARI

```

import numpy as np
import pandas as pd
veri =pd.read_csv("car.csv",encoding='unicode_escape')
veri_copy=veri.copy()

from sklearn import preprocessing
sayisallastirma = preprocessing.LabelEncoder()

veri_copy["fiyat"]=sayisallastirma.fit_transform(veri_copy["fiyat"])
veri_copy["onarim"]=sayisallastirma.fit_transform(veri_copy["onarim"])

```

```

veri_copy["kapi sayisi"]=sayisallaştırma.fit_transform(veri_copy["kapi sayisi"])
veri_copy["kisi sayisi"]=sayisallaştırma.fit_transform(veri_copy["kisi sayisi"])
veri_copy["bagaj boyutu"]=sayisallaştırma.fit_transform(veri_copy["bagaj boyutu"])
veri_copy["Guvenlik"]=sayisallaştırma.fit_transform(veri_copy["Guvenlik"])
veri_copy["satis "]=sayisallaştırma.fit_transform(veri_copy["satis "])

girisler=np.array(veri_copy.drop(["satis "],axis=1))
cikis=np.array(veri_copy["satis "])

from sklearn.model_selection import train_test_split
giris_egitim, giris_test, satis_egitim, satis_test = train_test_split(girisler,cikis,
test_size=0.3,random_state=109)

from sklearn.naive_bayes import CategoricalNB
model = CategoricalNB()
model.fit(giris_egitim, satis_egitim)
satis_tahmin= model.predict(giris_test)

from sklearn.metrics import confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt
cm= confusion_matrix(satis_test,satis_tahmin)
index = ['Kolay','Normal','Zor','Çok kolay']
columns = ['Kolay','Normal','Zor','Çok kolay']
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("Modelin Doğruluğu:",metrics.accuracy_score(satis_test, satis_tahmin))

```



4. ADIM: İLERLET

İlerlet adımda öğrencilerin gruplar halinde çalışarak, gerçekleştirilen uygulama üzerinde kendi düzenlemelerini yapmaları ve elde edilen sonuçlar hakkında notlar almaları istenir. Bu amaçla 30 dakikalık bir çalışma süresi verilir. Çalışma süresi esnasında öğrencilerin yaptıkları düzenlemeler uzaktan gözlemlenir, düzenlemelere esnasında sorusu olan öğrencilerin soruları cevaplandırılıp, yardım edilir.



5. ADIM: DEĞERLENDİRME

Derste edinilen bilgiler soru-cevap aşaması gerçekleştirilmek suretiyle pekiştirilir.

Öğrencilere şu sorular sorulabilir:

- Gerçekleştirilen Makine Öğrenmesi uygulamasından elde ettiğiniz bilgi ve beceriler neler olmuştur?
- Bayes öğrenme algoritması hakkındaki görüşleriniz nelerdir?
- Uyguladığınız Python kodlarında veri hazırlığı ve Makine Öğrenmesi modelinin oluşturularak eğitim-test süreçlerinden geçirilmesi aşamalarına dair görüşleriniz nelerdir?
- Benzeri bir uygulamayı gerçekleştirebileceğiniz problemler neler olabilir?

2. HAFTA – 3. GÜN – 1., 2., 3. DERSLER: KARAR AĞAÇLARI İLE COVID-19 HASTALIĞI TAHMİN ETME

DERS PLANI

DERS ETİKETLERİ



KONU: Karar Ağaçları ile COVID-19 Hastalığı Tahmin Etme



SINIF: 12-14 Yaş



SÜRE: 180 dk.



HAZIR BULUNUŞLUK: İngilizce eğitimi, Yapay Zeka kavramı ve uygulamaları hakkında temel bilgi, Makine Öğrenmesi hakkında temel bilgi, Python programlama dili hakkında temel bilgi



ANAHTAR KELİMELER: Yapay Zeka, Algoritma, Kodlama, Python, Karar Ağaçları



BECERİLER:



Tahmin



Sorgulama



Grup çalışması



Günlük hayat bağlantısı kurma



KAZANIMLAR:

Bu derste öğrenciler;

- Karar ağaçları algoritması hakkında bilgi ve beceri sahibi olurlar.
- Python ile karar ağaçları üzerinden sınıflandırmaya dayanan bir Makine Öğrenmesi uygulamasını gerçekleştirirler.
- Python ortamında medikal görüntü içeren bir veri seti üzerinden sınıflandırmaya dayalı bir Makine Öğrenmesi çözümünün nasıl gerçekleştirilebileceği hakkında bilgi ve beceri sahibi olurlar.



DERS MATERİYALLERİ:

Malzeme	Miktar
Yazı Tahtası	Öğrenci sayısı ve sınıf büyüğüğe göre en az bir adet
Projeksiyon	Öğrenci sayısı ve sınıf büyüğüğe göre en az bir adet
Defter, Kalem	Öğrenci sayısı kadar
Bilgisayar	Öğrenci sayısının yarısı kadar
Tablet	Öğrenci sayısının yarısı kadar
Python Anaconda Framework	Her bilgisayarda kurulu olacak şekilde



ÖĞRETMEN HAZIRLIK ÇALIŞMALARI:

Okunabilecek Kaynaklar:

Kaggle. (2020). *Decision-Tree Classifier Tutorial*. Çevirmiçi: <https://www.kaggle.com/code/prashant111/decision-tree-classifier-tutorial> (Erişim 7 Haziran 2024).

Kaggle. (2019). *A Guide to Decision Trees for Beginners*. Çevirmiçi: <https://www.kaggle.com/code/vipulgandhi/a-guide-to-decision-trees-for-beginners> (Erişim 8 Haziran 2024).

İzlenebilecek Kaynaklar:

Intuitive Machine Learning. (2021). Decision Tree: Important things to know. Çevirmiçi: <https://www.youtube.com/watch?v=JcI5E2Ng6r4> (Erişim 8 Haziran 2024).

Code Cube. (2023). Karar Ağaçları. Çevirmiçi: <https://www.youtube.com/watch?v=nAZmASdhmJQ> (Erişim 8 Haziran 2024).

ATÖLYE HAZIRLIK ÇALIŞMALARI:

Öğretmen;

- Dersten önce ders materyallerini hazırlamalıdır.
- Python kodlama için kullanılan Spyder editörünün genel kullanımına aşina olmalıdır.

- Derse konu olan veri set(ler)ini ve Python kodlarını önceden inceleyerek üzerinde çalışmış olmalıdır.



KAYNAKÇA:

Alan, M. A. (2014). Karar Ağaçlarıyla Öğrenci Verilerinin Sınıflandırılması. *Ataturk University Journal of Economics & Administrative Sciences*, 28(4), 101.

Dehghani, A. A., Movahedi, N., Ghorbani, K., & Eslamian, S. (2023). Decision tree algorithms. In *Handbook of Hydroinformatics*. Elsevier.

Köse, U., Özsoy, K., & Aksoy, B. (2023). *Yapay Zeka – Ortaokul ve Lise Kitapları*. Deneyap Teknoloji Atölyeleri, TÜBİTAK Yayıncılık.

Schmid, H. (2010). Decision trees. *The Handbook of Computational Linguistics and Natural Language Processing*, 180-196.

Uzun, E. (2023). Decision Tree (Karar Ağacı): ID3 Algoritması – Classification (Sınıflama). Erdinç Uzun Web Sayfası. Çevirmiçi: https://erdincuzun.com/makine_ogrenmesi/decision-tree-karar-agaci-id3-algoritmasi-classification-siniflama/ (Erişim 8 Haziran 2024).

GENEL BAKIŞ:

1) Harekete Geç: Öğrencilere uygulamaya konu karar ağaçları algoritması anlatılır ve Python kodlama ile Makine Öğrenmesi sürecine yönelik ön hazırlık gerçekleştirilir.

2) Keşfet: Öğrencilerle birlikte ilgili Makine Öğrenmesi modelinin çalıştırılmasına dair kodlamaların yapılması sağlanır.

3) Üret: Öğrencilerle birlikte çalıştırılan Makine Öğrenmesi modelinin elde ettiği sonuçların değerlendirilmesine yönelik kodlamaların yapılması sağlanır.

4) İlerlet: Öğrencilerin gruplar halinde çalışarak, gerçekleştirilen uygulama üzerinde kendi düzenlemelerini yapmaları ve elde edilen sonuçlar hakkında notlar almaları istenir.

5) Değerlendir: Derste edinilen bilgiler soru-cevap aşaması gerçekleştirilmek suretiyle pekiştirilir.

UYGULAMA



1. ADIM: HAREKETE GEÇ

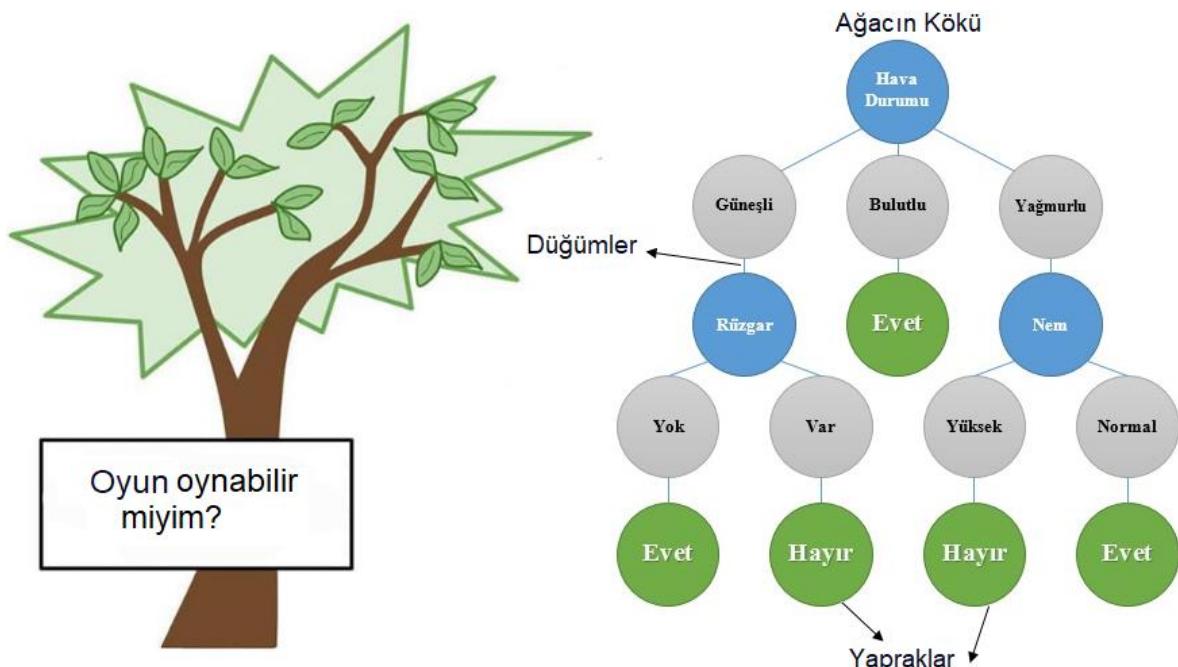
Harekete Geç adımı kapsamında öğrencilere uygulamaya konu olan karar ağaçları algoritması anlatılır ve Python kodlama ile Makine Öğrenmesi sürecine yönelik ön hazırlık gerçekleştirilir.

Karar Ağaçları Algoritması / Tekniği

Karar ağaçları, sınıfları bilinen örnek veriden karar düğümleri (decision nodes) ve yaprak düğümlerinden (leaf nodes) oluşan ağaç şekilli bir karar algoritmasıdır (Alan, 2014; Köse vd.,

2023). Karar ağaçları algoritması, veri setini bölüp küçülterek geliştirilen bir yöntemdir. Karar düğümleri bir veya birden fazla dallan meydana gelebilir. İlk düşüme kök düşüm (root node) denir. Karar ağacı algoritmaları hem metinsel hem de sayısal verilerden oluşabilir (Uzun, 2023).

Şekil 116'da gösterildiği gibi havanın durumuna göre öğrencilerin okulun bahçesinde oyun oynayıp oynamayacağını belirlemek için karar ağaçları yapısı gösterilmiştir. Burada karar ağacının algoritmasının kökü, hava durumudur. Karar ağacının düğümleri ise güneş, bulut, yağmur, rüzgar ve havanın nemi de koşul ifadeleridir. Karar ağacının yaprakları ise okulun bahçesinde oyun oynayıp oynamayacağını belirtir.



Şekil 116. Karar ağaçlarının yapısı (Web Kaynağı 4.1)

Karar ağaçları avantaj ve dezavantajları şu şekilde sıralanabilmektedir (Köse vd., 2023):

Avantajlar

- Ağaç yapılarını yorumlaması 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. Böylece 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. Aşırı öğrenme eğitim için kullanılan verinin çok iyi öğrenilip, başka verilere başarılı cevaplar verilememesi anlamına gelmektedir. Bu açıdan aşırı öğrenme, bir bakıma Makine Öğrenmesi algoritmasının eğitim verisi ezberlemesi demektir.

Karar ağacı tekniğini kullanarak verinin sınıflanması, öğrenme ve sınıflama olmak üzere iki basamaklı bir işlemidir. Yaygın olarak bilinen karar ağacı algoritmaları şu şekildedir (Köse vd., 2023; Dehghani vd., 2023; Schmid, 2010):

- **ID3 Karar Ağacı Algoritması:** ID3, Yapay Zeka alanında ilk kez çalışmaya başlayan öğrencilerin karar ağacılarının temel çalışma şeklini kolayca öğrenebilecekleri ideal bir algoritmadır. ID3 karar ağaç algoritmasının C4.5 ve C5.0 isminde iki tane versiyonu sıkılıkla 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.
- **C&RT Karar Ağacı Algoritması:** Gini indeksi (dizini) veya Gini katsayısı, İtalyan istatistikçi Corrado Gini tarafından 1912'de geliştirilen istatistiksel bir ölçütür. Gini'ye dayalı ikili bölmeye göre çalışan bir karar ağacı algoritmasıdır. Bu algoritmada en son veya ucta 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.
- **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 etkileşimi 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 bölmeye işlemini gerçekleştirir.
- **SPRINT Karar Ağacı Algoritması:** SPRINT algoritması 1996 yılında Shafer, Agrawal ve Mehta tarafından geliştirilip entropiye dayanmaktadır. SPRINT karar ağacı 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 ağacı yapısı bu şekilde oluşur. Bu algoritmada her bir değişken için ayrı bir değişken listesi hazırlanır. Bölme işlemi tek bir özelliğin değerine göre saptanır.
- **SLIQ Karar Ağacı Algoritması:** SLIQ karar ağacı algoritması 1996 yılında Agrawal, Mehta ve Rissanen tarafından geliştirilmiştir. Bu algoritma Gini teknigi ile nicel ve nitel veri tipleri kullanır. SLIQ algoritmasında verilerin sıralanmasında en iyi dallara ayırmaya teknigi için oldukça önemlidir. Bu algoritma hızlı ölçüm yapan bir sınıflandırıcıya ve hızlı ağaç budama algoritmasına sahiptir.



2. ADIM: KEŞFET

Bu adımda öğrencilerle birlikte ilgili Makine Öğrenmesi modelinin çalıştırılmasına dair kodlamaların yapılması sağlanır.

Öğrenciler, ilk olarak karar ağacı ile Bilgisayar Tomografi (BT) üzerinde COVID-19 hastalığı değerlendirme problemini anlar. Şekil 117'de gösterilen BT görüntüsüne göre COVID-19 hastalığı olup olmadığını modeller.

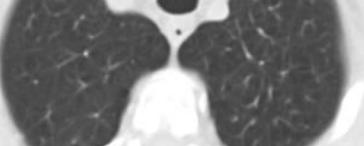


Şekil 117. BT görüntüsü.

Öğrenciler Tablo 3'de verilen karar ağaçları algoritması ile COVID-19 hastalığı için giriş çıkış parametresini belirler.

Tablo 3. Karar ağaçları algoritması için giriş ve çıkış parametreleri.

Veri Sayısı	GİRİŞ PARAMETRESİ	ÇIKIŞ PARAMETRESİ
	BT Görüntüsü	HASTALIK
1		0 (Pozitif)
2		0 (Pozitif)
...
1252		0 (Pozitif)

1253		1 (Negatif)
2481		1 (Negatif)

Bu uygulama kapsamında hastalara ait 2481 adet veri setinde BT görüntüsü giriş parametrelerine göre hastanın COVID-19 olup olmaması durumunu karar ağaçları algoritması ile tahminlenmesi amaçlanmıştır.

VERİ SETİ İÇİN İNDİRME LİNKİ

<https://www.kaggle.com/plameneduardo/sarscov2-ctscan-dataset>

Not: Veri seti ilgili linkten COVID ve non-COVID olarak ayrı şekilde indirilmektedir (Şekil 118).

Ad	Değiştirme tarihi	Tür	Boyut
COVID	19.08.2021 06:33	Dosya klasörü	
non-COVID	19.08.2021 06:33	Dosya klasörü	

Şekil 118. Veri seti klasörleri.

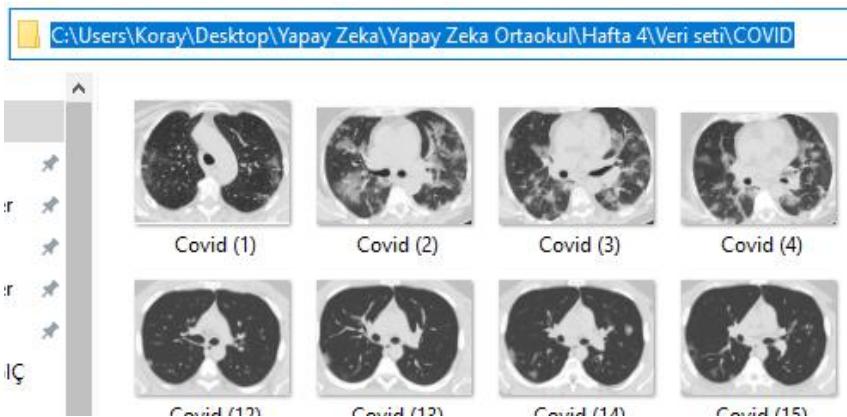
Öğrenci ilgili veri setini dosyasını basit bir karar ağaçları algoritması ile COVID-19 hastalığı teşhisine yönelik tahminleme işlemi yapar. Şekil 119'da gösterildiği gibi “veri seti” dosyasını yüklemek için gerekli kütüphaneleri ve komutları yazar.

The screenshot shows the Spyder Python IDE interface. The menu bar includes File, Edit, Search, Source, Run, Debug, Consoles, Projects, Tools, View, and Help. Below the menu is a toolbar with various icons for file operations like Open, Save, Copy, Paste, Find, and Run. The current file path is displayed as C:\Users\Koray\Desktop\Yapay Zeka\Yapay Zeka Ortaokul\Hafta 4\PTR.py. The code editor window contains Python code for reading and processing COVID-19 datasets:

```
1 import numpy as np
2 from PIL import Image as img #Pillow kütüphanesi çağrılması
3 import os #Operating system kütüphanesinin çağrılması
4 import pandas as pd
5
6 covidli="C:/Users/Koray/Desktop/Yapay Zeka/Yapay Zeka Ortaokul/Hafta 4/Veri seti/COVID/"
7 covid_olmayan="C:/Users/Koray/Desktop/Yapay Zeka/Yapay Zeka Ortaokul/Hafta 4/Veri seti/non-COVID/"
```

Şekil 119. Gerekli kütüphaneleri ve komutların kod gösterimi.

Öğrencilere COVID-19 hastalığı ile ilgili hazırlanacak örnekte ilk olarak dizi işlemlerinde kullanılan “NUMPY” kütüphanesinden, bilgisayarlı görüp işlemlerinden kullanılan “PILLOW (PIL)” kütüphanesinden ve verilerin çekileceği klasörler ile ilgili işlemlerde kolaylık sağlayan “OS” kütüphanesinden ve veri setlerinin analizlerinde kullanılan “PANDAS” kütüphanesinden kısaca bahsedilir (Bu aşamada önceki haftalarda edinilen bilgiler pekiştirilebilir). Python programlama dilinde 6. ve 7. satırda kod yazarken veri seti klasörü konumu her bir öğrencinin bilgisayarında farklılık göstereceği belirtilir. Bu nedenle öğrencilere Şekil 120’de gösterildiği gibi veri setinin konum adresi belirleme öğretilir.



Şekil 120. Veri seti klasör konumu.

Öğrenciler Şekil 121’de görüldüğü gibi operating system kütüphanesini kullanarak COVID-19 ve COVID-19 olmayan görüntülerin yer aldığı klasördeki tüm BT görüntülerini döngü kurarak (for döngüsü) bulunduğu klasör konumlarını yol değişkeninden okur.

```

Spyder (Python 3.8)
File Edit Search Source Run Debug Consoles Projects Tools View Help
D:\Users\Koray\Desktop\Yapay Zeka\Yapay Zeka Ortaokulu\Hafta 4\|DTR.py
DTR.py*
1 import numpy as np
2 from PIL import Image as img #Pillow kütüphanesi çağrılması
3 import os #Operating system kütüphanesinin çağrılması
4 import pandas as pd
5
6 covidli="C:/Users/Koray/Desktop/Yapay Zeka/Yapay Zeka Ortaokulu/Hafta 4/Veri seti/COVID/"
7 covid_olmayan="C:/Users/Koray/Desktop/Yapay Zeka/Yapay Zeka Ortaokulu/Hafta 4/Veri seti/non-COVID/"
8
9 def dosya(yol):
10     return [os.path.join(yol,f) for f in os.listdir(yol)]

```

Şekil 121. BT görüntülerinin konumları liste halinde çağrıımı kod ekranı.

Şekil 122’de gösterildiği gibi COVID-19 olmayan verileri dönüştürmek için “**klasor_adi**” ve “**sinif_adi**” parametrelerini içeren “**veri_donusturme**” isminde bir fonksiyon tanımlar.

14. kod satırında COVID-19 ve COVID-19 olmayan görüntülerini bulunduğu klasörlerden tek tek okumak için “**goruntuler**” isminde bir değişken oluşturulur.

16. Kod satırında COVID-19 ve COVID-19 olmayan görüntüleri etiketlemek için “**goruntu_sinif**” isminde boş bir dizi oluşturulur.

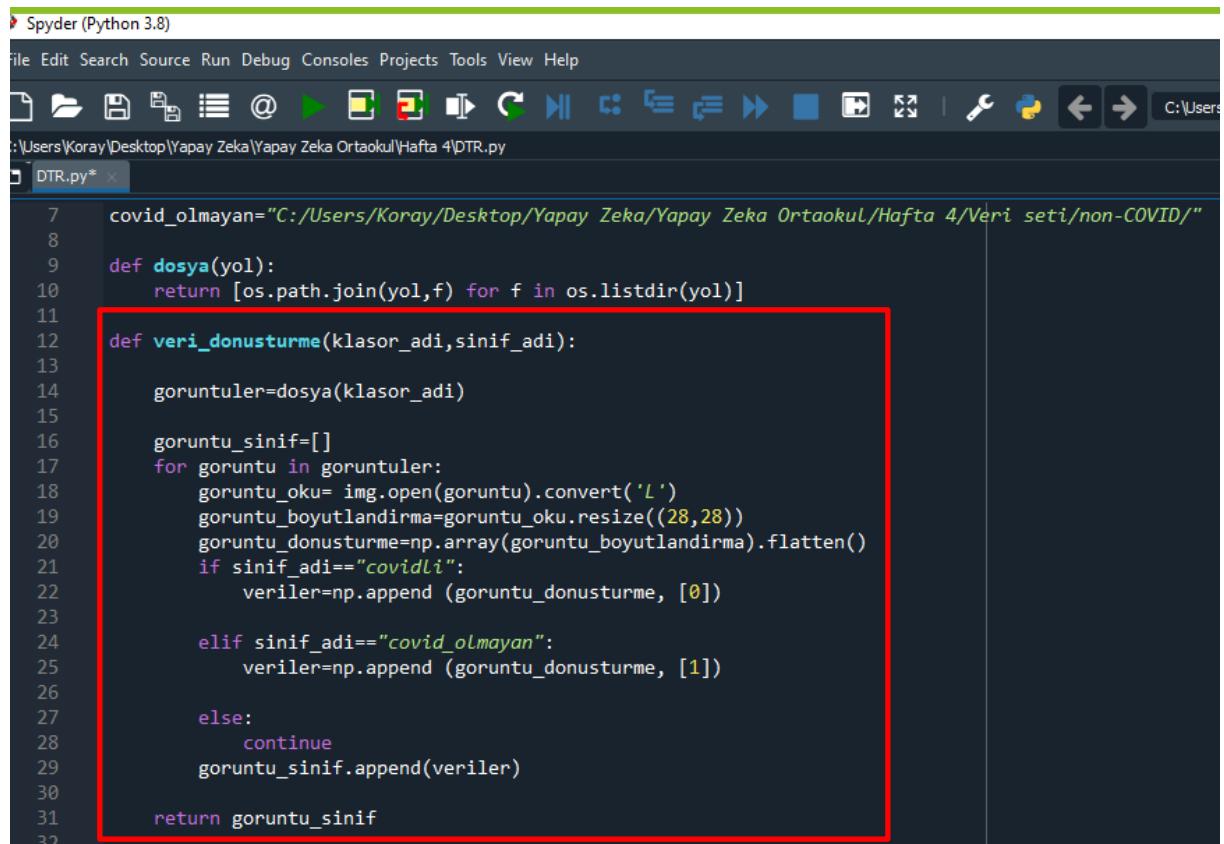
17. kod satırda COVID-19 ve COVID-19 olmayan tüm görüntüleri işlemek için bir “**for**” döngüsü oluşturur.

18. Satırda tüm BT görüntüleri okunarak “**.convert('L')**” özelliği kullanılarak gri tonlamalı görüntülere dönüştürür.

19. Kod satırında “**goruntu_boyutlandırma**” değişkeni kullanılarak tüm BT görüntüleri “**.resize**” komutu kullanılarak 28x28 piksel boyutuna küçültülür (Söz konusu küçültmenin görüntü verilerini daha hızlı işlemek için gerçekleştirildiği öğrencilere açıklanır).

20. Kod satırında iki boyutlu olan BT görüntülerini karar ağaçları ile eğitebilmek için “**.flatten**” özelliği kullanılarak görüntüler 784 elemanlı tek boyutlu bir diziye (vektör) dönüştürür (Öğrenciye 784 rakamının 28x28 görüntü boyutuna düşürülen iki boyutlu bir görüntü dizisinin tek boyuta düşürülmesi ile $28 \times 28 = 784$ elde edildiği açıklanır).

21-29 kod satırları arasında, COVID-19 ve COVID-19 olmayan BT görüntüleri etiketlenmiştir. Etiketleme işleminde COVID-19 olanlar “**0**”, COVID-19 olmayan ise “**1**” eğer COVID-19 ve COVID-19 olmayan bir BT görüntüsü geldiğinde ise etiketleme yapılmamıştır. 31. Kod satırında ise etiketleme yapılan değişken olan “**goruntu_sinif**” değişkeni ile etiketleme işlemi tamamlanmıştır.



```
Spyder (Python 3.8)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\Koray\Desktop\Yapay Zeka\Yapay Zeka Ortaokul\Hafta 4\DTI.py
DTR.py* [ ]
7     covid_olmayan="C:/Users/Koray/Desktop/Yapay Zeka/Yapay Zeka Ortaokul/Hafta 4/Veri seti/non-COVID/"
8
9     def dosya(yol):
10        return [os.path.join(yol,f) for f in os.listdir(yol)]
11
12    def veri_donusturme(klasor_adi,sinif_adi):
13
14        goruntuler=dosya(klasor_adi)
15
16        goruntu_sinif=[]
17        for goruntu in goruntuler:
18            goruntu_oku= img.open(goruntu).convert('L')
19            goruntu_boyutlandırma=goruntu_oku.resize((28,28))
20            goruntu_donusturme=np.array(goruntu_boyutlandırma).flatten()
21            if sinif_adi=="covidli":
22                veriler=np.append (goruntu_donusturme, [0])
23
24            elif sinif_adi=="covid_olmayan":
25                veriler=np.append (goruntu_donusturme, [1])
26
27            else:
28                continue
29            goruntu_sinif.append(veriler)
30
31    return goruntu_sinif
32
```

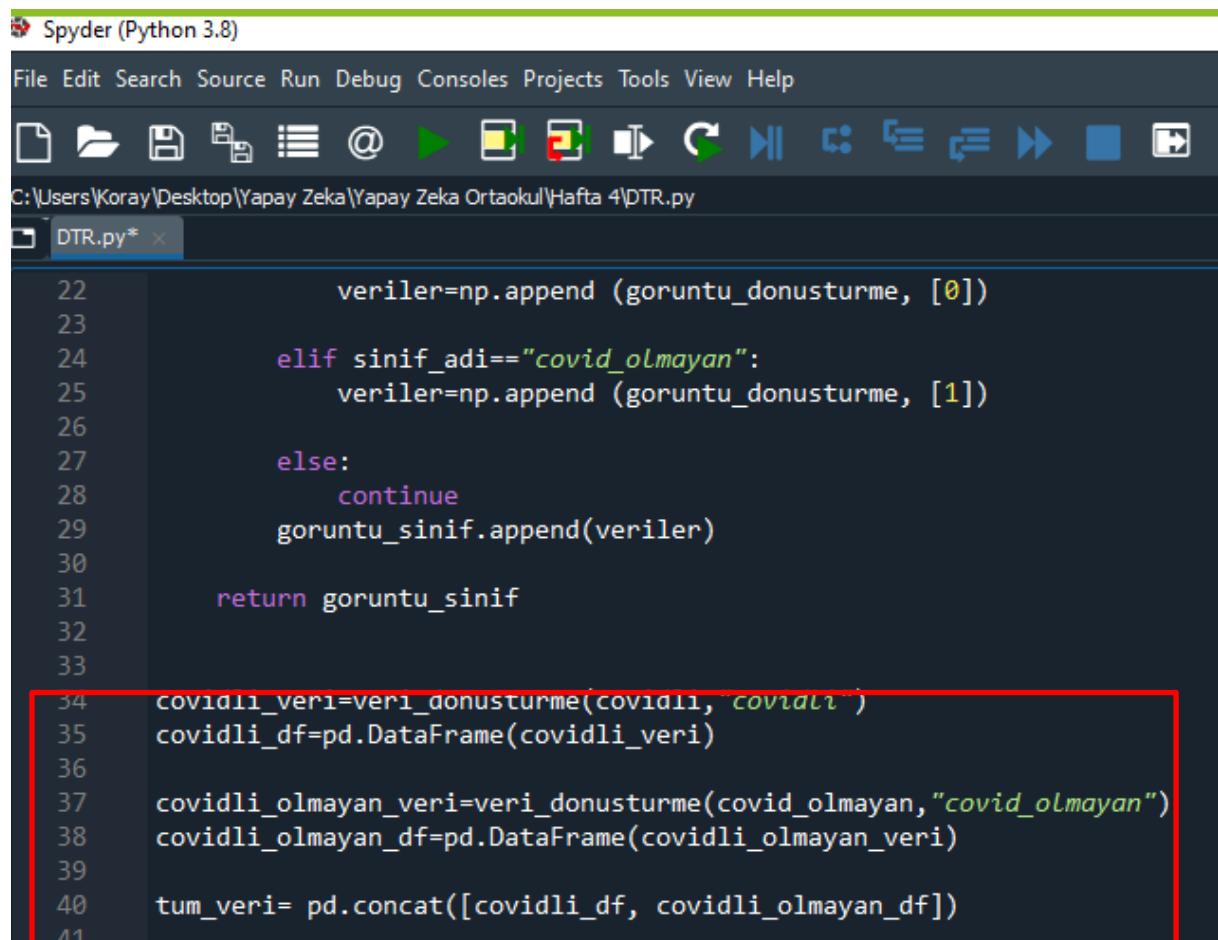
Şekil 123. Veri setindeki metin değerlerinin sayısallaştırılması kod ekranı.

Öğrenciler Şekil 124'te gösterilen 34. kod satırlarında “covidli” veri değişkeni ile oluşturulmuş durumdaki “veri_donusturme” fonksiyonuna giderek covidli olarak etiketlenen verileri “covidli_veri” değişkenine aktarır.

35. kod satırında ise covidli veriler pandas kütüphanesinin “.DataFrame” özelliği ile veriler daha sade ve anlaşılır hale dönüştürülmüştür.

37. ve 38 kod satırlarında, 34. ve 35. kod satırlarında covidli veriler için yapılan işlemler covidli olmayanlar içinde gerçekleştirilmiştir.

40. Kod satırında ise 34., 35. ve 37., 38. kod satırındaki covidli ve covid_olmayan veriler Pandas kütüphanesinin “.concat” özelliği kullanılarak birleştirilerek “tüm_veri” değişkenine aktarılmıştır.



The screenshot shows the Spyder Python 3.8 IDE interface. The menu bar includes File, Edit, Search, Source, Run, Debug, Consoles, Projects, Tools, View, Help. The toolbar has various icons for file operations. The current file is DTR.py, located at C:\Users\Koray\Desktop\Yapay Zeka\Yapay Zeka Ortaokul\Hafta 4\DTR.py. The code editor contains the following Python code:

```
22     veriler=np.append (goruntu_donusturme, [0])
23
24     elif sinif_adi=="covid_olmayan":
25         veriler=np.append (goruntu_donusturme, [1])
26
27     else:
28         continue
29     goruntu_sinif.append(veriler)
30
31     return goruntu_sinif
32
33
34     covidli_veri=veri_donusturme(covidli, "covidli")
35     covidli_df=pd.DataFrame(covidli_veri)
36
37     covidli_olmayan_veri=veri_donusturme(covid_olmayan, "covid_olmayan")
38     covidli_olmayan_df=pd.DataFrame(covidli_olmayan_veri)
39
40     tum_veri= pd.concat([covidli_df, covidli_olmayan_df])
41
```

A red rectangular box highlights the code from line 34 to 40, which performs the concatenation of the two DataFrames.

Şekil 124. BT görüntülerin etiketlenerek birleştirilmesi.

Öğrenciler 43. kod satırında arasında pandas kütüphanesi çağırır.

44. Kod satırında “sklearn” kütüphanesi içerisinde yer alan “DecisionTreeClassifier” ile karar ağaçları algoritmaları ile eğitim gerçekleştirebilmek için “sklearn.tree” kütüphanesini çağırır.

45. kod satırında ise, veri setindeki eğitim ve test olarak ayırbilmek için “sklearn.model_selection” kütüphanesinde yer alan “train_test_split” kütüphanesini çağırır.

46. Kod satırında “sklearn” kütüphanesi içerisinde yer alan “metrics” kütüphanesi kullanılarak model değerlendirilir.

47. Kod satırında ise “**sklearn.model_selection**” kütüphanesinde yer alan “**GridSearchCV**” algoritmasını çağırır. “**GridSearchCV**” algoritması modelin hiper parametreleri değiştirilerek model doğruluğunu artırmak için kullanılan bir algoritmadır.

49. ve 50. Kod satırlarında ise “**.flatten**” özelliği ile 784 elemanlı tek boyutlu bir vektöre dönüştürülen görüntüler “**Giris**” ve “**Cikis**” değişkenlerine aktarılmıştır.

51. satırda ise, veri setindeki eğitim ve test verilerini %80 eğitim, %20 test olacak şekilde “**test_size=0,2**” komutu kullanarak rastgele ayırrır. Burada “**random_state=1**” ifadesi her eğitim tekrarında alınacak verilerin aynı kalmasını sağlar.

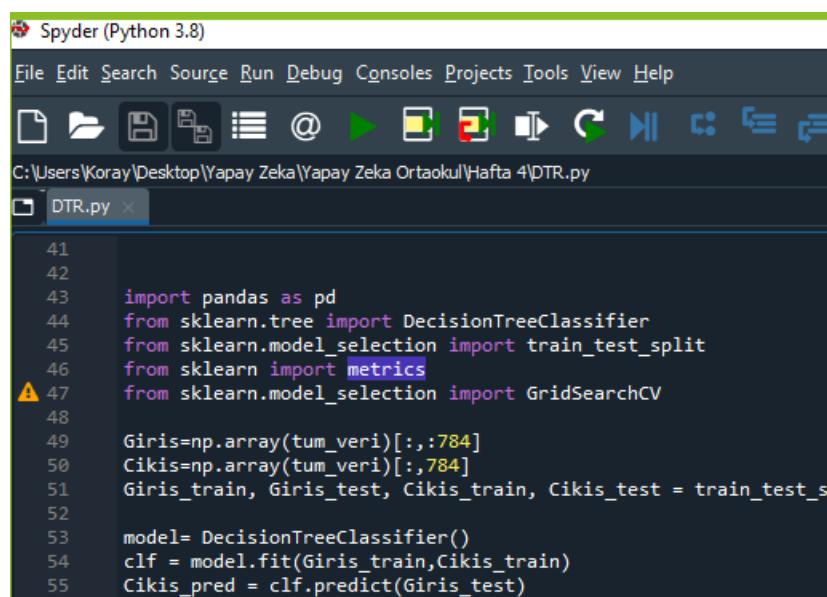
```
43 import pandas as pd
44 from sklearn.tree import DecisionTreeClassifier
45 from sklearn.model_selection import train_test_split
46 from sklearn import metrics
47 from sklearn.model_selection import GridSearchCV
48
49 Giris=np.array(tum_veri)[:,784]
50 Cikis=np.array(tum_veri)[:,784]
51 Giris_train, Giris_test, Cikis_train, Cikis_test = train_test_split(Giris, Cikis, test_size=0.2, random_state=1)
```

Şekil 125. Veri setinin eğitim ve test olarak rastgele ayrılması için kod ekranı

Öğrenciler Şekil 126'da gösterildiği gibi 53. kod satırında karar ağaçları algoritması “**DecisionTreeClassifier**” sınıflandırıcı ile model değişkenine aktarır.

54. Kod satırında ise BT görüntülerine göre COVID-19 hastalığın 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.

55. Kod satırında ise karar ağaçları modelinden elde edilen eğitim sonuçlarını “**Giris_test**” verilerine göre “**.predict**” özelliği kullanılarak tahmin edip, sonucu “**Cikis_pred**” değişkenine aktarır.



The screenshot shows the Spyder Python 3.8 IDE interface. The title bar says "Spyder (Python 3.8)". The menu bar includes File, Edit, Search, Source, Run, Debug, Consoles, Projects, Tools, View, and Help. Below the menu is a toolbar with various icons. The current file is "DTR.py". The code editor contains the following Python code:

```
41
42
43 import pandas as pd
44 from sklearn.tree import DecisionTreeClassifier
45 from sklearn.model_selection import train_test_split
46 from sklearn import metrics
47 from sklearn.model_selection import GridSearchCV
48
49 Giris=np.array(tum_veri)[:,784]
50 Cikis=np.array(tum_veri)[:,784]
51 Giris_train, Giris_test, Cikis_train, Cikis_test = train_test_split(Giris, Cikis, test_size=0.2, random_state=1)
52
53 model= DecisionTreeClassifier()
54 clf = model.fit(Giris_train,Cikis_train)
55 Cikis_pred = clf.predict(Giris_test)
```

Şekil 126. Karar ağaçları algoritmasında eğitim ve tahminleme işlemleri için kod ekranı.



3. ADIM: ÜRET

Öğrencilerle birlikte çalıştırılan Makine Öğrenmesi modelinin elde ettiği sonuçların değerlendirilmesine yönelik kodlamaların yapılması sağlanır.

Karar Ağacı Modelin Başarım Oranı

Öğrenciler yürüt aşamasında eğitmiş oldukları karar ağaçları modeline ait doğruluk sonucunu ve grafiklerini bu aşamada gerçekleştirir.

57. Kod satırında karar ağaçları ile eğitilen BT görüntülerinin tahmin sonuçlarını konsol ekranına aktaran ilgili kodu yazar.

59. kod satırında grafik çizmek için “`matplotlib.pyplot`” kütüphanesini çağırır.

60. Kod satırında ise, “`Cikis_test`” verilerini “`Cikis_pred`” ile karşılaştırıp ROC eğrisini çizebilmek için “`.roc_curve`” özelliği kullanılarak yanlış pozitif oranı (ypo), doğru pozitif oranı (dpo) ve threshold (esik değeri) parametrelerini belirler.

61. Kod satırında ise metrics kütüphanesinin “`.auc`” özelliğini kullanarak yanlış pozitif oranı (ypo), doğru pozitif oranı (dpo) değerlerini hesaplayarak “`roc_auc`” değişkenine aktarır.

Öğrencilere Makine Öğrenmesi’nde iki veya daha fazla sınıflandırma modelinin performansının ölçümlünde doğruluk, kesinlik duyarlılık, f-puanı ve alıcı çalışma karakteristik (Receiver Operating Characteristic-ROC) eğrisinin kullanıldığı anlatır. Sınıflandırma modellerinde kullanılan dört farklı durum şu şekildedir:

- **Doğru Pozitif Oranı (DPO):** Doğru olarak belirlenen pozitif sınıfların tahmin sayısı
- **Doğru Negatif Oranı (DNO):** Doğru olarak belirlenen negatif sınıfların tahmin sayısı
- **Yanlış Pozitif Oranı (YPO):** Yanlış olarak belirlenen pozitif sınıfların tahmin sayısı
- **Yanlış Negatif Oranı (YNO):** Yanlış olarak belirlenen negatif sınıfların tahmin sayısı

Öğrenciler, 62-70. satırları arasında ise ROC eğrisi için başlık, yanlış pozitif oranın (ypo), doğru pozitif oranına (dpo) göre çizimi, etiketlenmesi, hangi renkte çizileceği, hangi sayısal aralıkta çizileceği ve x ve y eksen isimlerine göre ROC eğrisini çizen komutları yazar.

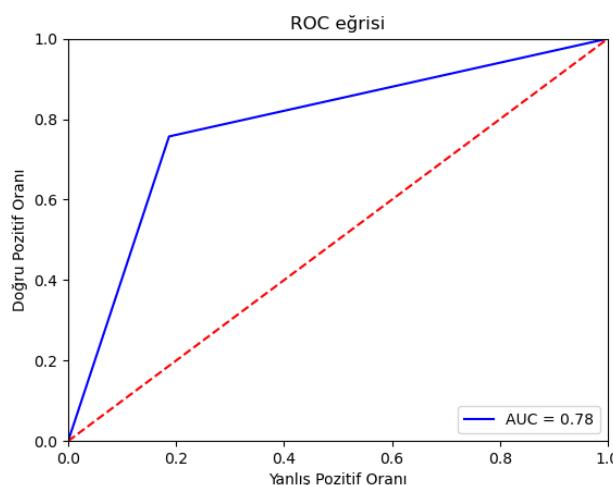
```

41
42
43     import pandas as pd
44     from sklearn.tree import DecisionTreeClassifier
45     from sklearn.model_selection import train_test_split
46     from sklearn import metrics
47     from sklearn.model_selection import GridSearchCV
48
49     Giris=np.array(tum_veri)[:,784]
50     Cikis=np.array(tum_veri)[:,784]
51     Giris_train, Giris_test, Cikis_train, Cikis_test = train_test_split(Giris, Cikis)
52
53     model= DecisionTreeClassifier()
54     clf = model.fit(Giris_train,Cikis_train)
55     Cikis_pred = clf.predict(Giris_test)
56
57     print("Doğruluk:",metrics.accuracy_score(Cikis_test, Cikis_pred))
58
59     import matplotlib.pyplot as cizim
60     ypo, dpo, threshold = metrics.roc_curve(Cikis_test, Cikis_pred)
61     roc_auc = metrics.auc(ypo, dpo)
62     cizim.title('ROC eğrisi')
63     cizim.plot(ypo, dpo, 'b', label = 'AUC = %0.2f' % roc_auc)
64     cizim.legend(loc = 'lower right')
65     cizim.plot([0, 1], [0, 1], 'r--')
66     cizim.xlim([0, 1])
67     cizim.ylim([0, 1])
68     cizim.ylabel('Doğru Pozitif Oranı')
69     cizim.xlabel('Yanlış Pozitif Oranı')
70     cizim.show()

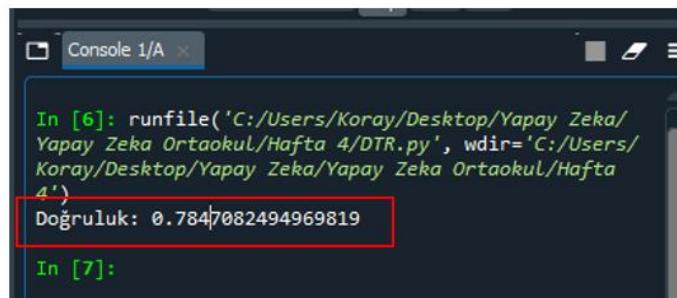
```

Şekil 127. ROC eğrisinin çizimi için kod ekranı.

Öğrenciler çeşitli BT görüntülerine göre COVID-19 hastalığı olup olmadığını karar ağaçları algoritması ile tahmin edebilen uygulama örneği için Python kodlarını yazar. Öğrenciler hazırlamış oldukları Python kodlarını çalıştırıldıklarında Şekil 128'de ve Şekil 129'da gösterildiği gibi ROC eğrisini ve modelin başarısını (doğruluk oranı) görüntülerler. Böylece, karar ağaçları algoritması sayesinde, 100 görüntü içerisinde 78 görüntünün doğru başırm oranı (%78) ile tahminlendiğini görürler (Bu aşamada ROC eğrisinde görülen detaylar öğrencilere anlatılır).



Şekil 128. Elde edilen ROC eğrisi.



Şekil 129. Elde edilen doğruluk oranı.

UYGULAMANIN PYTHON KODLARI (Başarım: %78)

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

covidli="covid_veri-seti/COVID/"
covid_olmayan="covid_veri-seti/non-COVID/"

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=="covidli":
            veriler=np.append (goruntu_donusturme, [0])
        elif sinif_adi=="covid_olmayan":
            veriler=np.append (goruntu_donusturme, [1])
        else:
            continue
```

```

goruntu_sinif.append(veriler)

return goruntu_sinif


covidli_veri=veri_donusturme(covidli,"covidli")
covidli_df=pd.DataFrame(covidli_veri)

covidli_olmayan_veri=veri_donusturme(covid_olmayan,"covid_olmayan")
covidli_olmayan_df=pd.DataFrame(covidli_olmayan_veri)

tum_veri= pd.concat([covidli_df, covidli_olmayan_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)

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

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

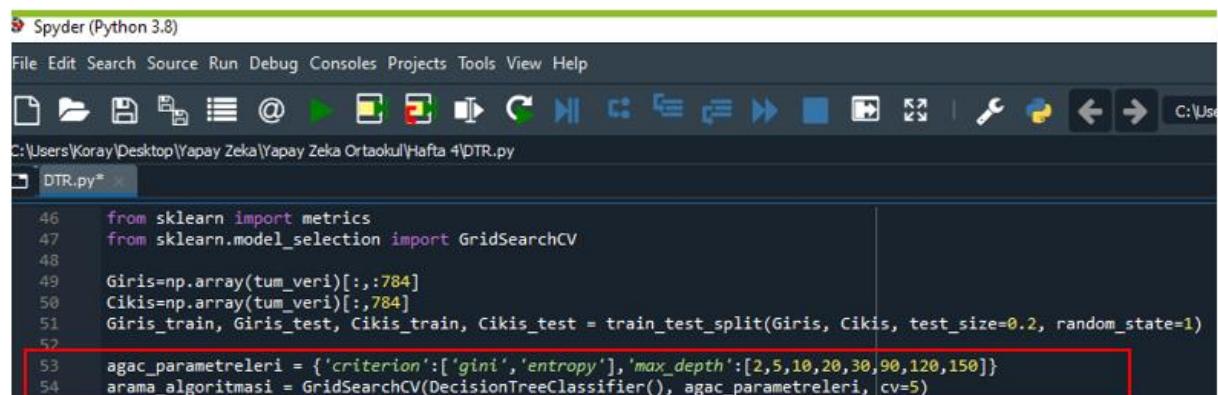
import matplotlib.pyplot as cizim
ypo, dpo, threshold = metrics.roc_curve(Cikis_test, Cikis_pred)
roc_auc = metrics.auc(ypo, dpo)
cizim.title('ROC eğrisi')
cizim.plot(ypo, dpo, 'b', label = 'AUC = %0.2f' % roc_auc)

```

```
cizim.legend(loc = 'lower right')
cizim.plot([0, 1], [0, 1],'r--')
cizim.xlim([0, 1])
cizim.ylim([0, 1])
cizim.ylabel('Doğru Pozitif Oranı')
cizim.xlabel('Yanlış Pozitif Oranı')
cizim.show()
```

Modelin Başarım Oranının Yükseltilmesi

Önceki uygulamada elde edilen başarım oranını yükseltmek için kod üzerinde birtakım güncellemeler gerçekleştiriliyor. Buna göre öğrenciler, 53. 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 “**ağac_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 gerçekleştirilmesi ve over fitting (aşırı öğrenme) engellemektir. Öğrenciler 54. kod satırında ise, GridSearchCV algoritması kullanılarak “**ağac_parametreleri**” sözlüğünde yer alan veriler 5 parçaaya ayrılarak (verileri yüzdesel olarak 100 kabul edilirse, rastgele olarak verileri 0-20, 20-40, 40-60, 60-80 ve 80-100 olmak üzere) en iyi sonucu veren parametre değerleri karar ağaçları algoritmasının eğitiminde kullanılır.



```

46     from sklearn import metrics
47     from sklearn.model_selection import GridSearchCV
48
49     Giris=np.array(tum_veri)[:,784]
50     Cikis=np.array(tum_veri)[:,784]
51     Giris_train, Giris_test, Cikis_train, Cikis_test = train_test_split(Giris, Cikis, test_size=0.2, random_state=1)
52
53     agac_parametreleri = {'criterion':['gini', 'entropy'], 'max_depth':[2,5,10,20,30,90,120,150]}
54     arama_algoritmasi = GridSearchCV(DecisionTreeClassifier(), agac_parametreleri, cv=5)

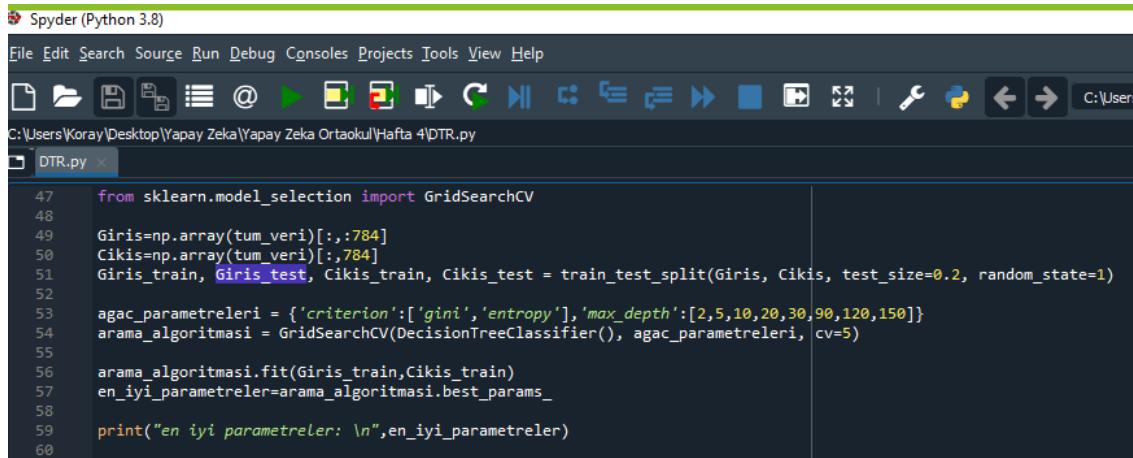
```

Şekil 130. Karar ağaçları hiper parametrelerin tanımı ve GridSearchCV algoritmasının uygulanması.

Öğrenciler Şekil 131'de gösterildiği gibi 56. Kod satırında BT görüntülerine göre COVID-19 hastalığın tahmini durumu için oluşturmuş olduğu karar ağaçları algoritmasını kullanarak “**arama_algoritması.fit**” komutu yazarak giriş eğitim verilerine göre çıkış eğitim verilerinin eğitimi gerçekleşir.

57. kod satırında ise, GridSearchCV algoritmasının “**.best_params_**” özelliği kullanılarak en iyi hiper parametreleri “**en_iyi_parametreler**” değişkenine aktarılır.

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



```
Spyder (Python 3.8)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\Koray\Desktop\Yapay Zeka\Yapay Zeka Ortaokul\Hafta 4\DTR.py
DTR.py

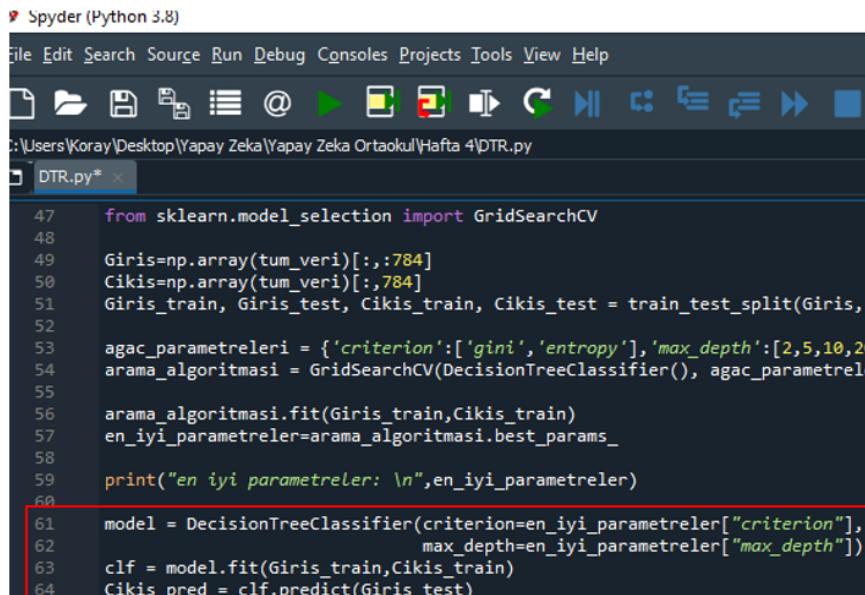
47 from sklearn.model_selection import GridSearchCV
48
49 Giris=np.array(tum_veri)[:,784]
50 Cikis=np.array(tum_veri)[:,784]
51 Giris_train, Giris_test, Cikis_train, Cikis_test = train_test_split(Giris, Cikis, test_size=0.2, random_state=1)
52
53 agac_parametreleri = {'criterion':['gini','entropy'],'max_depth':[2,5,10,20,30,90,120,150]}
54 arama_algoritması = GridSearchCV(DecisionTreeClassifier(), agac_parametreleri, cv=5)
55
56 arama_algoritması.fit(Giris_train,Cikis_train)
57 en_iyi_parametreler=arama_algoritması.best_params_
58
59 print("en iyi parametreler: \n",en_iyi_parametreler)
60
```

Şekil 131. GridSearchCV algoritması ile karar ağaçları modeli için en uygun belirlenen hiper parametrelerin belirlenmesi kod ekranı.

Öğrenciler Şekil 132'de gösterildiği gibi 61. ve 62. Kod satırında belirlenen kriterler (Gini, Entropy) ve maksimum derinlik (max_depth) parametrelerine göre karar ağaç modeli oluşturur.

63. Kod satırında ise BT görüntülerine göre COVID-19 hastalığın 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.

64. Kod satırında ise karar ağaçları modelinden elde edilen eğitim sonuçlarını “**Giris_test**” verilerine göre “**.predict**” özelliği kullanılarak tahmin edip, sonucu “**Cikis_pred**” değişkenine aktarır.



```
Spyder (Python 3.8)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\Koray\Desktop\Yapay Zeka\Yapay Zeka Ortaokul\Hafta 4\DTR.py
DTR.py

47 from sklearn.model_selection import GridSearchCV
48
49 Giris=np.array(tum_veri)[:,784]
50 Cikis=np.array(tum_veri)[:,784]
51 Giris_train, Giris_test, Cikis_train, Cikis_test = train_test_split(Giris,
52
53 agac_parametreleri = {'criterion':['gini','entropy'],'max_depth':[2,5,10,20
54 arama_algoritması = GridSearchCV(DecisionTreeClassifier(), agac_parametrele
55
56 arama_algoritması.fit(Giris_train,Cikis_train)
57 en_iyi_parametreler=arama_algoritması.best_params_
58
59 print("en iyi parametreler: \n",en_iyi_parametreler)
60
61 model = DecisionTreeClassifier(criterion=en_iyi_parametreler["criterion"],
62                                 max_depth=en_iyi_parametreler["max_depth"])
63 clf = model.fit(Giris_train,Cikis_train)
64 Cikis_pred = clf.predict(Giris_test)
```

Şekil 132. Karar ağaçları algoritmasında eğitim ve tahminleme işlemleri için kod ekranı.

UYGULAMANIN PYTHON KODLARI (Başarım: %81)

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

covidli="covid_veri-seti/COVID/"
covid_olmayan="covid_veri-seti/non-COVID/"

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=="covidli":
            veriler=np.append (goruntu_donusturme, [0])
        elif sinif_adi=="covid_olmayan":
            veriler=np.append (goruntu_donusturme, [1])
        else:
            continue
        goruntu_sinif.append(veriler)
    return goruntu_sinif

covidli_veri=veri_donusturme(covidli,"covidli")
covidli_df=pd.DataFrame(covidli_veri)

covidli_olmayan_veri=veri_donusturme(covid_olmayan,"covid_olmayan")
covidli_olmayan_df=pd.DataFrame(covidli_olmayan_veri)
```

```

tum_veri= pd.concat([covidli_df, covidli_olmayan_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))

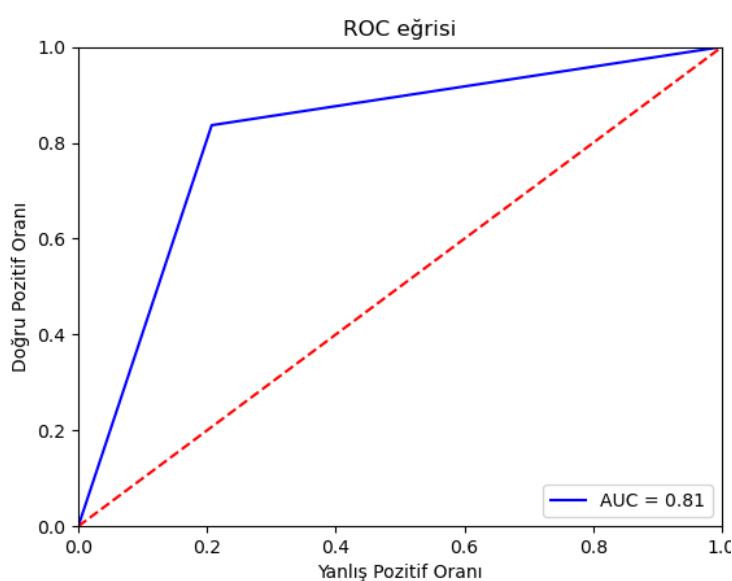
import matplotlib.pyplot as cizim
ypo, dpo, threshold = metrics.roc_curve(Cikis_test, Cikis_pred)
roc_auc = metrics.auc(ypo, dpo)

```

```

cizim.title('ROC eğrisi')
cizim.plot(ypo, dpo, 'b', label = 'AUC = %0.2f' % roc_auc)
cizim.legend(loc = 'lower right')
cizim.plot([0, 1], [0, 1],'r--')
cizim.xlim([0, 1])
cizim.ylim([0, 1])
cizim.ylabel('Doğru Pozitif Oranı')
cizim.xlabel('Yanlış Pozitif Oranı')
cizim.show()

```



Şekil 133. Elde edilen ROC eğrisi.

Şekil 134. Karar ağaçları modelin en iyi kriterler ve maksimum derinlik hiper parametrelerine göre doğruluk oranı.

Karar ağaçları algoritması ile eğitimi esnasında modellerin doğrulukları birbirinden farklı değerler çıkabilecektir. Birbirinden farklı değerler çıkması genellikle veri seti ile ilgili bir durumdur. Kod satırında yer alan “random_state=109” kod satırında 109 rakamını değiştirek (örneğin “0”, “1”, vs.) modelin doğruluğun değiştiği gözlemlenir ve tartışılar. Bu noktada öğrencilere durumu göstermek adına en yüksek doğruluk elde edilinceye kadar model eğitimi gerçekleştirilebilecektir.



4. ADIM: İLERLET

İlerlet adımda öğrencilerin gruplar halinde çalışarak, gerçekleştirilen uygulama üzerinde kendi düzenlemelerini yapmaları ve elde edilen sonuçlar hakkında notlar almaları istenir. Bu amaçla 30 dakikalık bir çalışma süresi verilir. Çalışma süresi esnasında öğrencilerin yaptıkları düzenlemeler uzaktan gözlemlenir, düzenlemelere esnasında sorusu olan öğrencilerin soruları cevaplandırılıp, yardım edilir.



5. ADIM: DEĞERLENDİRME

Derste edinilen bilgiler soru-cevap aşaması gerçekleştirilmek suretiyle pekiştirilir.

Öğrencilere şu sorular sorulabilir:

- Gerçekleştirilen Makine Öğrenmesi uygulamasından elde ettiğiniz bilgi ve beceriler neler olmuştur?
- Karar ağaçları algoritması hakkındaki görüşleriniz nelerdir?
- Uyguladığınız Python kodlarında veri hazırlığı ve Makine Öğrenmesi modelinin oluşturularak eğitim-test süreçlerinden geçirilmesi aşamalarına dair görüşleriniz nelerdir?
- Benzeri bir uygulamayı gerçekleştirebileceğiniz problemler neler olabilir?

2. HAFTA – 4. GÜN – 1., 2., 3. DERSLER: YAPAY SINİR AĞLARI İLE BİNA ENERJİ VERİMLİLİĞİNİN TAHMİNİ

DERS PLANI

DERS ETİKETLERİ



KONU: Yapay Sinir Ağları ile Bina Enerji Verimliliğinin Tahmini



SINIF: 12-14 Yaş



SÜRE: 180 dk.



HAZIR BULUNUŞLUK: İngilizce eğitimi, Yapay Zeka kavramı ve uygulamaları hakkında temel bilgi, Makine Öğrenmesi hakkında temel bilgi, Python programlama dili hakkında temel bilgi



ANAHTAR KELİMELER: Yapay Zeka, Algoritma, Kodlama, Python, Yapay Sinir Ağları



BECERİLER:



Tahmin



Sorgulama



Grup çalışması



Günlük hayat bağlantısı kurma



KAZANIMLAR:

Bu derste öğrenciler;

- Yapay sinir ağları hakkında bilgi ve beceri sahibi olurlar.
- Python ile yapay sinir ağları yardımıyla gerçekleştirilen bir regresyon tabanlı Makine Öğrenmesi uygulamasını gerçekleştirirler.
- Python ortamında çeşitli sayısal girdi değerlerini içeren bir veri seti üzerinden regresyona dayalı bir Makine Öğrenmesi çözümünün nasıl gerçekleştirilebileceği hakkında bilgi ve beceri sahibi olurlar.



DERS MATERİYALLERİ:

Malzeme	Miktar
Yazı Tahtası	Öğrenci sayısı ve sınıf büyüğününe göre en az bir adet
Projeksiyon	Öğrenci sayısı ve sınıf büyüğününe göre en az bir adet
Defter, Kalem	Öğrenci sayısı kadar
Bilgisayar	Öğrenci sayısının yarısı kadar
Tablet	Öğrenci sayısının yarısı kadar
Python Anaconda Framework	Her bilgisayarda kurulu olacak şekilde



ÖĞRETMEN HAZIRLIK ÇALIŞMALARI:

Okunabilecek Kaynaklar:

Bulutistan. (2023). *Sinir Ağları Nedir? Sinir Ağları Nasıl Çalışır?*. Çevrimiçi: <https://bulutistan.com/blog/sinir-aglari-nedir-sinir-aglari-nasil-calisir/> (Erişim 8 Haziran 2024).

Kaggle. (2022). *ANN-Projects + Neural Networks*. Çevrimiçi: <https://www.kaggle.com/code/mukeshmanral/ann-projects-neural-networks> (Erişim 8 Haziran 2024).

Keskenler, M. F., & Keskenler, E. F. (2017). Geçmişten günümüze yapay sinir ağları ve tarihçesi. *Takvim-i Vekayi*, 5(2), 8-18.

Öztemel, E. (2003). *Yapay sinir ağları*. Papatya Yayıncılık.

Yıldırım, E. (2020). *Yapay Sinir Ağı (Artificial Neural Network) Nedir?*. Veri Bilimi Okulu. Çevrimiçi: <https://www.veribilimiokulu.com/yapay-sinir-agı-artificial-neural-network-nedir/> (Erişim 9 Haziran 2024).

İzlenebilecek Kaynaklar:

Matematik ve Yapay Zeka Enstitüsü. (2023). Yapay Sinir Ağları - 1. Genel Bakış. Çevrimiçi: https://www.youtube.com/watch?v=2Udvp9JkXyQ&list=PLU8l6n1Iz_GGg2loPlxt2zQmX6bHaH6i6 (Erişim 9 Haziran 2024).

IBM Technology. (2022). Neural Networks Explained in 5 minutes. Çevirmiçi: <https://www.youtube.com/watch?v=jmmW0F0biz0> (Erişim 9 Haziran 2024).

ATÖLYE HAZIRLIK ÇALIŞMALARI:

Öğretmen;

- Dersten önce ders materyallerini hazırlamalıdır.
- Python kodlama için kullanılan Spyder editörünün genel kullanımına aşina olmalıdır.
- Derse konu olan veri set(ler)ini ve Python kodlarını önceden inceleyerek üzerinde çalışmış olmalıdır.



KAYNAKÇA:

Bottou, L. (1991). Stochastic gradient learning in neural networks. *Proceedings of Neuro-Nimes*, 91(8), 12.

Deperlioğlu, Ö., & Köse, U. (2024). *Python ile Makine Öğrenmesi: Temel Kavramlar – Sınıflandırma – Regresyon – Kümeleme*. Seçkin Yayıncılık.

Ketkar, N., & Ketkar, N. (2017). Stochastic gradient descent. *Deep learning with Python: A hands-on introduction*, 113-132.

Köse, U., Özsoy, K., & Aksoy, B. (2023). *Yapay Zeka – Ortaokul ve Lise Kitapları*. Deneyap Teknoloji Atölyeleri, TÜBİTAK Yayıncılık.

Öztemel, E. (2003). *Yapay sinir ağları*. Papatya Yayıncılık.

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

GENEL BAKIŞ:

- 1) Harekete Geç:** Öğrencilere uygulamaya konu yapay sinir ağları tekniği anlatılır ve Python kodlama ile Makine Öğrenmesi sürecine yönelik ön hazırlık gerçekleştirilir.
- 2) Keşfet:** Öğrencilerle birlikte ilgili Makine Öğrenmesi modelinin çalıştırılmasına dair kodlamaların yapılması sağlanır.
- 3) Üret:** Öğrencilerle birlikte çalıştırılan Makine Öğrenmesi modelinin elde ettiği sonuçların değerlendirilmesine yönelik kodlamaların yapılması sağlanır.
- 4) İlerlet:** Öğrencilerin gruplar halinde çalışarak, gerçekleştirilen uygulama üzerinde kendi düzenlemelerini yapmaları ve elde edilen sonuçlar hakkında notlar almaları istenir.
- 5) Değerlendir:** Derste edinilen bilgiler soru-cevap aşaması gerçekleştirilmek suretiyle pekiştirilir.

UYGULAMA

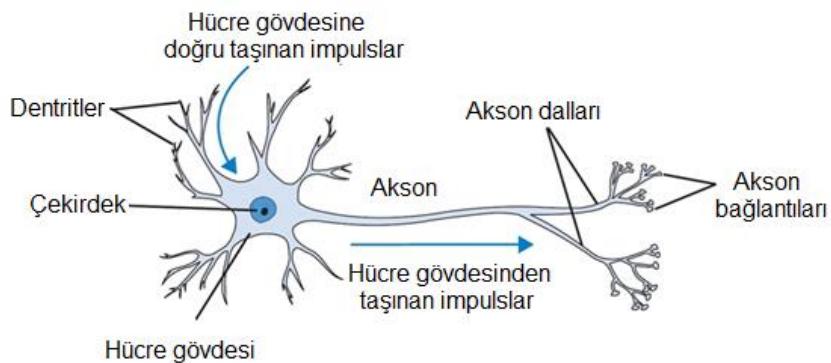


1. ADIM: HAREKETE GEÇ

Bu adım kapsamında öğrencilere uygulamaya konu yapay sinir ağları teknigi anlatılır ve Python kodlama ile Makine Öğrenmesi sürecine yönelik ön hazırlık gerçekleştirilir.

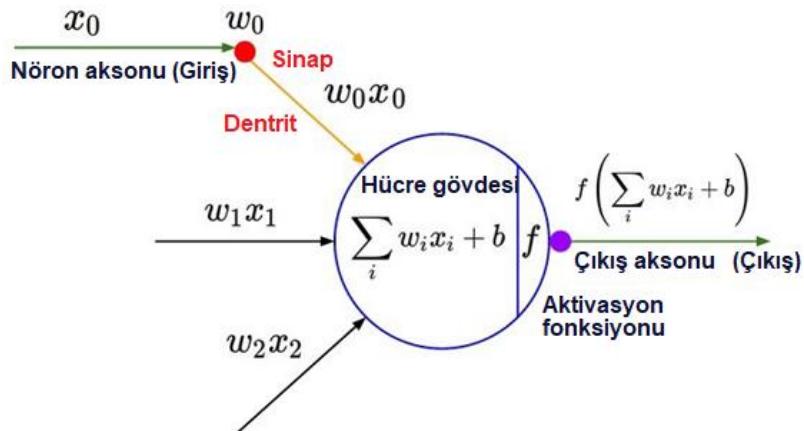
Yapay Sinir Ağları Tekniği

Canlıların davranışlarını inceleyip, matematiksel olarak modelleyip, benzer yapay modellerin üretilmesine “**sibernetik**” denir. Eğitilebilir, adaptif ve kendi kendine organize olup öğrenebilen ve değerlendirmeye yarayan yapay sinir ağları ile insan beyninin öğrenme yapısının modellenmesi amaçlanmıştır. Bu bakımdan Yapay Sinir Ağları teknigi, içerisindeki esnek matematiksel yapı sayesinde, Yapay Zeka'nın en güçlü tekniği olma unvanını elinde tutmaktadır. Yapay sinir ağları vasıtıyla tipki insanoğlunda olduğu gibi makinelerin eğitilmesi, öğrenmesi ve karar vermesi amaçlanmaktadır (Köse vd., 2023). Şekil 135'te insan sinir hücresinin (nöron) örnek görseli verilmiştir (Öğrencilere Şekil 135'te verilen insan sinir hücresinin yapısı, çalışma prensibi ve özellikleri kapsamlı bir biçimde anlatılır).



Şekil 135. İnsan sinir hücresinin yapısı.

Şekil 136'da görüldüğü gibi; yapay sinir ağları insan sinir hücrelerine benzer bir yapıda giriş değerlerine göre (x_0, x_1, x_2 vs.) ağırlıkları belirlenip (w_0, w_1, w_2, \dots vs.) çarpılarak elde edilen sonuçların toplanması ilkesine dayanmaktadır. Yapay sinir ağlarında istenilirse sabit bir bias değeri eklenerek belirlenen aktivasyon fonksiyonuna göre hem sınıflandırmada hem regresyon problemlerinde hem de kümeleme problemlerinde sıkılıkla kullanılmaktadır (Deperlioğlu, & Köse, 2024; Köse vd., 2023). Bu aşamada öğrencilere yapay sinir ağları teknigideki temel mantığı açıklamak için üzerinde çok çalıştığımız bir konuyu pekiştirme yaptıkça hatırlamamızı sağlayan, beyin hücreleri arasındaki artan elektriksel yükler anlatılır. Bu noktada üzerinde çok düşünülmeyen konuların hücreler arası elektriksel yüklerin azalması nedeniyle unutulacağı da ifade edilir. Yine beyin hücreleri arasındaki elektriksel bağların, yapay sinir ağları içerisindeki hücrede ağırlık değerleriyle karşılandığını açıklayarak, doğal ve yapay sinir hücresi arası esin kaynakları irdelenir.



Şekil 136. Temel yapay sinir hücresinin matematiksel modeli.

Temel bir yapay sinir ağı modelinin matematiksel denklemi Şekil 137'de 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.

$$y = W \times X + b$$

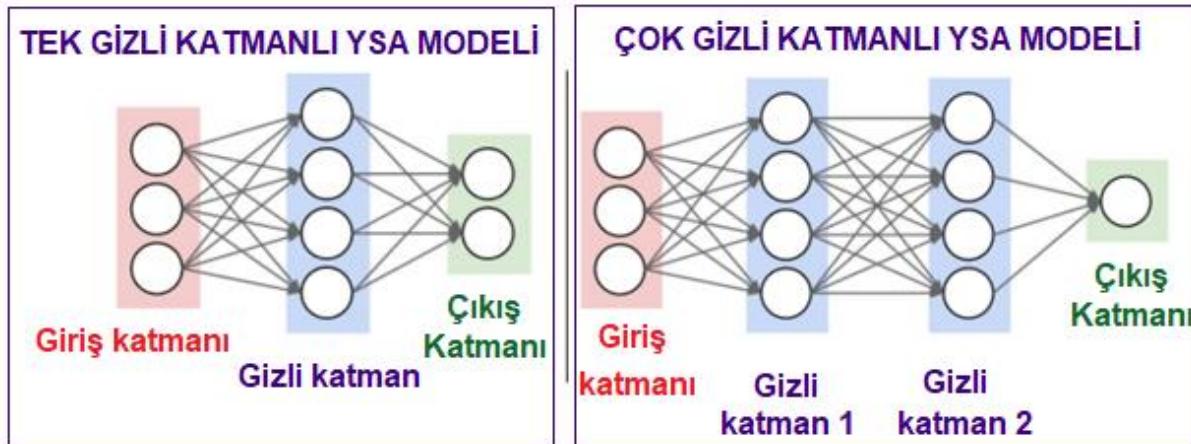
Şekil 137. Yapay sinir ağlarının matematiksel denklemi.

Tek ve Çok Katmanlı Yapay Sinir Ağ Modelleri

Yapay sinir ağları (YSA) tek ve çok katmanlı olmak üzere iki farklı model olarak tasarılanabilir. 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 138'de 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 genel olarak şöyle hesaplanabilmektedir:

- Giriş katmanındaki nöron sayısı (3) \times gizli katmandaki nöron sayısı (4) + gizli katmandaki nöron sayısı (4) \times 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 parametresi 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 parametresi hesaplanır.



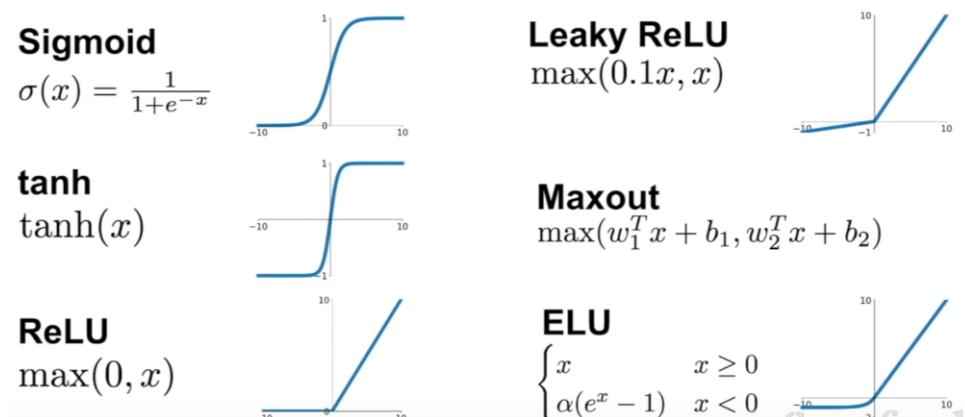
Şekil 138. Tek ve çok gizli katmanlı yapay sinir ağ model yapıları.

Yapay Sinir Ağlarında Nöronlar

Yapay sinir ağında katmanlar içinde kullanılan nöronların birbirleri ile ilişkisi yoktur. En önemli görevleri sistemde olan bilgiyi bir sonraki katmana ya da çıkış katmanına aktarma görevini üstlenirler. Ardı ardına gelen iki katmandaki nöronlar farklı aktivasyon değerlerine göre YSA modelin öğrenme seviyesini belirleyip aktarım işlemi gerçekleştirir. Yapay sinir ağlarında nöron sayısı, giriş parametresi ve katman sayısı modelin önemli parametrelerindendir (Köse vd., 2023; Öztemel, 2003). Pek çok hiper parametre kullanılarak çıkış performansı artırlabilir. W ağırlık vektörü ise düğüm/nöron sayısı (hücre), bias (b) değerleri de gelecek katmandaki düğüm sayısı kadar olmalıdır.

Yapay Sinir Ağlarında Aktivasyon Fonksiyonları

Yapay sinir ağı modelinde regresyon veya sınıflandırma probleminin çözümü için kurulan modelde farklı aktivasyon fonksiyonları kullanılabilir (Şekil 139) (Köse vd., 2023).



Şekil 139. Yapay sinir ağlarında sıkılıkla kullanılan aktivasyon fonksiyonları.

- **Sigmoid Fonksiyonu:** YSA modellerinde sıkılıkla 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 sınıf'a ait öğrenen olasılıksal bir değer üretir.
- **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.
- **Rectified Linear Unit / Rektifiye Doğrusal Birim (ReLU) Fonksiyonu:** YSA modellerinde ReLU aktivasyon fonksiyonu çok gizli katmanlarda sıkılıkla 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 modeli eğitir.
- **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 yönde de gerçekleşmesi sağlayan bir aktivasyon fonksiyonudur.
- **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 gönderilir.
- **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.

Yapay Sinir Ağları Kullanım Alanları

Günümüzde bilgisayarlar ve bilgisayar sistemleri yaşamımızın vazgeçilmez bir parçası haline gelmiştir. Yapay zekanın alt dallarından birisi olan Yapay Sinir Ağları, bilgisayarların derin öğrenme yöntemlerinde sıkılıkla kullanılan bir yapısıdır. 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 sıkılıkla kullanılmaktadır (Köse vd., 2023).



2. ADIM: KEŞFET

Bu adımda öğrencilerle birlikte ilgili Makine Öğrenmesi modelinin çalıştırılmasına dair kodlamaların yapılması sağlanır.

Öğrenciler, ilk olarak yapay sinir ağları ile farklı binalara ait rölatif sıkılık, binanın yüzey, duvar ve çatı alanı, binanın yüksekliği, bina yerlesimi (oryantasyon), cam alanı ve camın alan

dağılımı giriş parametrelerine göre ısıtma ve soğutma yükü değerlerini hesaplayıp değerlendirilme problemini anlar.

Öğrenciler Tablo 4'te verilen yapay sinir ağları ile bina enerji verimliliği analizi için giriş çıkış parametresini belirler.

Tablo 4. 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

İlgili uygulamada farklı binalara ait 768 adet veri setinde (Tsanas, & 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 değerlerini 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/raw/main/Hafta5/ENB2012_data.xlsx

Öğrencilere veri seti anlatılırken veri seti indirme linkinden indirilen farklı binalara ait veriler bağlamında, giriş ve çıkış parametrelerine ait şu tanımlamalar ifade edilir:

- 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

Kodlama aşamasında, 1 nolu kod satırında Yapay Zeka kütüphanesi olan **keras** kütüphanesinin **Sequential** fonksiyonu ile boş bir yapay sinir ağı modeli oluşturmak için gerekli olan sınıf çağrırlır.

2 ve 3 nolu kod satırlarında Yapay Zeka kütüphanesi olan **keras** kütüphanesinin **Dense**, **Input** ve **Dropout** fonksiyonu ile yapay sinir ağı tam bağlı katmanları tanımlamak için gerekli olan sınıf çağrılarıdır.

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

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

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

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

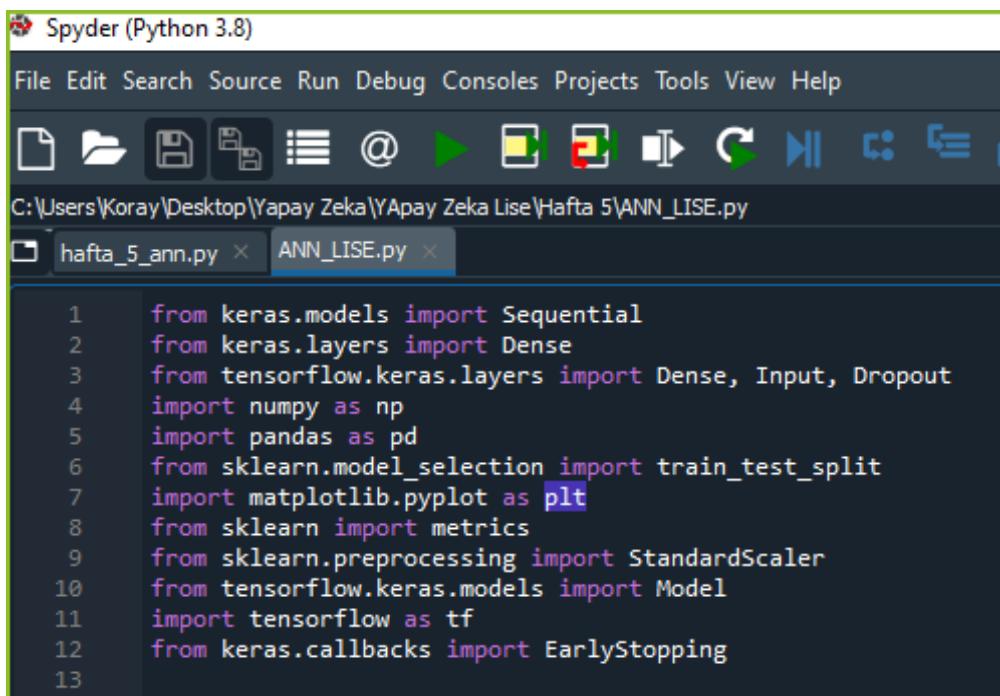
8 nolu kod satırında ise **sklearn** kütüphanesinden **metrics** fonksiyonu ile modelin değerlendirilebilmesi için gerekli fonksiyon çağrılr.

9 nolu kod satırında ise veri ölçeklendirme işlemi için **sklearn.preprocessing** kütüphanesinden **StandardScaler** fonksiyonu çağrılr.

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

11 nolu kod satırında ise Yapay Zeka kütüphanesi olan **tensorflow** kütüphanesi çağrılr.

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



The screenshot shows the Spyder Python 3.8 IDE interface. The title bar says "Spyder (Python 3.8)". The menu bar includes File, Edit, Search, Source, Run, Debug, Consoles, Projects, Tools, View, Help. Below the menu is a toolbar with various icons. The current file path is "C:\Users\Koray\Desktop\Yapay Zeka\YApay Zeka Lise\Hafta 5\ANN_LISE.py". There are two tabs open: "hafta_5_ann.py" and "ANN_LISE.py", with "ANN_LISE.py" being the active tab. The code editor displays the following Python code:

```
1  from keras.models import Sequential
2  from keras.layers import Dense
3  from tensorflow.keras.layers import Dense, Input, Dropout
4  import numpy as np
5  import pandas as pd
6  from sklearn.model_selection import train_test_split
7  import matplotlib.pyplot as plt
8  from sklearn import metrics
9  from sklearn.preprocessing import StandardScaler
10 from tensorflow.keras.models import Model
11 import tensorflow as tf
12 from keras.callbacks import EarlyStopping
13
```

Şekil 140. Gerekli kütüphanelerin çağrılmaması.

Öğrenciler Şekil 141'de görüldüğü veri seti ile ilgili düzenlemeleri 14-19 kod satırları arasında gerçekleştirir.

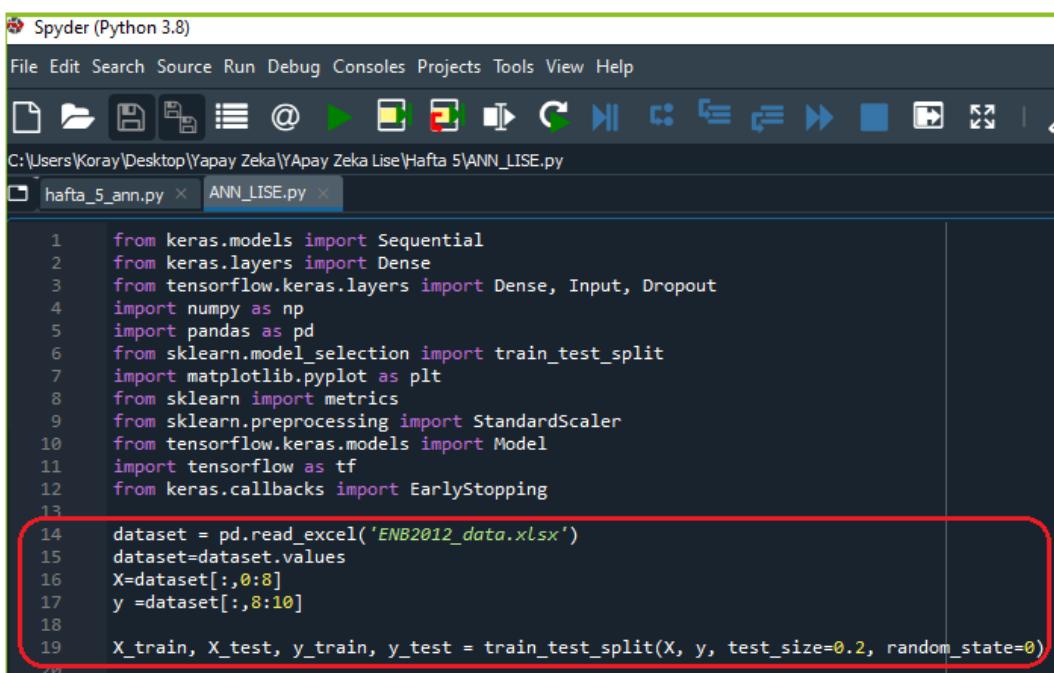
14 kod satırında pandas kütüphanesinin **read_excel** özelliğini kullanarak veri seti ile aynı konumda olan kod dosyasının içerisindeki 'ENB2012_data.xlsx' veri dosyasını okuyarak dataset değişkenine aktarır.

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

16 kod satırında dataset değişkenindeki veri setindeki ilk sekiz sütunu (`[:,0:8]`) giriş parametresi olarak ayarlayarak "X" değişkenine aktarır. Kod satırında yer alan (`[:,0:8]`) ifadesinde virgülden önce yer alan ":" ifadesinin yazılması sebebi veri setindeki tüm satırları okumak içindir. "**0:8**" ifadesi ise veri setinde yer alan ilk sekiz sütun olan giriş parametresi olarak yer olmasını sağlar.

17 kod satırında dataset değişkenindeki 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 ayırrır.



The screenshot shows the Spyder Python 3.8 IDE interface. The menu bar includes File, Edit, Search, Source, Run, Debug, Consoles, Projects, Tools, View, Help. The toolbar has various icons for file operations like Open, Save, Run, and Stop. The current working directory is C:\Users\Koray\Desktop\Yapay Zeka\Yapay Zeka Lise\Hafta 5\ANN_LISE.py. Two tabs are open: hafta_5_ann.py and ANN_LISE.py. The code editor contains the following Python script:

```
1  from keras.models import Sequential
2  from keras.layers import Dense
3  from tensorflow.keras.layers import Dense, Input, Dropout
4  import numpy as np
5  import pandas as pd
6  from sklearn.model_selection import train_test_split
7  import matplotlib.pyplot as plt
8  from sklearn import metrics
9  from sklearn.preprocessing import StandardScaler
10 from tensorflow.keras.models import Model
11 import tensorflow as tf
12 from keras.callbacks import EarlyStopping
13
14 dataset = pd.read_excel('ENB2012_data.xlsx')
15 dataset=dataset.values
16 X=dataset[:,0:8]
17 y =dataset[:,8:10]
18
19 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
```

A red rectangle highlights the code block from line 14 to line 19, which reads the Excel file, extracts the first eight columns as X, and the last two columns as y, then splits the data into training and testing sets.

Şekil 141. Veri setinin giriş, çıkış ve eğitim için bölünmesi kod ekranı.

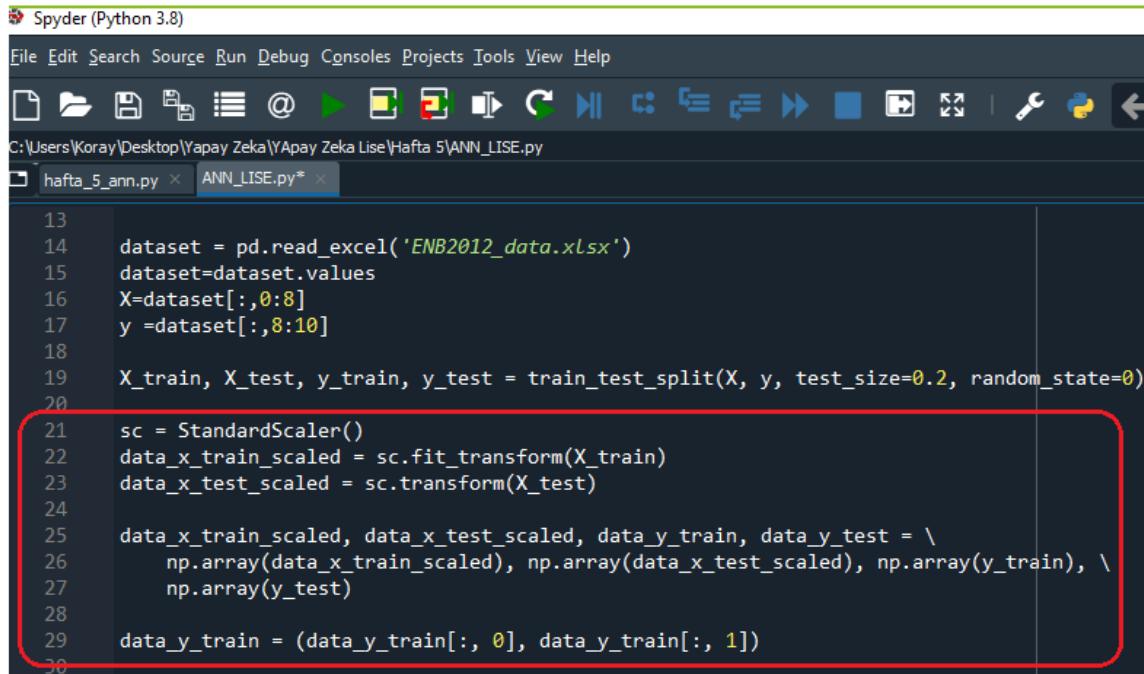
Öğrenciler 21-29 kod satırları arasında veri setinin ölçeklendirilmesi işlemini gerçekleştirir.

21 kod satırında 9. Kod satırında çağrılan **StandardScaler** fonksiyonu tanımlayarak sc değişkenine aktarılır.

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 dönüştürülmüştür.

29 kod satırında ise YSA ile eğitilecek modelden iki adet çıkış parametresi olacağı için “tuple” formatına dönüştürülmüştür. Buna göre ilgili tuple formatı kullanımı şu şekildedir: Kod satırında yer alan ([:,0]) ifadesinde virgülden önce yer alan “:” ifadesinin yazılması sebebi ile çıkışlardan ilk sütunu alıp ([:,1]) çıkışlardan ikinci sütunu alıp eşleştirmek için kullanılır.



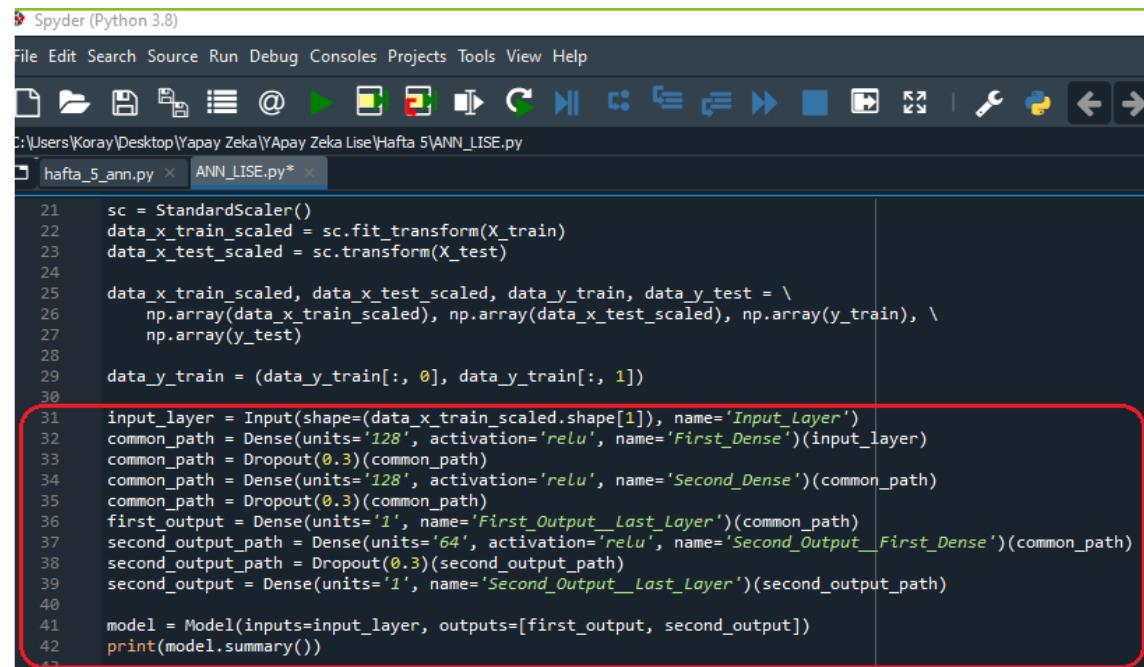
The screenshot shows the Spyder Python 3.8 IDE interface. The menu bar includes File, Edit, Search, Source, Run, Debug, Consoles, Projects, Tools, View, and Help. The toolbar contains various icons for file operations like Open, Save, and Run. The current working directory is C:\Users\Koray\Desktop\Yapay Zeka\Yapay Zeka Lise\Hafta 5\ANN_LISE.py. Two tabs are open: hafta_5_ann.py and ANN_LISE.py*. The code in hafta_5_ann.py is as follows:

```
13
14     dataset = pd.read_excel('ENB2012_data.xlsx')
15     dataset=dataset.values
16     X=dataset[:,0:8]
17     y =dataset[:,8:10]
18
19     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
20
21     sc = StandardScaler()
22     data_x_train_scaled = sc.fit_transform(X_train)
23     data_x_test_scaled = sc.transform(X_test)
24
25     data_x_train_scaled, data_x_test_scaled, data_y_train, data_y_test = \
26         np.array(data_x_train_scaled), np.array(data_x_test_scaled), np.array(y_train), \
27         np.array(y_test)
28
29     data_y_train = (data_y_train[:, 0], data_y_train[:, 1])
30
```

A red box highlights the scaling code from line 21 to 29.

Şekil 142. Veri setinin ölçeklendirilmesi.

Şekil 143'te gösterildiği gibi 31-39 kod satırları arasında yer alan YSA modelinin yapısının oluşumu ile ilgili katmanlar bilgileri verilmiştir.



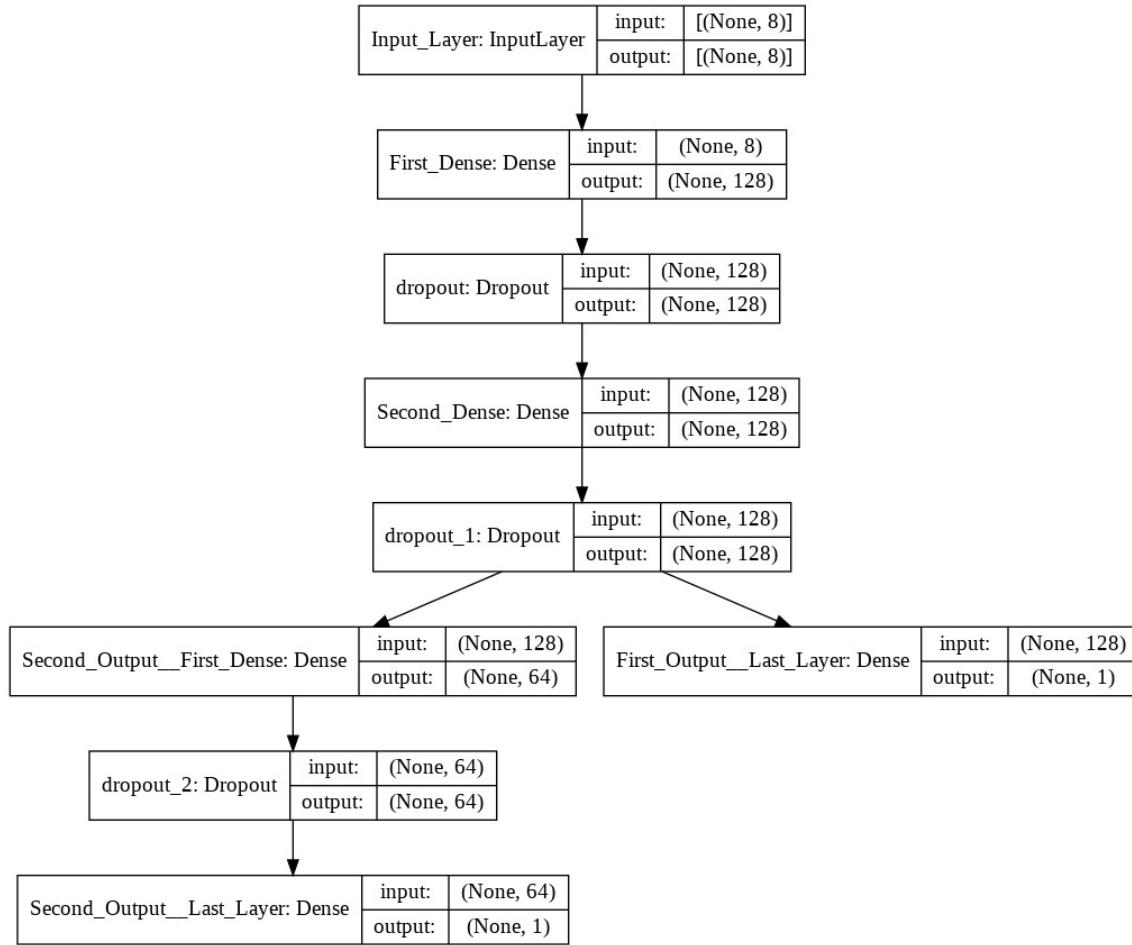
The screenshot shows the Spyder Python 3.8 IDE interface. The menu bar includes File, Edit, Search, Source, Run, Debug, Consoles, Projects, Tools, View, and Help. The toolbar contains various icons for file operations like Open, Save, and Run. The current working directory is C:\Users\Koray\Desktop\Yapay Zeka\Yapay Zeka Lise\Hafta 5\ANN_LISE.py. Two tabs are open: hafta_5_ann.py and ANN_LISE.py*. The code in ANN_LISE.py is as follows:

```
21     sc = StandardScaler()
22     data_x_train_scaled = sc.fit_transform(X_train)
23     data_x_test_scaled = sc.transform(X_test)
24
25     data_x_train_scaled, data_x_test_scaled, data_y_train, data_y_test = \
26         np.array(data_x_train_scaled), np.array(data_x_test_scaled), np.array(y_train), \
27         np.array(y_test)
28
29     data_y_train = (data_y_train[:, 0], data_y_train[:, 1])
30
31     input_layer = Input(shape=(data_x_train_scaled.shape[1]), name='Input_Layer')
32     common_path = Dense(units='128', activation='relu', name='First_Dense')(input_layer)
33     common_path = Dropout(0.3)(common_path)
34     common_path = Dense(units='128', activation='relu', name='Second_Dense')(common_path)
35     common_path = Dropout(0.3)(common_path)
36     first_output = Dense(units='1', name='First_Output_Last_Layer')(common_path)
37     second_output_path = Dense(units='64', activation='relu', name='Second_Output_First_Dense')(common_path)
38     second_output_path = Dropout(0.3)(second_output_path)
39     second_output = Dense(units='1', name='Second_Output_Last_Layer')(second_output_path)
40
41     model = Model(inputs=input_layer, outputs=[first_output, second_output])
42     print(model.summary())
43
```

A red box highlights the model definition code from line 31 to 42.

Şekil 143. Yapay sinir ağları modelinin kurulumu için kod ekranı.

31. kod satırında giriş parametresi ile eşit nöron sayılı giriş katmanı oluşturulur.
32. kod satırında 128 nöronlu ve Relu aktivasyon fonksiyonlu ilk gizli katmanı oluşturarak 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 katman 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 kullanarak YSA modelin giriş ve çıkış katmanları tanımlanarak “**model**” değişkenine aktarılmıştır.



Şekil 144. Yapay sinir ağları model yapısının algoritmik gösterimi.

Şekil 144'te gösterildiği gibi 42. kod satırında oluşturulan yapay sinir ağı modelindeki nöronların, giriş çıkış parametrelerine ait özellikleri “**.summary**” fonksiyonu ile görülmektedir (Şekil 145).

Layer (type)	Output Shape	Param #	Connected to
<hr/>			
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)	(None, 64)	8256	dropout_1[0][0]
dropout_2 (Dropout)	(None, 64)	0	Second_Output_First(Dense)[0][0]
First_Output_Last(Dense)	(None, 1)	129	dropout_1[0][0]
Second_Output_Last(Dense)	(None, 1)	65	dropout_2[0][0]
<hr/>			
Total params:	26,114		
Trainable params:	26,114		
Non-trainable params:	0		

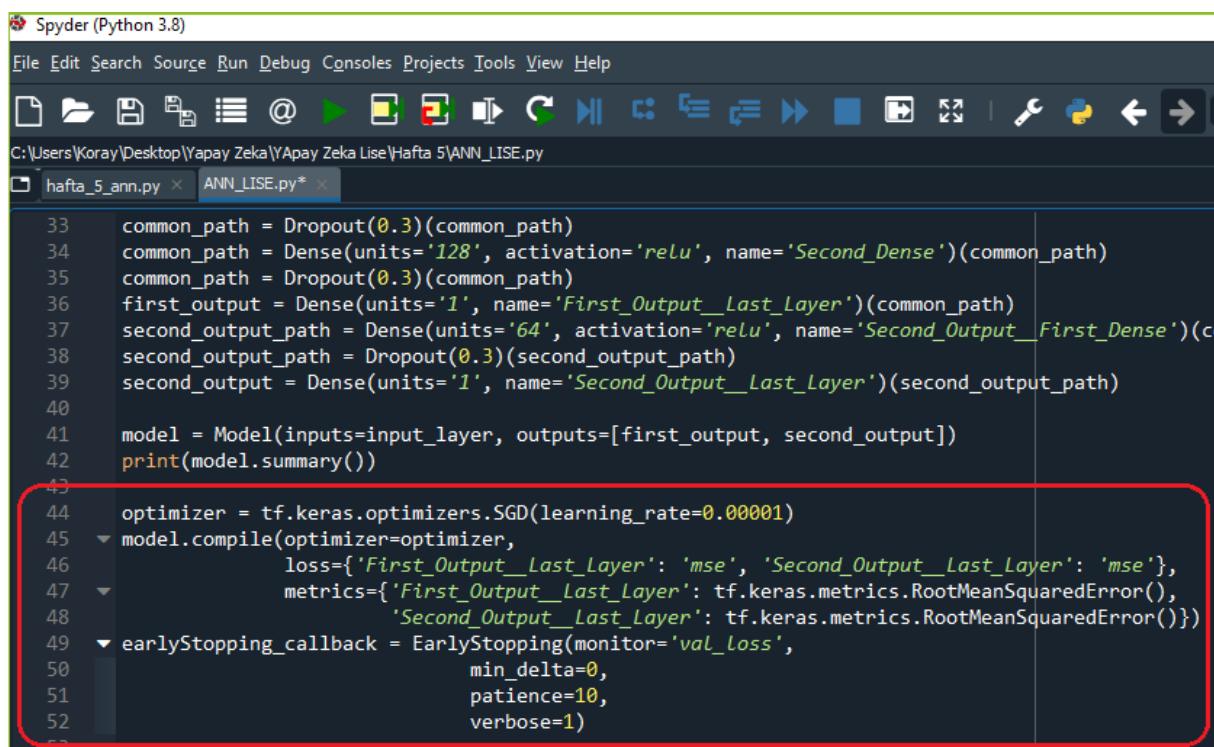
Şekil 145. Yapay sinir ağları modelinin özeti.

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. Makine öğrenmesinde hata oranını en aza indirmek kullanılan yöntemlerinden birisi de SGD optimizasyon yöntemidir. SGD (Stochastic Gradient Descent) tüm gradient’ler kullanmak yerine rastgele bir kısım gradient’le ağırlıkları değiştirilerek optimizasyon yapılmaktadır. Mevcut gradient’i ($\partial L / \partial w_t$), öğrenme katsayısı (a) ile çarparak mevcut ağırlığı (w_t) güncellemektedir (Bottou, 1991; Ketkar, & Ketkar, 2017):

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

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. Öğrencilere kod satırında yer alan “mse”, ifadesinin Makine Öğrenmesi modelinin ortalama kare hatasını bir dizi noktaya ne kadar yakın olduğunu belirleyen metrik olduğu ifade edilir. “RootMeanSquaredError (RMSE)” ifadesinin ise Makine Öğrenmesi modelinin tahminleyicinin tahmin ettiği değerler ile gerçek değerleri arasındaki uzaklığın bulunması adına kullanıldığı açıklanır.

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 modelinin doğruluğunun değişmez ise eğitim durdurulur. “**verbose=1**” ise eğitim esnasında eğitim sonuçları ekranda gösterir.



```

Spyder (Python 3.8)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\Koray\Desktop\Yapay Zeka\Yapay Zeka Lise\Hafta 5\ANN_LISE.py
hafta_5_ann.py ANN_LISE.py*
33 common_path = Dropout(0.3)(common_path)
34 common_path = Dense(units='128', activation='relu', name='Second_Dense')(common_path)
35 common_path = Dropout(0.3)(common_path)
36 first_output = Dense(units='1', name='First_Output_Last_Layer')(common_path)
37 second_output_path = Dense(units='64', activation='relu', name='Second_Output_First(Dense')(common_path)
38 second_output_path = Dropout(0.3)(second_output_path)
39 second_output = Dense(units='1', name='Second_Output_Last_Layer')(second_output_path)
40
41 model = Model(inputs=input_layer, outputs=[first_output, second_output])
42 print(model.summary())
43
44 optimizer = tf.keras.optimizers.SGD(learning_rate=0.00001)
45 model.compile(optimizer=optimizer,
46                 loss={'First_Output_Last_Layer': 'mse', 'Second_Output_Last_Layer': 'mse'},
47                 metrics={'First_Output_Last_Layer': tf.keras.metrics.RootMeanSquaredError(),
48                          'Second_Output_Last_Layer': tf.keras.metrics.RootMeanSquaredError()})
49 earlyStopping_callback = EarlyStopping(monitor='val_loss',
50                                         min_delta=0,
51                                         patience=10,
52                                         verbose=1)
53

```

Şekil 146. Yapay sinir ağı modelinin parametrelerinin tanımlanması.

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

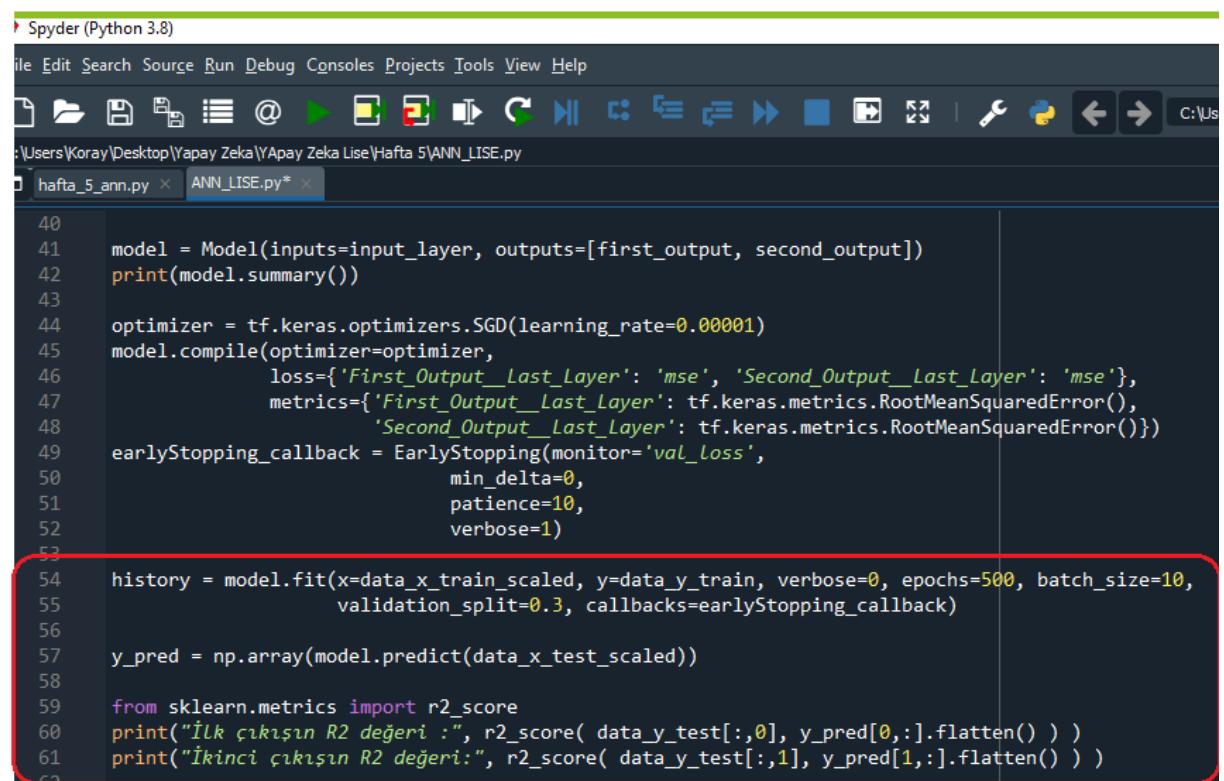
54-55. kod satırında “**model.fit**” komutu yazarak ölçeklendirilmiş giriş eğitim verilerine göre çıkış eğitim verileri için yapay sinir ağları ile eğitim gerçekleştirir. Burada her bir eğitim için “**batch_size**” fonksiyonuna aktarılan 10 değeri ile eğitme alınacak veri sayısı belirlenerek 500 adet eğitim (**epochs=500**), “**verbose=0**” ile her bir eğitimde elde edilecek sonuçları konsolda göstermez. Model eğitimden elde edilen sonuçları eğitim verilerinin %30 alarak (**validation_split=0,3**) modeli doğrulama işlemini gerçekleştirir. Callbacks özelliğinin Earlystopping_callback tanımlanarak eğitimin erken durma özelliğini 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 çevrilerek sonuçlar R^2 performans ölçüt kriterine göre doğruluk değerleri “**print**” komutu kullanılarak konsola yazdırılır.

Bu aşamada öğrencilere deneyisel verilerin doğrusal bir eğriye ne kadar uyumlu olduğunu belirlemeye “**Determinasyon katsayı/R²**” metriğinin kullanıldığı ifade edilir. Yapay zeka işlemlerinde elde edilen R^2 değerleri öğrencilerle birlikte tartışılr. $R^2 = 1$ olması durumunun deneyisel verilerin kusursuz bir doğrusal eğri sağlandığının kanıtı olarak anlatılır.



```

Spyder (Python 3.8)
File Edit Search Source Run Debug Consoles Projects Tools View Help
File Explorer IPython Terminal Kernel Help
C:\Users\Koray\Desktop\Yapay Zeka\Yapay Zeka Lise\Hafta 5\ANN_LISE.py
hafta_5_ann.py ANN_LISE.py*
40
41     model = Model(inputs=input_layer, outputs=[first_output, second_output])
42     print(model.summary())
43
44     optimizer = tf.keras.optimizers.SGD(learning_rate=0.00001)
45     model.compile(optimizer=optimizer,
46                     loss={'First_Output_Last_Layer': 'mse', 'Second_Output_Last_Layer': 'mse'},
47                     metrics={'First_Output_Last_Layer': tf.keras.metrics.RootMeanSquaredError(),
48                             'Second_Output_Last_Layer': tf.keras.metrics.RootMeanSquaredError()})
49     earlyStopping_callback = EarlyStopping(monitor='val_loss',
50                                             min_delta=0,
51                                             patience=10,
52                                             verbose=1)
53
54     history = model.fit(x=data_x_train_scaled, y=data_y_train, verbose=0, epochs=500, batch_size=10,
55                           validation_split=0.3, callbacks=earlyStopping_callback)
56
57     y_pred = np.array(model.predict(data_x_test_scaled))
58
59     from sklearn.metrics import r2_score
60     print("İlk çıkışın R2 değeri : ", r2_score( data_y_test[:,0], y_pred[0,:].flatten() ) )
61     print("İkinci çıkışın R2 değeri: ", r2_score( data_y_test[:,1], y_pred[1,:].flatten() ) )

```

Şekil 147. Yapay sinir ağları modelinde eğitim ve tahminleme işlemleri için kod ekranı.

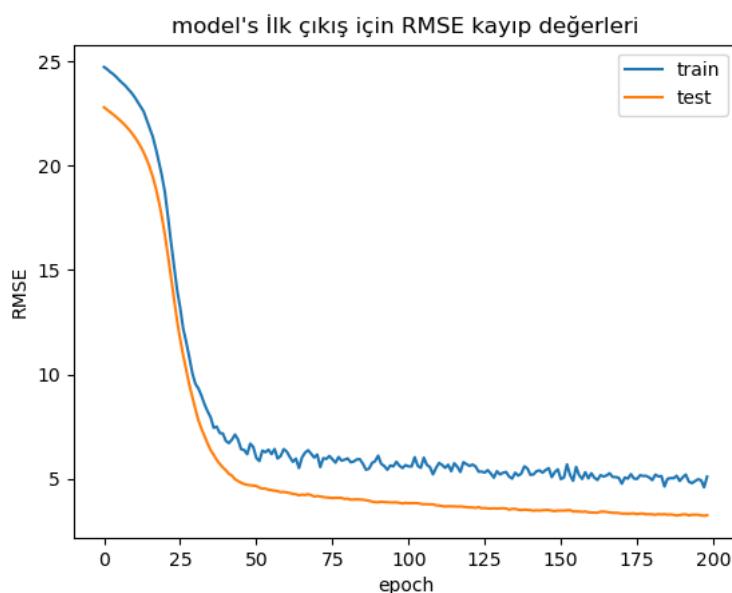


3. ADIM: ÜRET

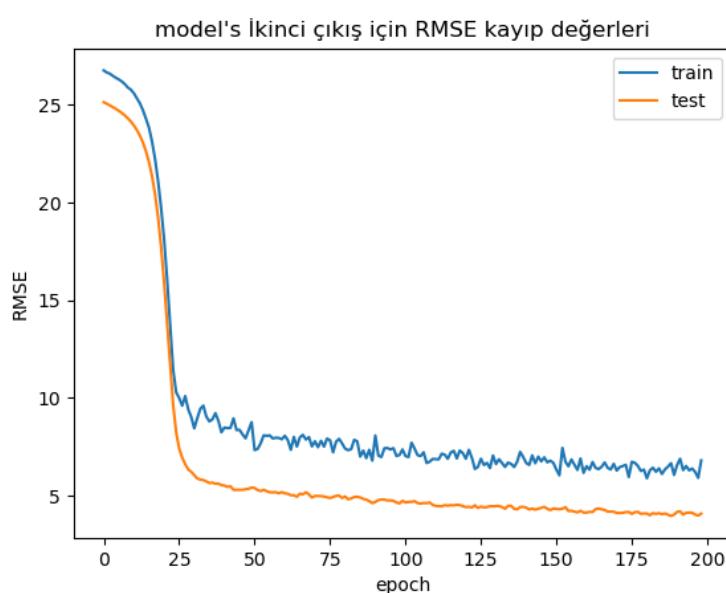
Öğrencilerle birlikte çalıştırılan Makine Öğrenmesi modelinin elde ettiği sonuçların değerlendirilmesine yönelik kodlamaların yapılması sağlanır.

Sonuçların Değerlendirilmesi

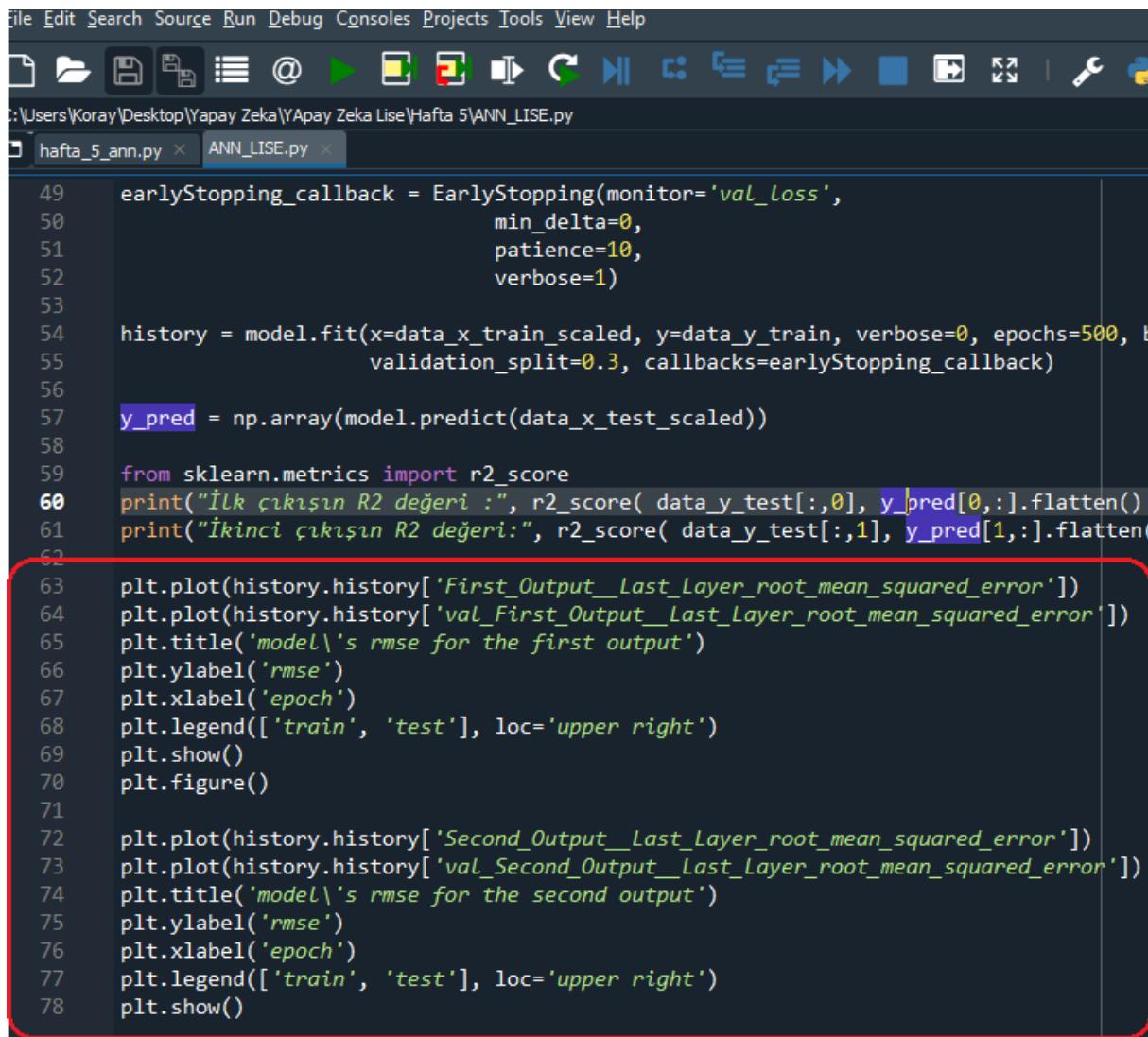
63.-70. ve 72-79. kod satırlarında ısıtma ve soğutma ait kayıp çıkış verilerine göre eğitim ve doğrulama grafiklerin çizdirilmiştir. Şekil 148'te ilk çıkış için RMSE kayıp değerleri, Şekil 149'da ise ikinci çıkış için RMSE kayıp değerlerine ait grafikler çizdirilmiştir. Grafikler incelediğinde eğitim ve test verilerindeki kayıp veri oranlarının azalması modelin doğru biçimde eğitildiğinin göstergesidir.



Şekil 148. İlk çıkış için RMSE kayıp değerleri grafiği.

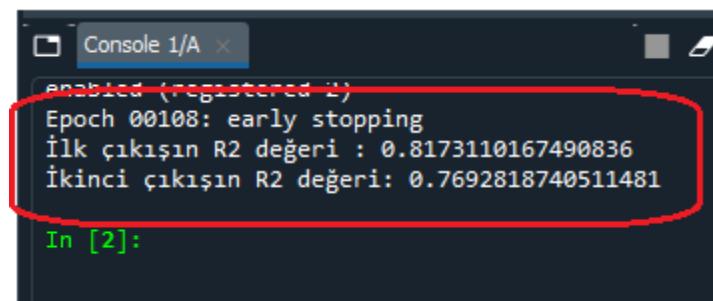


Şekil 149. İkinci çıkış için RMSE kayıp değerleri grafiği.



```
49     earlyStopping_callback = EarlyStopping(monitor='val_loss',
50                                             min_delta=0,
51                                             patience=10,
52                                             verbose=1)
53
54     history = model.fit(x=data_x_train_scaled, y=data_y_train, verbose=0, epochs=500, batch_size=32, validation_split=0.3, callbacks=earlyStopping_callback)
55
56     y_pred = np.array(model.predict(data_x_test_scaled))
57
58     from sklearn.metrics import r2_score
59     print("İlk çıkışın R2 değeri : ", r2_score( data_y_test[:,0], y_pred[0,:].flatten()))
60     print("İkinci çıkışın R2 değeri: ", r2_score( data_y_test[:,1], y_pred[1,:].flatten()))
61
62     plt.plot(history.history['First_Output_Last_Layer_root_mean_squared_error'])
63     plt.plot(history.history['val_First_Output_Last_Layer_root_mean_squared_error'])
64     plt.title('model\'s rmse for the first output')
65     plt.ylabel('rmse')
66     plt.xlabel('epoch')
67     plt.legend(['train', 'test'], loc='upper right')
68     plt.show()
69     plt.figure()
70
71     plt.plot(history.history['Second_Output_Last_Layer_root_mean_squared_error'])
72     plt.plot(history.history['val_Second_Output_Last_Layer_root_mean_squared_error'])
73     plt.title('model\'s rmse for the second output')
74     plt.ylabel('rmse')
75     plt.xlabel('epoch')
76     plt.legend(['train', 'test'], loc='upper right')
77     plt.show()
78
```

Şekil 150. YSA modelinden kayıp çıkış verilerine göre eğitim ve doğrulama grafiklerinin çizilmesi.



```
In [2]:
```

```
Epoch 00108: early stopping
İlk çıkışın R2 değeri : 0.8173110167490836
İkinci çıkışın R2 değeri: 0.7692818740511481
```

Şekil 151. YSA modeli ile pamuk tarlası sulama sistemi için doğruluk sonucu.

Öğrencilere yapay sinir ağları teknigi ile farklı binalara ait enerji verimliliği analizi için kurulan model için erken durdurma işleminin (Early stopping) 108. epoch (eğitim) de gerçekleştirildiği anlatılır.

UYGULAMANIN 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()

```



4. ADIM: İLERLET

İlerlet adımı kapsamında öğrencilerin gruplar halinde çalışarak, yapılan uygulama üzerinde kendi düzenlemelerini gerçekleştirmeleri ve yeni düzenlemeler ardından elde edilen sonuçlar hakkında notlar almaları istenir. Bu amaçla 30 dakikalık bir çalışma süresi verilir. Çalışma süresi esnasında öğrencilerin yaptıkları düzenlemeler uzaktan gözlemlenir, düzenlemelere esnasında sorusu olan öğrencilerin soruları cevaplandırılıp, yardım edilir.



5. ADIM: DEĞERLENDİRME

Derste edinilen bilgiler soru-cevap aşaması gerçekleştirilmek suretiyle pekiştirilir.

Öğrencilere şu sorular sorulabilir:

- Gerçekleştirilen Makine Öğrenmesi uygulamasından elde ettiğiniz bilgi ve beceriler neler olmuştur?
- Yapay sinir ağları tekniği hakkındaki görüşleriniz nelerdir?
- Yapay sinir ağları tekniğinin sınıflandırma, regresyon ve kümeleme gibi farklı Makine Öğrenmesi yöntemlerine uyumlu olması diğer Makine Öğrenmesi algoritmaları / tekniklerine göre ne gibi avantajları ortaya çıkarmaktadır?
- Uyguladığınız Python kodlarında veri hazırlığı ve Makine Öğrenmesi modelinin oluşturularak eğitim-test süreçlerinden geçirilmesi aşamalarına dair görüşleriniz nelerdir?
- Benzeri bir uygulamayı gerçekleştirebileceğiniz problemler neler olabilir?

2. HAFTA – 5. GÜN – 1., 2., 3. DERSLER: ENSEMBLE MODELLER İLE ŞEKER HASTALIĞI TAHMİNİ

DERS PLANI

DERS ETİKETLERİ



KONU: Ensemble Modeller ile Şeker Hastalığı Tahmini



SINIF: 12-14 Yaş



SÜRE: 180 dk.



HAZIR BULUNUŞLUK: İngilizce eğitimi, Yapay Zeka kavramı ve uygulamaları hakkında temel bilgi, Makine Öğrenmesi hakkında temel bilgi, Python programlama dili hakkında temel bilgi



ANAHTAR KELİMELER: Yapay Zeka, Algoritma, Kodlama, Python, Makine Öğrenmesi, Ensemble Modeller, Ensemble Öğrenme



BECERİLER:



Tahmin



Sorgulama



Grup çalışması



Günlük hayat bağlantısı kurma



KAZANIMLAR:

Bu derste öğrenciler;

- Ensemble model kavramı hakkında bilgi sahibi olurlar.
- Python ile ensemble modeller oluşturmak suretiyle şeker hastalığı hastalığı teşhisi üzerine uygulama gerçekleştirirler.
- Python ortamında ensemble model tabanlı Makine Öğrenmesi çözümlerinin geliştirilmesi hakkında bilgi ve beceri sahibi olurlar.



DERS MATERİYALLERİ:

Malzeme	Miktar
Yazı Tahtası	Öğrenci sayısı ve sınıf büyüklüğüne göre en az bir adet
Projeksiyon	Öğrenci sayısı ve sınıf büyüklüğüne göre en az bir adet
Defter, Kalem	Öğrenci sayısı kadar
Bilgisayar	Öğrenci sayısının yarısı kadar
Tablet	Öğrenci sayısının yarısı kadar
Python Anaconda Framework	Her bilgisayarda kurulu olacak şekilde



ÖĞRETMEN HAZIRLIK ÇALIŞMALARI:

Okunabilecek Kaynaklar:

Brownlee, J. (2021). *A Gentle Introduction to Ensemble Learning Algorithms*. Machine Learning Mastery. Çevrimiçi: <https://machinelearningmastery.com/tour-of-ensemble-learning-algorithms/> (Erişim 16 Haziran 2024).

Kaggle. (2020). *Ensemble Learning Techniques Tutorial*. Çevrimiçi: <https://www.kaggle.com/code/pavansanagapati/ensemble-learning-techniques-tutorial> (Erişim 16 Haziran 2024).

İzlenebilecek Kaynaklar:

Data Camp. (2020). Python Tutorial: Introduction to ensemble methods. Çevrimiçi: <https://www.youtube.com/watch?v=bN0OU7jeObI> (Erişim 16 Haziran 2024).

Simplilearn. (2020). Ensemble Learning | Ensemble Learning In Machine Learning | Machine Learning Tutorial. Çevrimiçi: <https://www.youtube.com/watch?v=WtWxOhhZWX0> (Erişim 16 Haziran 2024).

ATÖLYE HAZIRLIK ÇALIŞMALARI:

Öğretmen;

- Dersten önce ders materyallerini hazırlamalıdır.
- Python kodlama için kullanılan Spyder editörünün genel kullanımına aşina olmalıdır.

- Ensemble modeller ile Makine Öğrenmesi konusuna aşina olmalıdır.
- Derse konu olan veri set(ler)ini ve Python kodlarını önceden inceleyerek üzerinde çalışmış olmalıdır.



KAYNAKÇA:

Kyriakides, G., & Margaritis, K. G. (2019). *Hands-On Ensemble Learning with Python: Build highly optimized ensemble machine learning models using scikit-learn and Keras*. Packt Publishing Ltd.

Oshiro, T. M., Perez, P. S., & Baranauskas, J. A. (2012). How many trees in a random forest?. In *Machine Learning and Data Mining in Pattern Recognition: 8th International Conference, MLDM 2012, Berlin, Germany, July 13-20, 2012. Proceedings* 8 (pp. 154-168). Springer Berlin Heidelberg.

Zhang, C., & Ma, Y. (2012). *Ensemble machine learning* (Vol. 144). Springer.

GENEL BAKIŞ:

1) Harekete Geç: Öğrencilere uygulamaya konu ensemble modeller ile Makine Öğrenmesi hakkında bilgi verilir ve Python tabanlı ensemble öğrenme uygulamasının veri hazırlık aşamalarının kodlanması sağlanır.

2) Keşfet: Öğrencilerle birlikte Python tabanlı ensemble öğrenme sisteminin eğitim ve test süreçlerine yönelik adımların kodlanması sağlanır.

3) Üret: Öğrencilerle birlikte çalıştırılan ensemble öğrenme sisteminin elde ettiği sonuçların değerlendirilmesine yönelik kodlamaların yapılması sağlanır.

4) İlerlet: Öğrencilerin gruplar halinde çalışarak, gerçekleştirilen uygulama üzerinde kendi düzenlemelerini yapmaları ve elde edilen sonuçlar hakkında notlar almaları istenir.

5) Değerlendir: Derste edinilen bilgiler soru-cevap aşaması gerçekleştirilmek suretiyle pekiştirilir.

UYGULAMA



1. ADIM: HAREKETE GEÇ

Harekete geç adımında öğrencilere uygulamaya konu ensemble modeller ile Makine Öğrenmesi hakkında bilgi verilir ve Python tabanlı ensemble öğrenme uygulamasının veri hazırlık aşamalarının kodlanması sağlanır.

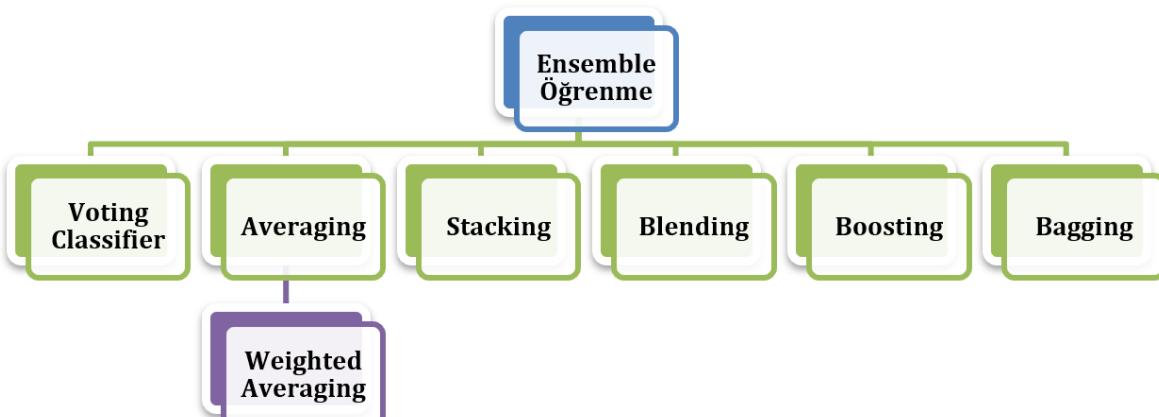
Ensemble Modeller ve Ensemble Öğrenme

Makine Öğrenmesi algoritmalarının sahip oldukları özellikler, hedef problemler içerisindeki dinamikler ve hatta veri setleri Yapay Zeka uygulamalarının başarısında doğrudan etkili

olmaktadır. Neticede Makine Öğrenmesi algoritmalarının her problemde benzeri başarı oranlarını yakalamaları kolay olmamakta, aynı problemde farklı algoritmaların denenmesi zaman almaktadır. Benzer şekilde böyle bir deneme yaklaşımı içerisinde olunması da sürdürülürken zor bir çözüm yoludur. Bu nedenle farklı Makine Öğrenmesi modellerinin aynı problem üzerinde uygulanmasına dayanan **ensemble modeller** alternatif bir uygulama yolu olarak bilinmektedir (Zhang, & Ma, 2017).

Ensemble modelleri ile gerçekleştirilen Makine Öğrenmesi uygulamaları aynı zamanda “**ensemble öğrenme (ensemble learning)**” olarak adlandırılmaktadır. Ensemble öğrenme zamanla farklı şekillerde tekniklere ayrılmıştır (Kyriakides, & Margaritis, 2019; Zhang, & Ma, 2017) (Şekil 152):

- **Voting Classifier (Oylamaya Dayalı Sınıflandırıcı):** Bu tür ensemble öğrenmede farklı modellerin aynı örnek için ürettiği tahminler arasında en çok görülen sınıf sistemin tahmini olarak kabul edilir.
- **Averaging (Ortalama) ve Weighted Averaging (Ağırlıklı Ortalama):** Averaging tabanlı ensemble öğrenmede farklı modellerin tahmin ortalamaları sistemin tahmini olarak kabul edilir. Weighted averaging tekniğinde ise ortalama hesaplamasına modeller için belirlenen ağırlıklar da katılır. Bu teknikler regresyon problemlerine ya da olasılıksal sınıf tahmini problemlerine daha uygundur.
- **Stacking (İstifleme / Bir Araya Toplama):** Bu ensemble öğrenme tekniğinde çeşitli taban modellerin tahminlerinden yola çıkarak genel tahmin yapan genelleyici bir model elde edilir.
- **Blending (Harmanlama):** Blending tabanlı ensemble öğrenmede stacking teknigine benzer bir süreç izlenerek beraber nihai sistem tahmini konusunda farklı bir yaklaşım olarak validation ve test veri setleri kullanılır.
- **Boosting (Güçlendirme):** Boosting ensemble öğrenmede önceki öğrenme süreçlerindeki zayıflıkların düzeltilmesi yoluyla model güçlendirme üzerinden sistem tahminin iyileştirildiği bir yaklaşım izlenir.
- **Bagging (Önyükleme ve Birleştirme):** Bu tür ensemble öğrenmede, orijinal veri setinden farklı alt veri setlerini kullanan modellerin elde ettiği tahminlerini baz alarak genel sistem tahminine ulaşılır.



Şekil 152. Ensemble öğrenme teknikleri.

Ensemble Modeller Yardımıyla Şeker Hastalığı Tahmini

Öğrencilerin öncelikli olarak şeker hastalığı veri setini indirmeleri ve ardından Python ortamında Pandas ve Numpy kütüphanelerinden destek almak suretiyle veri setini girdi ve çıktı değişkenleri altında ayırmaları sağlanır (Şekil 153).

VERİ SETİ İÇİN İNDİRME LINKİ

https://raw.githubusercontent.com/jaz-alli/Ensemble-learning-tutorial/master/data/diabetes_data.csv

```
7 import pandas as pd
8 import numpy as np
9
10 df = pd.read_csv("diabetes_data.csv")
11
12 X = df.drop(columns = ['diabetes'])
13 Y = df['diabetes']
```

Şekil 153. Şeker hastalığı veri seti düzenlemesi.

Öğrenciler ilgili dataframe değişkenlerini oluşturuktan sonra sklearn kütüphanesi yardımıyla veri setinin %70 eğitim ve %30 test oranlarında olacak şekilde ayrılması yönünde kodları yazar (Şekil 154).

```
15 from sklearn.model_selection import train_test_split
16 X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.3, stratify=Y)
```

Şekil 154. Şeker hastalığı veri seti düzenlemesi.



2. ADIM: KEŞFET

Bu adımda öğrencilerle birlikte Python tabanlı ensemble öğrenme sisteminin eğitim ve test süreçlerine yönelik adımların kodlanması sağlanır.

Python ortamında sklearn ile Makine Öğrenmesi uygulamaları kodlanırken model seçiminin kolaylaştırılan çeşitli kütüphane bileşenleri bulunmaktadır. Öğrencilere “**GridSearchCV**” bileşeni anlatılarak Şekil 155’de görüldüğü gibi ilgili bileşene çağrı yapılması yönünde kodun yazılması istenir.

Buradaki uygulamada ensemble modeller olarak kNN algoritması ve Random Forest (Rastgele Orman) algoritmaları seçilmiştir. Öğrencilere kNN algoritmasını daha önceki uygulamalardan bildikleri (Bkz. 1. Hafta – 4. Gün – 2., 3. Dersler), Random Forest algoritmasının ise Karar Ağaçları algoritmasının (Bkz. 2. Hafta – 3. Gün – 1., 2., 3. Dersler) bir tür genişletilmiş bir versiyonu olarak; çok sayıda ağacın verdiği kararları dikkate alarak tahmin yürüten bir Makine Öğrenmesi algoritması olduğu açıklanır (Oshiro vd., 2012). Bu noktada öncelikli olarak

Random Forest algoritmasının çağrısı yapılır ve GridSearchCV araması için 22. satırda görülen 8 farklı ağaç sayısı parametresinin deneneceği vurgulanır. İlgili tanımlama sonrası Random Forest algoritması “fit” fonksiyonu yardımıyla eğitilir ve en iyi model **best_estimator_** parametresi, en iyi modeli veren parametreler ise **best_params_** parametresi ile elde edilir.

```
18 from sklearn.model_selection import GridSearchCV
19
20 from sklearn.ensemble import RandomForestClassifier
21 random_forest = RandomForestClassifier()
22 param_random_forest = {'n_estimators': [25, 50, 75, 100, 125, 150, 175, 200]}
23 random_forest_GS = GridSearchCV(random_forest, param_random_forest, cv=5)
24 random_forest_GS.fit(X_train, Y_train)
25 random_forest_en_iyi = random_forest_GS.best_estimator_
26 print(random_forest_GS.best_params_)
```

Şekil 155. GridSearchCV çağrısı ve Random Forest algoritmasının kurularak en iyi model ile en iyi parametrelerin elde edilmesi.

Benzeri çağrı ikinci algoritma olan kNN için de yapılır. 30. satırda görüldüğü üzere, kNN algoritması için k değeri 1'den 30'a kadar olan değerler kapsamında denenmektedir (Şekil 156). Yapılan kodlamalar neticesinde kNN algoritması için de en iyi model ve parametre değerleri elde edilir.

```
28 from sklearn.neighbors import KNeighborsClassifier
29 knn = KNeighborsClassifier()
30 param_knn = {'n_neighbors': np.arange(1, 30)}
31 knn_GS = GridSearchCV(knn, param_knn, cv=5)
32 knn_GS.fit(X_train, Y_train)
33 knn_en_iyi = knn_GS.best_estimator_
34 print(knn_GS.best_params_)
```

Şekil 156. kNN algoritmasının kurularak en iyi model ile en iyi parametrelerin elde edilmesi.



3. ADIM: ÜRET

Öğrencilerle birlikte çalıştırılan ensemble öğrenme sisteminin elde ettiği sonuçların değerlendirilmesine yönelik kodlamaların yapılması sağlanır.

Nihai Sonuçların Elde Edilmesi

Öğrencilerin en iyi Random Forest ve en iyi kNN tespit oranlarını ekrana yazdırıldıktan sonra **sklearn.ensemble** bileşeni altından **VotingClassifier** bileşenini çağrılmaları sağlanır. VotingClassifier yardımıyla oylamaya dayanan bir ensemble öğrenme yapısı kurulmaktadır (Şekil 157).

```

36 print("Random Forest En İyi: {}".format(random_forest_en_iyi.score(X_test, Y_test)))
37 print("kNN En İyi: {}".format(knn_en_iyi.score(X_test, Y_test)))
38
39 from sklearn.ensemble import VotingClassifier
40 tahmin_modelleri=[('knn', knn_en_iyi), ('rf', random_forest_en_iyi)]
41 ensemble_model = VotingClassifier(tahmin_modelleri, voting='hard')

```

Şekil 157. VotingClassifier üzerinden ensemble öğrenme yaklaşımının kodlanması.

Son olarak ensemble model üzerinden eğitim süreci gerçekleştirilmek suretiyle en iyi tespit oranlarının ekrana yansıtılması ile voting (oylama) tekniğine dayalı ensemble yapılanmanın sonuçları elde edilir. Bu aşamada kodu tamami çalıştırıldığında farklı modellerin performansları ve nihai ensemble model sonuçları konsol ortamına yansıtılmaktadır (Şekil 158).

```

43 ensemble_model.fit(X_train, Y_train)
44 print("Nihai skor:",ensemble_model.score(X_test, Y_test))

```

(a)

```

{'n_estimators': 100}
{'n_neighbors': 14}
Random Forest En İyi: 0.7445887445887446
kNN En İyi: 0.7445887445887446
Nihai skor: 0.7532467532467533

```

(b)

Şekil 158. (a) Ensemble modellerin nihai değerlendirilmesine ilişkin kodlar
(b) bütün uygulama kodlarının çalıştırılması sonucu konsol görüntüsü.

Şekil 158b'de görüldüğü üzere, Random Forest algoritması için en uygun ağaç sayısı 100, kNN algoritması için en uygun k değeri 14 bulunurken, ensemble öğrenmede tahmin oranı 0.75 olarak elde edilmiştir.

UYGULAMANIN PYTHON KODLARI

```

import pandas as pd
import numpy as np

df = pd.read_csv("diabetes_data.csv")

X = df.drop(columns = ['diabetes'])
Y = df['diabetes']

from sklearn.model_selection import train_test_split

```

```

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.3,
stratify=Y)

from sklearn.model_selection import GridSearchCV

from sklearn.ensemble import RandomForestClassifier
random_forest = RandomForestClassifier()
param_random_forest = {'n_estimators': [25, 50, 75, 100, 125, 150, 175, 200]}
random_forest_GS = GridSearchCV(random_forest, param_random_forest,
cv=5)
random_forest_GS.fit(X_train, Y_train)
random_forest_en_iyi = random_forest_GS.best_estimator_
print(random_forest_GS.best_params_)

from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier()
param_knn = {'n_neighbors': np.arange(1, 30)}
knn_GS = GridSearchCV(knn, param_knn, cv=5)
knn_GS.fit(X_train, Y_train)
knn_en_iyi = knn_GS.best_estimator_
print(knn_GS.best_params_)

print("Random Forest En İyi: {}".format(random_forest_en_iyi.score(X_test,
Y_test)))
print("kNN En İyi: {}".format(knn_en_iyi.score(X_test, Y_test)))

from sklearn.ensemble import VotingClassifier
tahmin_modelleri=[('knn', knn_en_iyi), ('rf', random_forest_en_iyi)]
ensemble_model = VotingClassifier(tahmin_modelleri, voting='hard')

ensemble_model.fit(X_train, Y_train)
print("Nihai skor:",ensemble_model.score(X_test, Y_test))

```



4. ADIM: İLERLET

Öğrencilerin gruplar halinde çalışarak, gerçekleştirilen uygulama üzerinde kendi düzenlemelerini yapmaları ve elde edilen sonuçlar hakkında notlar almaları istenir.



5. ADIM: DEĞERLENDİRME

Derste edinilen bilgiler soru-cevap aşaması gerçekleştirilmek suretiyle pekiştirilir.

Öğrencilere şu sorular sorulabilir:

- Gerçekleştirilen Makine Öğrenmesi uygulamasından elde ettiğiniz bilgi ve beceriler neler olmuştur?
- Ensemble modeller ve ensemble öğrenmelarındaki görüşleriniz nelerdir?
- Ensemble öğrenme yaklaşımının avantaj ve dezavantajları neler olabilir?
- Benzeri bir uygulamayı gerçekleştireceğiniz problemler neler olabilir?