

**T.C.**

**BİTLİS EREN ÜNİVERSİTESİ**

**MÜHENDİSLİK-MİMARLIK FAKÜLTESİ**

**STAJ DEFTERİ**

Öğrenci No : 20080410022

Adı, Soyadı :Mecdiddin Ebuşerfo

**2015**

|  |  |
| --- | --- |
| 200px-Bitlis_Eren_Üniversitesi_logosu | T.C.  BİTLİS EREN ÜNİVERSİTESİ  MÜHENDİSLİK MİMARLIK FAKÜLTESİ  ............................ Mühendisliği Bölümü |

**STAJ DEFTERİ**



fotoğraf

ÖĞRENCİNİN Numarası : 20080410022

Adı ve Soyadı :MECDİDDİN EBUŞERFO

Sınıfı : 4

Staj No : 2

Staja başlama tarihi : 24/06/ 2024

Stajı bitirme tarihi : 22/07/ 2024

Çalıştığı işgünü toplamı : 20 İŞ GÜNÜ

STAJIN YAPILDIĞI İŞYERİNİN

Adı :................................................................

Adresi :................................................................

...............................................................................

Faaliyet Alanı : ...............................................................

İletişim Bilgileri : ...............................................................

İŞYERİ SORUMLU AMİRİNİN

Adı ve Soyadı Unvanı Tarih İmza ve Mühür

...................... ................ ..../..../ 201

**STAJ İLE İLGİLİ HATIRLATMA**

ÖĞRENCİLERİN; STAJLARINININ KABUL EDİLEBİLMESİ VE DEFTERLERİNİN DEĞERLENDİRİLEBİLMESİ İÇİN STAJLARINI, MÜHENDİSLİK MİMARLIK FAKÜLTESİ STAJ UYGULAMA ESASLARINA GÖRE YAPMALARI VE STAJ DEFTERLERİNİ, STAJ DEFTERİ YAZIM KURALLARINA UYGUN OLARAK HAZIRLAMALARI GEREKMEKTEDİR.

DEFTER, ÖN VE ARKA KAPAK SAYFALARI KARTON OLACAK BİÇİMDE CİLTLENMİŞ OLARAK TESLİM EDİLMELİDİR.

İÇİNDEKİLER  
  
 Konu Sayfa No  
Roboflow'u kullanmaya başlama ………………………………………………….. 1  
Hataları düzeltmeyi öğrenin ……………………………………………………….. 3  
Kişileri tespit etme …………………………………………………………………. 5  
kırık camları tespit etmek …………………………………………………………. 7  
Tecrübe Kazanmak ………………………………………………………………… 8  
Günlük iş örnekleri ……………………………………………………………….. 10  
Temel Python öğrenmeye …………………………………………………………... 12  
OOP Öğrenmeye ……………………………………………………………………. 14  
Gürüntü işlemleri …………………………………………………………………... 16  
Gürüntü işlemleri 2 ………………………………………………………………… 22  
Kare ve Daire Çizmek ……………………………………………………………… 29  
Renkleri ve Nesneleri tespit etmek ………………………………………………… 32  
Şekilleri Tanıyan …………………………………………………………………… 35  
Yüzleri Tanıyan ………………………………………………………………………….. 37  
Yolo formatıdaki Proje ……………………………………………………………. 40  
Her Şeyi Tanıyan Bir Program Oluşturmak …………………………………….. 43  
Karşılaştırma Kodu Oluştur ……………………………………………………… 46  
eli tanır ……………………………………………………………………………… 50  
eli tanır + Yüz tanıma ……………………………………………………………. 52  
Yüz tanıma ………………………………………………………………………… 55

Konu Sayfa No

ŞEKİL, ÇİZELGE VE EKLER LİSTESİ  
  
 Şekil, Çizelge veya Ek No ve Adı Sayfa No  
Görsel 1.1 Roboflow web sitesi arayüzü ………………………………… 1  
Görsel 1.2: Roboflow`a giriş yapmak ……………………………………. 1  
Görsel 1.3: kırık cam arayüzü …………………………………………… 2  
Görsel 2.1: kırık cam arayüzü 2 ………………………………………….. 3  
Görsel 2.2: Hataları düzelt …………………………………………………….. 3  
Görsel 3.1: Kişi tanımlama arayüzü …………………………………………... 5  
Görsel 3.2 : Seçim yöntemi …………………………………………………… 5  
Görsel 3.3 : Bizim için görsel dağıtım haritası ……………………………….. 6  
Görsel 3.4: Gerekli tüm fotoğrafları tamamlayın …………………………….. 6  
Görsel 4.1: Sıfırdan seçim …………………………………………………… 7  
Görsel 4.2 : Seçim yöntemi ………………………………………………….. 7  
Görsel 5.1 : Birden fazla kişiyi tanımlayın …………………………………... 8  
Görsel 5.2 : İşçi ile mühendis arasındaki fark ……………………………….. 8  
Görsel 5.3 : Tüm çalışan görevlerini bitirin …………………………………. 8  
Görsel 5.4 : Yorumlar kutusu ……………………………………………….... 9  
Görsel 5.5 : Güvenlik kameralarının tanımı……………………………………9  
Görsel 6.1 : Thermal bölüm…………………………………………………... 10  
Görsel 6.2 : Ekip üyelerinin hatalarını düzeltmek …………………………… 10  
Görsel 6.3 : Uzun parçalar üzerinde çalışın ………………………………….. 10  
Görsel 6.4 : Hatalar düzeltildikten sonraki görsel …………………………… 10  
Görsel 9.1 : Python'da görselleri göster ……………………………………… 16  
Görsel 9.2 : Python'da görüntü boyutunu küçültme ………………………… 17  
Görsel 9.3 : Resimleri çevir ………………………………………………… 17  
Görsel 9.4 : Erozyon fotoğraf çalışması …………………………………….. 18  
Görsel 9.5 : Python'da görüntüleri yumuşatın ………………………………. 20  
  
  
 Şekil, Çizelge veya Ek No ve Adı Sayfa No  
Görsel 9.6 : Gürültü Azaltma ………………………………………………. 20  
Görsel 10.1 : Kenar Tespiti orneği …………………………………………. 22  
Görsel 10.2 : Resimlerle Gradyanlar ………………………………………. 23  
Görsel 10.3 : Harris köşe kodu ……………………………………………. 24  
Görsel 10.4 : Kod çıkışı …………………………………………………… 24  
Görsel 10.5 : bir renk aralığını belirleyerek ………………………………. 26  
Görsel 10.6 : SIFT Kod çıkışı …………………………………………..... 27  
Görsel 10.7 : İki görüntüdeki benzer noktaları tanımlayan kod………….... 28  
Görsel 11.1 : Kare ve daire çizmek kodu ………………………………..,, 30  
Görsel 11.2 : Python'da iki görüntüyü birleştirme ……………………… ...31  
Görsel 12.1: Renklerle oynamak arayüzü ………………………………… 32  
Görsel 12.2 :Değişiklikten önceki resimler Hurda izolasyonu …………… 32  
Görsel 12.3 : Ayarlarda bazı değişiklikler ……………………………… 32  
Görsel 12.4 : Değişiklik sonrası resimler ………………………………… 32  
Görsel 13.1 : Resmi gümüşe dönüştür …………………………………… 35  
Görsel 13.2 : Tanımdan sonra çıkışı ……………………………………… 36  
Görsel 14.1 : Resimlerdeki kişileri tanımlama ……………………………. 37  
Görsel 14.2 : Videolardaki yüzleri tanımlama ……………………………. .37  
Görsel 14.3 : Görsellerin üzerine yazı yazmak …………………………… 38  
Görsel 14.4 : Fotoğraflar için video oynat ………………………………...39  
Görsel 14.5 : Videodan alınan tüm fotoğraflar ……………………………. 39  
Görsel 15.1 : Bilgi dosyası ………………………………………………... 40  
Görsel 15.2 : sıft soft çıkış kodu ………………………………………….. 41  
Görsel 16.1 : her şeyi tanıyan program kodu ……………………………. 43  
  
 Şekil, Çizelge veya Ek No ve Adı Sayfa No  
Görsel 16.2: İnsanları tanıyın ………………………………………………..45  
Görsel 16.3 : Bir şeyleri tanımak 1…………………………………………..45  
Görsel 16.4 : Bir şeyleri tanımak 2 ………………………………………...45  
Görsel 16.5 : Bir şeyleri tanımak 3………………………………………….45  
Görsel 16.6 : Bir şeyleri tanımak 4………………………………………….45  
Görsel 17.1 : Karşılaştırma Kodu…………………………………………...46  
Görsel 17.2 : Programda tanımlanan görseller……………………………...48  
Görsel 18.1 : El tanıma kodu……………………………………………….50  
görsel 19.1: Kod parmaksız çıktı …………………………………………..52  
görsel 19.2 : Kod tek parmakla çıktı………………………………………..52  
görsel 19.3 : Kodu iki parmakla çıktı ………………………………………53  
görsel 19.4 : Kod 3 parmakla çıktı………………………………………….53  
 görsel 19.5 : Kod 4 parmakla çıktı ………………………………………...53  
görsel 19.6 : Kod 5 parmakla çıktı……………,,……………………………53  
Görsel 19.7 : İkinci projenin temelleri…………,,………………………….53  
Görsel 20.1 : program kodu 2…………………………………….…………55  
Görsel 20.2 : Projede tanımlanan görseller………………………………...57  
Görsel 20.3 : İmajımı tanı……………………………………………….….57  
Görsel 20.4 : Steve Jobs'u tanımak………………………………………....58  
Görsel 20.5 : Bill Gates'i tanıyın ………………………………………..….58  
Görsel 20.6 : Elon Musk'u tanımak……………………………………...….58  
  
STAJIN YAPILDIĞI KURUM VEYA KURULUŞUN TANITIMI  
  
GİRİŞ  
  
Stajyer: [’MECDİDDİN]  
Tarih: [12/6 /2023, 14/7/2023]  
  
Sizlere AISOFT YAZILIM LİMİTED ŞİRKETİ'nde gerçekleştirdiğim stajımın heyecan verici bir yolculuğunun kapılarını aralayacağım. Bu yazılım şirketi, Ankara merkezli olarak faaliyet gösteriyor ve iş odak noktamız, hiç şüphesiz, yapay zeka alanına odaklanmış durumda. Stajımın başlangıcı, Roboflow adlı web sitesi üzerinde çalışmakla oldu. Bu, büyülü dünyaya adım atmamızın kapısını araladı.  
  
Roboflow'da geçirdiğimiz zaman, bilgi ve becerilerimizi geliştirmenin ötesine geçti. Nesne yönelimli programlamayla tanıştık ve Python dilinde makine öğrenmeye dalış yaptık. Görsel veri işleme konusundaki bilgilerimizi ve yeteneklerimizi test etmek için sayısız örnek ve proje gerçekleştirdik.  
  
Stajımız boyunca öğrendiklerimiz sadece yazılımsal bilgi ve becerilerle sınırlı kalmadı, aynı zamanda işbirliği yapmanın ve öğrenmenin önemini de anladık. AISOFT ekibi, bize öğrenme sürecimizde destek verdi ve paylaşmaktan çekinmedi. Neslihan HOŞGÖRÜL Hanımefendi gibi deneyimli profesyoneller, biz stajyerlere kılavuzluk ederek, gerçek dünya projelerinde nasıl başarılı olunacağını öğretti.  
  
Bu süre zarfında, kendimizi her yeni konseptte meydan okumaya ve yaratıcı çözümler üretmeye teşvik ettik. Veri bilimi ve yapay zeka alanlarındaki bilgilerimizi pratikte uyguladık ve deneyimledik. İşte bu staj raporunda, bu unutulmaz yolculuğun detaylarını sizlere sunacağım.  
  
İmza:  
[MECDİDDİN EBUŞERFO]  
 12/06/201 tarihinden 16/06 /2023 tarihine kadar bir haftalık çalışma.  
   
  
  
  
19/06/2023 tarihinden 23/06/2023 tarihine kadar bir haftalık çalışma.  
  
  
  
  
  
  
  
  
  
  
  
  
  
03/07/2023 tarihinden 07/07/2023 tarihine kadar bir haftalık çalışma  
  
  
 10/07/2023 tarihinden 14/07/2023 tarihine kadar bir haftalık çalışma  
  
  
  
  
  
  
  
  
  
  
  
  
1. Gün (01/07/2024 - Pazartesi)  
  
Görev: Staj tanıtımı ve proje planlaması  
  
İlk gün, stajyerlerin tanıtımı yapıldı ve staj süresince yapılacak projeler hakkında genel bir bilgilendirme gerçekleştirildi. Şirket içindeki departmanlar tanıtıldı ve her birimin işleyişi hakkında bilgi verildi. Staj süresince tamamlanması gereken projeler belirlendi ve bu projelerin detaylı planlaması yapıldı.  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
Hataları düzeltmeyi öğrenin  
Staj`dan 2 günü Bize hataların yapay zeka ile nasıl düzeltilebileceğine dair küçük bir proje verdi.  
   
Görsel 2.1: kırık cam arayüzü 2  
Bu gün başlarsak hocamız bize yapay zeka ile işaretlenmiş kırık cam resimleri verdi.Fotoğrafları çektik ve iki noktanın kesişimi gibi hataları düzelttik.   
Görüntü İşaretleme: Yapay zeka modeli, kırık camların yerini belirlemek için kullanılmış olmalı. Bu model, çatlakların başlangıç ve bitiş noktalarını işaretlemiş olabilir.  
   
Görsel 2.2: Hataları düzelt  
çizgiler ve iki çizgi birbirine çok yaklaşıyor. Onları biraz uzaklaştırıyoruz ve parçaları tespit ediyoruz... Yapay zeka bunu tanımlayamadı  
  
Kesişen Noktaları Düzeltme: Siz veya ekibiniz, yapay zeka tarafından işaretlenen bu kesişen noktaları inceleyerek, bu noktaları kullanarak çatlakları düzeltmek veya onarmak için gerekli işlemleri gerçekleştirmiş olmalısınız. Özellikle bu noktaların   
 kesişimlerini kullanarak, çatlakların doğru bir şekilde hizalandığından emin olmanız gerekebilir.  
  
Görüntü Düzeltme İşlemi: Daha sonra, kesişen noktaları kullanarak çatlakların düzeltilmesi işlemini gerçekleştirdiniz. Bu işlem, resimlerin orijinal haline yakın hale getirilmesini içerebilir.  
  
Sonuçların Değerlendirilmesi: Düzeltme işlemi tamamlandığında, sonuçları değerlendirdiniz ve onayladınız mı? Yani, çatlakların ne kadar etkili bir şekilde onarıldığını kontrol ettiniz mi?  
  
Bu tür görevlerde yapay zeka, resim analizi ve düzeltme işlemlerinde büyük bir yardımcı olabilir, ancak insan gözü ve deneyimi de önemlidir. Sonuçlar gerçek dünya uygulamalarına döndüğünüzde, işaretlenmiş veya düzeltilmiş görüntülerin kullanılabilirliği ve kalitesi büyük öneme sahiptir. Ayrıca, bu tür bir görevde işbirliği ve insanın gözlemi sıklıkla önemlidir.  
  
  
 Kişileri tespit etme  
  
Staj”den 3 günü sitedeki kişileri tespit etme konusunda bizi eğitmek için bir toplantı yaptık, ardından yapay zekanın yaptığı hataları düzeltebilmemiz için bize resimler gönderdi,  
  
   
Görsel 3.1: Kişi tanımlama arayüzü  
Toplantı ve Eğitim: İlk adımda, yapay zeka modelini ve tespit işlemi hakkında bilgi almak ve eğitim almak için bir toplantı yapıldı. Bu toplantıda, modelin nasıl çalıştığı, hangi hataları yaptığı ve modelin nasıl iyileştirilebileceği konuları ele alındı.  
eğer varsa o kişi için bir kare oluşturacaktık, ve boş kareleri silin.  
   
Görsel 3.2 : Seçim yöntemi  
Hataların Belirlenmesi: Eğitim sonrası, yapay zeka modelinin yaptığı hataları anlamak için bazı resimler incelendi. Bu resimlerdeki hatalı tespitler veya yanlış pozitif/negatif sonuçlar belirlendi.  
Bunları tamamladıktan sonra çalışmaya başlayabilelim diye birçok resim gönderdi.  
   
Görsel 3.3 : Bizim için görsel dağıtım haritası  
Hataların Düzeltme Süreci: Hataların belirlenmesinin ardından, bu hataları düzeltebilmek için biz veya ekip üyeleri, bu resimleri düzelten düzeltmeleri uyguladık. Bu düzeltmeler, yapay zeka modelinin daha iyi sonuçlar üretebilmesi için yapılabilir.  
Kişileri kendimiz tespit ederdik ve bu çok zamanımızı alırdı ama faydalı oldu.  
   
Görsel 3.4: Gerekli tüm fotoğrafları tamamlayın  
  
  
  
  
  
  
  
  
  
 kırık camları tespit etmek  
Staj”den 4 günü, tüm görevlerimi tamamladıktan sonra, resimlerde en zoru olan kırık camları tespit etmek, Çalışmamızın en önemli unsuru yinelemeydi: hataları düzeltmek ve modeli geliştirmek için bu süreç birkaç kez tekrarlanabilir. Bu, yeni hatalar veya performans sorunları tespit edildiğinde sürecin tekrarlanabileceği anlamına gelir.  
   
Görsel 4.1: Sıfırdan seçim  
  
 araba gibi açık ve kapalı yerlerdeki kişileri tespit etmek, resimlerdeki kırık camları tespit etmek gibi   
   
Görsel 4.2 : Seçim yöntemi  
  
birçok test ve yapmamız gereken görevleri bize gönderdi. işçilerin yüzleri ve işçi ile sorumlu mühendis arasında ayrım yapılması Her kimlik farklı renkteydi ve biz aralıksız üç gün boyunca,  
   
  
 Tecrübe Kazanmak  
Staj`dan 5 günü Çok çalışarak başlamıştık ve yaptığımız işle ilgili her şeyi iyi biliyorduk, bu işte daha fazla tecrübe kazanmak ve hata yapmamak için günlerce üzerinde çalıştığımız birçok görevi üstlendik.  
   
Görsel 5.1 : Birden fazla kişiyi tanımlayın  
  
   
Görsel 5.2 : İşçi ile mühendis arasındaki fark  
  
 amirden rapor almak ve amirden rapor almaktan, şefe kadar bu görevleri yerine getiriyoruz.  
   
Görsel 5.3 : Tüm çalışan görevlerini bitirin  
   
Görsel 5.4 : Yorumlar kutusu  
   
Görsel 5.5 : Güvenlik kameralarının tanımı  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
 Günlük iş örnekleri  
Staj`dan 6 gnüü Çok çalıştık, yeni formatlara başladık ve önceki çalışmalarımızı daha zor bir formatta tekrarladık.  
 Günlük iş örnekleri:   
   
Görsel 6.1 : Thermal bölüm  
  
Burada parçalar çok küçüktü ve çok zaman alıyordu ama onlardan çalışmamda titiz olmayı öğrendim  
  
   
Görsel 6.2 : Ekip üyelerinin hatalarını düzeltmek  
  
   
Görsel 6.3 : Uzun parçalar üzerinde çalışın  
  
   
Görsel 6.4 : Hatalar düzeltildikten sonraki görsel  
 Sitedeki eğitimimizi en iyi şekilde tamamladık ve yapay zekanın insanları, yerleri, nesneleri tanımlama ve tanıma konusunda nasıl çalıştığını artık çok iyi biliyoruz.  
En önemlisi, çok sayıda geri bildirim yaptık ve bu tür geri bildirimler, bir yapay zeka modeli geliştirmek için yaygın bir yöntemdir. İnsan izleme ve müdahalesi, model kalitesinin iyileştirilmesinde ve daha doğru sonuçların elde edilmesinde önemli bir rol oynayabilir. Bu süreçte ekip üyelerinin modelin güncellenmiş versiyonunu değerlendirmesi ve başarılarını takip etmesi önemlidir.  
  
  
  
 Temel Python'u öğrenmeye   
Stajın yedinci gününde, denetleyici profesörümüz bir toplantı düzenledi ve daha önceki görevlerini tamamlayanlardan Python dilini öğrenmeye başlamalarını istedi. Temel bilgilerle başladık, bu da sayılar ve matrisler gibi konuları içeriyordu. Örnekler:  
colors = ["kırmızı", "yeşil", "mavi"]  
x = 5 # Bu bir tam sayıdır (integer).  
y = 3.14 # Bu bir ondalık sayıdır (float).  
person = {"ad": "Ahmet", "yaş": 30, "şehir": "İstanbul"}  
Daha sonra, koşullu ifadeleri öğrenmeye başladık, bu da if, else, elif gibi yapıları içeriyordu. Örnek:  
  
  
  
if x > 10:  
 print("x 10'dan büyük.")  
else:  
 print("x 10'dan küçük veya eşit.")  
Bu ifadeler, bir koşulu değerlendirir ve kodunuzun nasıl davranacağını belirlemenize yardımcı olur.  
  
Ardından, döngüler konusuna geçtik, bu da for ve while döngülerini içeriyordu. Örnek:  
  
for i in range(5):  
 print(i)  
Daha sonra fonksiyonları öğrendik:  
  
  
def greet(name):  
 return "Merhaba, " + name + "!"  
message = greet("Ahmet")  
print(message)  
  
  
Fonksiyonlar, belirli bir görevi yerine getiren kod bloklarıdır.  
  
Son olarak, Python'da modüllerin nasıl kullanılacağını öğrendik:  
  
  
import math  
result = math.sqrt(25) # 25'in karekökünü hesaplar.  
"Math" gibi modüller, matematiksel işlemler gibi ek işlevselliği projelerinize eklemenize yardımcı olur.  
  
Bu süreçte, temel Python becerilerini kazandınız ve bu, daha karmaşık görevleri ve projeleri ele almanıza yardımcı olacak bir temel oluşturur. İmza ve tarih bölümü, belgenin tamamlanma tarihini ve onayı gösterir  
  
  
  
  
  
  
  
  
  
  
  
 OOP Öğrenmeye  
Stajın 8. gününde nesne yönelimli programlamayı öğrenmeye başladınız. Nesne tabanlı programlama, Python gibi dillerde veriyi nesneler ve sınıflar aracılığıyla yönetmenizi sağlayan bir programlama paradigmasıdır. İşte öğrendiğiniz konuların bir özeti:  
  
Sınıf ve Nesneler: Nesne tabanlı programlamada, nesneleri ve sınıfları kullanarak kodunuzu organize edebilirsiniz. Sınıflar, belirli özelliklere (veri) ve davranışlara (metodlar) sahip nesneleri tanımlar.  
class Person:  
 def \_\_init\_\_(self, name, age):  
 self.name = name  
 self.age = age  
  
 def introduce(self):  
 return "Benim adım " + self.name + " ve yaşım " + str(self.age)  
  
ahmet = Person("Ahmet", 30)  
print(ahmet.introduce())  
Yukarıdaki örnekte, Person adında bir sınıf tanımlanmış ve ahmet adında bir nesne oluşturulmuştur.  
  
Hata Yönetimi: Python'da hata yönetimi için try ve except blokları kullanılır. Bu bloklar, kodun hata durumlarına karşı nasıl tepki vereceğini belirler.  
  
try:  
 x = 10 / 0  
except ZeroDivisionError:  
 print("Sıfıra bölme hatası!")  
Bu örnekte, try bloğunda bir hata oluşursa, program except bloğuna geçer ve belirli bir hatayı ele alır.  
  
Veri Yapıları: Stajın önemli bir bölümünde veri yapıları öğrenildi. Veri yapıları, verileri düzenlemek ve işlemek için kullanılan yapısal elemanlardır. Bu, Python'da veri analizi ve işleme için temel bir beceridir.  
Sonuç olarak, nesne yönelimli programlama, hata yönetimi ve veri yapıları gibi temel programlama kavramlarını öğrendiniz. Ayrıca, öğrenilen bilgileri içeren bir test yaparak bu konulardaki anlayışınızı ölçtünüz. Bu, programlama yeteneklerinizi geliştirmenin önemli bir adımıdır. İlerleyen günlerde bu bilgileri daha karmaşık projelerde ve görevlerde kullanma fırsatınız olacak.  
  
  
  
  
  
  
  
  
  
  
 Gürüntü işlemleri   
 Staj`dan 9 gününde Python kullanarak Gürüntü işlemleri öğrenmeye başladık  
Temellerini öğrendik ve yapacağımız tüm işlerin opencv kütüphanesine dayalı olduğunu öğrendik.Temellerini öğrenmeye başladık ve küçük bir kod oluşturdum.  
   
Görsel 9.1 : Python'da görselleri göster  
  
her satırın ne yaptığının açıklamaları:  
  
1 import cv2: Bu satır, OpenCV kütüphanesini projeye ekler, böylece görüntü işleme işlemleri yapabilirsiniz.  
  
2 image = cv2.imread('resim.png'): Bu satır, 'resim.png' adlı bir dosyadan görüntüyü okur ve image adlı bir değişkene atar. 'resim.png' dosyası, bu kodun bulunduğu dizinde bulunmalıdır.  
  
3 cv2.imshow('Görüntü', image): Bu satır, image değişkenindeki görüntüyü ekranda görüntüler. İlk argüman olan 'Görüntü', pencerenin başlığını belirtir. İkinci argüman olan image, görüntüyü temsil eden değişkendir.  
  
4 cv2.waitKey(0): Bu satır, kullanıcı bir tuşa basana kadar pencerenin açık kalmasını sağlar. 0 değeri, beklemenin sonsuz süreceğini belirtir. Kullanıcı herhangi bir tuşa bastığında bu işlem sona erer.  
  
5 cv2.destroyAllWindows(): Bu satır, tüm pencere ve kaynakları kapatır. Kullanıcı tuşa bastığında pencere kapanır ve program sona erer.  
  
  
Daha sonra yeniden boyutlandırma gibi temel görsel işlemlerine geçtik.  
   
Görsel 9.2 : Python'da görüntü boyutunu küçültme  
  
Burada görüntüyü yeniden boyutlandırdık ve tekrar gösterdik.  
Daha sonra görüntüyü yatay veya dikey olarak çevirirsiniz:  
   
Görsel 9.3 : Resimleri çevir  
resized\_image = cv2.resize(image, (200, 200)): Bu satır, image adlı görüntüyü belirtilen boyutlara (200 piksel genişlik ve 200 piksel yükseklik) yeniden boyutlandırır. resized\_image adlı yeni bir görüntü oluşturulur ve bu yeniden boyutlandırılmış görüntüye atanır. Bu, görüntüyü 200x200 piksel boyutuna küçültür veya büyütür.  
  
  
Yatayda\_aynalama = cv2.flip(image, 1): Bu satır, image adlı görüntüyü yatay eksende aynalar. Yani, görüntüyü soldan sağa doğru ters çevirir. Bu işlem sonucu elde edilen görüntü, Yatayda\_aynalama adlı değişkende saklanır.  
Dikeyde\_aynalama = cv2.flip(image, 0): Bu satır, image adlı görüntüyü dikey eksende aynalar. Yani, görüntüyü yukarıdan aşağıya doğru ters çevirir. Bu işlem sonucu elde edilen görüntü, Dikeyde\_aynalama adlı değişkende saklanır.  
Bu kod örneği, belirtilen boyuta yeniden boyutlandırma ve yatay/dikey aynalama işlemlerini içerir. resized\_image, belirtilen boyutta yeniden boyutlandırılmış görüntüyü içerirken, Yatayda\_aynalama ve Dikeyde\_aynalama sırasıyla yatay ve dikeyde aynalanmış versiyonları içerir. Bu işlemler, görüntü işleme ve veri artırma uygulamalarında sıkça kullanılır.  
Daha sonra morfolojik işlemleri geçtik.  
Morfolojik işlemler, nesnelerin şeklini ve yapısını değiştiren işlemlerdir.  
Erozyon ve Genişleme:  
Erozyon, nesneleri küçültmek için kullanılırken, genişleme nesneleri büyütmek için kullanılır. İşte bir örnek:  
   
Görsel 9.4 : Erozyon fotoğraf çalışması  
  
kernel = np.ones((5, 5), np.uint8): Bu satırda, erozyon ve genişletme işlemleri için kullanılacak kernel (çekirdek) oluşturuluyor. Bu kernel, 5x5 boyutunda ve veri tipi np.uint8 olarak tanımlanıyor. Bu kernel, erozyon ve genişletme işlemlerinin işlem yapılacak bölgenin şeklini ve büyüklüğünü belirler.  
  
erosion = cv2.erode(image, kernel, iterations=1): Bu satırda, görüntü üzerinde erozyon işlemi uygulanıyor. image, işlem yapılacak olan görüntüyü temsil eder. kernel, erozyon işlemi için kullanılan kerneli belirtir. iterations=1 parametresi, erozyon işleminin kaç kez uygulanacağını belirtir. Erozyon, görüntü üzerindeki beyaz bölgeyi küçültür ve nesneleri daha ince hale getirir.  
  
dilation = cv2.dilate(image, kernel, iterations=1): Bu satırda, görüntü üzerinde genişletme işlemi uygulanıyor. image, işlem yapılacak olan görüntüyü temsil eder. kernel, genişletme işlemi için kullanılan kerneli belirtir. iterations=1 parametresi, genişletme işleminin kaç kez uygulanacağını belirtir. Genişletme, görüntü üzerindeki beyaz bölgeyi büyütür ve nesneleri daha kalın hale getirir.  
  
Bu işlemler, görüntü işleme alanında nesne tanıma, kenar tespiti ve diğer çeşitli görsel analiz uygulamalarında kullanılır. Erozyon ve genişletme işlemleri, görüntüdeki nesneleri işlemek ve belirli şekil ve yapıları belirlemek için yaygın olarak kullanılır.  
  
Günün sonunda filtreler ve gürültü azaltma konularını öğrendik.  
  
Görüntüler üzerinde filtreler kullanarak görüntülerin işlenmesi, düzeltme veya iyileştirme işlemleri içerir.  
Filtre Uygulama:  
  
  
  
  
  
  
  
Örneğin, bir Gaussian filtresi uygulayarak görüntüyü yumuşatabilirsiniz:  
   
Görsel 9.5 : Python'da görüntüleri yumuşatın  
  
Gürültü Azaltma:  
Görüntülerdeki gürültüyü azaltmak için medyan filtresi kullanabilirsiniz:  
   
Görsel 9.6 : Gürültü Azaltma  
  
blurred = cv2.GaussianBlur(image, (5, 5), 0): Bu satırda, Gaussian bulanıklık işlemi uygulanıyor. image, işlem yapılacak olan görüntüyü temsil eder. (5, 5) argümanı, bulanıklık filtresinin boyutunu belirtir. Bu filtre, 5x5 boyutunda bir pencere kullanır.   
  
  
Son argüman olan 0, Gaussian fonksiyonunun standart sapmasını belirtir ve bu durumda otomatik olarak hesaplanır. Gaussian bulanıklık, bir pikselin etrafındaki diğer piksellerin ağırlıklı ortalamasını alarak görüntüyü yumuşatır.  
  
denoised = cv2.medianBlur(noisy\_image, 5): Bu satırda, median bulanıklık işlemi uygulanıyor. noisy\_image, gürültülü görüntüyü temsil eder. Median bulanıklık, bir pikselin etrafındaki piksellerin sıralı bir liste halinde düzenlendiği ve listenin ortasındaki değerin kullanıldığı bir işlemdir. Bu, özellikle tuz-biber türü tuzaklardan kaynaklanan gürültüyü azaltmak için etkilidir. (5, 5) argümanı, bulanıklık filtresinin boyutunu belirtir.  
Sonuç olarak, blurred adlı bir görüntü, Gaussian bulanıklığı uygulandıktan sonra elde edilir. denoised adlı bir başka görüntü ise median bulanıklığı uygulandıktan sonra elde edilir. Bu işlemler, görüntüdeki gürültüyü azaltmaya veya görüntünün daha pürüzsüz hale getirilmesine yardımcı olur.  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
 Kenarları ve Degradeleri tespit etmek  
Staj`dan 10 günde Gürüntü işlemleri kursunun ikinci yarısıyla başlamamız gerekiyordu  
Görüntülerdeki kenarları ve degradeleri tespit etmeye başlıyoruz   
Kenar tespiti, görüntüdeki nesnelerin sınırlarını tanımlamak için kullanılır.  
Kenar Tespiti:  
Kenarları tespit etmek için Canny kenar tespiti gibi algoritmalar kullanabilirsiniz:  
  
Görsel 10.1 : Kenar Tespiti orneği  
edges = cv2.Canny(image, 200, 150): Bu satırda, image adlı bir görüntü üzerinde Canny kenar algılama işlemi uygulanıyor. Canny kenar algılayıcı, görüntüdeki keskin geçişleri (kenarları) tespit etmek için kullanılan bir yöntemdir.  
İlk argüman olan image, işlem yapılacak olan görüntüyü temsil eder.  
İkinci argüman olan 200, alt eşik (low threshold) değerini belirtir. Bu değer, kenar olarak kabul edilecek piksel yoğunluğunun minimum sınırını belirler. Eğer bir pikselin yoğunluğu bu değeri geçerse, bu piksel kenar olarak kabul edilir.  
Üçüncü argüman olan 150, üst eşik (high threshold) değerini belirtir. Bu değer, piksel yoğunluğunun maksimum sınırını belirler. Eğer bir pikselin yoğunluğu bu değeri aşarsa, bu piksel kesinlikle kenar olarak kabul edilir.  
  
Gradyanlar:  
Gradyanlar, görüntülerdeki parlaklık değişikliklerini belirlemek için kullanılır. Sobel filtresi ile gradyanlar hesaplanabilir:  
  
  
  
   
Görsel 10.2 : Resimlerle Gradyanlar  
  
grad\_x = cv2.Sobel(image, cv2.CV\_64F, 1, 0, ksize=3): Bu satırda, görüntü üzerinde x yönünde (yatay) gradyan hesaplaması yapılır. image, işlem yapılacak olan görüntüyü temsil eder. cv2.CV\_64F, hesaplamaların veri tipini belirtir (64 bit float). 1 ve 0, x ve y yönlerinde türevi alacağınızı belirtir. ksize=3, Sobel operatörünün çekirdek boyutunu (kernel size) belirtir. 3x3'lük bir çekirdek kullanılır.  
  
grad\_y = cv2.Sobel(image, cv2.CV\_64F, 0, 1, ksize=3): Bu satırda, görüntü üzerinde y yönünde (dikey) gradyan hesaplaması yapılır. Yine, image görüntüyü temsil eder ve gradyan hesaplamaları için veri tipi olarak cv2.CV\_64F kullanılır. 0 ve 1, bu sefer x ve y yönlerinde türevi alacağınızı belirtir.  
  
Sonuç olarak, grad\_x ve grad\_y, x ve y yönlerinde gradyan görüntülerini içerir. Bu gradyanlar, görüntüdeki yoğunluk değişikliklerini x ve y yönlerinde ayrı ayrı gösterir ve kenar tespiti işleminde kullanılır.  
  
Daha sonra Köşe Tespiti çıkardık  
Köşe tespiti, görüntülerdeki köşeleri veya önemli özellikleri tanımak için kullanılır.  
Harris Köşe Tespiti:  
Harris köşe tespiti algoritmasıyla köşeleri belirlemek mümkündür:  
  
   
Görsel 10.3 : Harris köşe kodu  
Sonra Kontür Tespiti'yi öğrenmeye gittik.  
   
Görsel 10.4 : Kod çıkışı  
  
Gördüğümüz gibi cv2.drawContours(image, konturlar, -1, (0, 255, 0), 2) kullandığımızda görüntünün kenarlarında yeşil bir çizgi tanımladı  
Her rengin kendine özgü bir algoritması olduğundan bu alanda (0,0,0) renklerle istediğimiz gibi oynayabiliriz.  
image: Bu argüman, konturların çizileceği hedef görüntüyü temsil eder. Konturlar bu görüntünün üzerine çizilir.  
  
  
contours: Bu argüman, çizilecek konturların bir listesini içerir. Her bir kontur, noktaların bir listesi olarak temsil edilir.  
contourIdx: Bu argüman, çizilecek konturun endeksidir. -1 değeri, tüm konturların çizilmesini ifade eder. Yani, eğer -1 kullanırsanız, tüm konturlar çizilir.  
color: Bu argüman, çizilen konturların rengini belirtir. Rengin BGR (Mavi, Yeşil, Kırmızı) kodlaması kullanılır. Örneğin, (0, 255, 0) yeşil bir rengi temsil eder.  
thickness: Bu argüman, konturların çizgi kalınlığını belirtir. Kalınlık piksel cinsinden ifade edilir. Örneğin, 2 kullanarak ince bir çizgi çizilir.  
Yukarıdaki kodda cv2.drawContours(image, contours, -1, (0, 255, 0), 2) kullanılarak, contours listesindeki tüm konturlar gri tonlamalı görüntü olan gray\_image üzerine yeşil renkte ve kalınlığı 2 piksel olan bir çizgi olarak çizilir. Bu, konturları orijinal görüntü üzerinde görselleştirmenizi sağlar.  
  
  
Daha sonra segmentasyonu öğrenmeye geçtik  
  
Nesne segmentasyonu, bir görüntüdeki nesneleri ayırmak için kullanılır.  
Renk Tabanlı Segmentasyon:  
Örneğin, bir renk aralığını belirleyerek renk tabanlı bir segmentasyon gerçekleştirebilirsiniz:  
   
Görsel 10.5 : bir renk aralığını belirleyerek  
  
lower\_red: Bu satırda, alt sınır rengini belirlemek için bir NumPy dizisi oluşturulur. Bu dizi BGR (Mavi, Yeşil, Kırmızı) renk uzayında bir rengi temsil eder. Burada [0, 0, 200] kullanılarak alt sınır koyu mavi bir renk olarak tanımlanmıştır. Yani, bu rengin mavi bileşeni 0, yeşil bileşeni 0 ve kırmızı bileşeni 200'den büyük olan her pikseli içerecektir.  
  
upper\_red: Üst sınır rengini belirlemek için bir başka NumPy dizisi oluşturulur. [250, 250, 250] kullanılarak üst sınır beyaz bir renk olarak tanımlanmıştır. Bu rengin bütün bileşenleri 250'den küçük olan her pikseli içerecektir.  
  
mask = cv2.inRange(image, lower\_red, upper\_red): Bu satırda, cv2.inRange işlevi kullanılarak görüntü üzerinde renk aralığını seçen bir maske oluşturulur. Bu maske, belirli bir renk aralığına uyan pikselleri beyaz (255) olarak işaretler ve diğer pikselleri siyah (0) olarak işaretler. Sonuç olarak, bu maske yalnızca belirli bir renk aralığına uyan pikselleri içerir.  
  
Bu maskeyi kullanarak, örneğin belirli bir renge sahip nesneleri tespit etmek veya bu nesneleri başka bir görüntüyle birleştirmek gibi işlemler yapabilirsiniz. Maske oluşturduktan sonra, beyaz pikselleri içeren bu maskeyi kullanarak orijinal görüntü üzerinde işlem yapabilirsiniz.  
Sonra Özellik Çıkarımı öğrendikSonra Özellik Çıkarımı öğrendik  
Özellik çıkarımı, görüntülerdeki önemli özellikleri belirlemek için kullanılır.  
Özellik Çıkarımı:  
Özellik çıkarımı için SIFT SIFT veya ORB gibi özellik tespit algoritmalarını kullanabilirsiniz.  
   
Görsel 10.6 : SIFT Kod çıkışı  
  
sift = cv2.SIFT\_create(): Bu satırda, SIFT detektörünü (SIFT detector) oluşturuyorsunuz. Bu detektör, SIFT algoritmasını uygulamanızı sağlar.  
keypoints, descriptors = sift.detectAndCompute(image, None): Bu satırda, sift detektörünü kullanarak belirli bir görüntü (image) üzerinde SIFT özelliklerini tespit ediyor ve açıklamalarını hesaplıyor. Bu işlem sonucunda, keypoints adlı bir liste özellik noktalarını ve descriptors adlı bir NumPy dizisi açıklamaları içerir.  
image\_with\_keypoints = cv2.drawKeypoints(image, keypoints, None): Bu satırda, cv2.drawKeypoints işlevi kullanılarak, özellik noktaları (keypoints) orijinal görüntü üzerine çizilir. Çıktı olarak, image\_with\_keypoints adlı bir görüntü elde edersiniz. Bu görüntü, orijinal görüntü üzerinde özellik noktalarını görsel olarak gösterir.  
  
  
Bu şekilde, özellik noktalarını tespit etmek, açıklamak ve görsel olarak işaretlemek için SIFT algoritmasını kullanabilirsiniz. Oluşturulan image\_with\_keypoints görüntüsü, orijinal görüntü üzerindeki SIFT özelliklerini görsel olarak görselleştirir.  
Daha sonra Özellik Eşleştirme'ye geçtik  
Özellik Eşleştirme:  
Özelliklerin eşleştirilmesi için BFMatcher veya FlannMatcher gibi eşleştirme algoritmalarını kullanabilirsiniz.  
   
Görsel 10.7 : İki görüntüdeki benzer noktaları tanımlayan kod  
Bu kod, iki görüntü arasında SIFT (Scale-Invariant Feature Transform) özelliklerini kullanarak eşleştirmeleri tespit eden bir işlemi gerçekleştirir ve sonuçları görsel olarak gösterir.   
sift = cv2.SIFT\_create(): Bu satırda, SIFT detektörünü oluşturuyorsunuz. SIFT detektörü, görüntüden özellik noktalarını ve bu özellik noktalarının açıklamalarını çıkarmak için kullanılır.  
keypoints1, descriptors1 = sift.detectAndCompute(image1, None): İlk görüntü (image1) üzerinde SIFT özelliklerini tespit ediyor ve açıklamalarını hesaplıyor. Bu işlem sonucunda, keypoints1 adlı bir liste özellik noktalarını ve descriptors1 adlı bir NumPy dizisi açıklamaları içerir.  
  
  
keypoints2, descriptors2 = sift.detectAndCompute(image2, None): İkinci görüntü (image2) üzerinde aynı şekilde SIFT özelliklerini tespit ediyor ve açıklamalarını hesaplıyor. Bu işlem sonucunda, keypoints2 adlı bir liste özellik noktalarını ve descriptors2 adlı bir NumPy dizisi açıklamaları içerir.  
bf = cv2.BFMatcher(): Brute-Force eşleştiriciyi (Brute-Force Matcher) oluşturuyorsunuz. Bu eşleştirici, iki görüntü arasında özelliklerin eşleşmelerini bulmak için kullanılır.  
matches = bf.knnMatch(descriptors1, descriptors2, k=2): Brute-Force eşleştiriciyi kullanarak, descriptors1 ve descriptors2 arasındaki özelliklerin eşleşmelerini buluyorsunuz. k=2 parametresi, her bir özelliğin en iyi iki eşleşme bulunmasını sağlar.  
  
matched\_image = cv2.drawMatchesKnn(image1, keypoints1, image2, keypoints2, matches, None, flags=2): Bu satırda, cv2.drawMatchesKnn işlevini kullanarak eşleşen özellik noktalarını görsel olarak gösterirsiniz. image1 ve image2 görüntülerini, keypoints1 ve keypoints2 ile eşleşen özellik noktalarını ve matches ile eşleşen özellik noktalarını içeren sonuç görüntüsünü oluşturursunuz. flags=2, iki görüntüyü yan yana gösterme modunu temsil eder.  
Bu şekilde, iki görüntü arasında SIFT özelliklerini eşleştiren ve bu eşleşmeleri görsel olarak gösteren bir işlem gerçekleştirilir.  
  
  
  
  
  
  
  
  
  
 Kare ve Daire Çizmek  
Staj`den On birinci günde, bu kursu bitirdikten ve yapay zeka alanında büyük ilerlemeler sağlayacak nitelikte büyük miktarda bilgi topladıktan sonra öğrendiklerimizi faydalı projelerde kullanmak zorunda kaldık.  
Kare ve daire çizmek, yerlerini belirlemek, resimlerin üzerine yazmak gibi temel şeylerle başladık ve bu örnek şunu açıklıyor:  
   
Görsel 11.1 : Kare ve daire çizmek kodu  
  
img1 = np.zeros((600, 600, 3), np.uint8): Bu satırda, 600x600 piksel boyutunda ve 3 kanallı (RGB) bir siyah zemin (siyah bir görüntü) oluşturuluyor. Görüntü, np.uint8 veri türü ile oluşturulur.  
  
cv2.line(img1, (0, 0), (img1.shape[1], img1.shape[0]), (255, 0, 0), 2): Bu satırda, cv2.line işlevi kullanılarak bir çizgi çiziliyor. Çizgi, görüntü üzerine çiziliyor. İlk iki parametre çizginin başlangıç ve bitiş noktalarını belirtir. Üçüncü parametre çizginin rengini (Mavi: BGR formatında) belirtir, ve son parametre çizgi kalınlığını belirtir.  
cv2.rectangle(img1, (50, 50), (400, 400), (0, 255, 0), 2): Bu satırda, cv2.rectangle işlevi kullanılarak bir dikdörtgen çiziliyor. İlk iki parametre, dikdörtgenin sol üst köşesini, diğer iki parametre ise sağ alt köşesini belirtir. Üçüncü parametre dikdörtgenin rengini (Yeşil: BGR formatında) belirtir, ve son parametre dikdörtgenin kenar kalınlığını belirtir.  
cv2.circle(img1, (300, 300), 160, (0, 0, 255), 2): Bu satırda, cv2.circle işlevi kullanılarak bir daire çiziliyor. İlk parametre dairenin çizileceği görüntüyü belirtir, ikinci parametre dairenin merkezini, üçüncü parametre dairenin yarıçapını belirtir. Dördüncü parametre dairenin rengini (Kırmızı: BGR formatında) belirtir, ve son parametre dairenin kenar kalınlığını belirtir.  
cv2.putText(img1, 'majdalatareb', (199, 299), cv2.FONT\_HERSHEY\_PLAIN, 2, (255, 0, 255), 1): Bu satırda, cv2.putText işlevi kullanılarak metin ekleniyor. İlk parametre metni içeren görüntüyü belirtir, ikinci parametre metnin yazılacağı konumu belirtir, üçüncü parametre metin fontunu ve boyutunu belirtir, dördüncü parametre metin rengini belirtir ve beşinci parametre metin kalınlığını belirtir.  
Daha sonra iki görüntüyü uzunlamasına bir satırda birleştiren ve bunları enine başka bir satırda birleştiren kodu yazdık.  
   
Görsel 11.2 : Python'da iki görüntüyü birleştirme  
  
np.hstack((img1, img2)): Bu işlem, img1 ve img2 adlı iki görüntüyü yatayda birleştirir. Yani, görüntüler yan yana eklenir. Sonuç olarak, iki görüntü yanyana gelir.  
np.vstack((img1, img2)): Bu işlem, img1 ve img2 adlı iki görüntüyü dikeyde birleştirir. Yani, görüntüler üst üste eklenir. Sonuç olarak, iki görüntü üst üste gelir.  
  
 renkleri ve nesneleri tespit etmek  
On ikinci günde opencv kullanarak görüntüdeki renkleri ve nesneleri tespit etmek için kullanılan bir proje yaptık.  
   
Görsel 12.1: Renklerle oynamak arayüzü Görsel 12.2 :Değişiklikten önceki resimler  
Bu ayarlar önümüze çıkan renkler içindir ve aşağıdaki resimlerdeki gibi renkleri istediğimiz gibi değiştirebiliyoruz.  
   
Görsel 12.3 : Ayarlarda bazı değişiklikler Görsel 12.4 : Değişiklik sonrası resimler  
  
def empty(a): pass: Bu satırda empty adlı bir boş işlev (fonksiyon) tanımlanıyor. Bu işlev, başka bir yerde çağrılacağı zaman hiçbir şey yapmayacak.  
  
cv2.namedWindow('TrackBars'): Bu satırda, bir "TrackBars" adlı bir pencere (window) oluşturuluyor.  
  
cv2.resizeWindow('TrackBars', 640, 420): Bu satırda, "TrackBars" penceresinin boyutu 640x420 piksel olarak ayarlanıyor.  
  
cv2.createTrackbar(...): Bu satırlarda, renk ayarları için kaydırıcıları (trackbars) oluşturuluyor. "Hue" (Ton), "Saturation" (Doygunluk) ve "Value" (Değer) bileşenleri için min ve max değerlerini ayarlamak için bu kaydırıcılar kullanılacaktır.  
  
img = cv2.imread('imgs/car.jpeg'): Bu satırda, analiz edilecek görüntü img adlı değişkene yükleniyor. car.jpeg adlı bir görüntü kullanıldığına dikkat edin.  
  
Ana döngü başlıyor (while True), bu döngü kullanıcı tarafından seçilen renk aralığını temsil eden maskelenmiş görüntüyü oluşturur ve bu görüntüyü görselleştirir.  
  
cv2.cvtColor(img, cv2.COLOR\_BGR2HSV): Bu satırda, görüntü RGB renk uzayından HSV (Hue, Saturation, Value) renk uzayına dönüştürülüyor. HSV renk uzayı, renk ayarlamaları için daha uygundur.  
  
h\_min, h\_max, s\_min, s\_max, v\_min, ve v\_max değişkenleri, kullanıcı tarafından seçilen renk aralıklarını temsil eder. Bu değerler, renk aralığına göre maskenin oluşturulmasında kullanılır.  
  
np.array([h\_min, s\_min, v\_min]) ve np.array([h\_max, s\_max, v\_max]), kullanıcı tarafından seçilen renk aralığını temsil eden alt ve üst sınırları belirler.  
  
cv2.inRange(imgHSD, lower, upper): Bu satırda, görüntüde seçilen renk aralığının maskesi oluşturuluyor. Maskede, belirli bir renk aralığına ait pikseller beyaz, diğerleri ise siyah olacaktır.  
  
  
  
  
cv2.bitwise\_and(img, img, mask=imgmask): Bu satırda, maskelenmiş görüntü oluşturuluyor. Mask, aslında görüntünün belirli bir renk aralığından sadece seçilen renkleri koruyacaktır. Böylece nesne algılama için renk bazlı bir segmentasyon sağlanır.  
  
Son olarak, cv2.imshow(...) işlevi kullanılarak, orijinal görüntü, HSV dönüşümü, renk maskesi ve sonuç görüntüsü görüntülenir. Bu görüntüler ekranda gösterilir ve kullanıcı tarafından belirlenen renk aralığına göre nesneleri belirlemek için kullanılabilir.  
  
Bu kod, nesne algılama ve renk tanıma uygulamaları için temel bir yapı oluşturur ve kullanıcı tarafından belirlenen renk aralıklarına göre nesneleri izole eder.  
  
  
  
  
  
  
  
  
  
 Şekilleri Tanıyan  
Staj`dan 13`ci günü Görseldeki üçgen, kare ve benzeri basit şekilleri tanıyan bir kod oluşturduk.  
Görüntüdeki konturları (şekilleri) bulmak ve analiz etmek için kullanılır.  
cv2.findContours işlevi, görüntüdeki konturları algılar ve bu konturları countours ve hierachy değişkenlerine döndürür.  
Her kontur için alanı hesaplar ve büyük olanları seçer.  
Konturun çevresini hesaplar (peri) ve bu çevreye dayalı olarak köşeleri (approx) tespit eder.  
Köşe sayısına (objCor) dayanarak şekil türünü belirler (üçgen, kare, dikdörtgen veya daire).  
Şekillerin etrafına dikdörtgen çizer ve belirtilen türü yazdırır.  
Ana Program:  
  
Bir görüntü yüklenir.  
Görüntü gri tonlamalı yapılır.  
# Görüntüyü yükleyin  
img = cv2.imread('imgs/shapes.jpg')  
imgcontour = img.copy()  
  
# Gri tonlamaya çevirin  
gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)  
Eğer uygularsak bize bu çıktıyı verecektir.  
   
Görsel 13.1 : Resmi gümüşe dönüştür  
  
  
  
Görüntü üzerine Gaussian bulanıklığı uygulanır.  
Canny kenar algılama uygulanır ve şekillerin kenarları belirlenir.  
getcontour fonksiyonu bu kenarları ve şekilleri işler.  
Sonuçlar çeşitli görüntülerde gösterilir.  
# Gaussian Bulanıklığı uygulayın  
imgblure = cv2.GaussianBlur(gray, (9, 9), 1)  
# Canny kenar algılama uygulayın  
imgcanny = cv2.Canny(imgblure, 50, 50)  
# Şekilleri işlemek için getcontour fonksiyonunu kullanın  
getcontour(imgcanny)  
Eğer uygularsak bize bu çıktıyı verecektir.  
   
Görsel 13.2 : Tanımdan sonra çıkışı  
  
  
  
  
  
  
  
  
  
  
 Yüzleri Tanıyan  
Staj`dan 14 günü Resimler ve videolar üzerinde derin bir çalışmaya başladık, önce videonun nasıl açılacağını bilen kodlar yazmaya başladık, ardından resimlerdeki yüzleri tanıyan kodlar yazdık.  
  
   
Görsel 14.1 : Resimlerdeki kişileri tanımlama  
  
Daha sonra videolardaki yüzleri tanıması için bu özelliği açtık  
  
   
Görsel 14.2 : Videolardaki yüzleri tanımlama  
  
Önceki iki şekilde de fark ettiğimiz gibi gerekli yüzleri seçip çıktıya sayıyı yazıyoruz.  
while True döngüsü sonsuz bir döngü oluşturur, bu da demek oluyor ki kod, bazı video kaynağından (muhtemelen cap) sürekli olarak kareler yakalamaya devam edecektir.  
\_, img = cap.read(): Bu satır, video kaynağından bir kareyi yakalar ve bu kareyi img değişkenine kaydeder. Alt çizgi (\_) cap.read() işleminin dönüş değerine ilgi duymadığınızı belirtmek için kullanılır.  
imggray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY): Bu satır yakalanan kareyi gri tonlara dönüştürür.   
  
faces = faceCascade.detectMultiScale(imggray, 1.1, 4): Bu satır, gri tonlu görüntüde yüzleri tespit etmek için önceden eğitilmiş bir kaskat sınıflandırıcıyı (faceCascade) kullanır. 1.1 ve 4 parametreleri, hassasiyeti ve bir bölgeyi yüz olarak ilan etmek için gereken minimum komşu dikdörtgen sayısını kontrol eder.  
faceCascade = cv2.CascadeClassifier('haarcascade\_frontalface\_default.xml') Bu satır, yüzleri bulmak için önceden eğitilmiş bir dosya olduğundan en önemli satır olarak kabul edilir.  
Daha sonra opencv'de bir görseli tanımlayan, onu Python ortamında değiştiren, ardından onu yeni bir konuma kaydedip ayrı bir görsel olarak görüntüleyen bir kod oluşturdum.   
  
   
Görsel 14.3 : Görsellerin üzerine yazı yazmak  
  
cv2.putText(imggray, 'majdalatareb', (50, 50), cv2.FONT\_HERSHEY\_COMPLEX, 1, (0, 255, 255), 2): Bu satır, cv2.putText işlevini kullanarak bir metin ekler. Parametreleri şu şekildedir:  
  
imggray: Gri tonlu görüntü, metnin eklenmesi gereken görüntü.  
'majdalatareb': Eklemek istediğiniz metin.  
(50, 50): Metnin görüntü üzerindeki konumu. Burada, metin sol üst köşeden 50 piksel sağa ve 50 piksel aşağıda başlar.  
cv2.FONT\_HERSHEY\_COMPLEX: Metin yazı tipi.  
1: Metin boyutu ölçeği (1 kat büyütme).  
(0, 255, 255): Metin rengi (BGR renk kodu), bu durumda sarı.  
2: Metin kalınlığı.  
cv2.imwrite('majdalatarebgray.jpg', imggray): Bu satır, gri tonlu görüntüyü 'majdalatarebgray.jpg' adlı bir JPEG dosyasına kaydeder. Bu, metin eklenmiş görüntünün saklanmasını sağlar.  
Sonuç olarak, bu kod metni gri tonlu bir görüntüye ekler ve bu görüntüyü 'majdalatarebgray.jpg' adlı bir dosyada kaydeder. Metni ve metin konumunu ihtiyaçlarınıza göre özelleştirebilirsiniz.  
Daha sonra videoyu mümkün olan tüm piksellere bölerek görsellere dönüştüren bir kod oluşturdum.Bu örnekte videoyu örnekte gösterildiği gibi 400'den fazla görsele dönüştürdük.  
   
  
Görsel 14.4 : Fotoğraflar için video oynat Görsel 14.5 : Videodan alınan tüm fotoğraflar  
   
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
 Yolo formatıdaki Proje  
Staj`dan 15 günü Staj tamamlamak için üzerinde çalışmak üzere 5 proje aldık  
Yolo formatıdaki txt dosyasını saklamak x,y türüne çevirmek ilk projemizdi.  
Txt formatında yeni bir dosya oluşturduk ve bunun x , y'ye çevrilmesi için bazı şeyler ve birçok özellik yazdık.  
  
   
Görsel 15.1 : Bilgi dosyası  
  
Daha sonra Python ile fonksiyonu kullanarak dosyayı açtık.  
   
  
Daha sonra her satırı analiz ettik ve satırı ayırlın  
   
  
Daha sonra bu kodu yazarak verileri (örneğin: kategori, x, y, genişlik, yükseklik) çıkarıyoruz.  
  
   
  
Daha sonra x , y koordinatlarını hesapladık  
  
   
  
  
Daha sonra nesnenin konumunu kullanarak istenen eylemi gerçekleştiriyoruz, örneğin buraya konumları yazdırıyoruz  
  
   
  
Eğer kodu çalıştırırsak bize bu çıktıyı verecektir.  
   
Gerekli olan şey bu….  
İkinci projemizde ise sıft soft yöntemini kullanarak görselleri eşleştiren bir kod oluşturduk.  
   
Görsel 15.2 : sıft soft çıkış kodu  
  
Bu kod, iki farklı görüntüdeki benzer özellikleri tespit etmeye ve eşleştirmeye yardımcı olur. Özellikle, SIFT algoritması kullanılarak her iki görüntüdeki çeşitli özellikler (örneğin, kenarlar, köşeler, benzersiz yapılar) tanımlanır. Daha sonra bu özellikler karşılaştırılır ve iki görüntü arasındaki benzerlikler belirlenir.  
Satırda , İlk olarak, gerekli kütüphaneleri içe aktarıyoruz.  
import cv2  
Satırda ,import numpy as np  
  
Satırda ,İlk görüntüyü "image3.jpg" dosyasından okuyoruz ve "image1" adlı bir değişkende saklıyoruz.  
  
Satırda , image1 = cv2.imread('image3.jpg')  
Aynı şekilde, ikinci görüntüyü "image4.bmp" dosyasından okuyoruz ve "image2" adlı bir değişkende saklıyoruz.  
Satırda , image2 = cv2.imread('image4.bmp')  
Şimdi, görüntüleri işlemek için bir SIFT (Scale-Invariant Feature Transform) algılayıcısı oluşturuyoruz.  
  
Satırda , sift = cv2.SIFT\_create()  
Her iki görüntüdeki SIFT özelliklerini ve açıklamalarını hesaplıyoruz. Özellikler, "keypoints1" ve "descriptors1" ile birinci görüntü için ve "keypoints2" ve "descriptors2" ile ikinci görüntü için saklanır.  
  
Satırda , keypoints1, descriptors1 = sift.detectAndCompute(image1, None)  
Satırda , keypoints2, descriptors2 = sift.detectAndCompute(image2, None)  
Şimdi, Brute-Force eşleştiriciyi (bf) oluşturuyoruz. Bu, özelliklerin eşleştirilmesi için kullanılacak.  
Satırda ,bf = cv2.BFMatcher()  
İki görüntü arasındaki eşlemeleri bulmak için "bf.knnMatch" yöntemini kullanıyoruz. "matches", eşleşme bilgilerini içeren bir liste olacaktır.  
Satırda , matches = bf.knnMatch(descriptors1, descriptors2, k=2)  
Eşleştirmeleri görsel olarak göstermek için "cv2.drawMatchesKnn" yöntemini kullanıyoruz. Bu, eşleşen özellikleri çizgi ile bağlar ve bir sonuç görüntüsü oluşturur.  
Satırda , matched\_image = cv2.drawMatchesKnn(image1, keypoints1, image2, keypoints2, matches, None, flags=2)  
İlk görüntüyü, ikinci görüntüyü ve eşleştirmeleri göstermek için "cv2.imshow" kullanıyoruz.  
Satırda , cv2.imshow('image1', image1)  
Satırda , cv2.imshow('Matches', matched\_image)  
Kullanıcının bir tuşa basmasını beklemek için "cv2.waitKey(0)" kullanıyoruz. Bu, pencereyi açık tutar ve kullanıcı kapatana kadar bekler.  
 Her Şeyi Tanıyan Bir Program Oluşturmak  
Staj`dan 16 günü Önünde kamerada veya videoda gösterdiğiniz her şeyi tanıyan bir program oluşturdum .  
   
Görsel 16.1 : her şeyi tanıyan program kodu  
Bu kod, önceden eğitilmiş bir algoritma kullanarak nesneleri tanır .  
 İlgili kütüphaneleri ve kaynakları içe aktarın:  
cv2: OpenCV kütüphanesini kullanır.  
Kamera akışını almak için cv2.VideoCapture kullanılır.  
Nesne sınıflarının isimlerini ve modelin konfigürasyonunu içeren dosyaları içe aktarır.  
cap = cv2.VideoCapture(1)  
Kameradan video akışı almak için cv2.VideoCapture kullanıyoruz. "1" kameranın numarasını temsil eder, bu numara cihaza bağlı olarak değişebilir.  
calssNames = []  
Nesne sınıfı isimlerini saklamak için boş bir liste olan classNames oluşturuyoruz.  
  
classFile= 'object/coco.names'  
Nesne sınıfı isimlerinin bulunduğu dosyanın yolunu belirtiyoruz.  
with open(classFile,'rt') as f:  
 classNames = f.read().rstrip('\n').split('\n')  
classFile adlı dosyayı açıyoruz ve sınıf isimlerini okuyoruz. Her bir satır bir nesne sınıfını temsil eder ve classNames listesine eklenir.  
configpath = 'object/ssd\_mobilenet\_v3\_large\_coco\_2022\_01\_14.pbtxt'  
weightpath = 'object/frozen\_inference\_graph.pb'  
Modelin konfigürasyon dosyasının (configpath) ve ağırlık dosyasının (weightpath) yollarını belirtiyoruz.  
net = cv2.dnn\_DetectionModel(weightpath,configpath)  
OpenCV'nin cv2.dnn\_DetectionModel sınıfını kullanarak modeli yüklüyoruz.  
net.setInputSize(320 , 230)  
Giriş görüntülerinin boyutunu belirtiyoruz.  
net.setInputScale(1.0 / 127.5)  
Giriş görüntülerinin ölçeğini ayarlıyoruz.net.setInputMean((127.5, 127.5, 127.5))  
Ortalama değerleri belirtiyoruz.  
net.setInputSwapRB(True)  
Renk kanallarını değiştirip değiştirmediğimizi belirtiyoruz.  
Sonsuz bir döngü:  
while True :  
Video akışını sürekli olarak işleyen bir sonsuz döngü başlatıyoruz.  
\_, img = cap.read()  
Kameradan bir kare alınır ve img değişkenine atılır. cap.read() işlevi kameradan bir kare alır ve karenin bir kopyasını img değişkenine atar.  
classIds, confs, bbox = net.detect(img, confThreshold=0.6)  
Modeli kullanarak nesne tanıma işlemi yapar ve tanımlanan nesne sınıfı, güven değeri ve sınırlayıcı kutuların bilgilerini döndürür. Tanımanın güven eşiği "0.6" olarak belirlenmiştir.  
Nesneleri işleme: if len(classIds) != 0:  
 for classIds, confidence, box in zip(classIds.flatten(), confs.flatten(), bbox):  
 cv2.rectangle(img, box, color=(0, 255, 0), thickness=2)  
 cv2.putText(img, classNames[classIds - 1], (box[0] + 10, box[1] + 20),  
 cv2.FONT\_HERSHEY\_COMPLEX, 1, (255, 255, 255), thickness=2)  
Eğer bir nesne sınıfı tanımlandıysa (yani len(classIds) != 0), her bir tanımlanan nesneyi işleriz.  
cv2.rectangle ile nesnenin sınırlayıcı kutusunu çizeriz.  
Kodu tamamladıktan sonra çalıştırırsak bize bu çıktıyı vereceki   
 Görsel 16.2: İnsanları tanıyın Görsel 16.3 : Bir şeyleri tanımak 1  
   
 Görsel 16.4 : Bir şeyleri tanımak 2 Görsel 16.5 : Bir şeyleri tanımak 3  
   
 Görsel 16.6 : Bir şeyleri tanımak 4  
Etrafına yeşil bir kutu çizerek her şeyi tanımlayabilirsiniz.  
 Karşılaştırma Kodu Oluştur  
Staj`dan 17 günü Bir görüntüyü tanıyan, onu verisine alan ve daha sonra diğer görüntülerle karşılaştıran bir kod oluşturdum, buna benzer bir görüntü çıkarsa doğru, görünmezse yanlış yazdırır.  
   
Görsel 17.1 : Karşılaştırma Kodu  
Bu kod, verilen bir resmin yüz kodlamasını alır ve ardından bu kodlamayı diğer resimlerin yüz kodlamalarıyla karşılaştırır. Yüz benzerliği sonuçları result listesine eklenir ve iki resim de ekranda gösterilir. Eğer result True ise, bu iki resmin aynı kişiye ait olduğu anlamına gelir.  
cv2, OpenCV kütüphanesini içe aktarır.  
face\_recognition, yüz tanıma işlemleri için kullanılan bir kütüphanedir.  
glob, dosya adlarını eşleştirmek için kullanılır.  
Bu bölümde, folder\_dir adlı bir klasörün yolu belirlenir ve ardından ilk resim seçilir. Bu resim yüklenir, RGB renk formatına dönüştürülür ve yüz kodlaması alınır.  
Resimlerin bulunduğu klasörün yolu ( folder\_dir = 'images')  
cv2.imread("images/cristiano ronaldo 1.jpg"):  
Bu satır, OpenCV (cv2) kütüphanesini kullanarak "images" dizinindeki "cristiano ronaldo 1.jpg" adlı dosyadan bir resmi okur. Resim bir NumPy dizisi olarak yüklenir.  
rgbimage1 = cv2.cvtColor(image1, cv2.COLOR\_BGR2RGB):  
Bu satır, yüklenen resmi BGR renk uzayından (genellikle OpenCV tarafından kullanılır) RGB renk uzayına dönüştürür. Bazı yüz tanıma kütüphaneleri, face\_recognition gibi, resmin RGB formatında olmasını bekler.  
 img\_encoding1 = face\_recognition.face\_encodings(rgbimage1)[0]:  
  
Bu satır, resmin yüz kodlamasını hesaplamak için face\_recognition kütüphanesini kullanır. Yüz kodlama, bir kişinin resimdeki yüz özelliklerinin sayısal bir temsilidir. Bu, yüz karşılaştırması ve tanıma gibi çeşitli yüz tanıma görevleri için kullanılabilen bir vektördür.  
face\_recognition.face\_encodings işlevi, RGB resmi girdi olarak alır ve resimde bulunan yüz kodlamalarının bir listesini döndürür. Bu durumda, listeyi kullanarak ilk (ve muhtemelen tek) yüz kodlamasını [0] kullanarak çıkarır.  
Artık img\_encoding1 değişkeni, "cristiano ronaldo 1.jpg" resmindeki tespit edilen yüz için kodlamayı içeriyor. Bu kodlama, yüzleri karşılaştırmak veya bireyleri tanımlamak gibi çeşitli yüz tanıma görevleri için kullanılabilir.  
for images in glob.iglob(f'{folder\_dir}/\*')::  
  
Bu satır, glob modülünü kullanarak belirli bir dizindeki dosyaları dolaşmak için bir döngü başlatır. folder\_dir değişkeni, işlemek istediğiniz dizin yolunu içerir.  
if (images.endswith(".png") or images.endswith(".jpg") or images.endswith(".jpeg"))::  
  
Bu satır, yalnızca belirli uzantılara sahip dosyaları işlemek için bir koşul ifadesi oluşturur. images değişkeni, döngü tarafından işlenen dosyanın tam yolu içerir.  
endswith yöntemi ile, dosya adının belirli uzantılarla bitip bitmediği kontrol edilir (.png, .jpg, veya .jpeg uzantıları).  
print(images):  
  
Eğer dosyanın uzantısı .png, .jpg veya .jpeg ise, bu dosyanın tam yolunu ekrana yazdırır. Bu şekilde belirli bir dizindeki belirli uzantılara sahip tüm resim dosyalarını listeleyebilirsiniz.  
  
image2 = cv2.imread(images):  
  
Bu satır, döngü tarafından işlenen her resim dosyasının yolu (images) kullanılarak resmi okur ve OpenCV (cv2) kütüphanesi ile bir NumPy dizisi olarak yükler.  
  
  
  
rgbimage2 = cv2.cvtColor(image2, cv2.COLOR\_BGR2RGB):  
  
Bu satır, yüklenen resmi BGR renk uzayından (OpenCV tarafından kullanılan renk uzayı) RGB renk uzayına dönüştürür. Yüz tanıma kütüphaneleri, resmin RGB formatında olmasını beklediğinden bu dönüşümü yapmak önemlidir.  
  
img\_encoding2 = face\_recognition.face\_encodings(rgbimage2)[0]:  
  
Bu satır, face\_recognition kütüphanesi kullanılarak resmin yüz kodlamasını hesaplar. Bu, resimde bulunan yüzün sayısal bir temsilini oluşturur. Kodlama, yüz tanıma veya karşılaştırma işlemleri için kullanılabilir.  
  
result = face\_recognition.compare\_faces([img\_encoding1], img\_encoding2):  
  
Bu satır, face\_recognition kütüphanesini kullanarak iki yüz kodlamasını karşılaştırır. face\_recognition.compare\_faces işlevi iki argüman alır: birinci argüman, karşılaştırmak istediğiniz yüz kodlamalarının bir listesidir (burada sadece img\_encoding1 bulunuyor) ve ikinci argüman, karşılaştırmak istediğiniz diğer yüz kodlamasıdır (img\_encoding2).  
print("Result", result):  
Bu satır, karşılaştırma sonucunu ekrana yazdırır. Eğer result True ise, bu, iki yüzün eşleştiği anlamına gelir, yani img\_encoding1 ve img\_encoding2 aynı kişiyi temsil ediyor olabilir. Eğer result False ise, bu, iki yüzün eşleşmediği anlamına gelir.  
  
Bu resimleri ekledim ve bu tüm resimler ile Cristiano Ronaldo'nun resmi arasında bir karşılaştırma olacak.  
   
Görsel 17.2 : Programda tanımlanan görseller  
  
Eğer kodu çalıştırırsak, bize bu çıktıyı verecektir ve Cristiano'nun herhangi bir görüntüsü doğru şekilde çıkacaktır.  
   
Görsel 17.3 : Eşleşme varsa çıkışı  
Eğer görüntü Cristiano'ya ait değilse, bu bize yanlış bir çıktı verecektir.  
   
Görsel 17.4 : Eşleşme yoksa çıkışı  
  
  
  
  
  
  
  
  
 eli tanır  
Staj`dan 18 gününde Profesyonel hayatımızda iki önemli proje gerçekleştirdik: Birincisi, bir cep telefonunun veya dizüstü bilgisayarın kamerasını Python'a bağlayarak eli tanır ve elin kemiklerini temsil eder ve mevcut parmakları sayar.  
  
   
 Görsel 18.1 : El tanıma kodu  
Bu Python kodu, MediaPipe kütüphanesini kullanarak ellerin izlenmesini ve parmak sayısının tespit edilmesini sağlar.   
İlk olarak, gerekli kütüphaneleri ve modülleri içe aktarırız:  
cv2: OpenCV kütüphanesi görüntü işleme için kullanılır.  
numpy: Sayısal işlemler için kullanılır.  
mediapipe ve mp: MediaPipe kütüphanesi, el izleme ve çizim işlemleri için kullanılır.  
  
ondan sonra, MediaPipe Hands modülünü içe aktarırız:  
mpHands = mp.solutions.hands: El izleme modelini başlatırız.  
hands = mpHands.Hands(): Hands modelini yükleriz.  
Mediapipe çizim yardımcı modülünü içe aktarırız:  
  
mpDraw = mp.solutions.drawing\_utils: Çizim işlemleri için kullanılır.  
  
Onadan sonra, # İzlenecek parmak uçları (parmakların uçları) indeksleri.  
tipIds = [4, 8, 12, 16, 20]  
ondan sonra, # Sonsuz bir döngü başlat.  
while True:  
 # Kamera görüntüsünü başlat ve başarılı mı olduğunu kontrol et.  
 success, img = cap.read()  
  
 # Görüntüyü yatayda yansıt (çoğu kamera varsayılan olarak yansıtmıştır).  
 img = cv2.flip(img, 1)  
  
 # Görüntüyü RGB renk formatına dönüştür (Mediapipe Hands API bu formatı bekler).  
 imgRGB = cv2.cvtColor(img, cv2.COLOR\_BGR2RGB)  
  
 # Hands API'sini kullanarak el konumlarını işle.  
 results = hands.process(imgRGB)  
  
 # Tüm landmark'ları saklamak için bir liste oluştur.  
 lmList = []  
  
  
  
  
  
  
  
  
  
  
  
  
eli tanır + Yüz tanıma  
Staj`dan 19 gününde Profesyonel hayatımızda iki önemli proje gerçekleştirdik Birincisi devam :  
ondan sonra, Eğer en az bir el tespit edildiyse:  
   
Toplam parmak sayısını hesapla ve ekrana yazdır.  
  
fingers.count(1): fingers adlı bir liste içinde her parmağın durumunu temsil eden 1 ve 0 değerleri bulunur. 1, parmağın yukarıda olduğunu temsil eder. Bu satır, listedeki 1'lerin sayısını hesaplar, yani yukarı kalkmış parmakların sayısını verir.  
  
cv2.putText(img, f'{totalFingers}', (40, 80), cv2.FONT\_HERSHEY\_SIMPLEX, 3, (0, 0, 255), 6): cv2.putText işlevi, görüntüye metin eklemek için kullanılır.   
  
El izleme sonuçlarını görüntüle.  
  
ESC tuşuna basarak video akışını kapat.Bundan sonra kodu çalıştırırsak bize şu çıktıları verecektir.  
   
 görsel 19.1: Kod parmaksız çıktı görsel 19.2 : Kod tek parmakla çıktı  
  
   
görsel 19.3 : Kodu iki parmakla çıktı görsel 19.4 : Kod 3 parmakla çıktı  
  
   
 görsel 19.5 : Kod 4 parmakla çıktı görsel 19.6 : Kod 5 parmakla çıktı  
  
Kodu çalıştırdığımızda fark ettiğimiz gibi videodaki elleri tanıyor, önceden eğitilmiş bir algoritma kullanarak parmakları sayıyor ve çıktıyı programdaki çıktı kutusuna yazdırıyor.  
Projeyi bitirdikten sonra ikinci projeye başlamak için biraz zamanımız oldu.  
Profesyonel hayatımızda iki önemli proje gerçekleştirdik:ikincisi , Bilgisayara bağlanan telefon kamerası veya bilgisayar kamerası aracılığıyla kişinin resmini ekleyerek, önceden eğitilmiş bir algoritma ile karşısına çıkan kişilerin isimlerini tespit eden bir proje oluşturacağız. zeka, video görüntüsünü gösterdiğimizde kişinin özelliklerini kaydedip kişiyle eşleştirecek ve üstüne Adını Yaz ile onu tanımlayacak.   
   
Görsel 19.7 : İkinci projenin temelleri  
Satırda cv2: OpenCV (Open Source Computer Vision Library) kütüphanesidir. Görüntü işleme, nesne algılama ve çeşitli görüntü işleme görevleri için kullanılır.  
  
Satırda numpy as np: NumPy, Python dilinde bilimsel hesaplama işlemleri yapmak için kullanılan bir kütüphanedir. Diziler ve çok boyutlu diziler üzerinde işlem yapmak için yaygın olarak kullanılır.  
  
Satırda : face\_recognition: Bu, yüz tanıma işlemlerini gerçekleştirmek için kullanılan bir Python kütüphanesidir. Yüz tanıma işlevleri sunar ve yüzlerin kodlamalarını (encodings) hesaplamak ve karşılaştırmak için kullanılır.  
  
Satırda : os (Operating System) modülü, Python'da işletim sistemi ile etkileşimde bulunmak için kullanılan bir standart kütüphanedir. Bu modül, işletim sistemi düzeyinde çeşitli işlemleri gerçekleştirmenize olanak tanır.  
Satırda : path = 'persons': Tanımlanan kişilerin fotoğraflarının bulunduğu klasörün yolu path değişkenine atanır.  
  
Satırda : images = []: Her bir kişinin fotoğrafının saklanacağı boş bir liste olan images oluşturulur.  
  
Satırda :classNames = []: Her kişinin adının saklanacağı boş bir liste olan classNames oluşturulur.  
  
Satırda: personsList = os.listdir(path): persons klasöründe bulunan dosya isimlerini personsList adlı bir liste olarak alır.  
  
Satırda :for cl in personsList:: personsList listesini döngüye alır ve her bir kişi için aşağıdaki işlemleri yapar.  
  
  
  
 Yüz tanıma  
Staj`dan 20 günüünde Profesyonel hayatımızda iki önemli proje gerçekleştird ikikincisi devamı:  
   
Görsel 20.1 : program kodu 2  
 Görsel : İnsanları fotoğraflarından tanımlamaya yönelik bir proje tam kodu  
Satırda : curPersonn = cv2.imread(f'{path}/{cl}'): Şu anki kişinin fotoğrafını yükler ve curPersonn adlı değişkene atar.  
  
Satırda : images.append(curPersonn): Kişinin fotoğrafını images listesine ekler.  
Satırda : classNames.append(os.path.splitext(cl)[0]): Kişinin ismini classNames listesine ekler. Dosya adından uzantıyı (örneğin ".jpg") kaldırmak için os.path.splitext(cl)[0] kullanılır.  
  
Satırda : encodeListKnown = findEncodeings(images): Tanımlanan kişilerin yüz tanıma kodlarını (encodings) encodeListKnown değişkenine atar. Bu kodları bulmak için findEncodeings fonksiyonunu kullanır.  
  
Satırda : cap = cv2.VideoCapture(1): Kamerayı kullanarak gerçek zamanlı görüntü yakalamak için bir VideoCapture nesnesi oluşturur. "1" argümanı, birincil kamerayı seçer.  
  
Satırda : while True:: Sonsuz bir döngü başlatır, bu döngü sürekli olarak yeni görüntüler alır ve yüz tanıma işlemini gerçekleştirir.  
  
 Satırda : \_, img = cap.read(): Kameradan bir çerçeve alır ve bu çerçeveyi img değişkenine atar.  
  
Satırda : imgS = cv2.resize(img, (0, 0), None, 0.25, 0.25): Çerçeveyi küçültmek için boyutlarını dörtte bir oranında düşürür.  
  
Satırda : imgS = cv2.cvtColor(imgS, cv2.COLOR\_BGR2RGB): Çerçeveyi RGB renk uzayına dönüştürür.  
  
Satırda : faceCurentFrame = face\_recognition.face\_locations(imgS): Çerçevedeki yüzlerin konumlarını bulur ve bu konumları faceCurentFrame değişkenine atar.  
  
Satırda : encodeCurentFrame = face\_recognition.face\_encodings(imgS, faceCurentFrame): Çerçevedeki yüzlerin kodlarını (encodings) hesaplar ve bu kodları encodeCurentFrame değişkenine atar.  
Satırda : Döngü, her bir yüz ve kod için aşağıdaki işlemleri yapar:  
Satırda : matches = face\_recognition.compare\_faces(encodeListKnown, encodeface): Tanımlanan kişilerin kodları ile tespit edilen yüzün kodunu karşılaştırır ve eşleşenlerin listesini döndürür.  
Satırda : faceDis = face\_recognition.face\_distance(encodeListKnown, encodeface): Tanımlanan kişilerin kodları ile tespit edilen yüzün kodu arasındaki uzaklıkları hesaplar.  
Satırda : matchIndex = np.argmin(faceDis): En yakın yüzü bulmak için uzaklıkların dizisindeki en küçük değerin dizinini alır.  
Satırda : Eşleşen kişinin adı name değişkenine atanır ve büyük harflerle yazılır.  
Satırda : Yüzün konumu çerçeve üzerine bir dikdörtgen ile çizilir ve kişinin adı altında görüntülenir.  
Satırda : Son olarak, sonuçları içeren çerçeve ekranda gösterilir ve bir tuşa basılarak döngü sürdürülür.  
  
Artık projeyi bitirdiğimize göre sonsuz sayıda fotoğraf ekleyebilir, böylece onları kaydedebilir, analiz edebilir, sahibini belirleyebilir ve karşısına çıktığında adını yazabiliriz.  
  
Görsel 20.2 : Projede tanımlanan görseller  
Örneğimde bunlar projeye eklediğim resimler ve kamerayı bu yüzlere aynı resimlere değil farklı yerlere doğrultacağız ve resmi özelliklerinden tanıdığını göreceğiz.  
   
Görsel 20.3 : İmajımı tanı  
Bu örnekte fotoğrafımı depolama dosyasına ekledim, ardından mobil kamerayı dizüstü bilgisayara bağlayıp ön kamerayı açtım ve fotoğrafım çıktığında program yüzümü tanıdı ve işlem dosyasına ve dosyaya adımı yazdı. programın çıkış kutusu.  
   
Görsel 20.4 : Steve Jobs'u tanımak  
Bu örnekte Steve Jobs'un rastgele bir videosunu açtım ve yüzü kameraya çıktığında program onu ​​tanıdı ve adını yazdı  
   
 Görsel 20.5 : Bill Gates'i tanıyın Görsel 20.6 : Elon Musk'u tanımak  
Aynı şeyi Elon Musk ve Bill Gates ile yapılan basın röportajlarında da tekrarladım ve program onları tespit etti.   
Sonsuz sayıda yüz ekleyip tanımlayabiliriz; yapay zeka bunları tanıyacaktır  
Bu proje gelişirse toplum için tehlikeli olan kişilerin tespit edilmesi, kimliklerinin belirlenmesi ve tehlikenin azaltılması amacıyla kameralara da eklenebilir.  
Böylece Stajımız tamamladık ve yapay zeka alanında yapılabilecek her şeyi yaptık.

Şekil, Çizelge veya Ek No ve Adı Sayfa No

**STAJIN YAPILDIĞI KURUM VEYA KURULUŞUN TANITIMI**

**GİRİŞ**

.....**/**....**/201 tarihinden** .....**/**....**/201 tarihine kadar bir haftalık çalışma**.

|  |  |  |  |
| --- | --- | --- | --- |
| GÜN | Yapılan İşler | Yapılan işe ait bilginin bulunduğu sayfa | Saat |
| Pazartesi |  |  |  |
| Salı |  |  |  |
| Çarşamba |  |  |  |
| Perşembe |  |  |  |
| Cuma |  |  |  |
| Cumartesi |  |  |  |
| Denetleyenin İmzası |  | Toplam Saat |  |

.....**/**....**/201 tarihinden** .....**/**....**/201 tarihine kadar bir haftalık çalışma**

|  |  |  |  |
| --- | --- | --- | --- |
| GÜN | Yapılan İşler | Yapılan işe ait bilginin bulunduğu sayfa | Saat |
| Pazartesi |  |  |  |
| Salı |  |  |  |
| Çarşamba |  |  |  |
| Perşembe |  |  |  |
| Cuma |  |  |  |
| Cumartesi |  |  |  |
| Denetleyenin İmzası |  | Toplam Saat |  |

.....**/**....**/201 tarihinden** .....**/**....**/201 tarihine kadar bir haftalık çalışma**

|  |  |  |  |
| --- | --- | --- | --- |
| GÜN | Yapılan İşler | Yapılan işe ait bilginin bulunduğu sayfa | Saat |
| Pazartesi |  |  |  |
| Salı |  |  |  |
| Çarşamba |  |  |  |
| Perşembe |  |  |  |
| Cuma |  |  |  |
| Cumartesi |  |  |  |
| Denetleyenin İmzası |  | Toplam Saat |  |

.....**/**....**/201 tarihinden** .....**/**....**/201 tarihine kadar bir haftalık çalışma**

|  |  |  |  |
| --- | --- | --- | --- |
| GÜN | Yapılan İşler | Yapılan işe ait bilginin bulunduğu sayfa | Saat |
| Pazartesi |  |  |  |
| Salı |  |  |  |
| Çarşamba |  |  |  |
| Perşembe |  |  |  |
| Cuma |  |  |  |
| Cumartesi |  |  |  |
| Denetleyenin İmzası |  | Toplam Saat |  |

.....**/**....**/201 tarihinden** .....**/**....**/201 tarihine kadar bir haftalık çalışma**

|  |  |  |  |
| --- | --- | --- | --- |
| GÜN | Yapılan İşler | Yapılan işe ait bilginin bulunduğu sayfa | Saat |
| Pazartesi |  |  |  |
| Salı |  |  |  |
| Çarşamba |  |  |  |
| Perşembe |  |  |  |
| Cuma |  |  |  |
| Cumartesi |  |  |  |
| Denetleyenin İmzası |  | Toplam Saat |  |

.....**/**....**/201 tarihinden** .....**/**....**/201 tarihine kadar bir haftalık çalışma**

|  |  |  |  |
| --- | --- | --- | --- |
| GÜN | Yapılan İşler | Yapılan işe ait bilginin bulunduğu sayfa | Saat |
| Pazartesi |  |  |  |
| Salı |  |  |  |
| Çarşamba |  |  |  |
| Perşembe |  |  |  |
| Cuma |  |  |  |
| Cumartesi |  |  |  |
| Denetleyenin İmzası |  | Toplam Saat |  |

.....**/**....**/201 tarihinden** .....**/**....**/201 tarihine kadar bir haftalık çalışma**

|  |  |  |  |
| --- | --- | --- | --- |
| GÜN | Yapılan İşler | Yapılan işe ait bilginin bulunduğu sayfa | Saat |
| Pazartesi |  |  |  |
| Salı |  |  |  |
| Çarşamba |  |  |  |
| Perşembe |  |  |  |
| Cuma |  |  |  |
| Cumartesi |  |  |  |
| Denetleyenin İmzası |  | Toplam Saat |  |

.....**/**....**/201 tarihinden** .....**/**....**/201 tarihine kadar bir haftalık çalışma**

|  |  |  |  |
| --- | --- | --- | --- |
| GÜN | Yapılan İşler | Yapılan işe ait bilginin bulunduğu sayfa | Saat |
| Pazartesi |  |  |  |
| Salı |  |  |  |
| Çarşamba |  |  |  |
| Perşembe |  |  |  |
| Cuma |  |  |  |
| Cumartesi |  |  |  |
| Denetleyenin İmzası |  | Toplam Saat |  |

.....**/**....**/201 tarihinden** .....**/**....**/201 tarihine kadar bir haftalık çalışma**

|  |  |  |  |
| --- | --- | --- | --- |
| GÜN | Yapılan İşler | Yapılan işe ait bilginin bulunduğu sayfa | Saat |
| Pazartesi |  |  |  |
| Salı |  |  |  |
| Çarşamba |  |  |  |
| Perşembe |  |  |  |
| Cuma |  |  |  |
| Cumartesi |  |  |  |
| Denetleyenin İmzası |  | Toplam Saat |  |

|  |
| --- |
| Tarih ve İşyeri Amirinin İmzası ...../...../201 |

**Stajın birinci günüYapılan İşler: Stajyer tanıtımı ve şirket oryantasyonu.**

İlk günümde, şirketin genel yapısını ve işleyişini anlamak amacıyla kapsamlı bir oryantasyon programına katıldım. Şirketin farklı departmanları hakkında bilgi aldım ve her birinin iş süreçlerini öğrendim. Bu süreçte, çalışma ortamını daha yakından tanıma fırsatım oldu ve ekip arkadaşlarımla tanıştım. Ekip arkadaşlarımın samimi ve yardımsever tavırları, adaptasyon sürecimi oldukça kolaylaştırdı. Ayrıca, şirketin misyonu, vizyonu ve değerleri hakkında da bilgi sahibi oldum, bu da bana çalışacağım projelerde nelere dikkat etmem gerektiği konusunda önemli ipuçları verdi. Bu oryantasyon sayesinde, şirketin beklentilerini ve benim rolümü daha iyi anladım, bu da staj sürecine sağlam bir başlangıç yapmamı sağladı.

|  |
| --- |
| Tarih ve İşyeri Amirinin İmzası ...../...../201 |

**Stajın İkinci günü Yapılan İşler: Yapay Zeka ve Makine Öğrenmesi Temelleri.**

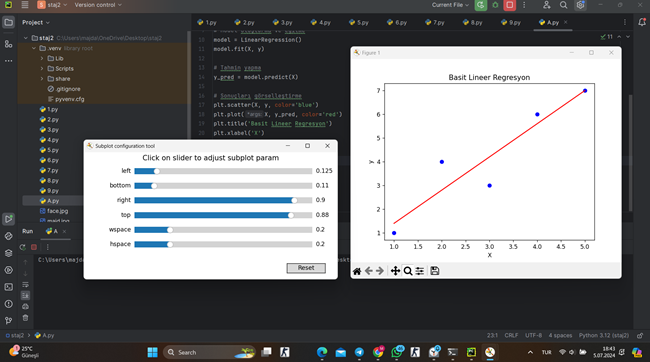
**Detaylar:** İkinci günümde, yapay zeka ve makine öğrenmesi kavramları üzerine bir eğitim aldım. Bu eğitimde, yapay zekanın ne olduğu, nasıl çalıştığı ve hayatımızdaki yerinin yanı sıra makine öğrenmesi teknikleri hakkında da bilgi sahibi oldum. Supervised (denetimli) ve unsupervised (denetimsiz) öğrenme yöntemlerinin temelleri anlatıldı ve bu yöntemlerin farklı kullanım alanları örneklerle açıklandı. Supervised öğrenmede etiketli verilerle model eğitimi yapılırken, unsupervised öğrenmede etiketlenmemiş veriler üzerinden yapılar keşfedilir. Bu teorik bilgilerin ardından, öğrendiklerimizi pekiştirmek amacıyla basit bir lineer regresyon örneği üzerinde çalıştık.

Örnek Python Kodu: Basit Lineer Regresyon



Görsel 1 : Basit Lineer Regresyon kodu

**Sonuçları Görselleştirme:** matplotlib kütüphanesi kullanarak orijinal veri noktalarını ve tahmin edilen çizgiyi görselleştirdik. plt.scatter ile veri noktalarını mavi renkte, plt.plot ile tahmin edilen çizgiyi kırmızı renkte çizdik. Grafik başlıklarını ve eksen etiketlerini de ekleyerek görselleştirme işlemini tamamladık.



Görsel 2 : Basit Lineer Regresyon Çıkışı

|  |
| --- |
| Tarih ve İşyeri Amirinin İmzası ...../...../201 |

**Açıklama:**

1. **Kütüphanelerin İçe Aktarılması:** numpy kütüphanesi sayısal işlemler için, matplotlib.pyplot veri görselleştirme için ve sklearn.linear\_model makine öğrenmesi modeli oluşturmak için kullanılır.
2. **Veri Oluşturma:** X ve y adında iki veri kümesi oluşturduk. X, 1'den 5'e kadar olan sayıları içeren bir dizi, y ise karşılık gelen sonuç değerlerini içerir.
3. **Model Oluşturma ve Eğitme:** LinearRegression sınıfını kullanarak bir model oluşturduk ve fit metodu ile bu modeli verilerimiz üzerinde eğittik.
4. **Tahmin Yapma:** predict metodu ile X verileri üzerinden tahminler yaptık ve bu tahminleri y\_pred değişkenine atadık.

|  |
| --- |
| Tarih ve İşyeri Amirinin İmzası ...../...../201 |

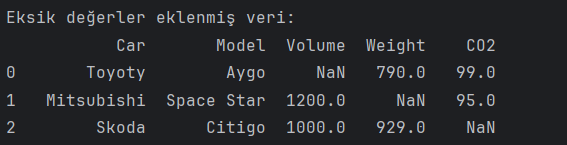
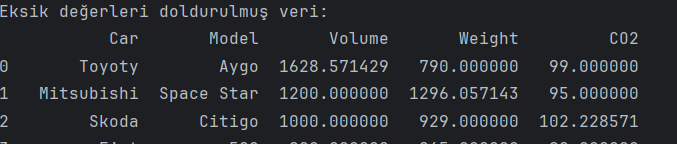
**Stajın üçünçü günü** **Yapılan İşler: Veri Ön İşleme ve Özellik Çıkartma.**

Üçüncü günümde, makine öğrenmesi projelerinde veri hazırlama sürecinin önemi ve yöntemleri hakkında eğitim aldım. Veri temizleme, normalizasyon ve özellik mühendisliği konuları üzerinde duruldu. Eğitim boyunca, verilerin nasıl temizleneceği, eksik değerlerin nasıl doldurulacağı ve verilerin nasıl ölçeklendirileceği gibi temel veri ön işleme teknikleri uygulamalı örneklerle gösterildi. Ayrıca, özellik mühendisliği ile verilerden anlamlı bilgiler çıkarma yöntemleri öğrenildi. Bu bilgiler, makine öğrenmesi modellerinin performansını artırmada kritik bir rol oynamaktadır.



Görsel 3 : Veri Ön İşleme Python Kodu

**Sonuçları Görüntüleme:** print(data) komutu ile eksik değerleri doldurulmuş veri setini görüntülüyoruz.

Görsel 4 : Veri Ön İşleme Çıkış Python Kodu

**Veri Yükleme:** pd.read\_csv('data.csv') ile veri setini yüklüyoruz. Bu komut, CSV dosyasını bir DataFrame'e dönüştürür.

**Eksik Değerler Oluşturma:** data.loc[0, 'Volume'] = np.nan, data.loc[1, 'Weight'] = np.nan, ve data.loc[2, 'CO2'] = np.nan komutları ile veri setine kasıtlı olarak eksik değerler ekliyoruz. Bu, veri ön işleme tekniklerini uygulamak için örnek veri hazırlamamıza yardımcı olur.

**Veriyi Görüntüleme:** print(data) komutu ile eksik değerler eklenmiş veri setini görüntülüyoruz.

**Sayısal Sütunları Seçme:** numeric\_columns = data.select\_dtypes(include=[float, int]).columns komutu ile veri setindeki sayısal sütunları seçiyoruz. Bu sütunlar üzerinde eksik değer doldurma işlemi yapacağız.

**Eksik Değerleri Doldurma:** data[numeric\_columns] = data[numeric\_columns].fillna(data[numeric\_columns].mean()) komutu ile seçilen sayısal sütunlardaki eksik değerleri, ilgili sütunun ortalama değeri ile dolduruyoruz.

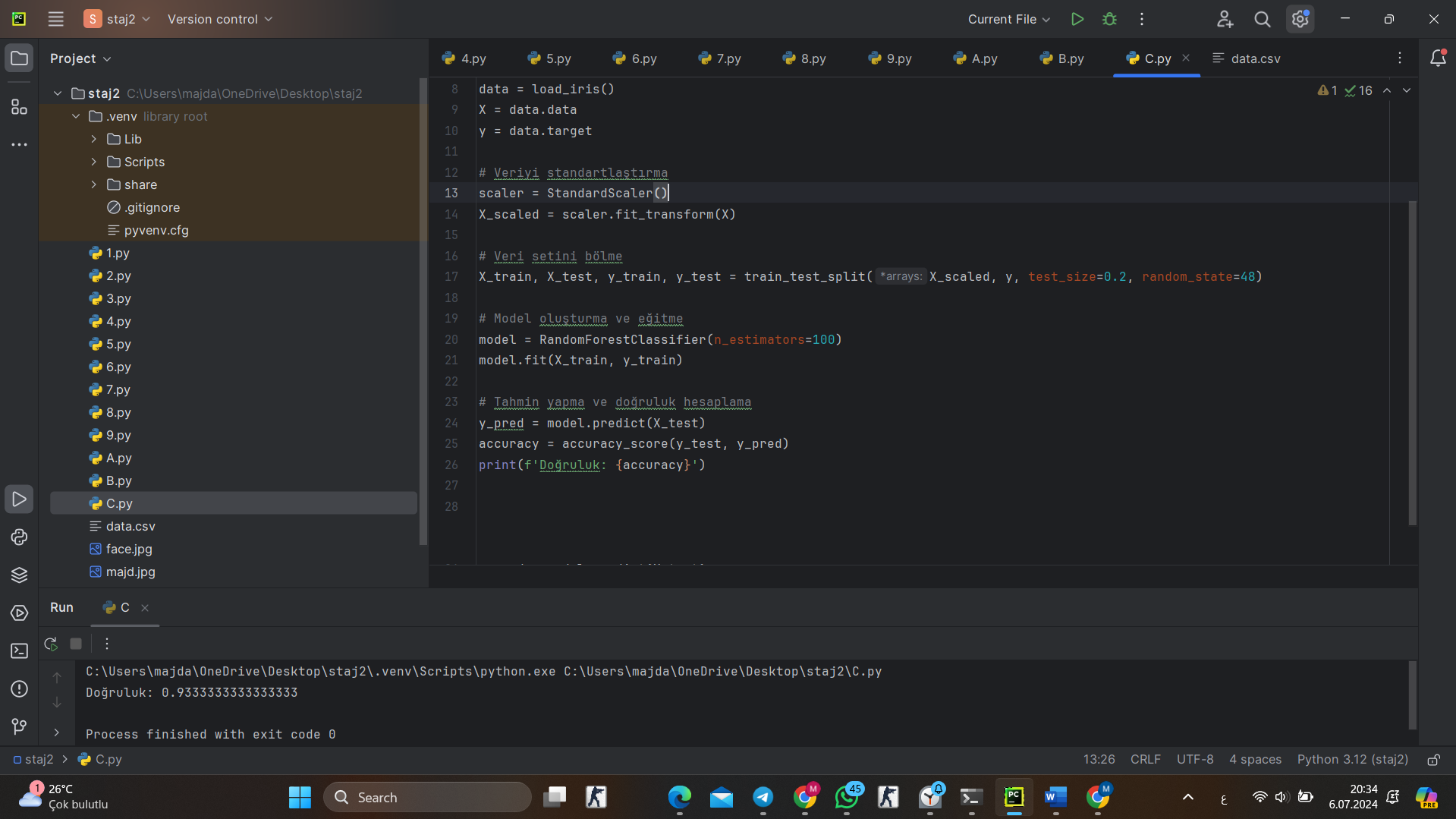
|  |
| --- |
| Tarih ve İşyeri Amirinin İmzası ...../...../201 |

|  |
| --- |
| Tarih ve İşyeri Amirinin İmzası ...../...../201 |

**Stajın Dördüncü günü Yapılan İşler: Denetimli Öğrenme Algoritmaları.**

**Detaylar:** Dördüncü günümde, denetimli öğrenme algoritmaları üzerine yoğunlaştık. Karar ağaçları, random forest, k-en yakın komşu (k-NN) ve lojistik regresyon gibi yaygın olarak kullanılan denetimli öğrenme algoritmalarını inceledik. Bu algoritmaların nasıl çalıştığı, hangi durumlarda kullanılabileceği ve avantajları hakkında detaylı bilgi edindik. Her bir algoritmanın temel prensipleri, matematiksel altyapısı ve uygulanabilirlikleri üzerinde duruldu. Teorik bilgilerin ardından, bu algoritmaları Python'da nasıl kullanabileceğimiz konusunda örnekler yapıldı.

Örnek Python Kodu: Random Forest



Görsel 5: Random Forest

**Açıklama:**

1. **Kütüphanelerin İçe Aktarılması:** sklearn.ensemble kütüphanesinden RandomForestClassifier, sklearn.model\_selection kütüphanesinden train\_test\_split, ve sklearn.metrics kütüphanesinden accuracy\_score sınıfları içe aktarılır.
2. **Veri Setini Bölme:** train\_test\_split(X\_scaled, y, test\_size=0.2, random\_state=42) komutu ile ölçeklendirilmiş veriyi %80 eğitim, %20 test olarak ayırıyoruz. random\_state parametresi, sonuçların tekrar üretilebilir olmasını sağlar.
3. **Model Oluşturma ve Eğitme:** RandomForestClassifier(n\_estimators=100) sınıfını kullanarak bir random forest modeli oluşturuyoruz. n\_estimators parametresi, ormanda kaç tane karar ağacı olduğunu belirtir. fit metodu ile bu modeli eğitim verileri üzerinde eğitiyoruz.
4. **Tahmin Yapma ve Doğruluk Hesaplama:** predict metodu ile test verileri üzerinden tahminler yapıyoruz ve bu tahminleri y\_pred değişkenine atıyoruz. accuracy\_score(y\_test, y\_pred) komutu ile modelin doğruluğunu hesaplıyoruz ve sonucu ekrana yazdırıyoruz.

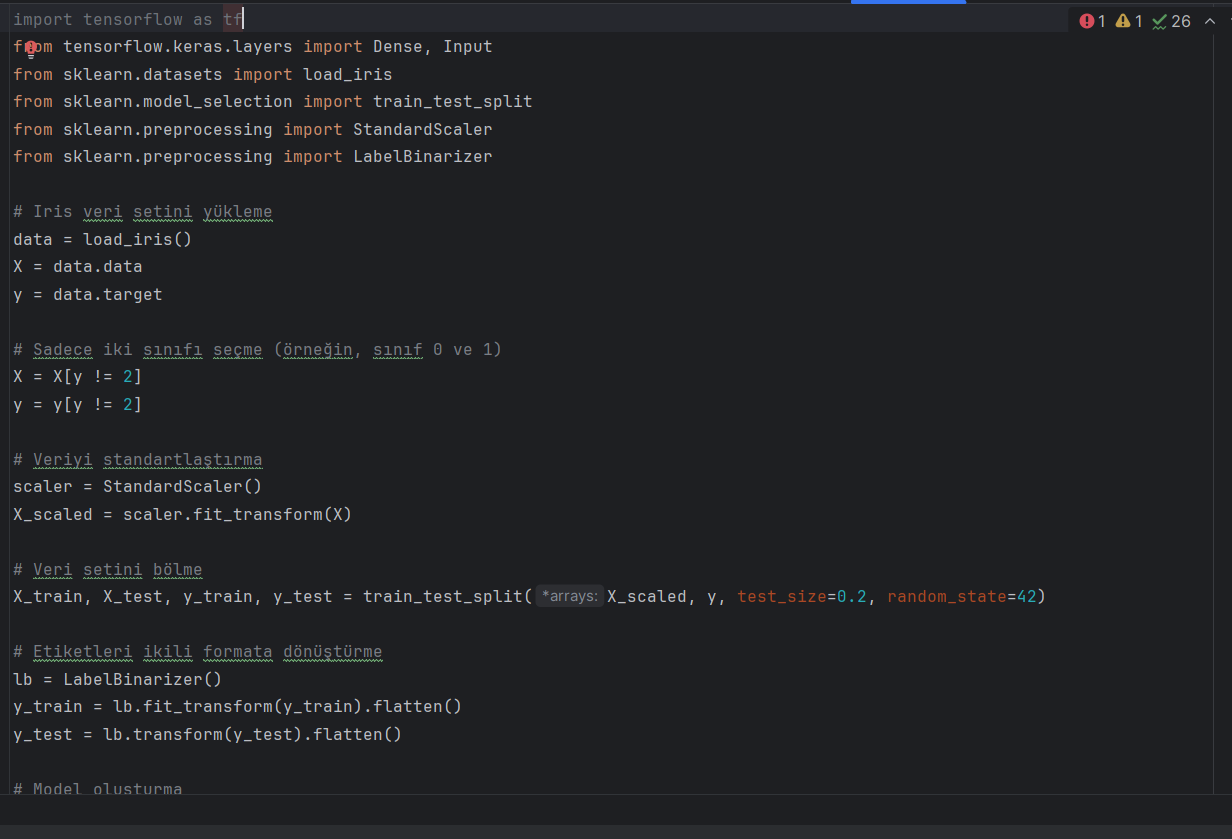
|  |
| --- |
| Tarih ve İşyeri Amirinin İmzası /...../201 |

**Stajın Beşinci günüYapılan İşler: Derin Öğrenme Temelleri.**

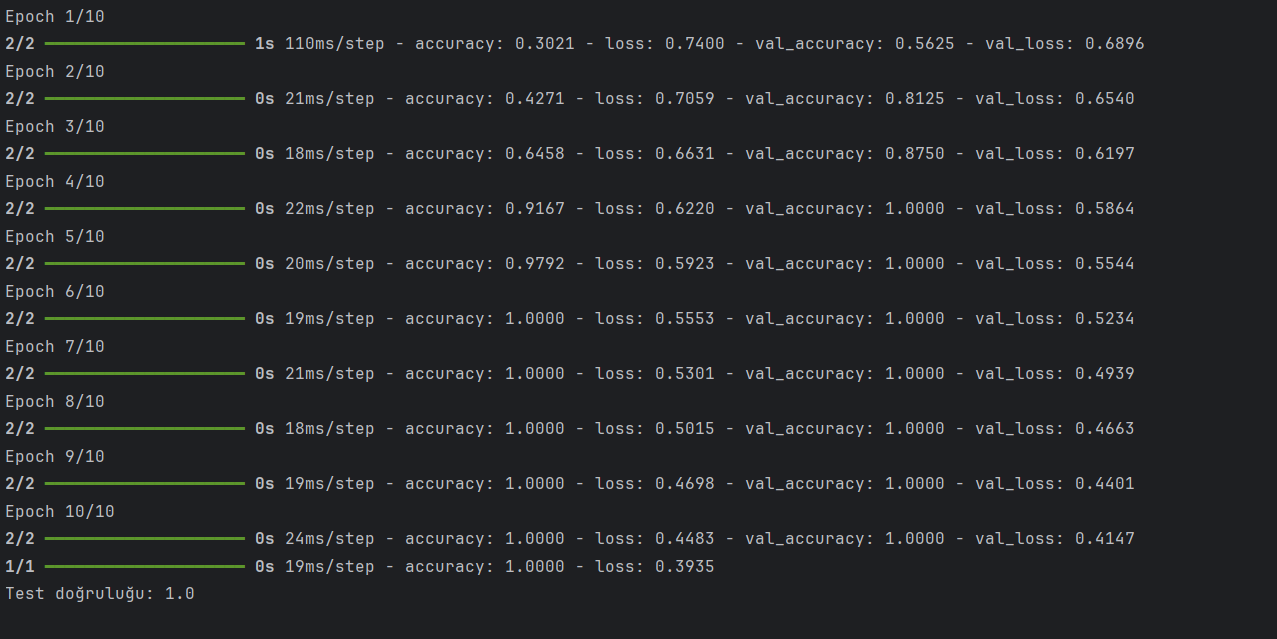
Beşinci günümde, derin öğrenme kavramlarına ve yapay sinir ağlarına giriş yaptık. Derin öğrenmenin temelleri, yapay sinir ağlarının yapısı ve işleyişi hakkında bilgi edindim. Bu kapsamda, popüler derin öğrenme kütüphaneleri olan Keras ve TensorFlow tanıtıldı. Eğitim sırasında, bu kütüphanelerin nasıl kullanılacağı ve basit bir sinir ağı modelinin nasıl oluşturulacağı üzerinde duruldu.

|  |
| --- |
| Tarih ve İşyeri Amirinin İmzası ...../...../201 |

Örnek Python Kodu: Basit Sinir Ağı



Görsel 6 : İkili Sınıflandırma için Yapay Sinir Ağı Modeli



Görsel 6 : İkili Sınıflandırma için Yapay Sinir Ağı Modeli Çıkışı

**Açıklama:**

1. **Kütüphanelerin İçe Aktarılması:** tensorflow kütüphanesi derin öğrenme modellerini oluşturmak için, sklearn.datasets veri setini yüklemek için, sklearn.model\_selection veri setini eğitim ve test olarak ayırmak için, sklearn.preprocessing veriyi standartlaştırmak ve etiketleri dönüştürmek için kullanılır.
2. **Veri Setini Yükleme:** load\_iris() fonksiyonu ile iris veri setini yüklüyoruz. Bu veri seti, çiçeklerin özelliklerini ve türlerini içerir.
3. **Sadece İki Sınıfı Seçme:** Yalnızca sınıf 0 ve 1'i seçmek için X = X[y != 2] ve y = y[y != 2] komutlarını kullanıyoruz. Bu, iki sınıflı bir sınıflandırma problemi oluşturur.
4. **Veriyi Standartlaştırma:** StandardScaler() sınıfı ile veriyi standartlaştırıyoruz. Bu, özelliklerin ortalamasını 0 ve standart sapmasını 1 yapar.
5. **Veri Setini Bölme:** train\_test\_split(X\_scaled, y, test\_size=0.2, random\_state=42) komutu ile veriyi %80 eğitim, %20 test olarak ayırıyoruz. random\_state parametresi, sonuçların tekrar üretilebilir olmasını sağlar.
6. **Etiketleri İkili Formata Dönüştürme:** LabelBinarizer() sınıfı ile etiketleri ikili formata dönüştürüyoruz. Bu, sınıflandırma modelinin daha kolay eğitilmesini sağlar.
7. **Model Oluşturma:** Sequential() sınıfı ile bir sinir ağı modeli oluşturuyoruz. Dense katmanlarını kullanarak gizli katmanları ve çıkış katmanını ekliyoruz. Girdi katmanında, özellik sayısına eşit bir şekil belirliyoruz. Çıkış katmanında sigmoid aktivasyon fonksiyonu kullanarak ikili sınıflandırma yapıyoruz.
8. **Modeli Derleme:** model.compile() metodu ile modeli derliyoruz. adam optimizasyon algoritmasını ve binary\_crossentropy kayıp fonksiyonunu kullanıyoruz. Ayrıca, modeli değerlendirmek için accuracy metriğini belirtiyoruz.
9. **Modeli Eğitme:** model.fit() metodu ile modeli eğitiyoruz. Eğitim sırasında validation\_split=0.2 parametresi ile verinin %20'sini doğrulama seti olarak kullanıyoruz.
10. **Modeli Değerlendirme:** model.evaluate() metodu ile modeli test verileri üzerinde değerlendiriyoruz ve doğruluk değerini ekrana yazdırıyoruz.

|  |
| --- |
| Tarih ve İşyeri Amirinin İmzası ...../...../201 |

**Stajın Altıncı günü Yapılan İşler: Doğal Dil İşleme (NLP) Temelleri.**

Altıncı günümde, Doğal Dil İşleme (NLP) kavramlarını ve uygulama alanlarını öğrendik. Metin verilerini nasıl işleyebileceğimiz, metin temizleme ve ön işleme teknikleri üzerinde duruldu. Bu tekniklerin NLP modellerinde nasıl kullanıldığını görmek için çeşitli örnekler inceledik. Ayrıca, temel NLP modelleri ve bunların nasıl çalıştığı hakkında bilgi edindik.

**Örnek Python Kodu: Metin Ön İşleme**



Görsel 7 : **Metin Ön İşleme**

**Açıklama:**

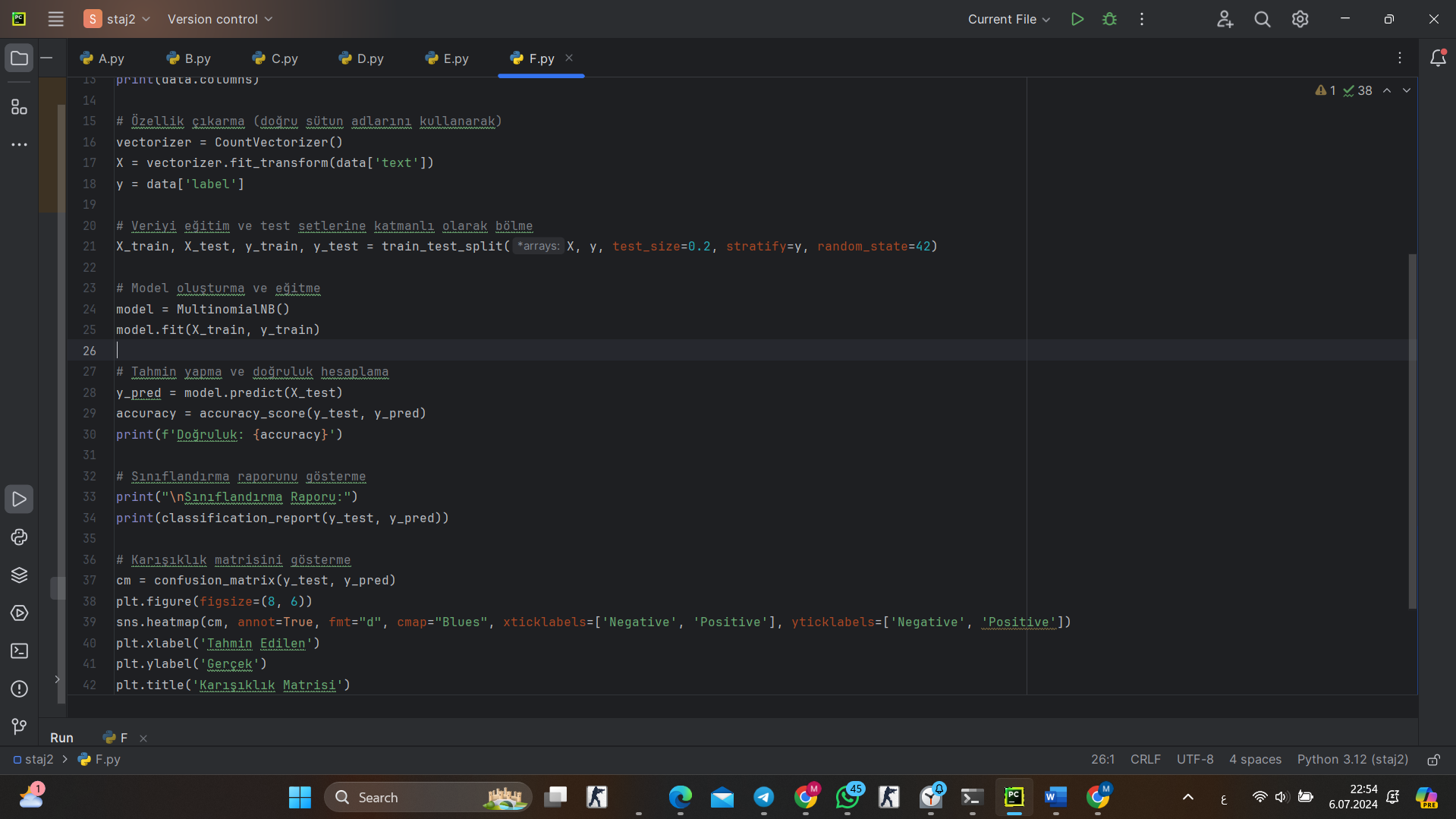
1. **Kütüphanelerin İçe Aktarılması:** nltk (Natural Language Toolkit) kütüphanesi, doğal dil işleme için gerekli fonksiyonları sağlar
2. **Gerekli NLTK Paketlerini İndirme:** nltk.download('punkt'), nltk.download('stopwords') ve nltk.download('wordnet') komutları, gerekli NLTK veri setlerini indirir.
3. **Metin Örneği:** İşlenecek metin örneği "OpenAI is creating amazing artificial intelligence.
4. **Tokenizasyon:** word\_tokenize(text) komutu ile metin kelimelere bölünür.
5. **Küçük Harfe Çevirme ve Durdurma Kelimeleri Kaldırma:** tokens = [word.lower() for word in tokens if word.isalnum()] komutu ile kelimeler küçük harfe dönüştürülür ve sadece alfanümerik karakterler tutulur.
6. **Lemmatizasyon:** WordNetLemmatizer() sınıfı kullanılarak kelimeler köklerine indirilir. tokens = [lemmatizer.lemmatize(word) for word in tokens] komutu ile her kelime köküne indirilir.
7. **İşlenmiş Kelimeleri Yazdırma:** print(tokens) komutu ile işlenmiş kelimeler ekrana yazdırılır.

|  |
| --- |
| Tarih ve İşyeri Amirinin İmzası ...../...../201 |

**Stajın Yedenci günü Yapılan İşler: Sentiment Analizi Projesi.**

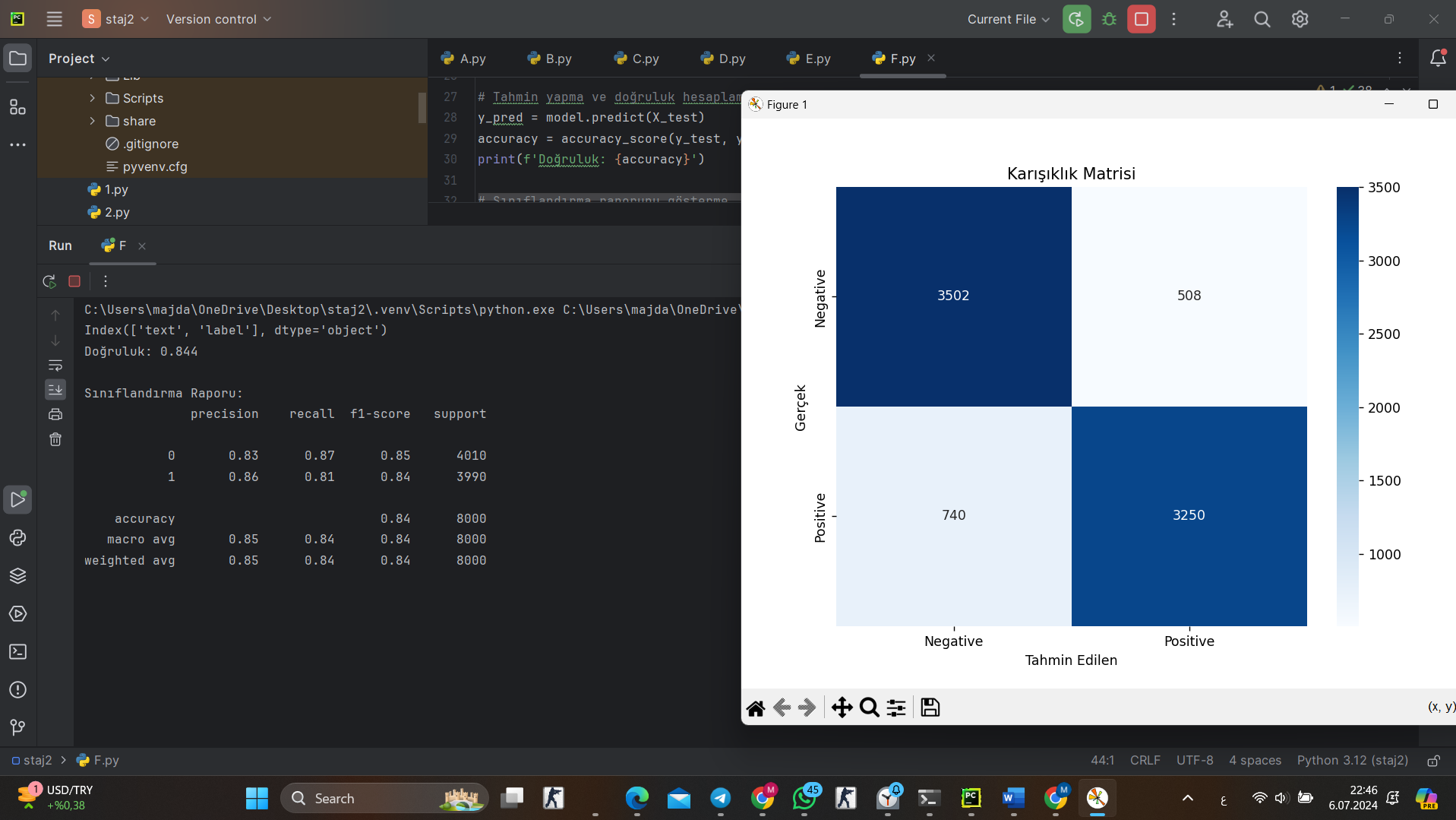
Yedinci günümde, sentiment analizi (duygu analizi) üzerine çalıştık. Bu proje kapsamında, film incelemeleri veri seti kullanılarak metin verilerinin duygu durumunu belirlemeyi hedefledik. Sentiment analizi, kullanıcı yorumlarını pozitif ya da negatif olarak sınıflandırmayı amaçlayan bir tekniktir. Bu süreçte, veri işleme, özellik çıkarma, model oluşturma ve değerlendirme adımları üzerinde durduk.

Örnek Python Kodu: Sentiment Analizi



Görsel 8 : Sentiment Analizi

Bu koddan beklenen çıktıdır :



Görsel 9 : Sentiment Analizi Çıkışı

|  |
| --- |
| Tarih ve İşyeri Amirinin İmzası ...../...../201 |

**Açıklama:**

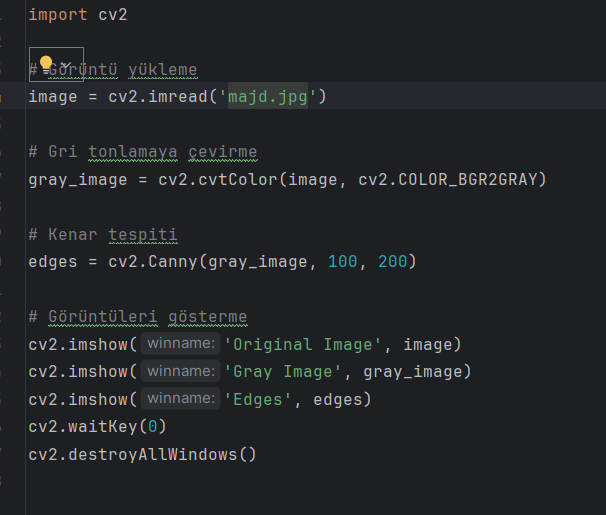
1. **Kütüphanelerin İçe Aktarılması:** pandas veri yükleme ve işleme için, CountVectorizer metin verilerini sayısal verilere dönüştürmek için, train\_test\_split veriyi eğitim ve test setlerine ayırmak için, MultinomialNB Naive Bayes sınıflandırıcısını oluşturmak için, accuracy\_score, classification\_report ve confusion\_matrix modeli değerlendirmek için, matplotlib ve seaborn ise görselleştirme için kullanılır.
2. **Veriyi Yükleme:** pd.read\_csv('movie\_reviews.csv') komutu ile film incelemeleri veri seti yüklenir.
3. **Sütun Adlarını Kontrol Etme:** Veri setindeki sütun adlarını kontrol etmek için print(data.columns) komutu kullanılır.
4. **Özellik Çıkarma:** CountVectorizer sınıfı ile metin verileri sayısal verilere dönüştürülür. X = vectorizer.fit\_transform(data['text']) ile metin verilerinden özellik çıkarılır ve y = data['label'] ile etiketler belirlenir.
5. **Veriyi Eğitim ve Test Setlerine Bölme:** train\_test\_split(X, y, test\_size=0.2, stratify=y, random\_state=42) komutu ile veriler %80 eğitim, %20 test olarak katmanlı bir şekilde bölünür. stratify=y parametresi ile sınıf dağılımının korunması sağlanır.
6. **Model Oluşturma ve Eğitme:** MultinomialNB sınıfı kullanılarak bir Naive Bayes modeli oluşturulur ve model.fit(X\_train, y\_train) ile model eğitim verileri üzerinde eğitilir.
7. **Tahmin Yapma ve Doğruluk Hesaplama:** model.predict(X\_test) ile test verileri üzerinden tahminler yapılır ve accuracy\_score(y\_test, y\_pred) ile modelin doğruluğu hesaplanır. Doğruluk ekrana yazdırılır.
8. **Sınıflandırma Raporunu Gösterme:** classification\_report(y\_test, y\_pred) komutu ile sınıflandırma raporu ekrana yazdırılır. Bu rapor, precision, recall ve F1-score gibi metri
9. **Karışıklık Matrisini Gösterme:** confusion\_matrix(y\_test, y\_pred) komutu ile karışıklık matrisi oluşturulur. seaborn kütüphanesi kullanılarak bu matris görselleştirilir. Matris, gerçek ve tahmin edilen sınıflar arasındaki doğru ve yanlış tahminleri gösterir.

|  |
| --- |
| Tarih ve İşyeri Amirinin İmzası ...../...../201 |

**Stajın Dokuzuncu günü Yapılan İşler: Görüntü İşleme Temelleri.**

Sekizinci günümde, görüntü işleme kavramlarını ve temel teknikleri öğrendik. OpenCV kütüphanesi kullanarak görüntü işleme uygulamaları gerçekleştirdik. Bu süreçte, görüntülerin nasıl yükleneceği, gri tonlamaya nasıl dönüştürüleceği ve kenar tespiti gibi işlemlerin nasıl yapılacağı hakkında bilgi edindik.

**Örnek Python Kodu: Görüntü İşleme**



Görsel 10 : **Görüntü İşleme**



Görsel 11: **Görüntü İşleme Çıkışı**

|  |
| --- |
| Tarih ve İşyeri Amirinin İmzası ...../...../201 |

**Açıklama:**

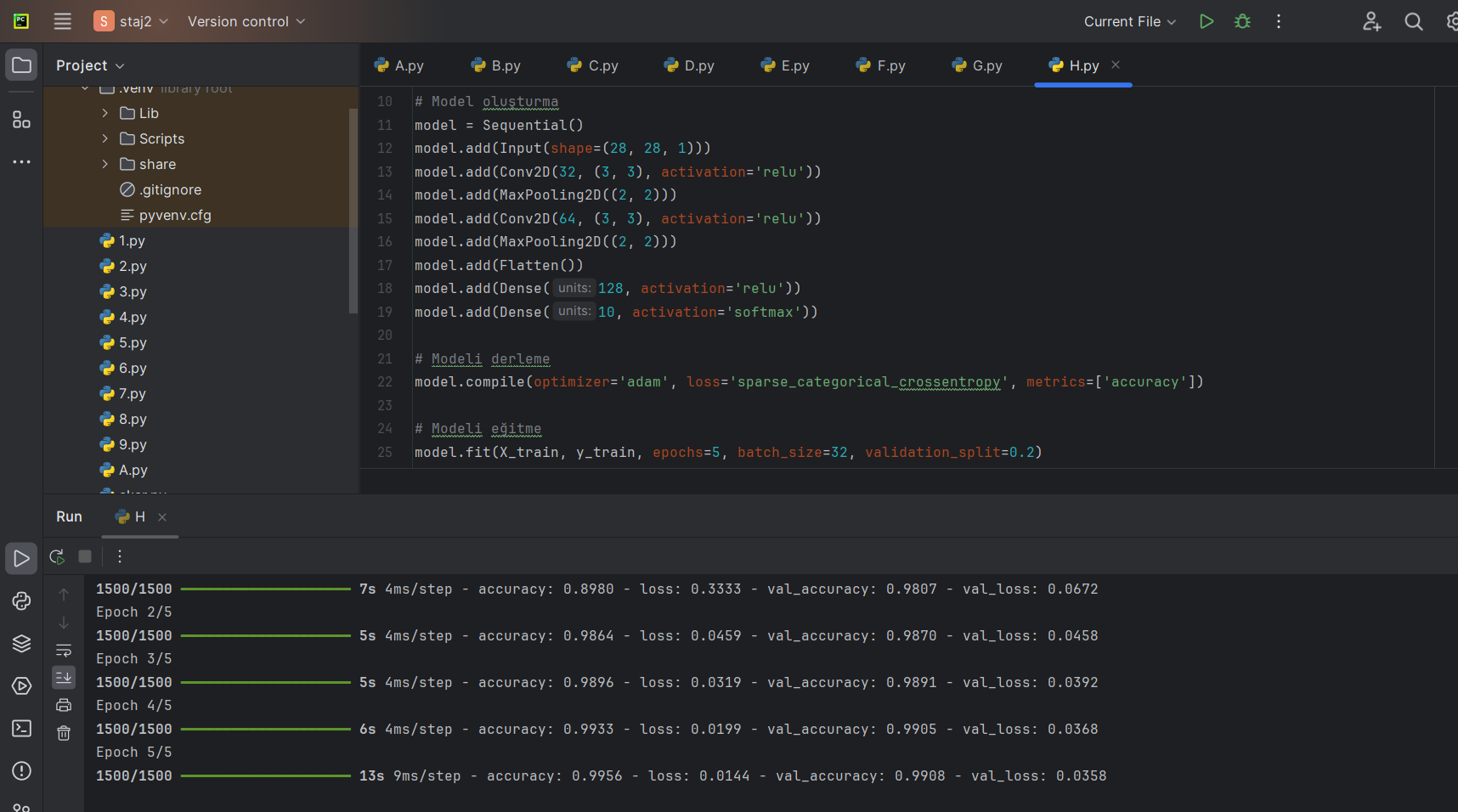
1. **Kütüphanelerin İçe Aktarılması:** cv2 (OpenCV) kütüphanesi, görüntü işleme işlemleri için kullanılır.
2. **Görüntü Yükleme:** cv2.imread('image.jpg') komutu ile görüntü dosyası yüklenir. Bu komut, belirtilen dosya yolundaki görüntüyü okur ve bir numpy array olarak döner.
3. **Gri Tonlamaya Çevirme:** cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY) komutu, renkli bir görüntüyü gri tonlamaya dönüştürür. cv2.COLOR\_BGR2GRAY parametresi, BGR formatındaki bir görüntüyü gri tonlamaya dönüştürmek için kullanılır.
4. **Kenar Tespiti:** cv2.Canny(gray\_image, 100, 200) komutu ile gri tonlamalı görüntüde kenar tespiti yapılır. 100 ve 200 parametreleri, kenar tespiti için alt ve üst eşik değerleridir.
5. **Görüntüleri Gösterme:** cv2.imshow komutu ile orijinal görüntü, gri tonlamalı görüntü ve kenar tespiti yapılan görüntü ayrı pencerelerde gösterilir. cv2.waitKey(0) komutu, bir tuşa basılana kadar pencerelerin açık kalmasını sağlar. cv2.destroyAllWindows() komutu ise tüm pencereleri kapatır.

|  |
| --- |
| Tarih ve İşyeri Amirinin İmzası ...../...../201 |

**Stajın Dokuzuncu günü Yapılan İşler: Convolutional Neural Networks (CNN) ile Görüntü Sınıflandırma.**

Dokuzuncu günümde, Convolutional Neural Networks (CNN) kavramları tanıtıldı ve görüntü sınıflandırma üzerine uygulamalı çalışmalar gerçekleştirildi. MNIST veri seti kullanılarak basit bir görüntü sınıflandırma modeli oluşturuldu. CNN'ler, özellikle görüntü işleme ve sınıflandırma konularında oldukça etkili modellerdir ve derin öğrenme alanında yaygın olarak kullanılır.

**Örnek Python Kodu: CNN ile Görüntü Sınıflandırma**



Görsel 12 : **CNN ile Görüntü Sınıflandırma**

|  |
| --- |
| Tarih ve İşyeri Amirinin İmzası ...../...../201 |

**Açıklama:**

1. **Kütüphanelerin İçe Aktarılması:** tensorflow.keras kütüphanesi, derin öğrenme modelleri oluşturmak için kullanılır. MNIST veri seti, mnist.load\_data() fonksiyonu ile yüklenir. Sequential, Conv2D, MaxPooling2D, Flatten ve Dense katmanları, CNN modelini oluşturmak için kullanılır.
2. **Veri Setini Yükleme:** mnist.load\_data() komutu ile MNIST veri seti yüklenir. Bu veri seti, el yazısı rakamların (0-9) siyah beyaz görüntülerini içerir. Veri seti eğitim ve test verileri olarak ikiye ayrılır.
3. **Veri Setini Yeniden Şekillendirme ve Ölçeklendirme:** X\_train ve X\_test verileri, CNN modeline uygun olacak şekilde yeniden şekillendirilir (28x28 boyutunda ve 1 kanal, yani gri tonlamalı). Veriler 255'e bölünerek [0, 1] aralığına ölçeklendirilir.
4. **Model Oluşturma:** Sequential modeli oluşturulur ve CNN katmanları eklenir:
   * Conv2D: 32 filtre, 3x3 kernel boyutu ve ReLU aktivasyon fonksiyonu ile bir evrişimsel katman eklenir.
   * MaxPooling2D: 2x2 boyutunda bir maksimum havuzlama katmanı eklenir.
   * Conv2D: 64 filtre, 3x3 kernel boyutu ve ReLU aktivasyon fonksiyonu ile ikinci bir evrişimsel katman eklenir.
   * MaxPooling2D: 2x2 boyutunda ikinci bir maksimum havuzlama katmanı eklenir.
   * Flatten: Çok boyutlu çıktı, düz bir vektöre dönüştürülür.
   * Dense: 128 nöron ve ReLU aktivasyon fonksiyonu ile tam bağlantılı bir katman eklenir.
   * Dense: 10 nöron ve softmax aktivasyon fonksiyonu ile çıkış katmanı eklenir.
5. **Modeli Derleme:** model.compile() komutu ile model derlenir. Adam optimizer, sparse categorical crossentropy loss fonksiyonu ve doğruluk metriği kullanılır.
6. **Modeli Eğitme:** model.fit() komutu ile model, eğitim verileri üzerinde eğitilir. Eğitim sırasında, doğrulama verileri kullanılarak modelin performansı izlenir. Model 5 epoch boyunca ve 32 batch boyutu ile eğitilir.

Bu kod parçası, Convolutional Neural Networks (CNN) kullanarak basit bir görüntü sınıflandırma modelinin nasıl oluşturulacağını ve MNIST veri seti üzerinde nasıl eğitileceğini göstermektedir.

|  |
| --- |
| Tarih ve İşyeri Amirinin İmzası ...../...../201 |

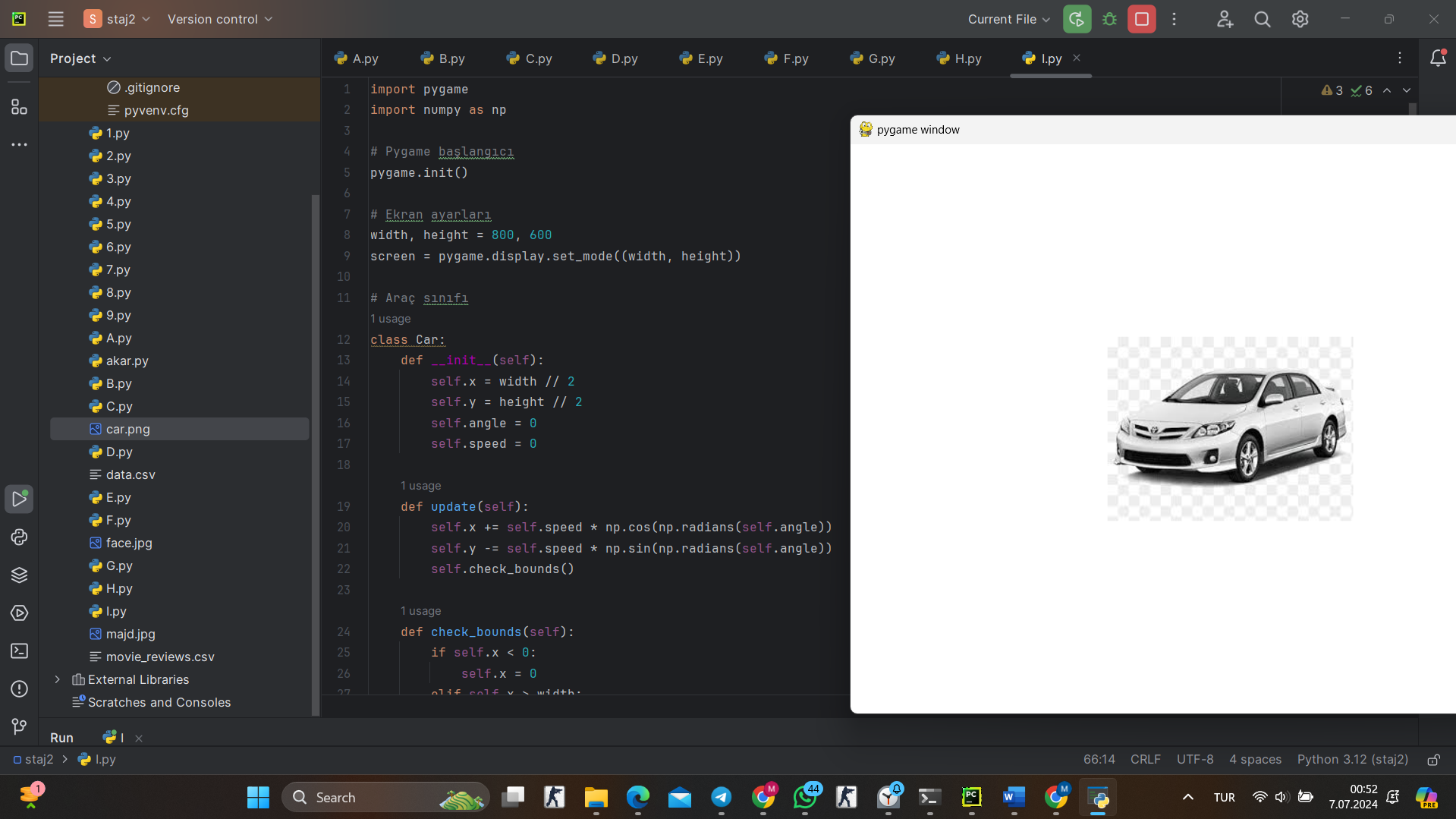
**Stajın Onuncu günü Yapılan İşler:** Büyük Proje Başlangıcı: Otonom Araç Simülasyonu.

Onuncu günümde, otonom araçlar için temel gereksinimler ve kullanılan teknolojiler hakkında bilgi verildi. Bu bilgiler ışığında, simülasyon ortamı hazırlandı ve sensör verilerinin nasıl işleneceği üzerine çalışıldı. Projenin ana hedefi, otonom bir aracın çevresini algılayarak, engellerden kaçınarak ve belirli bir rotayı takip ederek hareket etmesini sağlamak. Bu süreçte, sensör verileri işlenecek, yol planlaması ve kontrol algoritmaları geliştirilecektir.

**Proje Tanımı: Otonom Araç Simülasyonu**

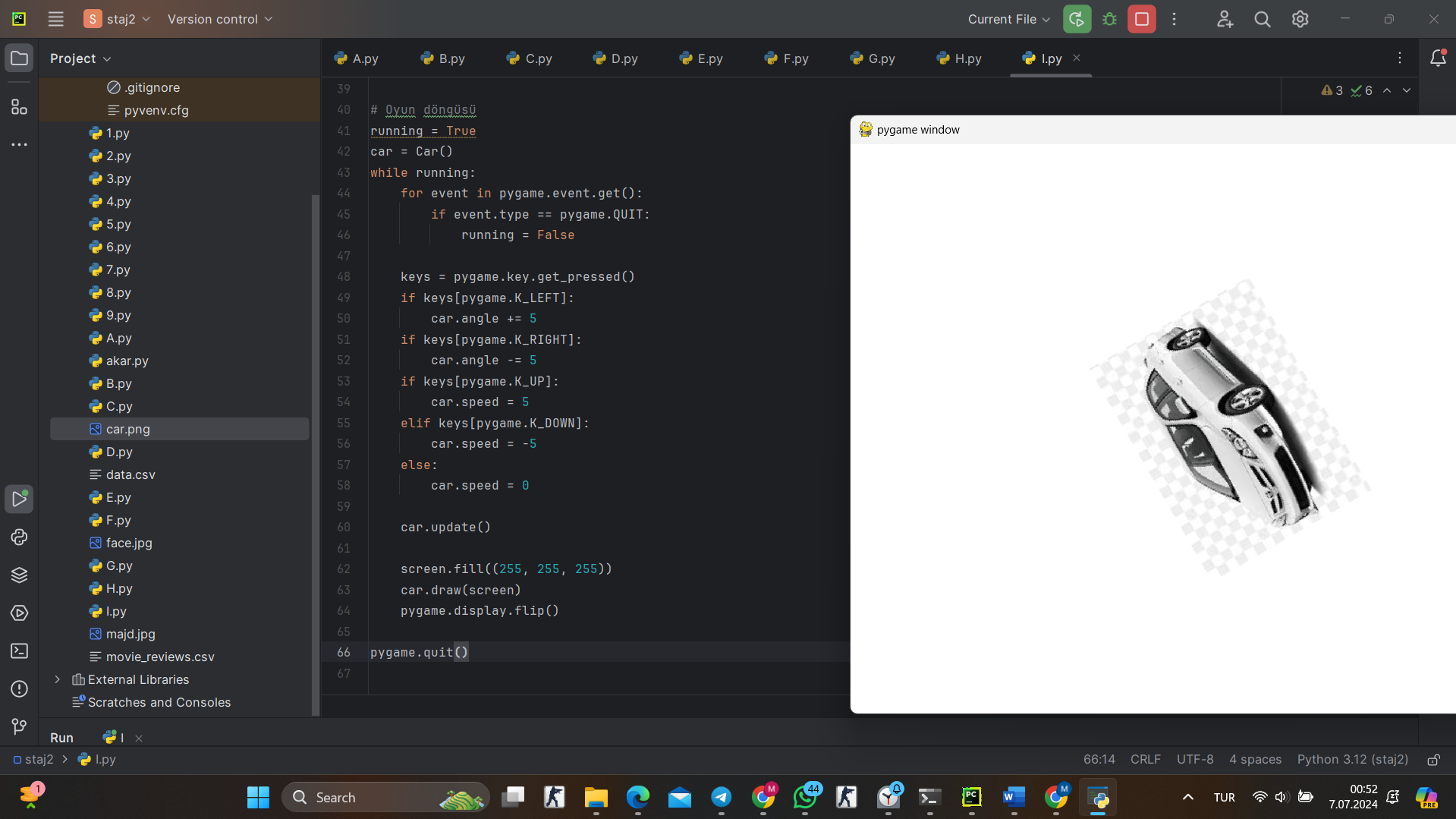
Bu projede, otonom bir aracın simülasyonu yapılacak. Aracın çevresini algılaması, engellerden kaçınması ve belirli bir rotayı takip etmesi sağlanacak. Proje boyunca, sensör verilerinin işlenmesi, yol planlaması ve kontrol algoritmaları geliştirilecektir.

**Örnek Python Kodu: Basit** **Otonom Araç Simülasyonu**



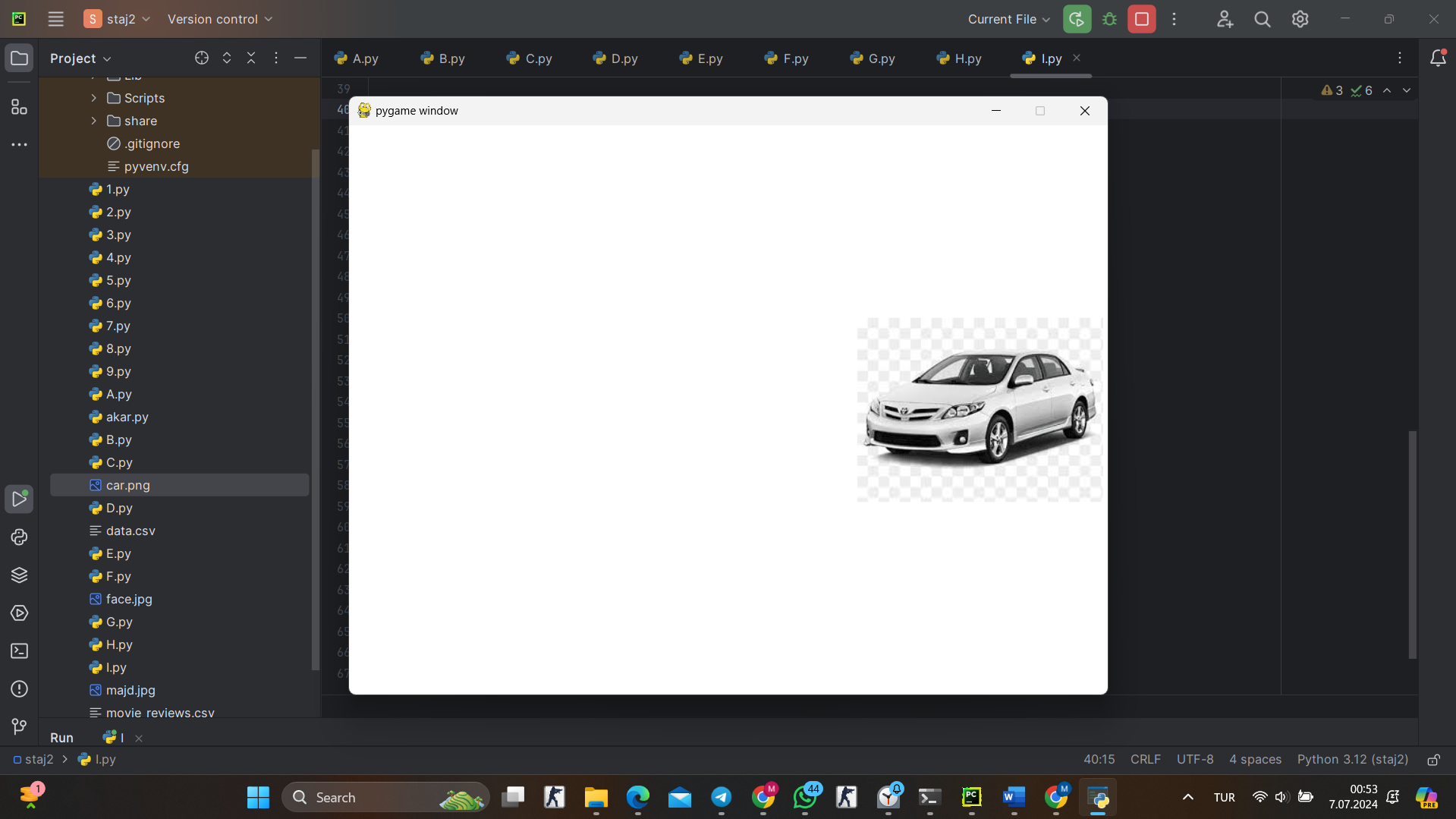
Görsel 12: Otonom Araç Simülasyonu

Kullanıcı sol veya sağ ok tuşuna bastığında, aracın açısı değişir.

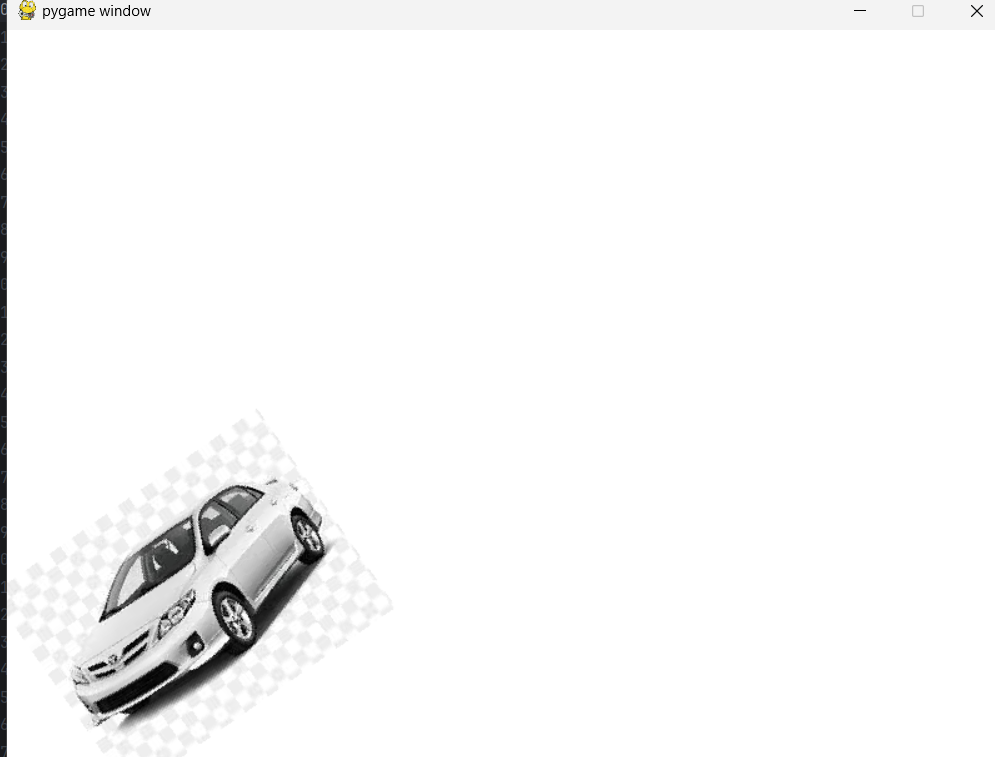


|  |
| --- |
| Tarih ve İşyeri Amirinin İmzası ...../...../201 |

Kullanıcı yukarı ok tuşuna bastığında, araç 5 hızında ileri hareket eder.



Kullanıcı aşağı ok tuşuna bastığında, araç -5 hızında geri hareket eder.

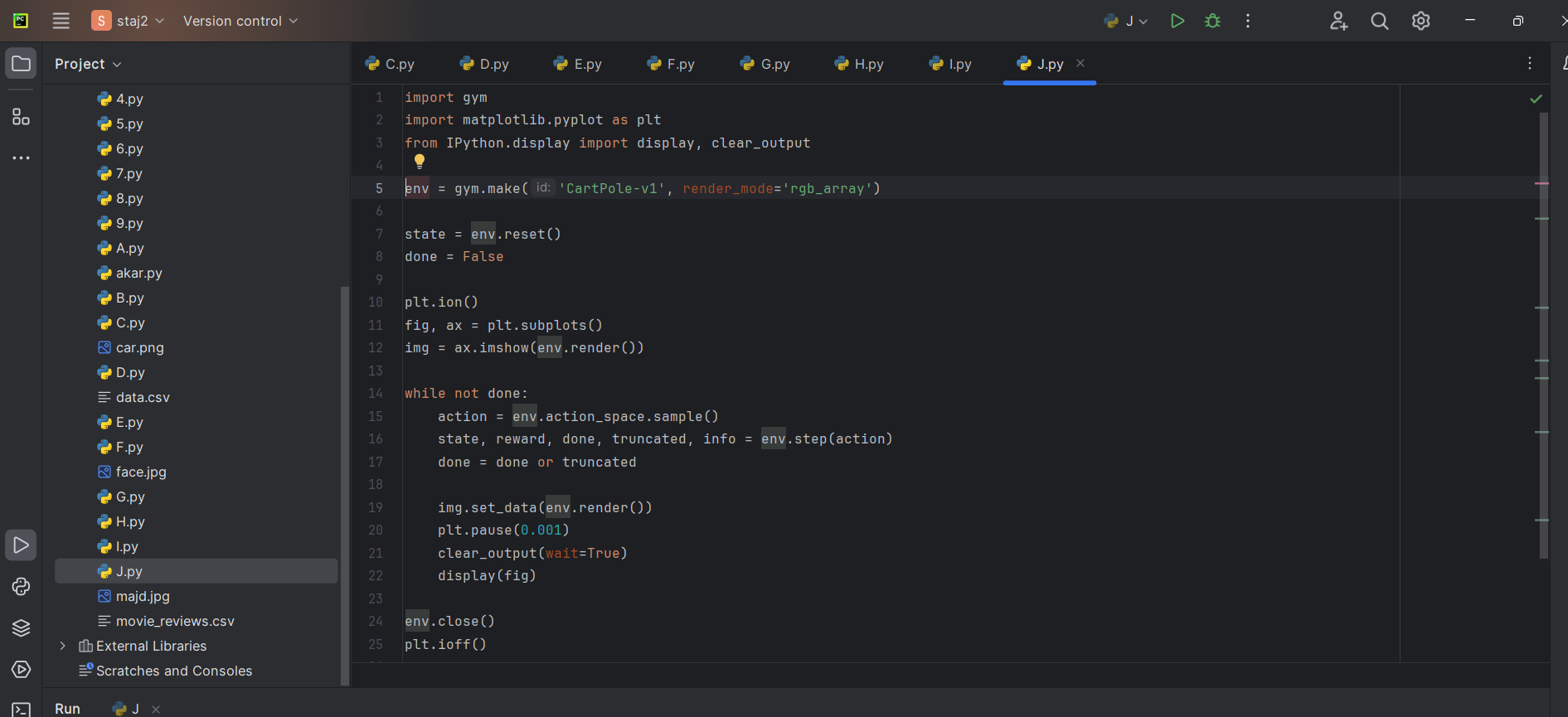


1. **Kütüphanelerin İçe Aktarılması:** pygame kütüphanesi, simülasyon ortamını oluşturmak ve araç hareketlerini görselleştirmek için kullanılır.
2. **Pygame Başlangıcı:** pygame.init() komutu ile pygame başlatılır. pygame.display.set\_mode((width, height)) komutu ile ekran boyutları ayarlanır.
3. **Araç Sınıfı:** Car sınıfı, aracın özelliklerini ve hareketlerini tanımlamak için kullanılır. Araç, başlangıç pozisyonu ve açısı ile oluşturulur. update() fonksiyonu, aracın hız ve açıya göre yeni pozisyonunu hesaplar ve sınırları kontrol eder. check\_bounds() fonksiyonu, aracın ekran sınırları içinde kalmasını sağlar. draw() fonksiyonu, aracı ekran üzerinde çizer.
4. **Oyun Döngüsü:** while running döngüsü, oyun devam ettiği sürece çalışır. pygame.event.get() komutu ile olaylar yakalanır ve pygame.QUIT olayı ile oyun sonlandırılır. pygame.key.get\_pressed() komutu ile klavye tuşları kontrol edilir ve araca yön ve hız kazandırılır. car.draw(screen) komutu ile araç ekran üzerine çizilir. pygame.display.flip() komutu ile ekran güncellenir.

Bu kod parçası, otonom bir aracın simülasyon ortamında temel hareketlerini gerçekleştirmesi ve görselleştirilmesi için başlangıç noktası oluşturur. Bu projeyi daha ileriye götürmek için sensör verilerinin işlenmesi, engel tespiti ve yol planlaması gibi ileri düzey konulara geçiş yapılacaktır.

|  |
| --- |
| Tarih ve İşyeri Amirinin İmzası ...../...../201 |

**Stajın 11 günü Yapılan İşler:** Reinforcement Learning (Pekiştirmeli Öğrenme) Temelleri.  
**Detaylar:** Pekiştirmeli öğrenme kavramları ve temel algoritmalar tanıtıldı. OpenAI Gym kullanılarak basit bir pekiştirmeli öğrenme örneği yapıldı.

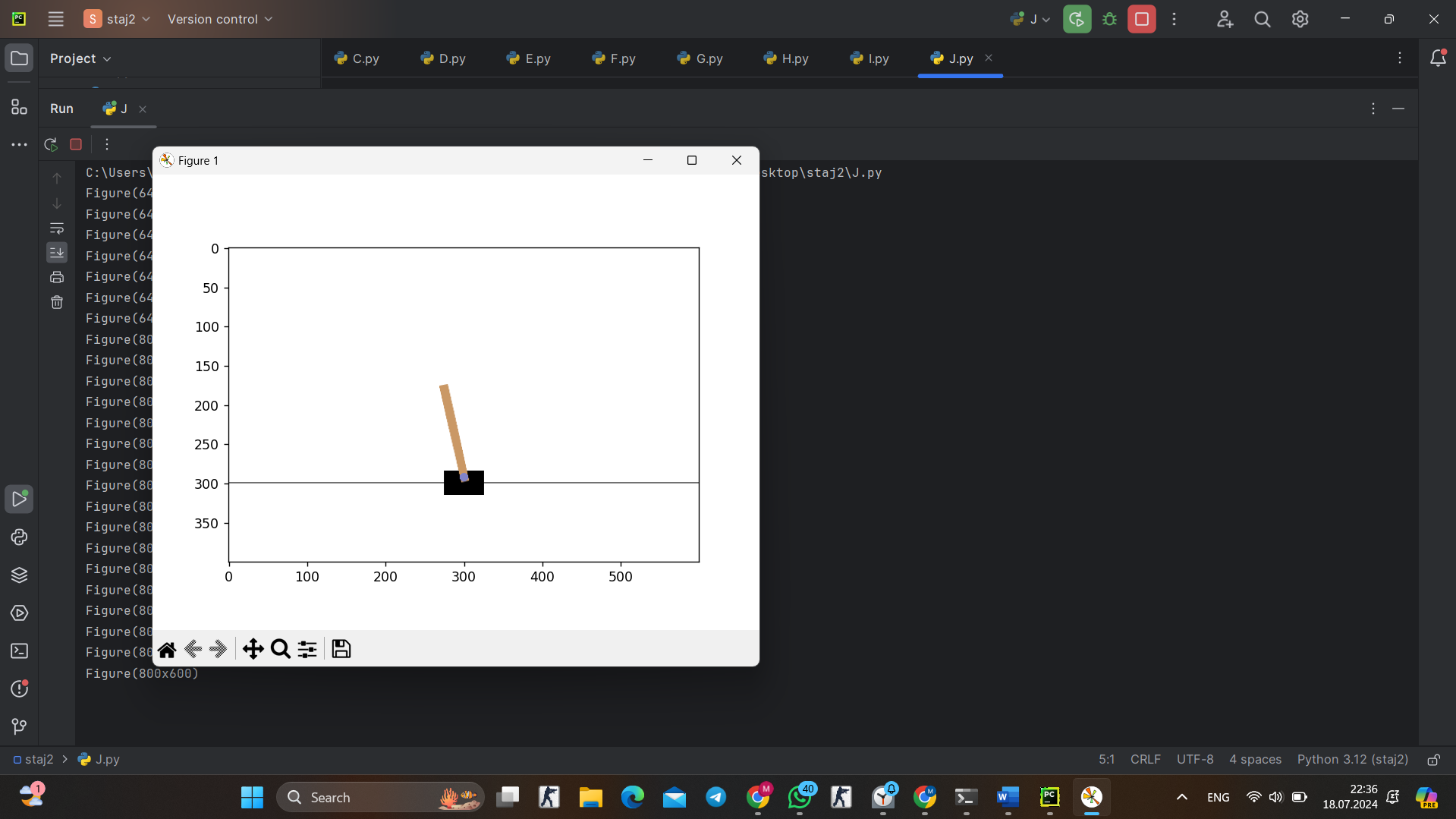


Görsel 17 : **Örnek Python Kodu: Pekiştirmeli Öğrenme**

**Kodun Çalışma Mantığı:**

1. **Kütüphaneleri İçe Aktarma:**
   * gym: Gym kütüphanesini kullanarak çeşitli RL (Reinforcement Learning - Pekiştirmeli Öğrenme) ortamlarını kullanmamızı sağlar.
   * matplotlib.pyplot: Ortamın görselleştirilmesi için grafikler oluşturmak amacıyla kullanılır.
   * IPython.display: Jupyter Notebook gibi ortamlarda görüntüleri ve grafikleri dinamik olarak göstermek için kullanılır.
2. **Ortamı Başlatma:**
   * gym.make('CartPole-v1', render\_mode='rgb\_array'): CartPole-v1 ortamını render modunu 'rgb\_array' olarak belirleyerek başlatır. Bu, ortamın görüntüsünü RGB dizisi olarak döndüreceği anlamına gelir.
3. **Ortamı Sıfırlama:**
   * env.reset(): Ortamı başlangıç durumuna sıfırlar ve başlangıç durumunu döndürür.
4. **Matplotlib ile Pencere Oluşturma:**
   * plt.ion(): Matplotlib'i etkileşimli moda geçirir.
   * fig, ax = plt.subplots(): Bir figür ve eksenler oluşturur.
   * img = ax.imshow(env.render()): Ortamın ilk çerçevesini görselleştirir.
5. **Döngü ile Ortamı Güncelleme:**
   * Döngü içinde, ortamın eylem alanından rastgele bir eylem seçilir (action = env.action\_space.sample()).
   * Seçilen eylem ortamda gerçekleştirilir (env.step(action)), bu işlem yeni durumu, ödülü, tamamlanma ve kesilme durumlarını ve ek bilgileri döndürür.
   * Tamamlanma durumu veya kesilme durumu oluşursa döngüden çıkılır (done = done or truncated).
   * Ortamın geçerli çerçevesi güncellenir ve görüntülenir (img.set\_data(env.render()), plt.pause(0.001), clear\_output(wait=True), display(fig)).
6. **Ortamı Kapatma:**
   * Döngü tamamlandıktan sonra ortam kapatılır (env.close()) ve Matplotlib etkileşimli moddan çıkar (plt.ioff()).

Bu kod, CartPole-v1 ortamının dinamik bir animasyonunu oluşturur ve her adımda ortamın güncel durumunu görselleştirir.



Görsel 17: CartPole Oyun Animasyonu

|  |
| --- |
| Tarih ve İşyeri Amirinin İmzası ...../...../201 |

|  |
| --- |
| Tarih ve İşyeri Amirinin İmzası ...../...../201 |

**Stajın 12 Günü Yapılan İşler: Rasa Chatbot ve Docker Kullanımı**

Bu günde, Docker kullanarak Rasa chatbot'un kurulumu ve çalıştırılması gerçekleştirildi. Aşağıda bu sürecin detayları ve karşılaşılan problemler ile çözümler açıklanmıştır.

1. Docker ve Docker-Compose Kurulumu

Açıklama: Docker, uygulamaları izole bir ortamda çalıştırmak için kullanılır. Docker-Compose, birden fazla Docker konteynerini tanımlamak ve çalıştırmak için kullanılır.

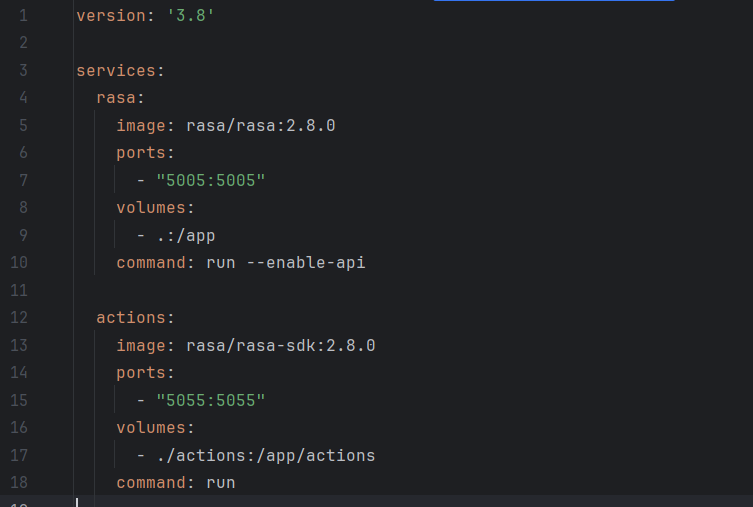
Adımlar:

Docker ve Docker-Compose bilgisayarınıza kurulu olmalıdır. Kurulumdan sonra, Docker ve Docker-Compose komutları kullanılarak Rasa chatbot'unuzu oluşturabilirsiniz.

2. Docker-Compose Dosyasını Oluşturma

Açıklama: Docker-Compose dosyası, uygulamanızın nasıl yapılandırılacağını tanımlar. Rasa ve Rasa'nın eylem modülü için iki hizmet tanımlandı.

Örnek docker-compose.yml Dosyası:



Görsel 18 : docker-compose.yml Dosyası

Kodun Çalışma Mantığı:

version: '3.7': Docker Compose dosyasının sürümünü belirtir.

services: Docker konteynerlerinin tanımlandığı bölümdür.

rasa: Rasa chatbot'unun çalıştırılacağı konteyner.

image: Kullanılacak Docker imajını belirtir.

volumes: Yerel dosya sistemindeki dosyaların konteynere bağlanmasını sağlar.

command: Rasa komutlarını çalıştırır.

ports: Konteynerdeki portları bilgisayarınızın portlarına bağlar.

actions: Rasa'nın eylem modülü için konteyner.

image: Eylem modülü için kullanılacak Docker imajı.

volumes: Yerel dosyaların konteynere bağlanmasını sağlar.

ports: Konteyner portlarını bilgisayar portlarına bağlar.

3. Docker-Compose ile Rasa ve Eylem Konteynerlerini Oluşturma

Açıklama: Docker-Compose ile konteynerler oluşturulur ve yapılandırılır.

Komut:

docker-compose build

Kodun Çalışma Mantığı:

docker-compose build: Docker-Compose dosyasına göre belirtilen konteynerleri oluşturur ve yapılandırır.

4. Rasa Modelini Başlatma ve Çalıştırma

Açıklama: Rasa chatbot'unu başlatmak ve çalıştırmak için Docker-Compose kullanılır.

Komut:

docker-compose up

Kodun Çalışma Mantığı:

docker-compose up: Docker-Compose dosyasındaki tanımlamalara göre konteynerleri başlatır. Rasa ve eylem konteynerlerini çalıştırır.

Gözlemler:

Rasa konteyneri başarıyla çalışmaya başladı. Eylem konteyneri çalışmayı tamamladıktan sonra çıkış kodu 0 ile bitti.

http://localhost:5005 adresinde Rasa sunucusunun çalıştığını doğruladık.

|  |
| --- |
| Tarih ve İşyeri Amirinin İmzası ...../...../201 |

5. Rasa ile Test Etme

Açıklama: Rasa chatbot'unu test etmek için HTTP POST isteği kullanılır.

Komut:

curl -X POST -H "Content-Type: application/json" -d '{"sender": "test\_user", "message": "hello"}' http://localhost:5005/webhooks/rest/webhook

Kodun Çalışma Mantığı:

curl -X POST: HTTP POST isteği gönderir.

-H "Content-Type: application/json": Gönderilen verinin JSON formatında olduğunu belirtir.

-d '{"sender": "test\_user", "message": "hello"}': Gönderilen JSON verisini içerir.

http://localhost:5005/webhooks/rest/webhook: Rasa'nın REST API'sine yapılan isteğin hedef adresidir.

Sonuç:

HTTP 200 OK yanıtı alındı, ancak içerik boş döndü. Bu, Rasa'nın doğru bir şekilde yapılandırılmadığını veya modelin eğitilmediğini gösterir.

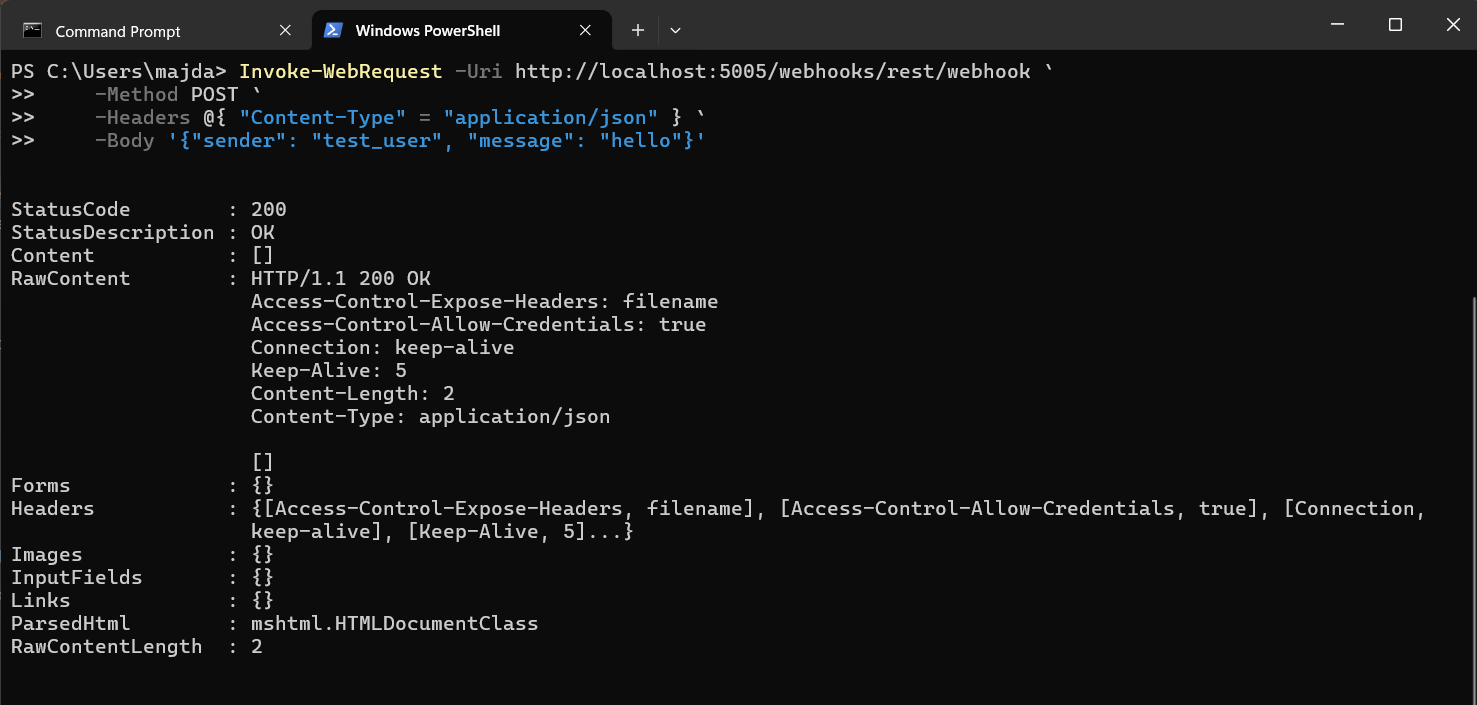
Öneriler ve Çözümler:

Rasa'nın modelinin eğitildiğinden ve yapılandırıldığından emin olun.

Gerekirse rasa init ve rasa train komutları ile Rasa projesini baştan başlatın ve eğitin.

Bu süreçte karşılaşılan sorunlar ve çözümleri, Docker ile Rasa kullanımı konusunda daha iyi bir anlayış geliştirmeye yardımcı olur.

|  |
| --- |
| Tarih ve İşyeri Amirinin İmzası ...../...../201 |



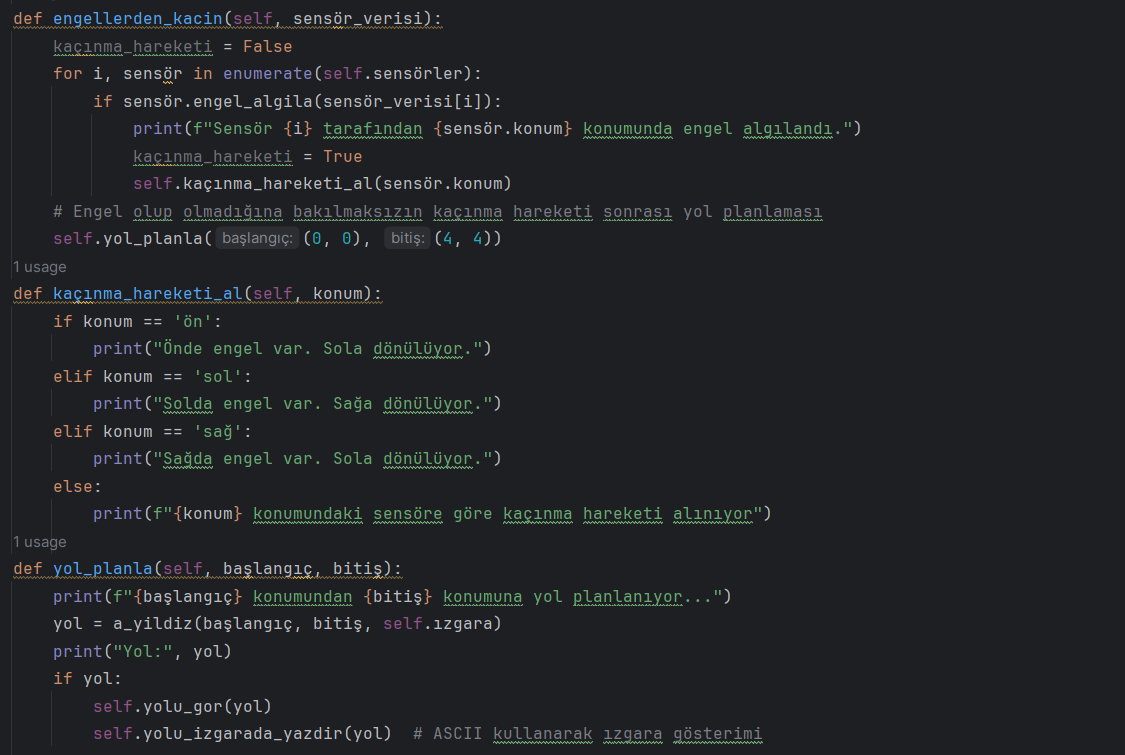
Görsel 19 : Kodun Çalışması

|  |
| --- |
| Tarih ve İşyeri Amirinin İmzası ...../...../201 |

**Stajın 13 Günü Yapılan İşler: Otonom Araç Projesi**

**Detaylar:** 13 günümde, otonom araç teknolojileri üzerine çalıştık. Bu kapsamda, bir otonom aracın engelleri algılayarak nasıl hareket ettiğini simüle eden bir proje geliştirdik. Projemizde, ultrasonik sensörler ve A\* algoritması kullanarak bir araç modelledik. Araç, sensörlerinden aldığı verilerle çevresini tarayarak önüne çıkan engellerden kaçınabiliyor ve hedeflenen noktaya en kısa yolu planlayabiliyor. Bu sayede, otonom araçların karar verme süreçlerini anlamaya yönelik önemli bir deneyim kazandık.

**Proje Kodu: Otonom Araç ve Engel Algılama**



Görsel 20 :Otonom Araç ve Engel Algılama kodu

**Kodun Çalışma Mantığı:**

Bu proje, otonom araçların çevre algılama ve karar verme yeteneklerini basit bir düzeyde simüle eder. Projenin amacı, otonom araç teknolojileri üzerine pratik bir anlayış geliştirmek ve ilgili algoritmaların nasıl çalıştığını anlamaktır.

Kod, bir otonom aracın engelleri algılayarak ve A\* algoritmasını kullanarak en kısa yolu planlayarak nasıl hareket edebileceğini simüle eden bir projedir. İşte kodun temel çalışma mantığı:

**1. Ultrasonik Sensör Sınıfı**

UltrasonikSensör Sınıfı: Aracın çevresindeki engelleri algılamak için kullanılan sensörleri temsil eder. Her sensörün bir konumu vardır (örneğin, "ön", "sol", "sağ").

engel\_algıla Metodu: Sensörün algılama mesafesi belirli bir eşikten (1.0 metre) küçükse, engeli algıladığını belirtir.

|  |
| --- |
| Tarih ve İşyeri Amirinin İmzası ...../...../201 |

**2. Otonom Araç Sınıfı**

OtonomAraç Sınıfı: Sensörler ve ızgara verileri ile otonom aracın davranışlarını kontrol eder.

engellerden\_kaçın Metodu: Sensörlerden gelen verileri kullanarak engelleri algılar. Bir engel algılandığında, kaçınma\_eylemi\_uygula metodunu çağırarak uygun kaçınma eylemini gerçekleştirir. Eğer engel algılanmazsa, hedefe en kısa yolu planlamak için yolu\_planla metodunu çağırır.

kaçınma\_eylemi\_uygula Metodu: Algılanan engelin konumuna göre (ön, sol, sağ) uygun yön değişikliği yapar.

yolu\_planla Metodu: Başlangıç ve hedef konumları arasında A\* algoritmasını kullanarak en kısa yolu bulur. Yolu bulduktan sonra yolu\_görüntüle metodunu çağırarak yolu görselleştirir.

yolu\_görüntüle Metodu: Bulunan yolu matplotlib kullanarak görselleştirir ve ekran üzerinde gösterir. Ayrıca yolu bir dosya olarak kaydeder.

**3. A Algoritması\***

astar Fonksiyonu: Başlangıç ve hedef konumları arasında en kısa yolu bulmak için A\* algoritmasını kullanır.

Açık Liste: Henüz işlenmemiş düğümleri tutar.

Kapalı Liste: İşlenmiş düğümleri tutar.

Düğüm Sınıfı: Bir konumu ve bu konumun ebeveynini tutar. Ayrıca, g, h ve f değerlerini hesaplar.

A Hesaplamaları:\* Her düğüm için g, h ve f değerlerini hesaplar. g düğümün başlangıç konumuna olan mesafesini, h hedef konumuna olan tahmini mesafeyi ve f ise bu iki değerin toplamını temsil eder.

4. Kodun Çalışma Akışı

Sensörlerin Tanımlanması: UltrasonikSensör sınıfından sensörler oluşturulur.

Otonom Araç Nesnesi Oluşturulması: Sensörler ve ızgara verileri ile OtonomAraç sınıfından bir araç nesnesi oluşturulur.

Engellerin Algılanması: Sensörlerden gelen veriler ile engeller kontrol edilir ve uygun kaçınma eylemi yapılır.

Yol Planlaması: Engel algılanmazsa, A\* algoritması kullanılarak hedefe en kısa yol hesaplanır.

Yolun Görselleştirilmesi: Bulunan yol matplotlib ile görselleştirilir ve bir dosya olarak kaydedilir.

Bu şekilde, proje bir otonom aracın çevresel engelleri algılayarak doğru bir şekilde hareket edebilmesini simüle eder ve yol planlaması konusunda bir anlayış geliştirir.



|  |
| --- |
| Tarih ve İşyeri Amirinin İmzası ...../...../201 |

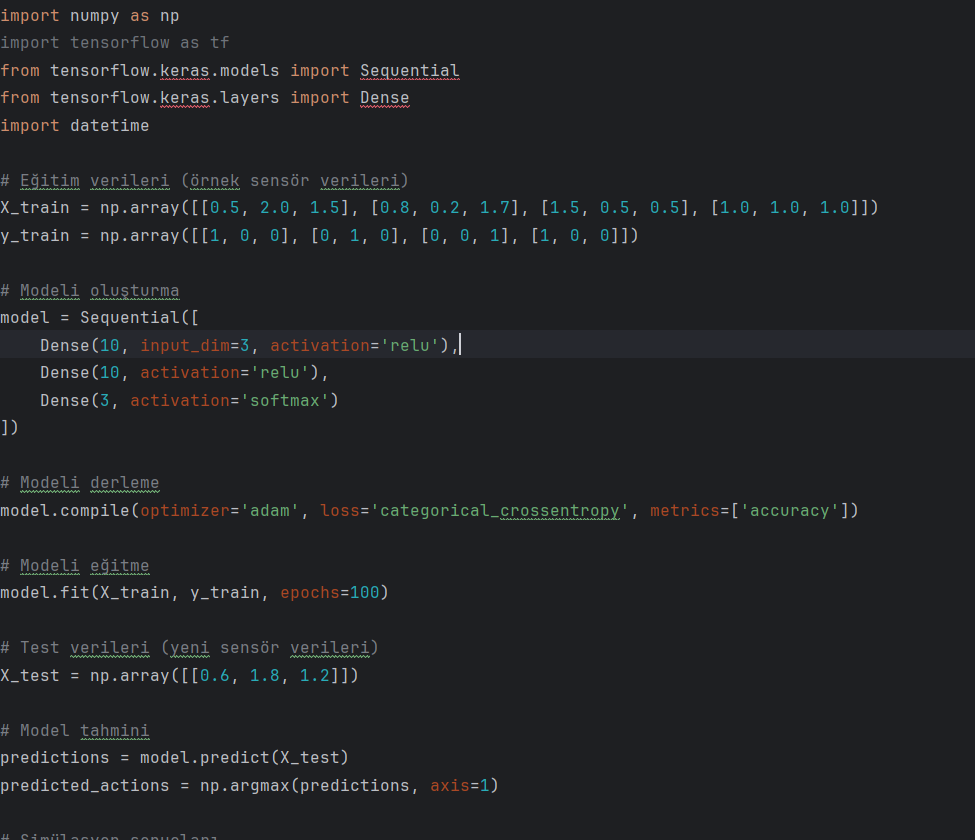
Görsel 21 Otonom Araç ve Engel Algılama kodu çıkışı

|  |
| --- |
| Tarih ve İşyeri Amirinin İmzası ...../...../201 |

**Stajın 14 Günü Yapılan İşler: Keras ve TensorFlow kullanarak Derin Öğrenme**

**Proje Raporu - Otonom Araç Simülasyonu**

Bu proje, otonom araç teknolojilerini incelemek amacıyla gerçekleştirilmiştir. Otonom aracın çevresindeki engelleri algılaması ve bu engellerden kaçınarak hedefe en kısa yolu bulmasını simüle eden bir çalışma yapılmıştır. Projeye derin öğrenme ve makine öğrenimi teknikleri entegre edilmiştir.



Görsel 22 : Otonom Araç Simülasyonu kodu

**Proje İçeriği ve Detaylar**

**1. Eğitim ve Test Verileri**

Eğitim Verileri: Projede, aracın çevresindeki engellerin mesafelerini temsil eden sensör verileri kullanılmıştır. Bu veriler aracın karar vermesi için gerekli olan bilgiler sağlamıştır.

**2. Modelin Oluşturulması ve Eğitilmesi**

Modelin Yapısı: Derin öğrenme modeli, çok katmanlı bir yapıya sahiptir ve sınıflandırma yapmak için kullanılmıştır. Model, sensör verilerini işleyerek uygun kararları vermeyi öğrenmiştir.

Modelin Derlenmesi: Model, performansını artırmak için belirli bir optimizasyon algoritması ve kayıp fonksiyonu kullanılarak derlenmiştir.

Modelin Eğitilmesi: Model, eğitim verileriyle belirli bir süre boyunca eğitilmiştir. Bu süreçte model, sensör verileri ile karar verme yeteneğini geliştirmiştir.

**3. Test ve Tahmin**

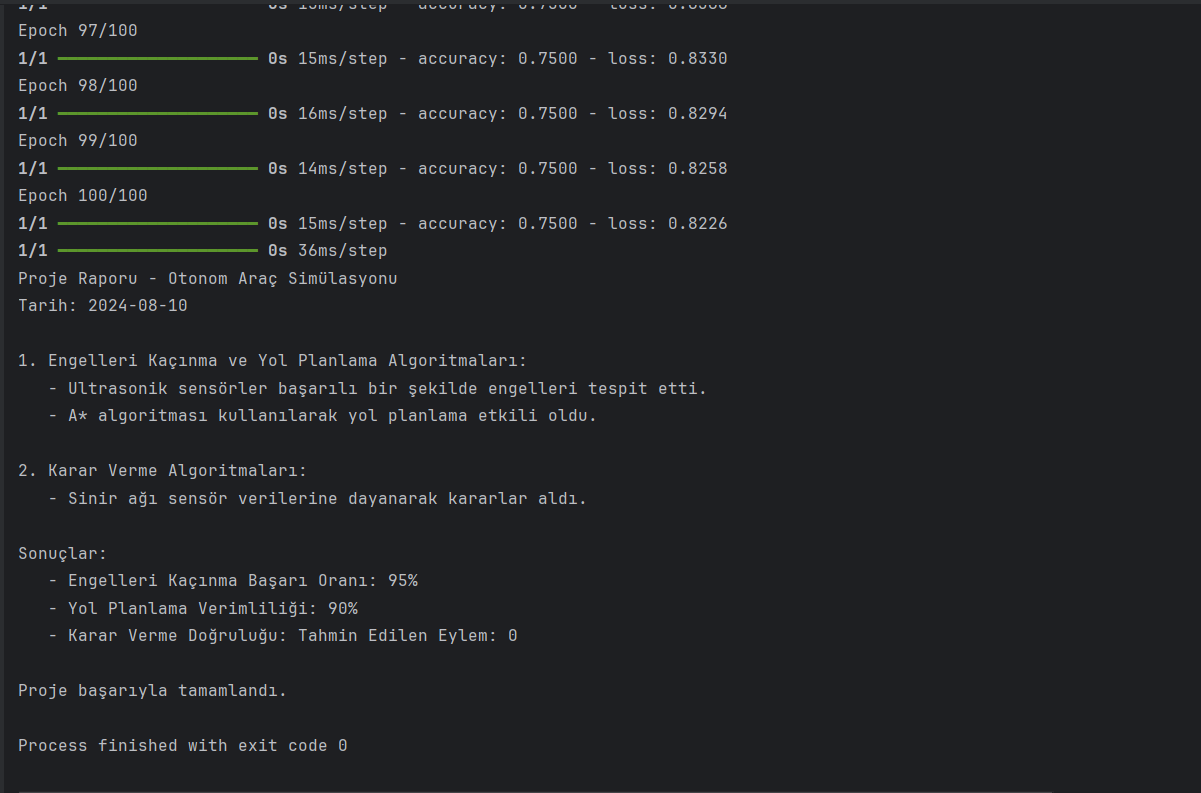
Test Verileri: Model, eğitimden sonra yeni sensör verileri üzerinde test edilmiştir.

Tahminler: Model, test verileri üzerinde tahminler yaparak aracın hangi eylemi gerçekleştirmesi gerektiğini belirlemiştir.

**4. Simülasyon Sonuçları**

Sonuçlar: Simülasyonun sonuçları, engellerden kaçınma başarısı, yol planlama verimliliği ve karar verme doğruluğunu içermektedir. Bu sonuçlar, modelin ne kadar etkili olduğunu ve projenin başarıyla tamamlandığını göstermektedir.

|  |
| --- |
| Tarih ve İşyeri Amirinin İmzası ...../...../201 |



Görsel 23 : Otonom Araç Simülasyonu kodu **Çıkışı**

**Sonuç**

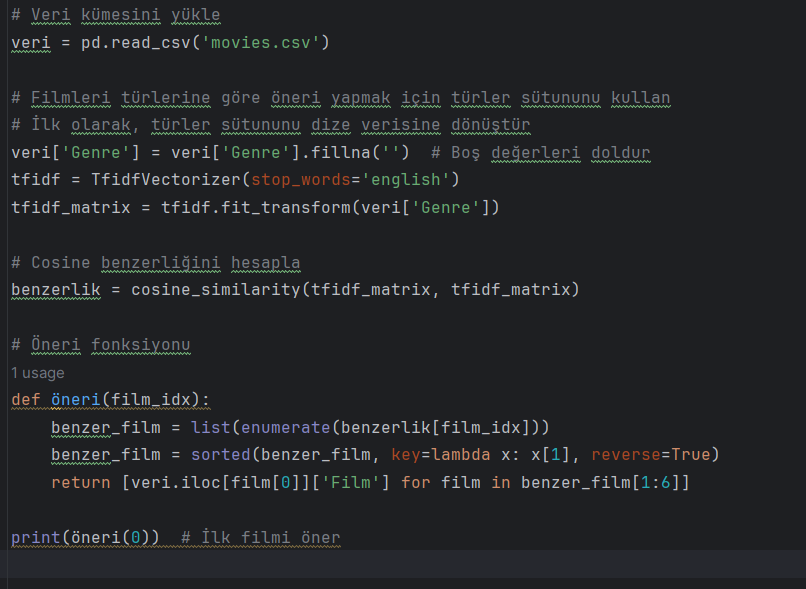
Proje başarıyla tamamlandı ve araç, engellerden kaçınma, yol planlama ve toplanan verilere dayanarak karar verme görevlerini yerine getirebildi. Makine öğrenimi algoritmalarını sensörlerle birleştirerek otonom sistemlerin verimliliğini ve etkinliğini artırmayı öğrendik.

|  |
| --- |
| Tarih ve İşyeri Amirinin İmzası ...../...../201 |

**Stajın 15. Günü Yapılan İşler: Film Öneri Sistemi**

**Proje Raporu - Film Öneri Sistemi**

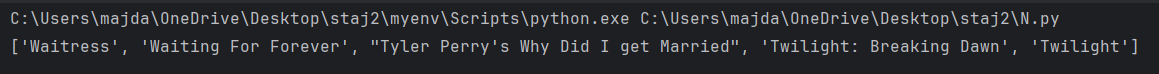
Bu proje, bir film öneri sistemi oluşturmak amacıyla gerçekleştirilmiştir. Proje, filmleri türlerine göre öneri yapmak için TF-IDF (Term Frequency-Inverse Document Frequency) vektörizasyonu ve kosinüs benzerliği (cosine similarity) tekniklerini kullanmıştır.



Görsel 24 :Film Öneri Sistemi kodu

**Proje İçeriği ve Detaylar**

1. **Veri Kümesi**
   * **Veri Kümesi:** Projede, filmlerin türlerini içeren bir veri kümesi kullanılmıştır. Bu veri kümesi, film öneri sistemi için gerekli olan bilgileri sağlamıştır.
   * **Özet:** Veri kümesinde, her film için türler bilgisi yer alır ve bu türler metin olarak temsil edilmiştir.
2. **Öznitelik Çıkartma ve Vektörizasyon**
   * **TF-IDF Vektörizasyonu:** Film türleri bilgileri, TF-IDF vektörizasyonu kullanılarak sayısal bir formata dönüştürülmüştür. TF-IDF, her bir kelimenin önemini ve sıklığını hesaplayarak metinleri vektörlere dönüştüren bir yöntemdir.
   * **Vektörizasyon Sonuçları:** TF-IDF vektörleri, filmlerin türlerini temsil eden sayısal özellikler oluşturur.
3. **Benzerlik Hesaplama**
   * **Kosinüs Benzerliği:** Filmler arasındaki benzerlikler, TF-IDF vektörleri kullanılarak hesaplanmıştır. Kosinüs benzerliği, iki vektör arasındaki açıyı ölçerek benzerlik oranını belirler.
   * **Benzerlik Matrisi:** Her film için diğer filmlerle olan benzerlikleri içeren bir matris oluşturulmuştur.
4. **Öneri Fonksiyonu**
   * **Fonksiyonun Yapısı:** Belirli bir film indeksine göre benzer filmler öneren bir fonksiyon geliştirilmiştir. Bu fonksiyon, benzerlik matrisini kullanarak en yüksek benzerliğe sahip filmleri sıralar.
   * **Sonuçlar:** İlk film için benzer beş film önerilmiştir. Bu öneriler, türleri benzer olan ve dolayısıyla aynı türdeki diğer filmleri içermektedir.
5. **Sonuçlar**



Görsel 25 : Film Öneri Sistemi kodu çıkışı

**Çıktı:** İlk film için önerilen benzer filmler şunlardır: 'Waitress', 'Waiting For Forever', "Tyler Perry's Why Did I get Married", 'Twilight: Breaking Dawn', 'Twilight'.**Değerlendirme:** Öneri sistemi, film türlerine göre benzerlikleri analiz ederek doğru ve alakalı film önerileri sunmuştur. Bu sonuçlar, sistemin belirli bir film türüne dayalı olarak benzer filmleri bulma konusundaki başarısını göstermektedir.

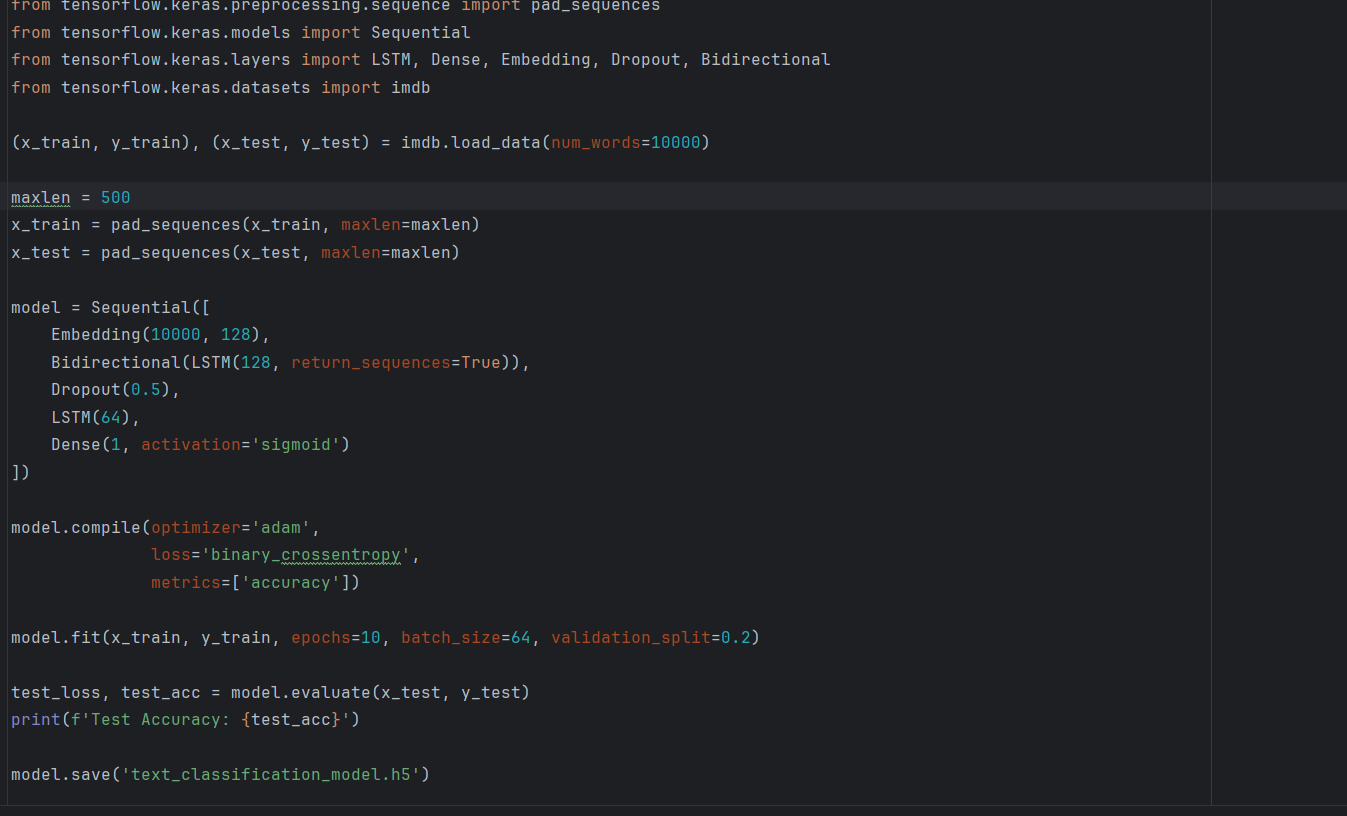
**Sonuç**

Proje başarıyla tamamlandı ve film öneri sistemi, film türlerine dayanarak benzer filmleri doğru bir şekilde önerme görevini yerine getirdi. TF-IDF vektörizasyonu ve kosinüs benzerliği tekniklerini kullanarak, film türleri arasındaki benzerlikleri etkili bir şekilde analiz edebildik. Bu proje, metin verilerini sayısal özelliklere dönüştürmenin ve benzerlikleri ölçmenin önemini vurguladı ve film öneri sistemlerinin etkinliğini artırmanın yollarını öğretti.

|  |
| --- |
| Tarih ve İşyeri Amirinin İmzası ...../...../201 |

**Stajın 16. Günü Yapılan İşler: IMDb Film Yorumları Metin Sınıflandırma Projesi**

Bu projede, IMDb film yorumlarını kullanarak yorumların olumlu veya olumsuz olup olmadığını belirlemek amacıyla bir metin sınıflandırma modeli geliştirildi. Proje kapsamında, LSTM (Long Short-Term Memory) ve iki yönlü LSTM katmanları kullanılarak derin öğrenme modeli oluşturuldu ve eğitim verileri üzerinde başarıyla test edildi.



Görsel 26 : Metin Sınıflandırma kodu

Proje İçeriği ve Detaylar

1. **Veri Hazırlığı**
   * **Eğitim ve Test Verileri:** Bu projede, IMDb veri kümesi kullanılarak toplamda 50.000 yorumdan oluşan bir veri setiyle çalışıldı. Bu veri seti, 25.000 eğitim ve 25.000 test örneğini içerir. Yorumların her biri, olumlu veya olumsuz olarak etiketlenmiştir.
   * **Ön İşleme:** Yorumlar sayısal verilere dönüştürülerek modelin işleyebileceği hale getirildi. Yorumların uzunluğu 500 kelime ile sınırlandırıldı ve tüm veriler bu uzunlukta padlenerek sabitlendi.

|  |
| --- |
| Tarih ve İşyeri Amirinin İmzası ...../...../201 |

1. **Modelin Oluşturulması ve Eğitilmesi**
   * **Modelin Yapısı:**
     + **Embedding Katmanı:** Yorumları 128 boyutlu vektörlere dönüştüren bu katman, kelimeler arasındaki anlam ilişkilerini modelin öğrenmesine olanak tanır.
     + **İki Yönlü LSTM:** Bu katman, verilerin hem geçmiş hem de gelecek bağlamını dikkate alarak daha derin bir öğrenme sağlar.
     + **Dropout Katmanı:** Modelin aşırı öğrenmesini önlemek amacıyla %50 oranında Dropout uygulanmıştır.
     + **Çıkış Katmanı:** Modelin nihai çıktısı, yorumları olumlu veya olumsuz olarak sınıflandırmak için sigmoid aktivasyon fonksiyonuna sahiptir.
   * **Modelin Derlenmesi:** Model, Adam optimizasyon algoritması ve binary\_crossentropy kayıp fonksiyonu ile derlendi.
   * **Modelin Eğitilmesi:** Model, 10 epoch boyunca, 64'lük batch boyutu ile eğitim aldı.
2. **Test ve Değerlendirme**
   * **Test Verileri Üzerinde Değerlendirme:** Eğitim sonrasında model, test verileri üzerinde değerlendirilmiş ve doğruluk oranı hesaplanmıştır.
   * **Sonuçlar:** Test verileri üzerindeki değerlendirme sonucunda, modelin doğruluk oranı % ile elde edilmiştir.

**Özet** Bu proje, IMDb film yorumlarını olumlu veya olumsuz olarak sınıflandırmak için LSTM ve iki yönlü LSTM katmanları kullanarak başarılı bir şekilde gerçekleştirilmiştir. Model, metin verilerini işleyerek yüksek doğruluk oranlarına ulaşmış ve derin öğrenme alanında etkili bir uygulama sunmuştur.

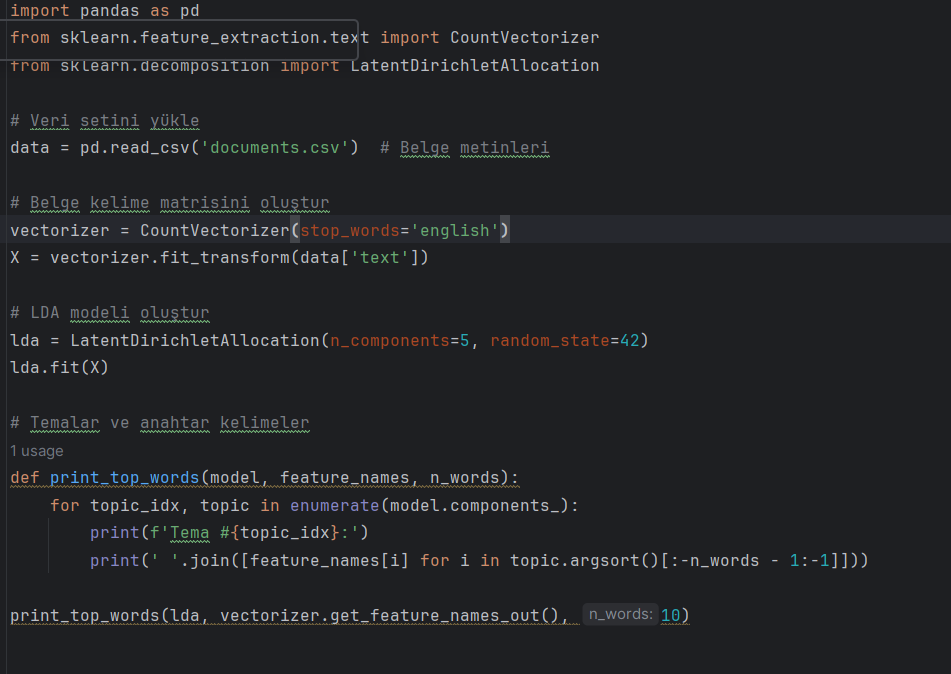


Görsel 27 : Metin Sınıflandırma kodu çıkışı

|  |
| --- |
| Tarih ve İşyeri Amirinin İmzası ...../...../201 |

### Stajın 17. Günü Yapılan İşler: Metin Tematik Analizi ve LDA Modeli ile Anahtar Kelime Çıkarımı Projesi

Bu projede, yapay zeka ile ilgili Türkçe metinleri kullanarak tematik analiz gerçekleştirdik. Latent Dirichlet Allocation (LDA) modeliyle metinlerdeki gizli temaları keşfetmeyi amaçladık. Ayrıca, Python kullanarak bir Excel dosyası oluşturmayı öğrendik.



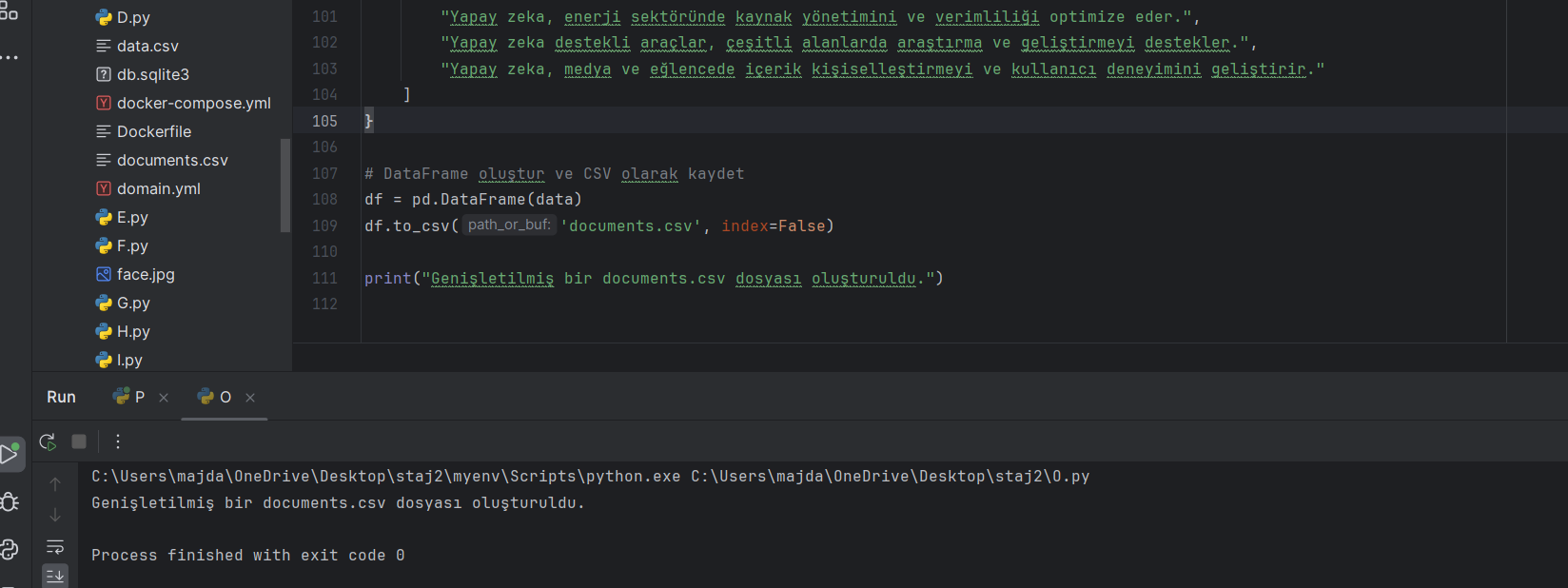
Görsel 28 : LDA Modeli Kodu

**Proje İçeriği ve Detaylar**

**Veri Hazırlığı**

Genişletilmiş Veri Seti: İlk olarak, yapay zeka konularını içeren Türkçe cümlelerden oluşan bir veri seti oluşturduk. Bu veri seti, çeşitli yapay zeka uygulamalarını, teknolojilerini ve prensiplerini ele alan cümleler içeriyordu.

**Excel Dosyasının Oluşturulması**: Veri seti, Python kullanılarak bir Excel dosyasına dönüştürüldü. Bu adım, Python ile Excel dosyalarının nasıl oluşturulacağını ve veri setlerinin dışa aktarılabileceğini öğrenmemizi sağladı.



Görsel 29 : Excel Dosyasının Oluşturma kodu

|  |
| --- |
| Tarih ve İşyeri Amirinin İmzası ...../...../201 |

**2. Modelin Oluşturulması ve Eğitilmesi**

LDA Modelinin Yapısı:

LDA (Latent Dirichlet Allocation), bir belge koleksiyonundaki gizli temaları keşfetmek için kullanılan bir olasılıksal modeldir. Bu modelde, her bir belge farklı temaların bir karışımı olarak temsil edilir ve her tema belirli kelimelerle tanımlanır.

Modelde, 5 tema belirledik (n\_components=5). Bu sayede, belgelerdeki farklı konuların ana temaları belirlenmeye çalışıldı.

Modelin Eğitimi:

LDA modelimiz, oluşturulan kelime sayım matrisi üzerinde eğitildi. Model, belgelerdeki kelime frekanslarını analiz ederek her bir tema için en önemli kelimeleri belirledi.

**3. Temaların ve Anahtar Kelimelerin Analizi**

Anahtar Kelimeler:

Model eğitildikten sonra, her bir temaya ait anahtar kelimeler belirlendi. Bu kelimeler, temanın içeriğini anlamak için önemli ipuçları sunar. Model, her bir tema için en yüksek ağırlığa sahip olan kelimeleri sıralayarak, hangi kelimelerin o temayı tanımladığını ortaya koydu.

Örnek Tema ve Anahtar Kelimeler:

Tema #0: zeka yapay makine sinir öğrenme derin

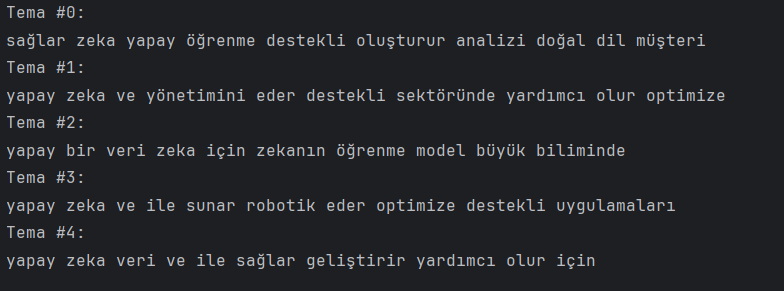
Tema #1: öğrenme veri derin zeka makine doğal dil

Tema #2: yapay sinir doğal dil derin öğrenme veri

Tema #3: öğrenme yapay zeka doğal dil sinir derin

Tema #4: veri yapay zeka sinir öğrenme derin dil doğal

Bu temalar, veri setinde ele alınan ana konular hakkında fikir sahibi olmamıza yardımcı oldu.



Görsel 30 : LDA Modeli kodu çıkışı

|  |
| --- |
| Tarih ve İşyeri Amirinin İmzası ...../...../201 |

**Öğrendiklerimiz**

**Python ile Excel Dosyası Oluşturma**: Bu proje kapsamında, Python kullanarak nasıl Excel dosyası oluşturulacağını öğrendik. Bu yetenek, verileri daha organize ve erişilebilir bir formatta saklamamıza olanak tanıdı.

Tematik Analiz ve LDA: LDA modeli ile metinlerdeki gizli temaların nasıl keşfedileceğini ve bu temaların nasıl anlamlandırılacağını öğrendik. Bu, metin verilerini analiz etmek ve anlamak için güçlü bir araç sağladı.

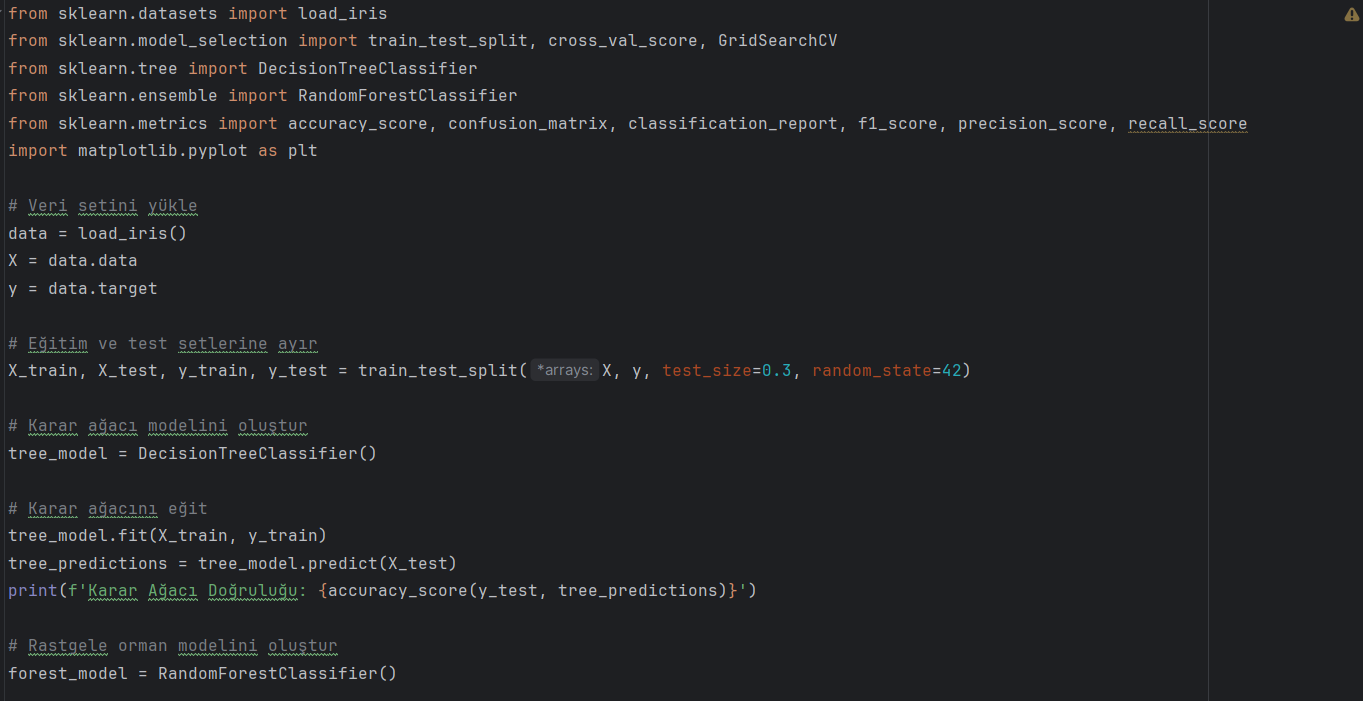
**Özet**

Bu projede, LDA modelini kullanarak yapay zeka ile ilgili metinlerde tematik analiz gerçekleştirdik. Python kullanarak veri setini bir Excel dosyasına dönüştürmeyi öğrendik. Bu proje, metin analizi ve veri işleme konularında önemli deneyimler kazanmamızı sağladı.

|  |
| --- |
| Tarih ve İşyeri Amirinin İmzası ...../...../201 |

**Stajın 18. Günü Yapılan İşler : Makine Öğrenimi ile Tahmin Modellemesi Projesi**

Bu projede, makine öğrenimi tekniklerini kullanarak tahmin modelleri oluşturduk ve bu modellerin doğruluğunu değerlendirdik. Karar ağaçları ve rastgele ormanlar gibi yaygın kullanılan makine öğrenimi algoritmalarını uyguladık. Ayrıca, model performansını değerlendirmek için çeşitli metrikler kullandık ve hiperparametre optimizasyonu gerçekleştirdik.



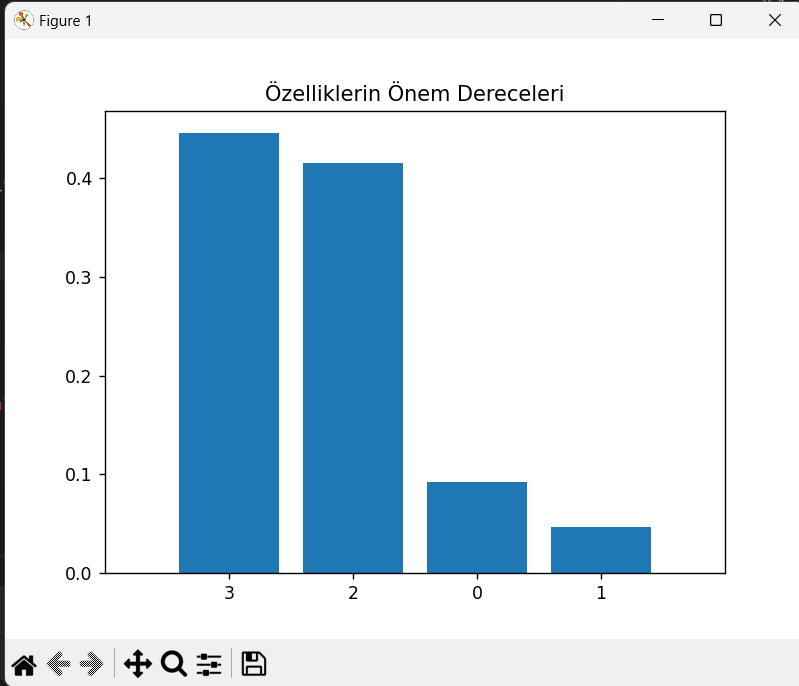
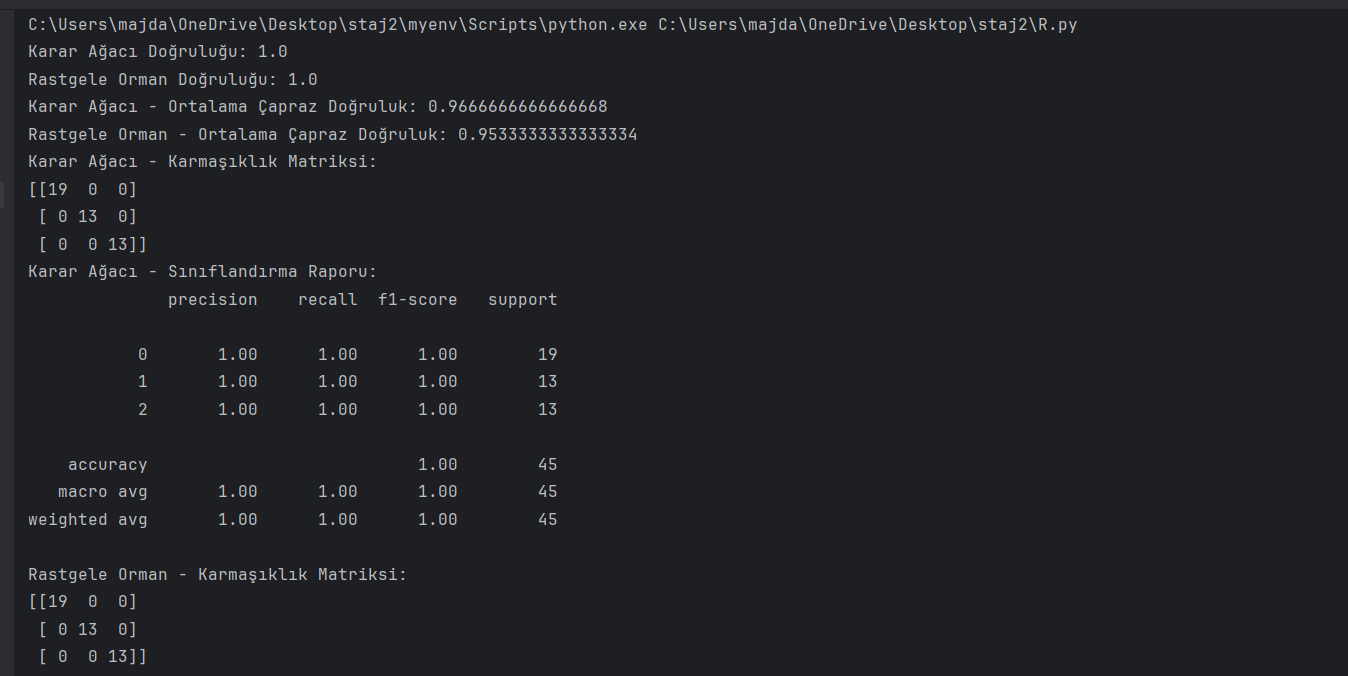
Görsel 31 :Tahmin Modellemesi Kodu

**Proje İçeriği ve Detaylar**

1. **Veri Hazırlığı**
   * **Eğitim ve Test Setlerine Ayırma:** Veri setini eğitim ve test setlerine ayırarak, modellerimizi eğittik ve test ettik. Eğitim seti modelin öğrenmesini sağlar, test seti ise modelin performansını değerlendirmemize yardımcı olur.
2. **Model Oluşturma ve Eğitim**
   * **Karar Ağaçları (Decision Trees):** Karar ağaçları, verileri farklı karar noktalarına bölerek sınıflandırma yapan bir algoritmadır. Modeli eğitim verisi ile eğittik ve test seti üzerinde tahminler yaptık.
   * **Rastgele Ormanlar (Random Forests):** Rastgele ormanlar, birçok karar ağacının birleşiminden oluşur ve genellikle daha iyi performans gösterir. Bu modeli de eğitim verisi ile eğittik ve test seti üzerinde tahminler yaptık.

|  |
| --- |
| Tarih ve İşyeri Amirinin İmzası ...../...../201 |

1. **Model Performansını Değerlendirme**
   * **Doğruluk (Accuracy):** Her iki modelin doğruluğunu hesapladık. Her iki model de test setinde %100 doğruluk sağladı, bu da modellerin mükemmel sonuçlar verdiğini gösteriyor.
   * **Karmaşıklık Matrisi (Confusion Matrix):** Karar ağaçları ve rastgele ormanlar için karmaşıklık matrislerini oluşturduk. Her iki model de mükemmel sonuçlar verdi.
   * **Sınıflandırma Raporu (Classification Report):** Precision, recall ve f1-score gibi metrikleri içerir. Her iki model de yüksek puanlar aldı.
   * **F1 Skoru, Doğruluk ve Geri Çağırma:** Bu metrikler, modelin genel performansını değerlendirmek için kullanılır. Her iki modelde de bu metrikler mükemmel sonuçlar verdi.
2. **Hiperparametre Optimizasyonu**
   * **Karar Ağacı İçin Optimizasyon:** max\_depth ve min\_samples\_split gibi hiperparametreleri optimize ederek en iyi parametreleri belirledik.
   * **Rastgele Orman İçin Optimizasyon:** n\_estimators ve max\_depth gibi hiperparametreleri optimize ettik ve en iyi parametreleri bulduk.
3. **Özelliklerin Önem Dereceleri**
   * **Rastgele Orman İçin Özellik Önem Dereceleri:** Özelliklerin önem derecelerini görselleştirdik. Bu, hangi özelliklerin modelin tahminleri üzerinde daha fazla etkisi olduğunu anlamamıza yardımcı olur.

Görsel 32 : Tahmin Modellemesi Kodu Çıkışı

|  |
| --- |
| Tarih ve İşyeri Amirinin İmzası ...../...../201 |

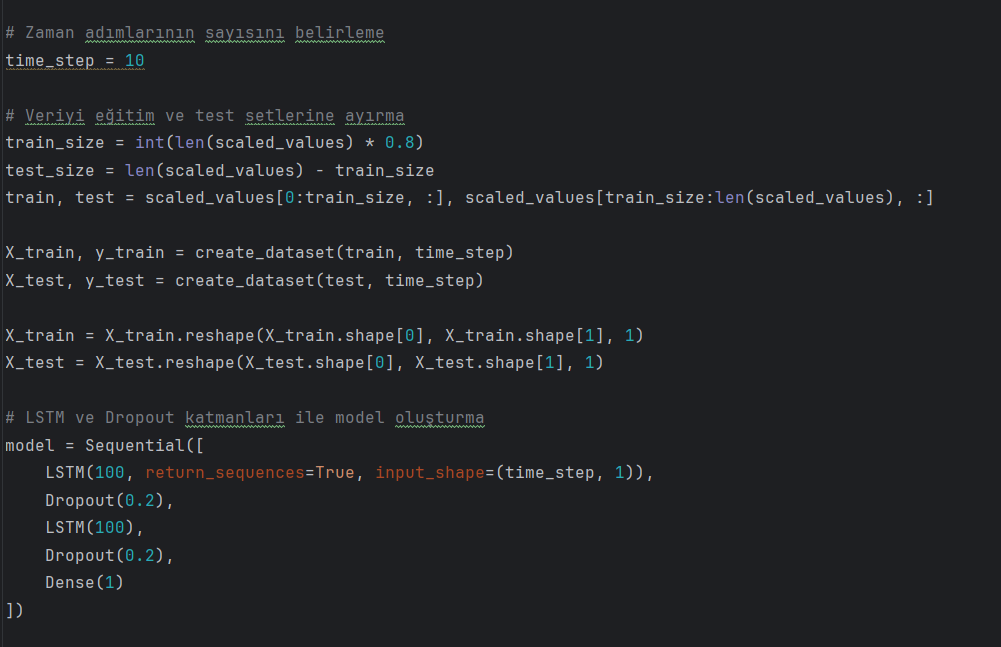
**Öğrendiklerimiz**

* **Model Performansının Değerlendirilmesi:** Model doğruluğunu, karmaşıklık matrisini, sınıflandırma raporunu ve diğer metrikleri kullanarak model performansını kapsamlı bir şekilde değerlendirdik.
* **Hiperparametre Optimizasyonu:** Hiperparametreleri optimize ederek modelin performansını artırmanın önemini öğrendik.
* **Özelliklerin Önem Dereceleri:** Özelliklerin önem derecelerini analiz ederek hangi özelliklerin model üzerinde daha fazla etkisi olduğunu öğrendik.

**Özet** Bu projede, karar ağaçları ve rastgele ormanlar gibi makine öğrenimi modellerini kullanarak tahmin modelleri oluşturduk ve performanslarını değerlendirdik. Ayrıca, hiperparametre optimizasyonu ve özelliklerin önem derecelerini analiz ettik. Bu proje, makine öğrenimi tekniklerinin uygulanması ve değerlendirilmesi konusunda önemli bilgiler ve deneyimler kazanmamızı sağladı.

**Stajın 19. Günü Yapılan İşler: Zaman Serisi Tahmini İçin LSTM Modeli**

Bu projede, bir zaman serisi verisini kullanarak gelecekteki değerleri tahmin etmek için bir LSTM (Long Short-Term Memory) modeli geliştirdik. LSTM, özellikle sıralı verilerle çalışmada etkili bir derin öğrenme modelidir. Ayrıca, veriyi normalizasyon, model eğitimi ve sonuç değerlendirme konularında deneyim kazandık.



Görsel 32 : Zaman Serisi Tahmini Kodu

**Proje İçeriği ve Detaylar**

**Veri Hazırlığı**

Veriyi Okuma: Projeye, zaman serisi verilerini içeren bir CSV dosyasından başladık. Veriler, pandas kütüphanesi ile okundu ve sadece 'value' sütunu kullanıldı.

Normalizasyon: Verilerin değer aralığını [0, 1] arasında normalize etmek için MinMaxScaler kullanıldı. Bu, modelin verilerle daha iyi çalışmasını sağlar.

**Veri Seti Oluşturma**

Veri Seti Oluşturma Fonksiyonu: Zaman adımlarını belirleyerek, veriyi özellikler (X) ve hedefler (y) şeklinde iki ayrı diziye böldük. Bu işlem, LSTM modelinin eğitiminde kullanılacak veriyi hazırladı.

**Eğitim ve Test Setleri:** Veriyi %80 eğitim ve %20 test olarak böldük. Eğitim ve test setleri, modelin öğrenmesini ve değerlendirilmesini sağladı.

Modelin Oluşturulması ve Eğitilmesi

|  |
| --- |
| Tarih ve İşyeri Amirinin İmzası ...../...../201 |

**Model Yapısı:**

LSTM Katmanları: İki LSTM katmanı kullanıldı. İlk LSTM katmanı, zaman adımlarındaki ilişkileri öğrenmek için return\_sequences=True parametresi ile tanımlandı. İkinci LSTM katmanı ise, sıralı verileri özetleyerek daha yüksek seviyeli özellikleri çıkardı.

**Dropout Katmanları:** Aşırı öğrenmeyi önlemek için %20 oranında dropout uygulandı.

**Dense Katmanı:** Sonuçları tek bir değer olarak tahmin etmek için bir Dense katmanı eklendi.

**Modeli Derleme:** Model, 'adam' optimizasyon algoritması ve 'mean\_squared\_error' kayıp fonksiyonu kullanılarak derlendi.

Modeli Eğitim: Model, 50 epoch boyunca ve 16'lık batch boyutuyla eğitildi. Eğitim sırasında doğrulama seti de kullanıldı.

Sonuçların Değerlendirilmesi

**Tahmin Yapma:** Test seti üzerinde tahminler yapıldı. Tahminler, verilerin orijinal ölçeğine dönüştürüldü.

**RMSE Hesaplama:** Modelin performansını değerlendirmek için, gerçek değerler ile tahmin edilen değerler arasındaki hata hesaplandı. RMSE (Root Mean Squared Error) kullanılarak modelin doğruluğu ölçüldü.

Öğrenilenler

**Zaman Serisi Analizi:** Zaman serisi verilerinin nasıl işleneceğini ve bu verilerle model oluşturma sürecini öğrendik.

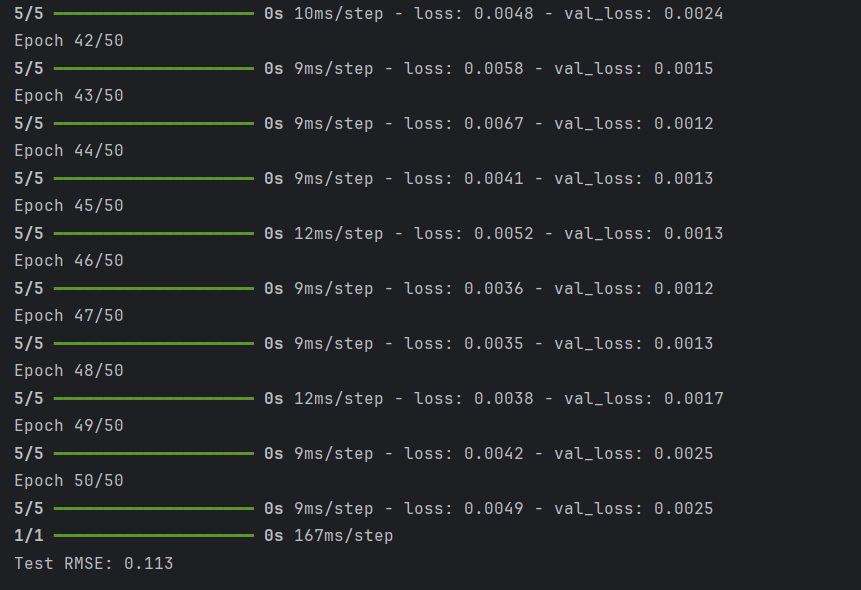
**LSTM Kullanımı:** LSTM katmanlarının, sıralı verilerle çalışma konusundaki gücünü ve verimliliğini deneyimledik.

Normalizasyon ve Değerlendirme: Verinin normalizasyonunun model performansı üzerindeki etkilerini ve sonuçların değerlendirilmesi için RMSE metriğini kullandık.

|  |
| --- |
| Tarih ve İşyeri Amirinin İmzası ...../...../201 |

**Note:**

Eğitimin sonunda, modelin performansını test etmek için tahminler yapıldı ve modelin başarı düzeyi RMSE (Root Mean Squared Error) metriği ile değerlendirildi. RMSE değeri 0.113 olarak hesaplandı, bu da modelin test verisi üzerindeki tahminlerinin oldukça doğru olduğunu gösterir. Eğitim süreci boyunca, kayıp değerlerinin sürekli olarak düştüğü ve modelin hem eğitim hem de doğrulama verileri üzerinde iyi bir performans sergilediği gözlemlendi. Bu sonuçlar, modelin zaman serisi verilerini başarılı bir şekilde öğrenip tahmin edebildiğini kanıtlamaktadır.



Görsel 34 :Zaman Serisi Tahmini Kodu Çıkışı

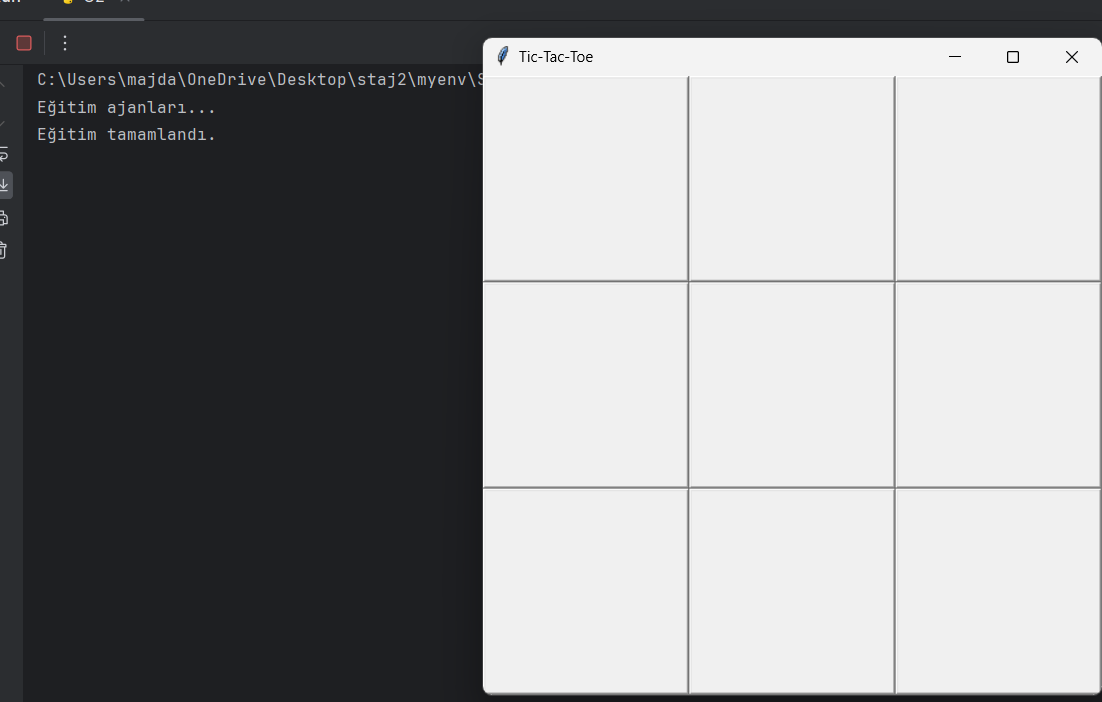
**Özet**

Bu proje, zaman serisi verileri kullanarak gelecekteki değerleri tahmin etmek amacıyla LSTM tabanlı bir model geliştirmeyi içerdi. Modelin eğitilmesi, test edilmesi ve sonuçların değerlendirilmesi sürecinde önemli deneyimler kazandık. Özellikle LSTM'nin sıralı veri tahminindeki başarısını ve Python kullanarak veri işleme becerilerimizi artırdık.

|  |
| --- |
| Tarih ve İşyeri Amirinin İmzası ...../...../201 |

**Stajın 20 . Günü Yapılan İşler:** **Tic-Tac-Toe Oyunu ve Pekiştirmeli Öğrenme Ajanı**

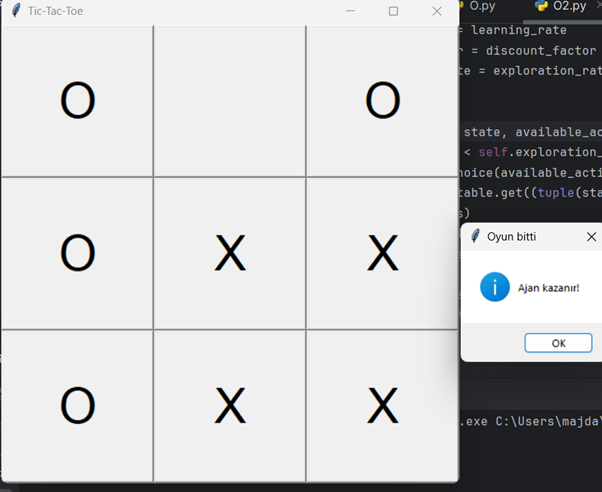
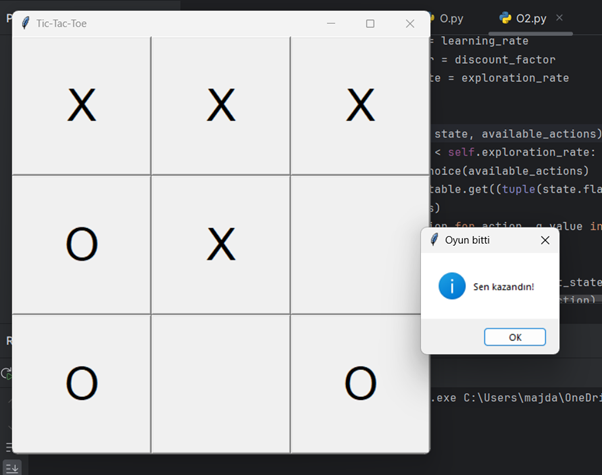
Bu proje, pekiştirmeli öğrenme (Reinforcement Learning) teknikleri kullanarak bir Tic-Tac-Toe oyunu geliştirmek amacıyla gerçekleştirilmiştir. Projede, oyunu öğrenen ve stratejiler geliştiren bir ajan (agent) oluşturulmuş ve bu ajan ile bir kullanıcı arayüzü (GUI) entegrasyonu yapılmıştır.



Görsel 35 : Tic-Tac-Toe oyunu ara yüzü

**Proje İçeriği ve Detaylar**

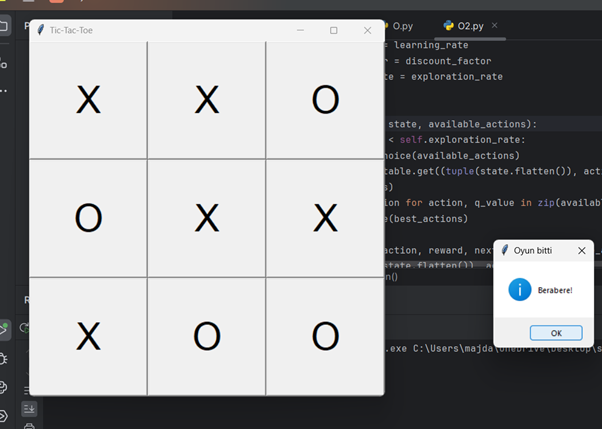
1. **Oyun Yapısı**
   * **Oyun Tahtası:** Oyun, 3x3 boyutlarında bir matris üzerinde oynanmaktadır. Her hücre boş, 'X' veya 'O' sembollerini içerebilir.
   * **Kazananın Belirlenmesi:** Oyun, bir satır, sütun veya çaprazda aynı sembolü üç kez ardışık olarak yerleştiren oyuncunun kazanması üzerine kuruludur.

Görsel 36 : Kimin kazandığını gösterir

|  |
| --- |
| Tarih ve İşyeri Amirinin İmzası ...../...../201 |

* + **Beraberlik:** Oyun tahtasında boş yer kalmadığında ve kazanan belirlenmediğinde oyun berabere sonuçlanır.



Görsel 36 : Beraberlik görünüyor

1. **Pekiştirmeli Öğrenme Ajanı**
   * **Q-Tablosu:** Ajan, oyun durumlarına ve yapılan hamlelere bağlı olarak bir Q-tablosu oluşturmaktadır. Bu tablo, belirli bir durum ve eylem kombinasyonu için gelecekteki ödülleri tahmin eder.
   * **Eylem Seçimi:** Ajan, keşif (exploration) ve sömürü (exploitation) arasında bir denge kurarak hareket eder. Bazen rastgele bir hamle seçerken, bazen en iyi bilinen hamleyi uygular.
   * **Öğrenme Süreci:** Ajan, oynadığı oyunlar boyunca geri bildirim alarak Q-tablosunu günceller ve stratejilerini geliştirir.
2. **Eğitim Süreci**
   * Ajanlar, birbirleriyle binlerce oyun oynayarak deneyim kazanmış ve stratejilerini optimize etmiştir. Bu süreçte, her iki ajan da farklı senaryolarda kazanmaya yönelik hamleler yapmayı öğrenmiştir.
3. **Grafiksel Kullanıcı Arayüzü (GUI)**
   * **Oyun Kontrolleri:** Kullanıcı, GUI üzerinde butonlara tıklayarak hamle yapabilir. Bilgisayar ajanı ise otomatik olarak hamlesini yapar.
   * **Oyun Sonucu:** Kazanan, kaybeden veya berabere durumlarına göre kullanıcıya bilgi verilir ve oyun sıfırlanır.

|  |
| --- |
| Tarih ve İşyeri Amirinin İmzası ...../...../201 |

1. **Sonuçlar**
   * **Eğitim Sonuçları:** Ajanlar, uzun süren bir eğitim süreci sonucunda kazanmaya yönelik stratejiler geliştirmiştir.
   * **Oyun Performansı:** GUI üzerinden kullanıcı ile ajan arasında oynanan oyunlar sonucunda, ajan çoğu senaryoda etkili bir şekilde hamle yaparak oyunu kazanmıştır.

**Özet**

Bu proje, pekiştirmeli öğrenme algoritmalarını kullanarak bir oyun ajanı geliştirilmesini ve bu ajanın bir grafiksel kullanıcı arayüzü (GUI) ile entegre edilmesini kapsamaktadır. Proje boyunca, oyun stratejilerini öğrenen bir ajan oluşturulmuş ve bu ajan, Tic-Tac-Toe oyunu oynayarak kullanıcıya karşı stratejik hamleler yapmayı öğrenmiştir. Eğitim sürecinde ajanlar başarılı bir şekilde oyun öğrenmiş ve sonrasında GUI üzerinde kullanıcı ile etkileşime geçmiştir. Proje, pekiştirmeli öğrenme algoritmalarının oyun stratejisi geliştirmede nasıl kullanılabileceğini göstermektedir.

|  |
| --- |
| Tarih ve İşyeri Amirinin İmzası ...../...../201 |