KÜTAHYA DUMLUPINAR ÜNİVERSİTESİ



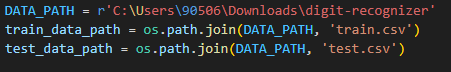
YÜKSEK DÜZEY PROGRAMLAMA PROJE RAPORU

KONU:DIGIT RECOGNIZER

HAZIRLAYAN:202113172008) BARIŞ ALTıN

DANIŞMAN:DOÇ.DR.HASAN TEMURTAŞ

KODLAR:



**DATA\_PATH = 'C:\\Users\\90506\\Downloads\\digit-recognizer'**:

Burada, dosya yolunun doğru bir şekilde tanımlandığına emin olmak için ters eğik çizgiler (\\) kullanmalısınız. Python'da tek bir eğik çizgi (\) özel karakter olarak kullanıldığından, iki eğik çizgi kullanarak doğru yolu belirtmek gerekir.

**train\_data\_path = os.path.join(DATA\_PATH, 'train.csv')**:

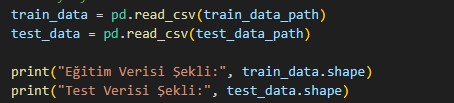
**os.path.join** fonksiyonu, platform bağımsız bir şekilde dosya yolunu birleştirir. Burada, DATA\_PATH ile train.csv dosyasının yolunu birleştiriyorsunuz.

**test\_data\_path = os.path.join(DATA\_PATH, 'test.csv')**:

Aynı şekilde, test.csv dosyasının tam yolunu oluşturuyorsunuz.

**print("Eğitim ve test veri yolları tanımlandı.")**:

Bu satırda, veri yollarının başarıyla tanımlandığını bildiren bir mesaj yazdırılır.



**train\_data = pd.read\_csv(train\_data\_path)**:

**pd.read\_csv()** fonksiyonu, belirtilen dosya yolundan (train\_data\_path) CSV dosyasını okur ve bir pandas DataFrame olarak yükler.

**train\_data** değişkeni, eğitim veri kümesini tutar.

**test\_data = pd.read\_csv(test\_data\_path)**:

Bu satır, test veri kümesinin bulunduğu test\_data\_path dosyasını okur ve **test\_data** adlı bir pandas DataFrame'e yükler.

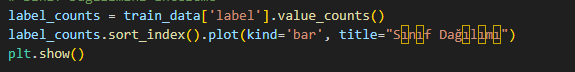
**print("Eğitim Verisi Şekli:", train\_data.shape)**:

**train\_data.shape** ifadesi, eğitim verisinin **satır ve sütun** sayısını verir. **shape** özelliği bir tuple döndürür: (satır sayısı, sütun sayısı).

Bu satır, eğitim veri kümesinin boyutlarını (satır ve sütun sayısını) ekrana yazdırır.

**print("Test Verisi Şekli:", test\_data.shape)**:

Aynı şekilde, test veri kümesinin boyutları ekrana yazdırılır.



**label\_counts = train\_data['label'].value\_counts()**:

**train\_data['label']** ifadesi, eğitim veri setindeki **label** sütununu alır.

**value\_counts()** fonksiyonu, her bir sınıfın (etiketin) kaç kez tekrar ettiğini sayar. Bu, sınıf dengesizliğini veya her sınıfın ne kadar temsil edildiğini görmek için kullanışlıdır.

**label\_counts.sort\_index().plot(kind='bar', title="Sınıf Dağılımı")**:

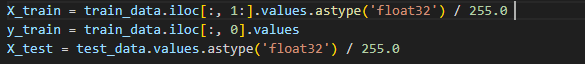
**sort\_index()** fonksiyonu, etiketlerin sırasını **arttıran** şekilde sıralar. (Sınıflar sırasıyla grafikte görünecek.)

**plot(kind='bar')** ifadesi, veriyi bir **çubuk grafik** (bar chart) olarak çizer.

**title="Sınıf Dağılımı"** ifadesi, grafiğin başlığını "Sınıf Dağılımı" olarak ayarlar.

**plt.show()**:

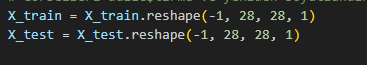
Matplotlib grafiği ekranda görüntülemek için **show()** fonksiyonu çağrılır.



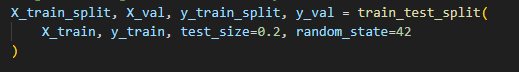
**X\_train**: Eğitim verisindeki **özellikleri** içerir (etiketler hariç). Görsel veriler 0 ile 1 arasında normalize edilmiştir.

**y\_train**: Eğitim verisindeki **etiketleri** içerir (ilk sütun).

**X\_test**: Test verisindeki **özellikleri** içerir. Görsel veriler yine 0 ile 1 arasında normalize edilmiştir.



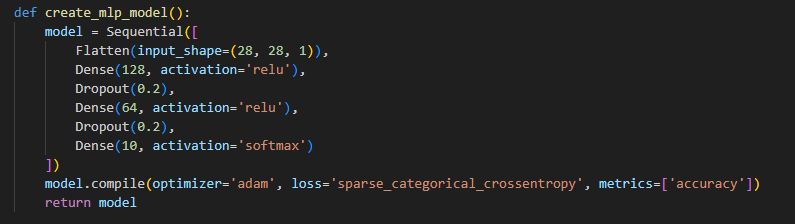
**görüntü verisini derin öğrenme modeline uygun formatta** (4D tensör) hazırlamak için kullanılır. Modelin giriş katmanına **28x28 boyutlarında, tek kanallı (grayscale) görüntüler** gönderilecektir.



**X\_train\_split, y\_train\_split**: Modeli eğitmek için kullanılacak veriler.

**X\_val, y\_val**: Modelin performansını doğrulamak için kullanılacak veriler.

modelin **aşırı öğrenmesini (overfitting)** engellemeye yardımcı olur.



**Flatten(input\_shape=(28, 28, 1))**:

Bu katman, **28x28x1 boyutundaki giriş verisini (görüntüyü)** tek boyutlu bir vektöre dönüştürür.

Görüntüler başlangıçta bir matris olarak gelir (28x28 piksel). Flatten katmanı, bu matrisin her bir elemanını bir vektör halinde sıralar.

**Dense(128, activation='relu')**:

Bu katman, **128 nöronlu** bir tam bağlantılı (fully connected) katmandır.

**ReLU** (Rectified Linear Unit) aktivasyon fonksiyonu kullanılır. ReLU, giriş değeri pozitifse giriş değeri, negatifse sıfır döner. Bu, ağın doğrusal olmayan ilişkileri öğrenmesini sağlar.

**Dropout(0.2)**:

**Dropout** katmanı, eğitim sırasında rastgele **%20** nöronun kapanmasını sağlar. Bu, modelin **aşırı öğrenmesini (overfitting)** engellemeye yardımcı olur ve daha iyi genelleme yapmasını sağlar.

**Dense(64, activation='relu')**:

İkinci bir **tam bağlantılı katman** olup 64 nörondan oluşur ve yine **ReLU** aktivasyon fonksiyonu kullanılır.

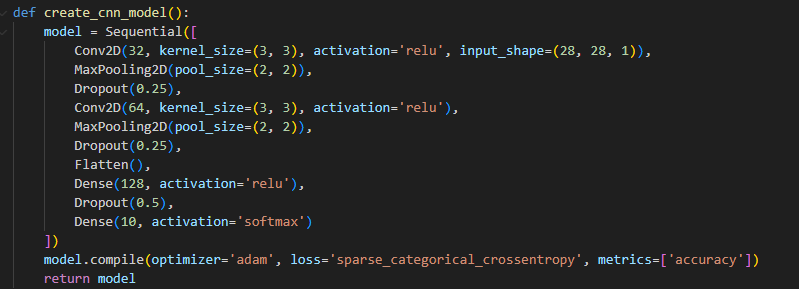
**Dropout(0.2)**:

Bu, yine **%20 dropout** uygulanmasını sağlar. Aşırı öğrenmeyi engellemeye devam eder.

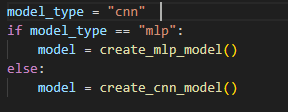
**Dense(10, activation='softmax')**:

Son katman, **10 nöron** içerir çünkü burada **10 sınıflı bir sınıflandırma problemi** vardır (örneğin, MNIST veri setindeki rakamlar gibi).

**Softmax** aktivasyon fonksiyonu kullanılır. Softmax, çıktıları **olasılık dağılımı** şeklinde dönüştürür. Yani, her sınıf için bir olasılık değeri verir ve bu olasılıkların toplamı 1 olur.



Bu fonksiyon, **Convolutional Neural Network (CNN)** modelini oluşturur. CNN, özellikle **görüntü sınıflandırma** gibi görevler için çok etkilidir. Modelde **evrişimli katmanlar (Conv2D)** ve **max pooling katmanları (MaxPooling2D)** yer alır. Ayrıca modelde **dropout** kullanılarak aşırı öğrenme (overfitting) engellenmeye çalışılır.



**odel\_type = "cnn"**:

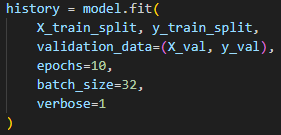
Burada, model tipi bir string değişken olarak belirleniyor. 'cnn' veya 'mlp' olabilir. Bu değişken, hangi tür modelin kullanılacağına karar vermek için kullanılır.

**if model\_type == "mlp":**:

Eğer model\_type değişkeni "mlp" değerine sahipse, create\_mlp\_model() fonksiyonu çağrılır ve MLP (Multi-Layer Perceptron) modelini oluşturur.

**else:**:

Eğer model\_type "mlp" değilse (yani "cnn" olduğu durumda), create\_cnn\_model() fonksiyonu çağrılır ve CNN (Convolutional Neural Network) modelini oluşturur.



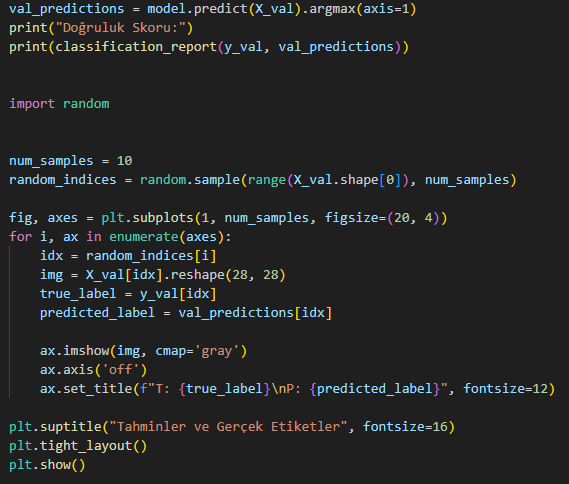
**X\_train\_split, y\_train\_split**: Bu, eğitim için kullanılan giriş verisi ve etiketler. Verileriniz train\_test\_split fonksiyonu ile eğitim ve doğrulama (validation) olarak ayrılmıştı, dolayısıyla X\_train\_split eğitim verilerini ve y\_train\_split ise etiketlerini içeriyor.

**validation\_data=(X\_val, y\_val)**: Eğitim sırasında doğrulama verisini belirtir. Bu veriler modelin her epok sonunda doğrulama kaybı ve doğruluk gibi metrikleri hesaplamasına olanak tanır.

**epochs=10**: Modelin eğitim süresi. Modelin 10 epok boyunca eğitilmesini belirtir. Bir epok, tüm eğitim verisinin bir kez modellenmesi anlamına gelir.

**batch\_size=32**: Eğitim sırasında verilerin kaçlı gruplar halinde işleneceğini belirtir. Bu, her bir "batch" başına modelin parametrelerini güncellemek için kaç örneğin kullanılacağını ifade eder.

**verbose=1**: Eğitim sırasında, eğitim sürecinin çıktı olarak gösterilmesini sağlar. verbose=1, her epokta eğitim ilerlemesini görsel olarak sunar .



val\_predictions = model.predict(X\_val).argmax(axis=1) satırı, modelin doğrulama setindeki tahminlerini alır ve her örnek için tahmin edilen sınıfı seçer.

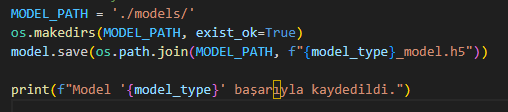
classification\_report(y\_val, val\_predictions) fonksiyonu, gerçek etiketler (y\_val) ve tahmin edilen etiketler (val\_predictions) arasındaki performansı değerlendirir ve doğruluk (accuracy), precision, recall, F1-skora dair metrikleri yazdırır.

random\_indices = random.sample(range(X\_val.shape[0]), num\_samples) ile doğrulama setindeki örneklerden rastgele 10 tanesini seçiyorsunuz.

fig, axes = plt.subplots(1, num\_samples, figsize=(20, 4)): 10 adet örnek için yatayda bir çizim alanı oluşturulur.

ax.imshow(img, cmap='gray'): Her bir örneği gri tonlama (gray) olarak görselleştiriyoruz. X\_val[idx] şekillendirilip görsel olarak çizilir.

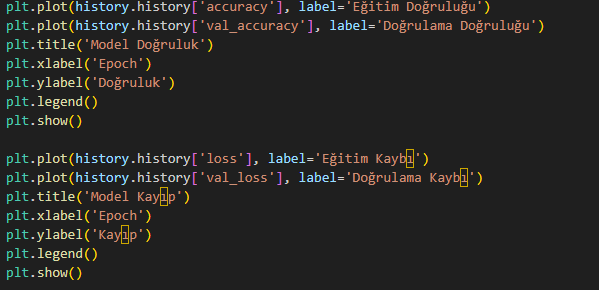
ax.set\_title(f"T: {true\_label}\nP: {predicted\_label}", fontsize=12): Her görselin başlığında "Gerçek Etiket" (T) ve "Tahmin Edilen Etiket" (P) gösterilmektedir.



MODEL\_PATH = './models/' ifadesi, modelin kaydedileceği dizinin yolunu belirler. Eğer bu dizin mevcut değilse, os.makedirs fonksiyonu ile otomatik olarak oluşturulacaktır.

model.save(os.path.join(MODEL\_PATH, f"{model\_type}\_model.h5")): Bu satırda, model.save fonksiyonu kullanılarak, model belirtilen dizine .h5 uzantılı olarak kaydedilir. Burada {model\_type} dinamik bir değişken olup, kullanmakta olduğunuz modelin türünü temsil eder.

print(f"Model {model\_type} başarıyla kaydedildi."): Modelin başarılı bir şekilde kaydedildiği hakkında kullanıcıya bilgi verir.



plt.plot(history.history['accuracy'], label='Eğitim Doğruluğu'): Bu satır eğitim doğruluğunu çiziyor.

plt.plot(history.history['val\_accuracy'], label='Doğrulama Doğruluğu'): Bu satır ise doğrulama doğruluğunu çiziyor.

plt.title('Model Doğruluk'): Grafik başlığı.

plt.xlabel('Epoch'): X ekseninde gösterilecek etiket. Burada 'Epoch' kullanılmıştır çünkü her epoch'ta modelin doğruluğu ölçülür.

plt.ylabel('Doğruluk'): Y ekseninde gösterilecek etiket.

plt.legend(): Grafik üzerindeki etiketleri gösterir.

plt.show(): Grafiği görüntüler.

plt.plot(history.history['loss'], label='Eğitim Kaybı'): Eğitim kaybı grafiği.

plt.plot(history.history['val\_loss'], label='Doğrulama Kaybı'): Doğrulama kaybı grafiği.

plt.title('Model Kayıp'): Grafik başlığı.

plt.xlabel('Epoch'): X eksenini belirler, burada 'Epoch' kullanılmıştır.

plt.ylabel('Kayıp'): Y eksenindeki etiket.

plt.legend(): Grafik üzerindeki etiketleri gösterir.

plt.show(): Grafik görüntülenir.

