Veri setinin bulunduğu dosya yolunu kullanarak pandas modülü ile dosyayı açıyoruz.

```
import pandas as pd
import keras  #keras kütüphanesi
import numpy as np
import re    #regex kütüphanesi

veriseti_yolu = "C:\\Users\\azsar\\OneDrive\\Masaüstü\\birlesik_veri_seti_yorum_yeni.csv"
#kullandığımız verisetinin .csv dosyası

veri = pd.read_csv(veriseti_yolu, sep=";", names=["yorum","deger"], encoding="utf-8")
#verisetini okuyup ";" (yani sütun) ifadesine göre etiketleri yorum ve deger olarak ayırıyor.

veri.head() #veriyi okuyoruz
```

Çıktı olarak veri setini görüyoruz.

Yorum ve değer şeklinde 2 sütun olarak ayrılmış durumda.

Yorum sütunu yorumları, değer sütunu yorumun olumlu veya olumsuz değerini belirliyor.

```
                                               yorum  deger
0  Bu markadan şaştığımda hep pişman oldum ses ka...    1.0
1  Ses kalitesi gayet tatmin edici. Fiyatına göre...    1.0
2  Ürün 2 ayda bozuldu. Hiç beğenmedim aşırı dand...    0.0
3  Kulaklığı takınca dış sesler baya azalıyor ve ...    1.0
4  Kulaklığın üst kısmında içinde metal var bu da...    1.0
```

Veri setinin sütunlarını yorum ve değer olarak ayırıyoruz.

```
yorumlar=veri["yorum"] #yorum sütun
degerler=veri["deger"] #deger sütun olarak ayrı ayrı atadık

yorumlar=yorumlar.str.lower() #bütün metni küçük harflere dönüştürdük

yorumlar.head() #yorumları okur
# degerler.head() #degerleri okur
```

Çıktı olarak sadece yorum sütununu alıyoruz. Bu sayede artık sadece yorum sütununda işlemler yapabiliriz.

```
Out[52]:
0    bu markadan şaştığımda hep pişman oldum ses ka...
1    ses kalitesi gayet tatmin edici. fiyatına göre...
2    ürün 2 ayda bozuldu. hiç beğenmedim aşırı dand...
3    kulaklığı takınca dış sesler baya azalıyor ve ...
4    kulaklığın üst kısmında içinde metal var bu da...
Name: yorum, dtype: object
```

Kullanacağımız metinlerde özel karakterler, boşluklar ve rakamlar olmamalı. Bu yüzden oluşturduğumuz fonksiyonda regex kullanarak metni temizliyoruz.

```
def metni_temizle(metin): #aldığı metini regex modülü ile özel karakterlerden temizleme fonksiyonu
    temiz_metin = re.sub(r'[^a-zA-ZğüşıöçĞÜŞİÖÇ\s]', '', str(metin))
    #regex türkçe karakterler ve normal harfler dışındaki her karakteri siler
    return temiz_metin

yorumlar=yorumlar.apply(metni_temizle) #yorumlar sütununa fonksiyonu uygular

yorumlar.head()
```

Çıktı olarak verideki özel karakterlerin ve rakamların silindiğini görüyoruz.

```
Out[54]:
0    bu markadan şaştığımda hep pişman oldum ses ka...
1    ses kalitesi gayet tatmin edici fiyatına göre ...
2    ürün  ayda bozuldu hiç beğenmedim aşırı dandik...
3    kulaklığı takınca dış sesler baya azalıyor ve ...
4    kulaklığın üst kısmında içinde metal var bu da...
Name: yorum, dtype: object
```

Model eğitebilmek için eğitim ve test verilerinin farklı olması gerekiyor. Bunun için sklearn modülü ile eğitim ve test verilerini ayırıyoruz. Veri setinin %80'i eğitim ve %20'sini test olarak belirliyoruz.

```
import sklearn.model_selection #scikit-learn kütüphanesi

X_egitim=yorumlar #x egitim verisine yorumları atar
y_egitim=degerler #y eğitim verisine degerleri atar

X_egitim,X_test,y_egitim,y_test=sklearn.model_selection.train_test_split(X_egitim,y_egitim,test_size=0.20,random_state=80)
#verisetinin yüzde 80ini x_eğitim'e eğitim için , yüzde 20sini x_teste test için atar ve verileri karıştırır.

print(X_egitim.shape) #verilerin boyutunu görmek için
print(X_test.shape)
print(y_egitim.shape)
print(y_test.shape)
```

Çıktı olarak eğitim ve test verilerinin %80 ve %20 şeklinde ayrıldığını görüyoruz.

Toplam 23.524 adet yorumun 18.819 adeti eğitim için ve 4705 adeti test için kullanılacak.

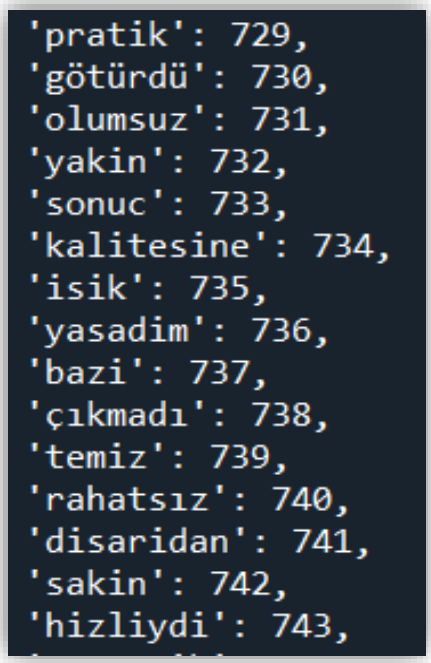
```
X eğitim verisi:  (18819,)
X test verisi  :  (4705,)
Y eğitim verisi:  (18819,)
Y test verisi  :  (4705,)
```

Metinleri işleyebilmemiz için tokenler haline getirmemiz gerekiyor. Bunun için Tokenizer fonksiyonunu kullanarak eğitim verisini tokenize ediyoruz.

```
from tensorflow.keras.preprocessing.text import Tokenizer #tokenizer modülü

tokenizer = Tokenizer() #tokenizer oluşturuyoruz. bu sayede metin tokenler olarak parçalanacak.

tokenizer.fit_on_texts(X_egitim) #x_egitim verisini tokenize eder.

kelime_tokenleri = tokenizer.word_index #tokenizerin parçaladığı x_eğitim tokenlerini gösterir.
kelime_tokenleri_uzunluk = len(kelime_tokenleri) #tokenlerin toplam sayısını verir.
kelime_tokenleri_uzunluk

kelime_tokenleri #tokenleri görmek için
```

Çıktıda görüldüğü gibi eğitim verilerindeki her bir kelimeye karşılık olarak bir sayı atandı.

Bu sayede veri sayılarla daha verimli şekilde işlenebilecek hale geldi.



Tokenler sequences modülü ile diziler haline getiriliyor. Bu sayede bir cümle artık bir token dizisi gibi kullanılabilir hale geliyor.

```python
from keras.preprocessing.sequence import pad_sequences #sequences modülü

X_egitim_dizisi = tokenizer.texts_to_sequences(X_egitim)
#x_eğitim verisne göre oluşturulan tokenlere sayısal değerler atanır. her token benzersiz olur.

X_test_dizisi = tokenizer.texts_to_sequences(X_test)
#aynı şey test verisi için de uygulanıyor

X_egitim_dizisi[10450] #örnek olarak 10450.verinin tokenleşmiş hali

max_uzunluk = max([len(x) for x in X_egitim_dizisi]) #token dizisinin en uzun tokeni
max_uzunluk
```

Çıktıda görüldüğü gibi örnek bir gösterim olarak eğitim verisindeki 10450. yorum token dizisi haline getirildi.

```
Out[126]: [92, 3, 952, 369, 4, 21310]
```

Oluşturulan her bir token dizisinin uzunluğu farklı bu model eğitimi sırasında istenmeyen bir durum. Bunun için token dizilerine en uzun token dizisinin uzunluğu olana kadar 0 eklenip aynı uzunlukta olmaları sağlanır.

```
sirali_x_egitim_dizisi=pad_sequences(X_egitim_dizisi,maxlen=max_uzunluk)
#oluşturulan token dizisinin sequences ile max_uzunluk a göre sıfır eklenerek hepsinin aynı boyutta olması sağlanıyor

sirali_x_test_dizisi=pad_sequences(X_test_dizisi,maxlen=max_uzunluk)
#aynı şey test verisi için de uygulanıyor

sirali_x_egitim_dizisi[10450] #örnek olarak 10450.verinin 0 eklenip tokenleştirilmiş hali
```

Çıktıda görüldüğü gibi 5 kelimeli bir token dizisi artık en uzun token dizisi ile aynı uzunluğa sahip oldu.

```
Out[134]:
array([    0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,    92,     3,   952,   369,     4, 21310])
```

Sinir ağlarının tasarımı kısmında keras modülünü kullanarak LSTM yapay sinir ağı oluşturduk. LSTM geçici olarak belleğine veri kaydedebilen bir ağ bu yüzden başarı oranı daha yüksek.

Embedding katmanında kelimeler gömülür. Giriş değeri olarak token sayısı kadar veri alır. Çıkış değeri ise değişkendir.

Dropout katmanı ise aşırı öğrenmeyi engellemek için belli bir oranda veriyi devre dışı bırakır.

Dense katmanı da sigmoid fonksiyonu kullanır bu sayede 0-1 arasında çıkış elde ederiz.

Optimizasyon Methodu olarak Adam Optimizier kullandık. Öğrenme oranını da 0.001 olarak ayarladık.

```
#sinir ağları bölümü
from keras.layers import Embedding, LSTM, Dense, Dropout #katmanların import edilmesi
from keras import Sequential
from keras.models import Sequential

optimizer_fonksiyonu=keras.optimizers.Adam(learning_rate=0.001)
#kullandığımız optimizasyon yöntemi. Adam optimizeri kullanıyoruz öğrenme oranı 0.001

model = Sequential()
#sequental modeli oluşturduk

model.add(Embedding(input_dim=kelime_tokenleri_uzunluk, output_dim=450))
#giriş olarak Embedding katmanı kullandık giriş sayısı oluştuduğumuz tokenlerin tomplam sayısı. çıkış ise değişken.

model.add(LSTM(150))
#LSTM uzun kısa süreli bellek 2.katman olarak kullandık. giriş parametresi değişken.

model.add(Dropout(0.2))
#aşırı öğrenme olmaması çıkışın %20sini devre dışı bıraktık. (deneysel olarak)

model.add(Dense(1, activation='sigmoid'))
#çıkış katmanı olarak Dense kullandık. sigmoid fonksiyonu ile çıkan 1 değeri 0-1 arasına sıkıştırdık.

model.compile(loss = 'binary_crossentropy', optimizer = optimizer_fonksiyonu, metrics=['accuracy'])
#modelin loss fonksiyonunu binary_crossentropy (ikili), optimizerini Adam, ve doğruluk değeri için accuracy kullandık.
model.summary() #model özeti
```

Modeli eğitmek için eğitim ve test verileri modele girilir. Epochs değeri eğitimin kaç çevrim olacağını ve batch_size değeri çevrim başına kullanılacak veri miktarını temsil eder.

```
#model eğitimi
model.fit(sirali_x_egitim_dizisi, y_egitim, validation_data=(sirali_x_test_dizisi, y_test), epochs=6, batch_size=64)
#fit komutu ile giriş değeri olarak sıralanmış x ve y eğitim verileri giriyor
#doğrulama olarak her turda x ve y test verileri ile eğitime göre doğrulama yapıyor
#epoch değeri kaç çevrim yapacağını belirliyor. (değişken)
#batch_size değeri her çevrimde kullanılacak veri miktarını belirliyor. bellek için önemli.
```

Çıktıda görüldüğü gibi 6 çevrim boyunca model eğitildi ve son validasyon doğruluk oranı %88.47 bulundu. Ortalama bir çevrim 55 saniye sürdü ve çevrimin doğruluk oranı 0.79 dan başlayarak 0.93 e kadar çıktı.

```
Epoch 1/6
295/295 ━━━━━━━━━━━━━━━━━━━━ 60s 191ms/step - accuracy: 0.7916 - loss: 0.3763 - val_accuracy: 0.8894 - val_loss: 0.2081
Epoch 2/6
295/295 ━━━━━━━━━━━━━━━━━━━━ 54s 183ms/step - accuracy: 0.9164 - loss: 0.1347 - val_accuracy: 0.8841 - val_loss: 0.2285
Epoch 3/6
295/295 ━━━━━━━━━━━━━━━━━━━━ 55s 186ms/step - accuracy: 0.9299 - loss: 0.0914 - val_accuracy: 0.8871 - val_loss: 0.2288
Epoch 4/6
295/295 ━━━━━━━━━━━━━━━━━━━━ 54s 184ms/step - accuracy: 0.9332 - loss: 0.0709 - val_accuracy: 0.8862 - val_loss: 0.2544
Epoch 5/6
295/295 ━━━━━━━━━━━━━━━━━━━━ 55s 185ms/step - accuracy: 0.9384 - loss: 0.0650 - val_accuracy: 0.8822 - val_loss: 0.2765
Epoch 6/6
295/295 ━━━━━━━━━━━━━━━━━━━━ 56s 191ms/step - accuracy: 0.9367 - loss: 0.0613 - val_accuracy: 0.8847 - val_loss: 0.3076
```

Oluşturulan fonksiyonda örnek olarak girilen cümleler tokenizasyon yapılarak modelin predict methodu ile tahmin edilir.

```python
def tahmin_yap(cumle): #kendi cümlelerimizle tahminde bulunmak için

    tahmin_cumlesi_tokeni = tokenizer.texts_to_sequences( [cumle] )
    #fonksiyona aldığımız cümleyi parçalayıyoruz (tokenleştiriyoruz)

    sirali_tahmin_cumlesi_tokeni = pad_sequences(tahmin_cumlesi_tokeni , max_uzunluk)
    #tokenleştirdiğimiz cümleyi padding yapıyoruz (0 ekleyerek sıralıyoruz)

    tahmin = model.predict(sirali_tahmin_cumlesi_tokeni)
    #modele tahmin yaptırıyoruz

    #tahmin değerine göre duyguyu sınıflanırıyor
    if tahmin>0 and tahmin<0.2:
        deger="Çok Kötü"
    elif tahmin>0.2 and tahmin<0.5:
        deger="Kötü"
    elif tahmin>0.4 and tahmin<0.6:
        deger="Ortalama"
    elif tahmin>0.6 and tahmin<0.8:
        deger="İyi"
    elif tahmin>0.8 and tahmin<1:
        deger="Çok İyi"

    tahmin = float(tahmin[0]) #tahmin değeri floatlaştırma

    print("Cumle: ", cumle)
    print(f"Tahmin Degeri: {(tahmin):6f}")
    print("Yorum Duygusu: ", deger)
    print("--------------------")

#örnek cümleler
cumle1="beklentimi tam olarak karşılamadı"
cumle2="oldukça iyi bir makine"
cumle3="kulaklığın kablo kalitesi iyi ama ses kalitesi biraz düşük"
cumle4="bu telefon hakkında ne diyeceğimden emin değilim idare eder bence"
cumle5="bu bilgisayarı bedava versen bile tercih etmem"
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 66ms/step
Cumle:  beklentimi tam olarak karşılamadı
Tahmin Degeri: 0.159296
Yorum Duygusu:  Aşırı olumsuz
--------------------
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 37ms/step
Cumle:  oldukça iyi bir makine
Tahmin Degeri: 0.999116
Yorum Duygusu:  Aşırı olumlu
--------------------
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 38ms/step
Cumle:  kulaklığın kablo kalitesi iyi ama ses kalitesi biraz düşük
Tahmin Degeri: 0.442067
Yorum Duygusu:  Olumsuz
--------------------
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 41ms/step
Cumle:  bu telefon hakkında ne diyeceğimden emin değilim idare eder bence
Tahmin Degeri: 0.501833
Yorum Duygusu:  Ortalama
--------------------
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 47ms/step
Cumle:  bu bilgisayarı bedava versen bile tercih etmem
Tahmin Degeri: 0.000739
Yorum Duygusu:  Aşırı olumsuz
```