İMDB FİLM İNCELEMELERİ İLE DUYGU ANALİZİ PROJESİ

1. Muhammet Emin Akyüz-030718106
2. Veysel Hacı Hazar-030718108
3. Soner Karaevli-030716005
4. Yakup Yıldırım-030716034


```
from google.colab import drive
drive.mount('/gdrive')
%cd /gdrive
```

```
Mounted at /gdrive
/gdrive
```

```
!ls 'My Drive/COLABS/'
```

```
bank.csv            İMDB_Derin_Öğrenme_Projesi.ipynb  Test.ipynb
Deneme_001.ipynb    IMDB_Veriseti_v1.csv              veri_seti_3.csv
housepricedata.csv  IMDB_Veriseti_v2.csv
```

Kullanılacak Kütüphanelerimizi Ekleyelim

```
#KÜTÜPHANE BÖLÜMÜ
import matplotlib.pyplot as plt
import seaborn as sns
import tensorflow as tf
from keras.datasets import imdb
from keras.preprocessing.sequence import pad_sequences
from tensorflow.python.keras.preprocessing.sequence import pad_sequences
import numpy as np
import pandas as pd
from tensorflow.python.keras.preprocessing.text import Tokenizer
from tensorflow.python.keras.models  import load_model
from sklearn.model_selection import train_test_split
from tensorflow.python.keras.models import Sequential
from tensorflow.python.keras.layers import Dense, Embedding,LSTM,Dropout
from keras.layers import Dropout
from keras.optimizers import Adam
from keras.layers.recurrent import GRU
import plotly.express as px

from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfTransformer
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense,Activation,Dropout
from tensorflow.keras.callbacks import EarlyStopping
```

Bu Kısımda İMDB Veri Setimizden İlk 15 Verimizin inceleme ve duygu sütunlarını getirdik.

```
#Veri Yükleme
imdb_verisi = pd.read_csv('My Drive/COLABS/IMDB_Veriseti_v1.csv')
imdb_verisi.head(15)
```

	review	sentiment
0	One of the other reviewers has mentioned that ...	positive
1	A wonderful little production. The...	positive
2	I thought this was a wonderful way to spend ti...	positive
3	Basically there's a family where a little boy ...	negative
4	Petter Mattei's "Love in the Time of Money" is...	positive
5	Probably my all-time favorite movie, a story o...	positive
6	I sure would like to see a resurrection of a u...	positive
7	This show was an amazing, fresh & innovative i...	negative
8	Encouraged by the positive comments about this...	negative
9	If you like original gut wrenching laughter yo...	positive
10	Phil the Alien is one of those quirky films wh...	negative
11	I saw this movie when I was about 12 when it c...	negative
12	So im not a big fan of Boll's work but then ag...	negative
13	The cast played Shakespeare. Shakes...	negative
14	This a fantastic movie of three prisoners who	positive

```
#Veri Yükleme_2
imdb_verisi_2 = pd.read_csv('My Drive/COLABS/IMDB_Veriseti_v2.csv',low_memory=False)
imdb_verisi_2.head()
```

	imdb_title_id	title	original_title	year	date_published	genre	duration	country
0	tt0000009	Miss Jerry	Miss Jerry	1894	1894-10-09	Romance	45	USA
1	tt0000574	The Story of the Kelly	The Story of the Kelly Gang	1906	1906-12-26	Biography, Crime, Drama	70	Australia

Aşağıda yazdığımız imdb_verisi.isna() metodu ile NaN(Not a number) isimlendirilen verilerimizin ileride problem olmaması için bunları tespit ediyoruz.NaN değerlerinin olduğu satırlarda sonuç True ,olmayan sütunlarda ise False olarak dönüş yapıyor.

```
imdb_verisi.isna()
```

	review	sentiment
0	False	False
1	False	False
2	False	False
3	False	False
4	False	False
...	...	...
49995	False	False
49996	False	False
49997	False	False
49998	False	False
49999	False	False

50000 rows × 2 columns

```
imdb_verisi_2.isna()
```

	imdb_title_id	title	original_title	year	date_published	genre	duration	country	la
0	False	False	False	False	False	False	False	False	
1	False	False	False	False	False	False	False	False	
2	False	False	False	False	False	False	False	False	
3	False	False	False	False	False	False	False	False	

Print ile 50000 satırdan ve 2 sütundan oluştuğu öğrenildi.

```
print(imdb_verisi.shape)
```

```
(50000, 2)
```

85852	False	False	False	False	False	False	False	False

Print ile 85855 satırdan ve 22 sütundan oluştuğu öğrenildi.

85854	False	False	False	False	False	False	False	False

```
print(imdb_verisi_2.shape)
```

```
(85855, 22)
```

Duygu sayılarına value_counts() metodu ile bu kısımda bakalım.

```
imdb_verisi['sentiment'].value_counts()
```

```
positive    25000
negative    25000
Name: sentiment, dtype: int64
```

Ülke sayılarına value_counts() metodu ile bu kısımda baktık.

```
imdb_verisi_2['country'].value_counts()
```

```
USA                                                28511
India                                               6065
UK                                                  4111
Japan                                               3077
France                                              3055
                                                   ...
Canada, Germany, France, USA                           1
USA, Germany, Hungary, Poland, Bulgaria, Canada        1
Germany, UK, Georgia, France                           1
Ecuador, Mexico                                        1
Italy, France, West Germany, Monaco                    1
Name: country, Length: 4907, dtype: int64
```

Kullanılan dil sayılarına value_counts() metodu ile bu kısımda baktık.

```
imdb_verisi_2['language'].value_counts()
```

```
English                                                                 35939
French                                                                   3903
Spanish                                                                  2831
Japanese                                                                 2826
Italian                                                                  2731
                                                                        ...
```

```
Mandarin, Chinese, Hokkien, English                                          1
Turkish, Azerbaijani, Russian, English, Spanish, Japanese, French, Arabic    1
Italian, Occitan, French                                                     1
English, Romany, Romanian, Ukrainian                                         1
Finnish, German, Russian, Spanish                                            1
Name: language, Length: 4377, dtype: int64
```

Yazarların sayılarına value_counts() metodu ile bu kısımda baktık.

```
imdb_verisi_2['writer'].value_counts()
```

```
Jing Wong                                84
Kuang Ni                                 45
Woody Allen                              40
Erdogan Tünas                            35
Leonardo Benvenuti, Piero De Bernardi    34
                                         ..
David DeCoteau, Rolfe Kanefsky            1
José Luis Garci, María Lejárraga          1
Rory Kindersley, Drew Sherring-Hill       1
Greg Iles, Greg Iles                      1
Jirí Blazek, Jirí Menzel                  1
Name: writer, Length: 66859, dtype: int64
```

imdb_verisi.describe() ile istatistiksel sonuçların hesaplanmasını sağlıyor.

```
imdb_verisi.describe()
```

	review	sentiment
count	50000	50000
unique	49582	2
top	Loved today's show!!! It was a variety and not...	positive
freq	5	25000

imdb_verisi_2.describe() ile istatistiksel sonuçların hesaplanmasını sağlıyor.

```
imdb_verisi_2.describe()
```

	duration	avg_vote	votes	metascore	reviews_from_users	reviews_from_
count	85855.000000	85855.000000	8.585500e+04	13305.000000	78258.000000	74058
mean	100.351418	5.898656	9.493490e+03	55.896881	46.040826	27
std	22.553848	1.234987	5.357436e+04	17.784874	178.511411	58
min	41.000000	1.000000	9.900000e+01	1.000000	1.000000	1
25%	88.000000	5.200000	2.050000e+02	43.000000	4.000000	3
50%	96.000000	6.100000	4.840000e+02	57.000000	9.000000	8
75%	108.000000	6.800000	1.766500e+03	69.000000	27.000000	23
max	808.000000	9.900000	2.278845e+06	100.000000	10472.000000	999

Veri setimizin birçok bilgisine bununla ulaşılır.

```
imdb_verisi.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50000 entries, 0 to 49999
Data columns (total 2 columns):
 #   Column     Non-Null Count  Dtype 
---  ------     --------------  ----- 
 0   review     50000 non-null  object
 1   sentiment  50000 non-null  object
dtypes: object(2)
memory usage: 781.4+ KB
```

Sütun verilerinin bilgisini verir.

```
imdb_verisi.columns
```

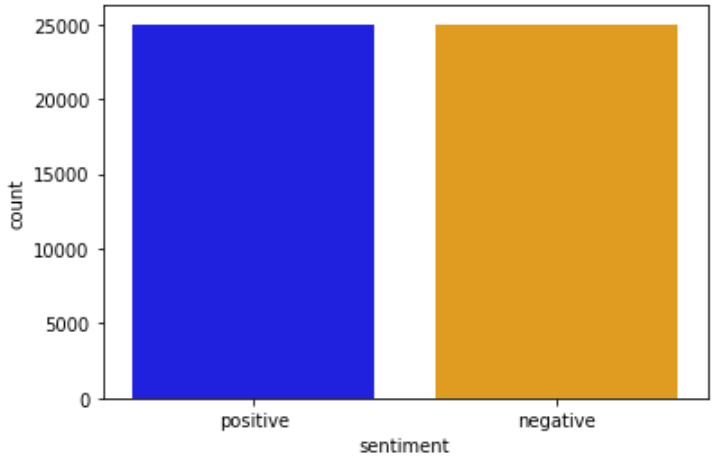
```
Index(['review', 'sentiment'], dtype='object')
```

Sentiment Sütununu Grafik olarak gösterdik.Verilen positive ve negatif değerler birbirine eşit olduğu görülüyor.

```
sns.countplot(imdb_verisi["sentiment"], palette = ["blue","orange"])
plt.show()
print(imdb_verisi.sentiment.value_counts())
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning:

Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional a
```



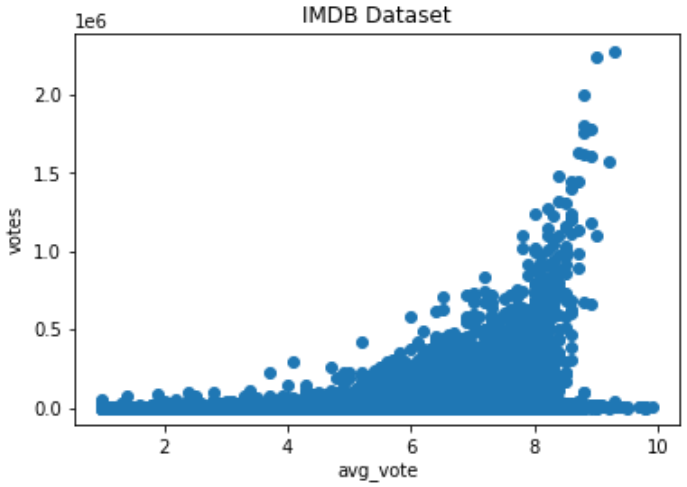

```
positive    25000
negative    25000
Name: sentiment, dtype: int64
```

IMDB 2. veri setimizin votes ve avg_note kısımlarını matplot kütüphanesi ile şekillendirdik.

```
fig, vyes = plt.subplots()
vyes.scatter(imdb_verisi_2['avg_vote'], imdb_verisi_2['votes'])
vyes.set_title('IMDB Dataset')
vyes.set_xlabel('avg_vote')
```

```
vyes.set_xlabel('avg_vote')
vyes.set_ylabel('votes')
```

```
Text(0, 0.5, 'votes')
```



IMDB 2. veri setimizin duration sütun kısmını matplot kütüphanesi ile çektik

Sentimentlerimizdeki positiflere 1 değeri ve negatiflere 0 değerini veriyoruz

```
imdb_verisi.sentiment = [ 1 if each == "positive" else 0 for each in imdb_verisi.sentiment ]
```

```
sentiment = imdb_verisi['sentiment'].values
sentiment
```

```
array([1, 1, 1, ..., 0, 0, 0])
```

```
imdb_verisi = imdb_verisi['review']
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(imdb_verisi,sentiment,test_size=0.25,random_state
```

İngilizce'de en çok kullanılan 18000 kelimeden oluşan bir sözlük oluşturuyoruz.

```
from tensorflow.python.keras.preprocessing.text import Tokenizer
tokenci=Tokenizer(num_words=18000)
tokenci.fit_on_texts(imdb_verisi)
```

Farklı uzunluklarda olan yorumlarımız RNN modelini eğitemez.Bu sebepten dolayıda cümleleri eşit boyutta yapmamız gerekiyor.

```
X_train_Tokens = tokenci.texts_to_sequences(X_train)
X_test_Tokens = tokenci.texts_to_sequences(X_test)
```

Cümlelerimizdeki her bir kelimenin,yukarıda tanımladığımız ve oluşturduğumuz 18000 kelimeden oluşan

```
#Verilerimizdeki her cümlemizin kelime sayımını alıp bir liste oluşturuyoruz.
token_1 = [len(tokens) for tokens in X_train_Tokens + X_test_Tokens]
token_1 = np.array(token_1)


#Token sayıları belirlenirken, ortalama etrafındaki değişiklikler dikkate alınarak sayılar belirlenni
token_2 = np.mean(token_1) + 2 * np.std(token_1)
token_2 = int(token_2)
token_2
```

```
561
```

token_2= Bu değerimiz verilerimizdeki cümlemizin dağılımını ve eğer varsada zıt uzunluklara sahip cümleleri ortalamaya indirmemize sağlayabileyecektir.

```
#Belirlediğimiz bu sayının verilerde yüzde kaçına ait olduğu bakılır.
np.sum(token_1 < token_2) / len(token_1)
```

```
0.94546
```

```
#Verilimiz belirtilen belirteç sayısına göre belirlenir.
X_train_Padd = pad_sequences(X_train_Tokens, maxlen=token_2)
X_test_Padd = pad_sequences(X_test_Tokens, maxlen=token_2)
```

```
X_train_Padd.shape
```

```
(37500, 561)
```

Görüldüğü gibi şekli 561 e ayarladık

```
idgrq= tokenci.word_index
ters_h = dict(zip(idgrq.values(), idgrq.keys()))

def ornek_cumle(tokens):
    kelimeler = [ters_h[token] for token in tokens if token!=0]
    yazı = ' '.join(kelimeler)
    return yazı


print(ornek_cumle(X_train_Padd[8]))
```

```
susan seems to have had a decent career with a few top notch credits under her belt i'm certain
```

```
print(X_train_Padd[8])
```

```
[    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     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     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     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  2815   184     5    25
    66     3   540   628    16     3   171   342  2412   896   464    40
  6031   145   424  1253    59   141    36    11    19    60   184     5
    25    92 12605   145    21    27     4    93     7     7   198   107
   126   113     8   299   368   297    71    10   115     8     1   111
   138    14     9     6   571   186  2815    35     6  3370     5    94
     9     8     1   181     4   207     2     6  1689     5    77    39
    42   231     5    76    46    59    57  7135     1  4494  8117     4
     3   186  1790    35     6   587    41     4    24  1079   122     1
  2503  2037   253    31  3114     7     7   186   439  8196     1   246
   241     8     1    19     8   192    27     4     1   246   113  2801
   198   107     8     3   194    55    54   581   237    21   781  1721
     7     7    10    80   131    12   759    15     1   355     6     3
    49    27     8  4513     1  1077  2503  1994  1818   117     1    19
    13   666   319    18    40   158    16    40   421  3248   154  4015
    53   145    21   247    59    66    12    72  2850   833     5   158
    16     7     7   163   714     2   474    63     3    52   186  1459
   159  1135     2   642     2   376     8     1   538    14     3   920
  9355     7     7    43    22    37  3541   782    22   233   847   140
    11    15     1   738   657  1240     5  9645  7850]
```

Birinci LSTM modelimiz ile derin öğrenme yaptık.

```
#LSTM modelimiz ile öğretelim.
model = Sequential()
model.add(Embedding(input_dim=18000,output_dim=60,input_length=token_2,name='embedding_layer1'))

model.add(LSTM(units=14, return_sequences=True))
model.add(Dropout(0.2))

model.add(LSTM(units=7, return_sequences=True))
model.add(Dropout(0.2))

model.add(LSTM(units=3))
model.add(Dropout(0.2))

model.add(Dense(1, activation='sigmoid'))

model.compile(loss='binary_crossentropy',optimizer=Adam(lr=1e-3),metrics=['accuracy'])
```

```
    /usr/local/lib/python3.7/dist-packages/tensorflow/python/keras/optimizer_v2/optimizer_v2.py:375
```

```
The `lr` argument is deprecated, use `learning_rate` instead.
```

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_layer1 (Embedding) (None, 561, 60)           1080000
_________________________________________________________________
lstm (LSTM)                  (None, 561, 14)           4200
_________________________________________________________________
dropout (Dropout)            (None, 561, 14)           0
_________________________________________________________________
lstm_1 (LSTM)                (None, 561, 7)            616
_________________________________________________________________
dropout_1 (Dropout)          (None, 561, 7)            0
_________________________________________________________________
lstm_2 (LSTM)                (None, 3)                 132
_________________________________________________________________
dropout_2 (Dropout)          (None, 3)                 0
_________________________________________________________________
dense (Dense)                (None, 1)                 4
=================================================================
Total params: 1,084,952
Trainable params: 1,084,952
Non-trainable params: 0
_________________________________________________________________
```

```
history_1 = model.fit(X_train_Padd,Y_train, validation_split=0.28, epochs=6, batch_size=1500, shuffle
```

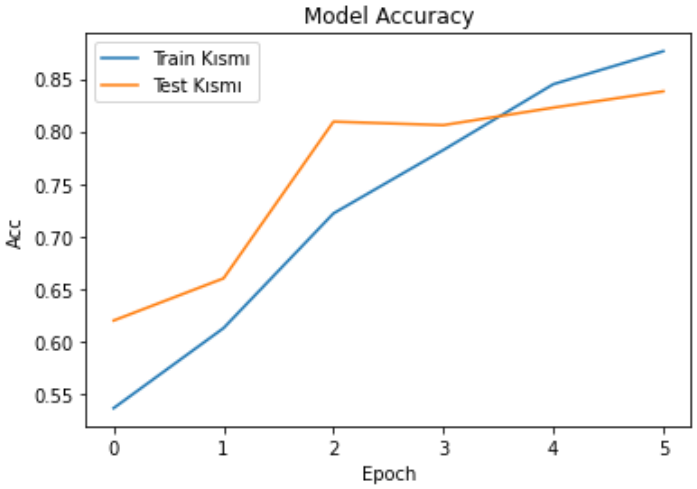
```
Epoch 1/6
18/18 [==============================] - 116s 6s/step - loss: 0.6925 - accuracy: 0.5364 - val_l
Epoch 2/6
18/18 [==============================] - 116s 7s/step - loss: 0.6846 - accuracy: 0.6126 - val_l
Epoch 3/6
18/18 [==============================] - 108s 6s/step - loss: 0.6190 - accuracy: 0.7219 - val_l
Epoch 4/6
18/18 [==============================] - 109s 6s/step - loss: 0.5725 - accuracy: 0.7823 - val_l
Epoch 5/6
18/18 [==============================] - 108s 6s/step - loss: 0.5109 - accuracy: 0.8449 - val_l
Epoch 6/6
18/18 [==============================] - 108s 6s/step - loss: 0.4621 - accuracy: 0.8763 - val_l
```

```
result_1 = model.evaluate(X_test_Padd,Y_test)
```

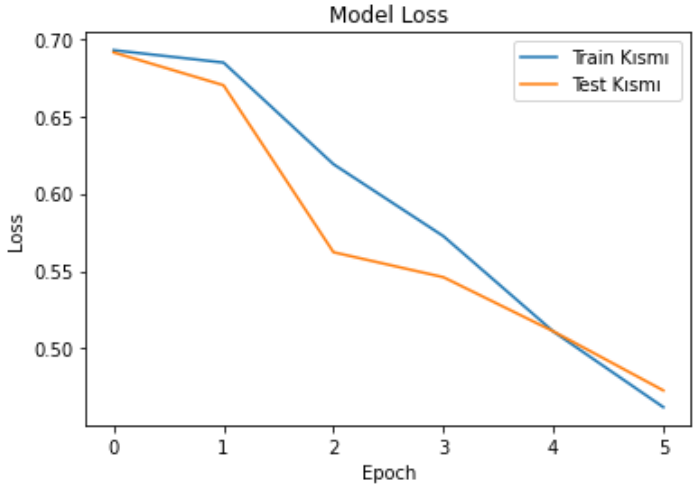
```
391/391 [==============================] - 66s 166ms/step - loss: 0.4755 - accuracy: 0.8333
```

Result_1'de başarı oranımız yüzde 84 yakaladık.

```
plt.figure()
plt.plot(history_1.history["accuracy"], label = "Train Kısmı" )
plt.plot(history_1.history["val_accuracy"], label= "Test Kısmı")
plt.title("Model Accuracy")
plt.ylabel("Acc")
plt.xlabel("Epoch")
plt.legend()
plt.show()
```




```
plt.figure()
plt.plot(history_1.history["loss"], label= "Train Kısmı ")
plt.plot(history_1.history["val_loss"], label= "Test Kısmı")
plt.title("Model Loss")
plt.ylabel("Loss")
plt.xlabel("Epoch")
plt.legend()
plt.show()
```



İkinci LSTM modelimizi deniyoruz

```
model_2=Sequential()

model_2.add(Embedding(input_dim=20000, output_dim=129,input_length=token_2))
model_2.add(LSTM(units=60, activation='tanh'))
model_2.add(Dense(units=1, activation='sigmoid'))

adam= tf.keras.optimizers.Adam(learning_rate=0.0001)

model_2.compile(optimizer=adam, loss='binary_crossentropy', metrics=['accuracy'])


model_2.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
```

```
embedding (Embedding)        (None, 561, 129)          2580000
_________________________________________________________________
lstm_3 (LSTM)                (None, 60)                45600
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 61
=================================================================
Total params: 2,625,661
Trainable params: 2,625,661
Non-trainable params: 0
_________________________________________________________________
```

```
history_2=model_2.fit(X_train_Padd, Y_train, epochs=5,validation_split=0.34, batch_size=128, shuffle=
```

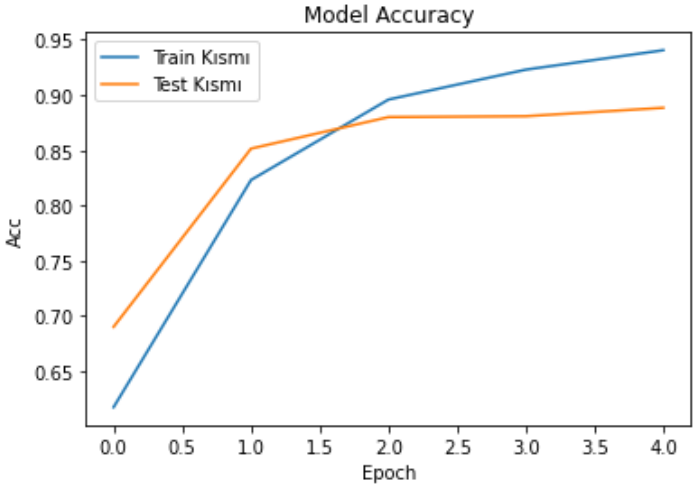
```
Epoch 1/5
194/194 [==============================] - 342s 2s/step - loss: 0.6787 - accuracy: 0.6175 - val
Epoch 2/5
194/194 [==============================] - 341s 2s/step - loss: 0.4442 - accuracy: 0.8231 - val
Epoch 3/5
194/194 [==============================] - 340s 2s/step - loss: 0.2981 - accuracy: 0.8960 - val
Epoch 4/5
194/194 [==============================] - 336s 2s/step - loss: 0.2296 - accuracy: 0.9230 - val
Epoch 5/5
194/194 [==============================] - 334s 2s/step - loss: 0.1839 - accuracy: 0.9406 - val
```

```
result_2 = model_2.evaluate(X_test_Padd,Y_test)
```

```
391/391 [==============================] - 48s 123ms/step - loss: 0.2764 - accuracy: 0.8893
```

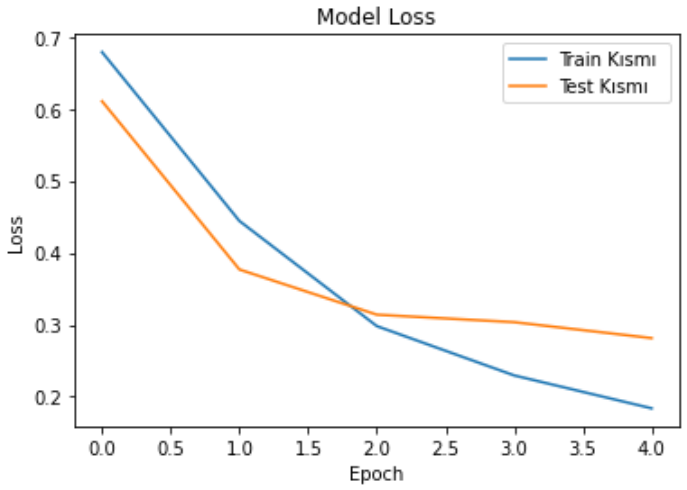
Result_2 sonucumuzda yüzde 89 başarı oranı yakaladık.

```
plt.figure()
plt.plot(history_2.history["accuracy"], label = "Train Kısmı" )
plt.plot(history_2.history["val_accuracy"], label= "Test Kısmı")
plt.title("Model Accuracy")
plt.ylabel("Acc")
plt.xlabel("Epoch")
plt.legend()
plt.show()
```




```
plt.figure()
plt.plot(history_2.history["loss"], label= "Train Kısmı ")
plt.plot(history_2.history["val_loss"], label= "Test Kısmı")
plt.title("Model Loss")
```

```
plt.ylabel("Loss")
plt.xlabel("Epoch")
plt.legend()
plt.show()
```



GRU katmanı ile deneme yaptık.

```
model_3=Sequential()

model_3.add(Embedding(input_dim=20000, output_dim=129,input_length=token_2))
model_3.add(GRU(units=60, activation='tanh'))
model_3.add(Dense(units=1, activation='sigmoid'))

adam= tf.keras.optimizers.Adam(learning_rate=0.0001)

model_3.compile(optimizer=adam, loss='binary_crossentropy', metrics=['accuracy'])
```

```
model_3.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 561, 129)          2580000
_________________________________________________________________
module_wrapper (ModuleWrappe (None, 60)                34200
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 61
=================================================================
Total params: 2,614,261
Trainable params: 2,614,261
Non-trainable params: 0
_________________________________________________________________
```

```
history_3=model_3.fit(X_train_Padd, Y_train, epochs=5,validation_split=0.34, batch_size=128, shuffle=
```

```
Epoch 1/5
194/194 [==============================] - 291s 1s/step - loss: 0.6869 - accuracy: 0.5928 - val
Epoch 2/5
194/194 [==============================] - 289s 1s/step - loss: 0.4668 - accuracy: 0.7950 - val
Epoch 3/5
194/194 [==============================] - 291s 1s/step - loss: 0.2553 - accuracy: 0.9017 - val
Epoch 4/5
194/194 [==============================] - 291s 1s/step - loss: 0.1921 - accuracy: 0.9319 - val
```

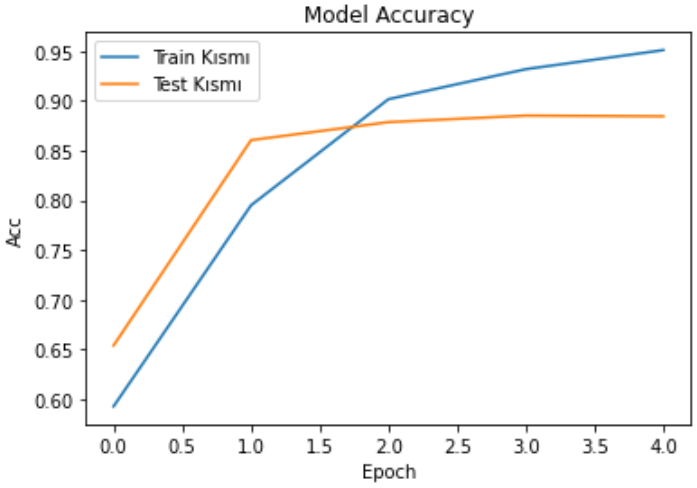
```
Epoch 5/5
194/194 [==============================] - 305s 2s/step - loss: 0.1508 - accuracy: 0.9510 - val
```

```
result_3 = model_3.evaluate(X_test_Padd,Y_test)
```

```
391/391 [==============================] - 40s 103ms/step - loss: 0.2844 - accuracy: 0.8848
```

Result_3 sonucumuzda yüzde 89 başarı oranı yakaladık.

```
plt.figure()
plt.plot(history_3.history["accuracy"], label = "Train Kısmı" )
plt.plot(history_3.history["val_accuracy"], label= "Test Kısmı")
plt.title("Model Accuracy")
plt.ylabel("Acc")
plt.xlabel("Epoch")
plt.legend()
plt.show()
```




```
plt.figure()
plt.plot(history_3.history["loss"], label= "Train Kısmı ")
plt.plot(history_3.history["val_loss"], label= "Test Kısmı")
plt.title("Model Loss")
plt.ylabel("Loss")
plt.xlabel("Epoch")
plt.legend()
plt.show()
```

İkinci veri seti

```
imdb_verisi_2 = pd.read_csv('My Drive/COLABS/IMDB_Veriseti_v2.csv',low_memory=False)
imdb_verisi_2.head(3)
```

	imdb_title_id	title	original_title	year	date_published	genre	duration	country	la
0	tt0000009	Miss Jerry	Miss Jerry	1894	1894-10-09	Romance	45	USA	
1	tt0000574	The Story of the Kelly Gang	The Story of the Kelly Gang	1906	1906-12-26	Biography, Crime, Drama	70	Australia	
2	tt0001892	Den sorte drøm	Den sorte drøm	1911	1911-08-19	Drama	53	Germany, Denmark	

Genre sütunu ile ikinci veri seti görselleştirdik.

```
Visualization = px.pie(values=imdb_verisi_2['genre'].value_counts(),
             names=imdb_verisi_2['genre'].value_counts().index,title='Genre sütununun içeriği')

Visualization.show()
```

Genre sütununun içeriği

```
imdb_verisi_2.nunique()#Her satırda benzersiz içerik bulmak için kullanılır.
```

```
imdb_title_id            85855
title                    82094
original_title           80852
year                       113
date_published           22012
genre                     1257
duration                   266
country                   4907
language                  4377
director                 34733
writer                   66859
production_company       32050
actors                   85729
description              83611
avg_vote                    89
votes                    14933
budget                    4642
usa_gross_income         14857
worlwide_gross_income    30414
metascore                   99
reviews_from_users        1213
reviews_from_critics       595
dtype: int64
```

```
imdb_verisi_2.isnull().sum()#NaN mevcut olan her sütunun sayısını verecektir.
```

```
imdb_title_id                0
title                        0
original_title               0
year                         0
date_published               0
genre                        0
duration                     0
country                     64
language                   833
director                    87
writer                    1572
production_company        4455
actors                      69
description               2115
avg_vote                     0
votes                        0
budget                   62145
usa_gross_income         70529
worlwide_gross_income    54839
metascore                72550
reviews_from_users        7597
reviews_from_critics     11797
dtype: int64
```

```
imdb_verisi_2.country.fillna("Country Unvailable",inplace=True)
```

```
imdb_verisi_2.isnull().sum()
```

```
imdb_title_id                0
title                        0
```

```
original_title              0
year                        0
date_published              0
genre                       0
duration                    0
country                     0
language                  833
director                   87
writer                   1572
production_company       4455
actors                     69
description              2115
avg_vote                    0
votes                       0
budget                  62145
usa_gross_income        70529
worlwide_gross_income   54839
metascore               72550
reviews_from_users       7597
reviews_from_critics    11797
dtype: int64
```

Üçüncü veri seti

```
veri_seti_3 = pd.read_csv('My Drive/COLABS/veri_seti_3.csv')
print(veri_seti_3.shape)
veri_seti_3.head()
```

```
(7787, 12)
```

	show_id	type	title	director	cast	cou
0	s1	TV Show	3%	NaN	João Miguel, Bianca Comparato, Michel Gomes, R...	E
1	s2	Movie	7:19	Jorge Michel Grau	Demián Bichir, Héctor Bonilla, Oscar Serrano, ...	Me
2	s3	Movie	23:59	Gilbert Chan	Tedd Chan, Stella Chung, Henley Hii, Lawrence ...	Singa
3	s4	Movie	9	Shane Acker	Elijah Wood, John C. Reilly, Jennifer Connelly...	U St
4	s5	Movie	21	Robert Luketic	Jim Sturgess, Kevin Spacey, Kate Bosworth, Aar...	U St

```
veri_seti_3.nunique()#Her satırda benzersiz içerik bulmak için kullanılır.
```

```
show_id         7787
type               2
title           7787
director        4049
cast            6831
country          681
date_added      1565
release_year      73
rating            14
duration         216
listed_in        492
description     7769
dtype: int64
```

```
veri_seti_3.isnull().sum()#NaN mevcut olan her sütunun sayısını verecektir.
```

```
show_id            0
type               0
title              0
director        2389
cast             718
country          507
date_added        10
release_year       0
rating             7
duration           0
listed_in          0
description        0
dtype: int64
```

3.veri setimizdeki verileri temizleyeceğiz

Yukarıdaki sonuçlardan; director,cast,country,date_added ve rating sütunlarının eksik değerlere sahip olduğunu görebiliriz.İlk olarak,bu eksik değerleri ele alıyorum.

```
veri_seti_3.director.fillna("No Director",inplace=True)
veri_seti_3.cast.fillna("No Cast",inplace=True)
veri_seti_3.country.fillna("Country Unvailable",inplace=True)
veri_seti_3.dropna(subset=["date_added","rating"],inplace=True)
```

```
veri_seti_3.isnull().sum()
```

```
show_id         0
type            0
title           0
director        0
cast            0
country         0
date_added      0
release_year    0
rating          0
duration        0
listed_in       0
description     0
dtype: int64
```

İkinci veri setimizdeki ilk 5 satırımızdaki title ve ratings sütunlarını listeledik

```
yeni_ratings = pd.DataFrame({'Title':imdb_verisi_2.title,
                             'Rating':imdb_verisi_2.avg_vote})
yeni_ratings.drop_duplicates(subset=['Title','Rating'],inplace=True)

print(yeni_ratings.shape)
yeni_ratings.head(5)
```

(85733, 2)

	Title	Rating
0	Miss Jerry	5.0

```
ınner_verisi = yeni_ratings.merge(veri_seti_3,left_on='Title',right_on='title',how='inner')
ınner_verisi = ınner_verisi.sort_values(by='Rating',ascending=False)

print(ınner_verisi.shape)
ınner_verisi.head(5)
```

(2719, 14)

	Title	Rating	show_id	type	title	director	
987	Breakout	9.0	s1093	TV Show	Breakout	No Director	Jeanette Aw, Elvin Ng, Zhou Christop
976	Innocent	9.0	s3009	TV Show	Innocent	Seren Yüce	Ali Atay, Haluk Bilginer, Nur Sürer,
371	Schindler's List	8.9	s5431	Movie	Schindler's List	Steven Spielberg	Liam Neeson, Ben Kingsley, Fiennes, C
389	Pulp Fiction	8.9	s5003	Movie	Pulp Fiction	Quentin Tarantino	John Travolta, Samuel L. Jackson Thurn
1284	Inception	8.8	s2980	Movie	Inception	Christopher Nolan	Leonardo DiCaprio, Joseph Go Levitt, E

```
Yeni_Data=ınner_verisi[['Title','Rating','type']]

Yeni_Data.drop_duplicates(subset=['Title','Rating','type'],inplace=True)
print(Yeni_Data.shape)
Yeni_Data.head(5)
```

(2719, 3)
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:3: SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/i

	Title	Rating	type
987	Breakout	9.0	TV Show
976	Innocent	9.0	TV Show
371	Schindler's List	8.9	Movie
389	Pulp Fiction	8.9	Movie
1284	Inception	8.8	Movie

```
Filmler_Data = Yeni_Data[Yeni_Data.type == 'Movie']
TVS_Data =  Yeni_Data[Yeni_Data.type == 'TV Show']
print(Filmler_Data.shape)
print(TVS_Data.shape)
```

```
(2383, 3)
(336, 3)
```

```
Filmler_Data = Filmler_Data.drop(['type'], axis = 1)
```

```
Filmler_Data
```

	Title	Rating
371	Schindler's List	8.9
389	Pulp Fiction	8.9
1284	Inception	8.8
1651	Much Ado About Nothing	8.6
598	Koshish	8.6
...	...	...
831	The Vault	1.9
571	Himmatwala	1.7
1188	Pink	1.6
1672	Welcome to New York	1.6
2173	Aerials	1.6

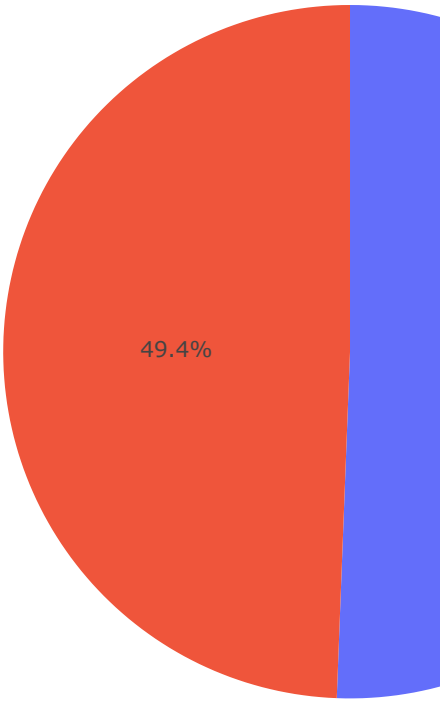
2383 rows × 2 columns

```
Filmler_Data['Polarity_Rating'] = Filmler_Data['Rating'].apply(lambda x:'Positive' if x > 6 else 'Neg
Filmler_Data
```

	Title	Rating	Polarity_Rating
371	Schindler's List	8.9	Positive
389	Pulp Fiction	8.9	Positive
1284	Inception	8.8	Positive
1651	Much Ado About Nothing	8.6	Positive
598	Koshish	8.6	Positive
...	...	...	...
831	The Vault	1.9	Negative
571	Himmatwala	1.7	Negative
1188	Pink	1.6	Negative
1672	Welcome to New York	1.6	Negative
2173	Aerials	1.6	Negative

2383 rows × 3 columns

```
sekil = px.pie(values=Filmler_Data['Polarity_Rating'].value_counts(),
               names=Filmler_Data['Polarity_Rating'].value_counts().index)
sekil.show()
```




```
pozitivs = Filmler_Data[Filmler_Data['Polarity_Rating'] == 'Positive']
negativs = Filmler_Data[Filmler_Data['Polarity_Rating'] == 'Negative']

print(pozitivs.shape)
print(negativs.shape)
```

```
(1206, 3)
(1177, 3)
```

```
ea = Filmler_Data[['Title','Polarity_Rating']]
ea
```

	Title	Polarity_Rating
371	Schindler's List	Positive
389	Pulp Fiction	Positive
1284	Inception	Positive
1651	Much Ado About Nothing	Positive
598	Koshish	Positive
...	...	...
831	The Vault	Negative
571	Himmatwala	Negative
1188	Pink	Negative
1672	Welcome to New York	Negative
2173	Aerials	Negative

2383 rows × 2 columns

```
one_kods = pd.get_dummies(ea["Polarity_Rating"])
```

```
one_kods = pd.get_dummies(ea[ 'Polarity_Rating' ])
ea.drop(['Polarity_Rating'],axis=1,inplace=True)
ea = pd.concat([ea,one_kods],axis=1)
ea
```

```
/usr/local/lib/python3.7/dist-packages/pandas/core/frame.py:4174: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/i
```

	Title	Negative	Positive
371	Schindler's List	0	1
389	Pulp Fiction	0	1
1284	Inception	0	1
1651	Much Ado About Nothing	0	1
598	Koshish	0	1
...	...	...	...
831	The Vault	1	0
571	Himmatwala	1	0
1188	Pink	1	0
1672	Welcome to New York	1	0
2173	Aerials	1	0

2383 rows × 3 columns

```
X= ea['Title'].values
y= ea.drop('Title',axis=1).values
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.10,random_state=42)
```

```
X_train
```

```
array(['Green Room', 'Kate & Leopold', 'Zoom', ..., 'Leap Year',
       'The Breaker Upperers', 'Romeo Akbar Walter'], dtype=object)
```

```
y_train
```

```
array([[0, 1],
       [0, 1],
       [0, 1],
       ...,
       [0, 1],
       [1, 0],
       [0, 1]], dtype=uint8)
```

```
vctr = CountVectorizer()
X_train = vctr.fit_transform(X_train)
X_test = vctr.transform(X_test)

sss = TfidfTransformer()
X_train =sss.fit_transform(X_train)
X_test = sss.transform(X_test)
X_train = X_train.toarray()
X_test = X_test.toarray()
```

```
X_test = X_test.toarray()


model_4 = Sequential()

model_4.add(Dense(units=12673,activation ='relu'))
model_4.add(Dropout(0.2))

model_4.add(Dense(units=4000,activation ='relu'))
model_4.add(Dropout(0.2))

model_4.add(Dense(units=500,activation ='relu'))
model_4.add(Dropout(0.2))

model_4.add(Dense(units=2,activation ='sigmoid'))

opwt=tf.keras.optimizers.Adam(learning_rate=0.001)
model_4.compile(loss='binary_crossentropy',optimizer=opwt,metrics=['binary_accuracy'])

erken_durs = EarlyStopping(monitor='val_loss',mode='min',verbose=1,patience=4)



wxy=model_4.fit(x=X_train, y=y_train, batch_size=50, epochs=80,validation_data=(X_test,y_test),verbos
wxy
```

```
Epoch 1/80
43/43 [==============================] - 52s 1s/step - loss: 0.6938 - binary_accuracy: 0.5196 -
Epoch 2/80
43/43 [==============================] - 40s 930ms/step - loss: 0.4208 - binary_accuracy: 0.816
Epoch 3/80
43/43 [==============================] - 40s 927ms/step - loss: 0.2460 - binary_accuracy: 0.892
Epoch 4/80
43/43 [==============================] - 40s 924ms/step - loss: 0.1695 - binary_accuracy: 0.897
Epoch 5/80
43/43 [==============================] - 40s 925ms/step - loss: 0.1402 - binary_accuracy: 0.902
Epoch 00005: early stopping
<tensorflow.python.keras.callbacks.History at 0x7f7a83f0be90>
```

```
result_4 = model_4.evaluate(X_test,y_test,batch_size=64,verbose=1)
print('Test Accuracy Başarı Oranı:',result_4[1])
```

```
4/4 [==============================] - 1s 249ms/step - loss: 1.1309 - binary_accuracy: 0.5418
Test Accuracy Başarı Oranı: 0.5418410301208496
```

```
cıktı = pd.DataFrame(wxy.history)

cıktı.loc[1:, ['loss','val_loss']].plot()
cıktı.loc[1:, ['binary_accuracy','val_binary_accuracy']].plot()

print(("En iyi Validation Loss: {:0.4f}"+\
       "\n En iyi Validation Accuracy: {:0.4f}")\
       .format(cıktı['val_loss'].min(),
               cıktı['val_binary_accuracy'].max()))
```

```
En iyi Validation Loss: 0.7140
 En iyi Validation Accuracy: 0.5418
```