Sosyal Medyada Bulunan Kötü Yorumların Sınıflandırılması

Ömer Faruk DEMİRCİ

01

Kullanılan Veri Seti

Veri Setinin İçeriği

Kullandığımız veri seti kullanıcılardan alınan rastgele twitteryorumlarından oluşuyor ve etiketlenmiş 5 adet sütun içeriyor.

```
In [3]: import pandas as pd

        df = pd.read_csv('Veri Seti.csv')
```

```
In [4]: df
```

Out[4]:

	COMMENT	INSULT	PROFANITY	SEXIST	RACIST	OTHER
0	Phil Spector bir lanet katil olduğunu Biliyoru...	1	0	0	0	0
1	Lan siktirin gidin AMK pozitif sik kafaları Gi...	0	1	0	0	0
2	Zpac Olmak İstiyorum Ja Rule bir Tupac Shakur ...	0	0	0	0	1
3	Ne yapılması gerekiyor Aradan sonra bu sayfaya...	0	0	0	0	1
4	Kabul Dream Chaser programı ile ilgili olmayan...	0	0	0	0	1
...	...	...	...	...	...	...
77795	İsmi kısaltmak için her kutu kendi başına bir ...	0	0	0	0	1
77796	ecek acak 31 marta kadar incelenir gorusulur d...	0	0	0	0	1
77797	Oh adamın Sen sadece yumuşak bir inegin Birkaç...	1	0	0	0	0
77798	Bu sayıf sitayi temizleyecegim hyyayayayayayay...	1	0	0	0	0
77799	Amanda aman putin 2036 ya kadar sen saltatha çık...	1	0	0	0	0

77800 rows × 6 columns

```
In [ ]:
```

Veri setindeki tüm değerlerin tam olduğunun kontrolü

In [3]:
```
import pandas as pd

df = pd.read_csv('Veri Seti.csv')
```

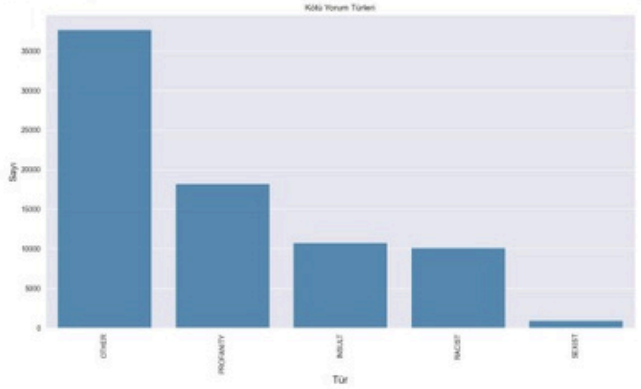
In [7]:
```
df.isnull().sum()
```

Out[7]:
```
COMMENT      0
INSULT       0
PROFANITY    0
SEXIST       0
RACIST       0
OTHER        0
dtype: int64
```

In [ ]:

Verinin Görselleştirilmesi

```
In [23]: import pandas as pd
         import seaborn as sns
         import matplotlib.pyplot as plt

         df = pd.read_csv('Veri Seti.csv')

In [27]: sentencetype_graph = df.iloc[:,1:].sum()

In [28]: sentencetype_graph

Out[28]: INSULT       10777
         PROFANITY    18252
         SEXIST         945
         RACIST       10163
         OTHER        37663
         dtype: int64

In [35]: plt.figure(figsize=(15, 8))
         temp = sns.barplot(x=ls.index, y=ls.values, alpha=0.8)
         plt.title('Kötü Yorum Türleri')
         plt.ylabel('Sayı', fontsize=14)
         plt.xlabel('Tür', fontsize=15)

         ticks = range(len(ls.index))
         temp.set_xticks(ticks)
         temp.set_xticklabels(ls.index, rotation=90, fontsize=10)

         plt.show()
```




```
In [ ]:

In [ ]:
```

02

Veri Üzerinde Yapılan İşlemler

Gerekli kütüphanelerin içe aktarılması ve veri setinin okunması

In [23]:
```
import os
import pandas as pd
import tensorflow as tf
import numpy as np
from matplotlib import pyplot as plt

df = pd.read_csv('Veri Seti.csv')
```

Verinin Hazırlanması

Veri setindeki yorumları ve etiketleri ayırdık:

In [23]:
```
from tensorflow.keras.layers import TextVectorization

X = df['COMMENT']
y = df[df.columns[1:]].values
```

X: Yorumların olduğu sütun.

y: Yorumlara ait etiketlerin olduğu sütunlar.

In [ ]:

Önişleme fonksiyonunun ve metin vektörleştirmekatmanının tanımlanıp verinin uyarlanması

In [23]:
```
MAX_FEATURES = 200000


def custom_standardization(input_text):
    input_text = tf.strings.lower(input_text)
    input_text = tf.strings.regex_replace(input_text, '[^a-zA-Z0-9 ]', '')
    return input_text


vectorizer = TextVectorization(max_tokens=MAX_FEATURES,
                               output_sequence_length=1800,
                               output_mode='int',
                               standardize=custom_standardization)

vectorizer.adapt(X.values)
vectorized_text = vectorizer(X.values)
```

In [7]:
```
vectorized_text
```

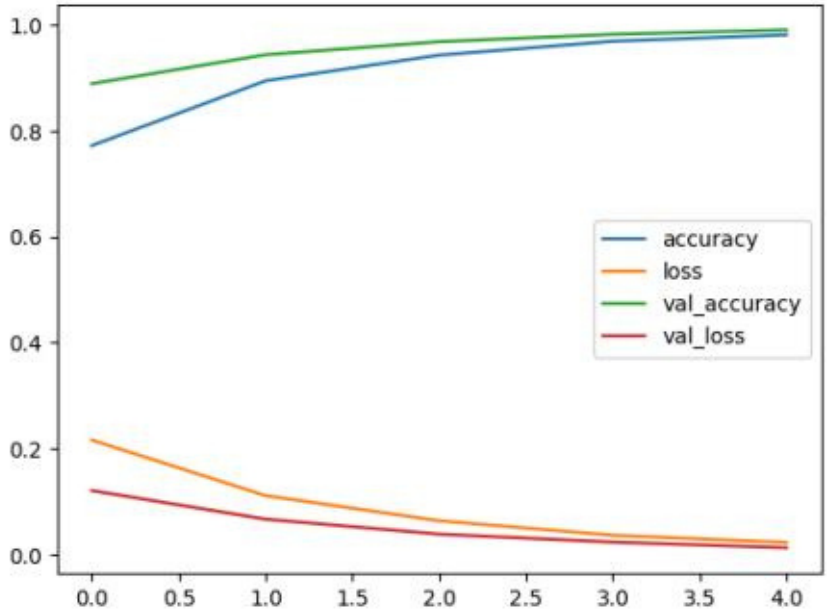
Out[7]:
```
<tf.Tensor: shape=(77800, 1800), dtype=int64, numpy=
array([[  7395, 103694,      2, ...,      0,      0,      0],
       [   216,   2534,    620, ...,      0,      0,      0],
       [ 34309,    138,  23617, ...,      0,      0,      0],
       ...,
       [   524,   2435,     28, ...,      0,      0,      0],
       [     4,   1651,   3396, ...,      0,      0,      0],
       [ 23038,   2327,   2355, ...,      0,      0,      0]], dtype=int64)>
```

Tensorflowveri kümesinin oluşturulması

In [ ]:
```
dataset = tf.data.Dataset.from_tensor_slices((vectorized_text, y))
dataset = dataset.cache()
dataset = dataset.shuffle(160000)
dataset = dataset.batch(16)
dataset = dataset.prefetch(8)

train = dataset.take(int(len(dataset) * 0.7))
val = dataset.skip(int(len(dataset) * 0.7)).take(int(len(dataset) * 0.2))
test = dataset.skip(int(len(dataset) * 0.9)).take(int(len(dataset) * 0.1))
```

CNN kullanılarak yapılan model eğitimi ve sonuçları

In [23]:
```python
from tensorflow.keras.layers import Conv1D, GlobalMaxPooling1D

# Modeli tanımla
cnn_model = Sequential()
cnn_model.add(Embedding(MAX_FEATURES + 1, 32))
cnn_model.add(Conv1D(128, 5, activation='relu'))
cnn_model.add(GlobalMaxPooling1D())
cnn_model.add(Dense(128, activation='relu'))
cnn_model.add(Dense(256, activation='relu'))
cnn_model.add(Dense(128, activation='relu'))
cnn_model.add(Dense(5, activation='sigmoid'))

# Modeli derle
cnn_model.compile(optimizer='adam',
                  loss='binary_crossentropy',
                  metrics=['accuracy'])


# Modeli eğit
cnn_history = cnn_model.fit(train, epochs=5, validation_data=val)

# Eğitim geçmişini görselleştir
plt.figure(figsize=(8, 5))
pd.DataFrame(cnn_history.history).plot()
plt.show()

# Test seti üzerinde modeli değerlendir
cnn_results = cnn_model.evaluate(test)
```

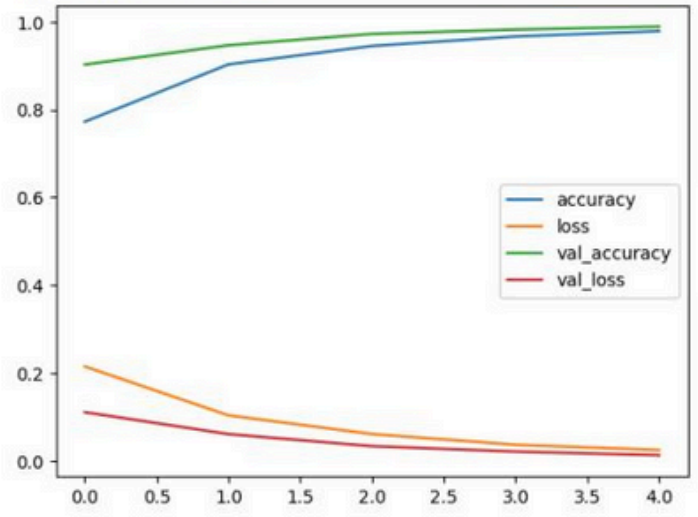



```
486/486 ━━━━━━━━━━━━━━━━━━━━ 2s 4ms/step - accuracy: 0.9899 - loss: 0.0126
Test Loss: 0.012930726632475853
Test Accuracy: 0.9903549551963806
```

LSTM kullanılarak yapılan model eğitimi ve sonuçları

In [ ]:
```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dropout, Bidirectional, Dense, Embeddi

model = Sequential()
model.add(Embedding(MAX_FEATURES + 1, 32))
model.add(Bidirectional(LSTM(32, activation='tanh')))
model.add(Dense(128, activation='relu'))
model.add(Dense(256, activation='relu'))
model.add(Dense(128, activation='relu'))
model.add(Dense(5, activation='sigmoid'))

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

history = model.fit(train, epochs=5, validation_data=val)

plt.figure(figsize=(8, 5))
pd.DataFrame(history.history).plot()
plt.show()
```

```
Epoch 5/5
3404/3404 ━━━━━━━━━━━━━━━━━━━━ 1068s 314ms/step - accuracy: 0.9797 - loss: 0.0238
- val_accuracy: 0.9894 - val_loss: 0.0131
<Figure size 800x500 with 0 Axes>
```



Lojistik

Regresyon

kullanılarak sonuçları yapılan model

eğitimi ve

In [25]:

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.multioutput import MultiOutputClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, precision_score, recall_score
import pandas as pd

df = pd.read_csv('Veri Seti.csv')

X = df['COMMENT']
y = df[df.columns[1:]]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_

vectorizer = TfidfVectorizer(max_features=20000)
X_train_vec = vectorizer.fit_transform(X_train)
X_test_vec = vectorizer.transform(X_test)

model = MultiOutputClassifier(LogisticRegression(max_iter=1000))

model.fit(X_train_vec, y_train)

y_pred = model.predict(X_test_vec)

accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='micro')
recall = recall_score(y_test, y_pred, average='micro')

print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
```

```
Accuracy: 0.7366323907455012
Precision: 0.8841117250775868
Recall: 0.7506426735218509
```

Kerasile modelin performansının değerlendirilmesi

In [ ]:
```
from tensorflow.keras.metrics import Precision, Recall, CategoricalAccuracy

pre = Precision()
re = Recall()
acc = CategoricalAccuracy()

for batch in test.as_numpy_iterator():
    X_true, y_true = batch
    yhat = model.predict(X_true)

    y_true = y_true.flatten()
    yhat = yhat.flatten()

    pre.update_state(y_true, yhat)
    re.update_state(y_true, yhat)
    acc.update_state(y_true, yhat)

print(f'Precision: {pre.result().numpy()}, Recall: {re.result().numpy()}, Accura
```

GirilenYorumların Duyarlılık Analizi ve GradioKullanıcı Arayüzüyle Tahmini

In [ ]:

```
def score_comment(comment):
    comment = custom_preprocess(comment)
    vectorized_comment = vectorizer([comment])
    results = model.predict(vectorized_comment)

    text = ''
    for idx, col in enumerate(df.columns[1:]):
        text += '{}: {}\n'.format(col, results[0][idx] > 0.5)

    return text

interface = gr.Interface(fn=score_comment,
                         inputs="text",
                         outputs="text")

interface.launch(share=True)
```

comment

Bu satıcıya salak demek az kalır! Bu kalitesiz çöpü bana sattığı için gerçekten salak olduğunu düşünüyorum. Bir daha asla bu satıcıdan bir şey almam sadece kandırmaya çalışıyorlar!

Clear | Submit

output

INSULT: True
PROFANITY: False
SEXIST: False
RACIST: False
OTHER: False

Flag

comment

Bu ürünü aldığıma gerçekten memnunum! Kalitesi beni şaşırttı ve fiyatı da gayet makul. Kesinlikle tavsiye ederim.

Clear | Submit

output

INSULT: False
PROFANITY: False
SEXIST: False
RACIST: False
OTHER: True

Flag

comment

Kız başına sokağa çıkma

Clear | Submit

output

INSULT: False
PROFANITY: False
SEXIST: True
RACIST: False
OTHER: False

Flag

comment

Senin ne işin var lan burada

Clear | Submit

output

INSULT: False
PROFANITY: True
SEXIST: False
RACIST: False
OTHER: False

Flag

comment

Bugün sokakta yürürken, birinin zenci bir adama yardım ettiğini gördüm.

Clear | Submit

output

INSULT: False
PROFANITY: False
SEXIST: False
RACIST: True
OTHER: False

Flag