Nick Johnson CS 4395.001 Text Classification

Find a text classification data set that interests you

https://www.kaggle.com/datasets/tirendazacademy/fifa-world-cup-2022-tweets

Describe the data set and what the model should be able to predict

I've chosen a dataset containing tweets that are about the 2022 FIFA World Cup. The model should be able to predict whether each tweet has a neutral, positive, or negative sentiment.

Create a graph showing the distribution of the target classes

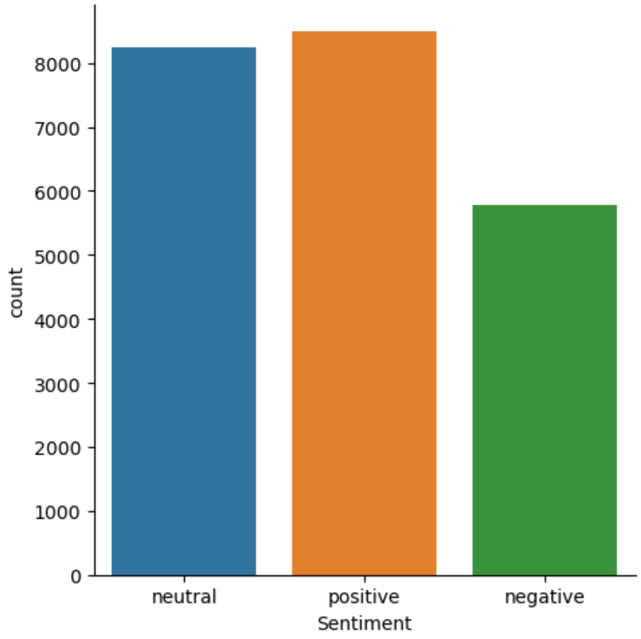
```
#Category Plot of target class

#showing how many neutral, positive, and negative tweets we have

import seaborn as sb

sb.catplot(x="Sentiment", kind="count", data=df)
```

```
<seaborn.axisgrid.FacetGrid at 0x7f5376cee810>
```



Naive Bayes

```
vectorizer = TfidfVectorizer(stop_words=stopwords, binary=True)

# define X and y
X = vectorizer.fit_transform(df.Tweet)
y = df.Sentiment

#divide into train/test
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, train_size=0.8, random_state=1234)


#choosing the BernoulliNB here
from sklearn.naive_bayes import BernoulliNB

bernoulli = BernoulliNB()
bernoulli.fit(X_train, y_train)
```

```
    BernoulliNB()
```

```
# make predictions on the test data
pred = bernoulli.predict(X_test)

# print confusion matrix
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test, pred)
```

```
    array([[ 811,  237,  123],
           [ 236, 1033,  376],
           [ 102,  293, 1294]])
```

```
#evaluating on the test data
print('accuracy score: ', accuracy_score(y_test, pred))
print('precision score: ', precision_score(y_test, pred, average='micro'))
print('recall score: ', recall_score(y_test, pred, average='micro'))
print('f1 score: ', f1_score(y_test, pred, average='micro'))
```

```
    accuracy score:  0.6965593784683685
    precision score:  0.6965593784683685
    recall score:  0.6965593784683685
    f1 score:  0.6965593784683685
```

Logistic Regression

```
# imports needed to do Logistic Regression

import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, log_loss


#read in the csv
df = pd.read_csv('/kaggle/input/fifa-world-cup-2022-tweets/fifa_world_cup_2022_tweets.csv', header=0, usecols=[4,5], encoding='lat

# define X and y
X = df.Tweet
y = df.Sentiment

# divide into train/test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, train_size=0.8, random_state=1234)

# vectorizer
vectorizer = TfidfVectorizer(binary=True)
X_train = vectorizer.fit_transform(X_train)  # fit and transform the train data
X_test = vectorizer.transform(X_test)        # transform the test data

#train on the train data
classifier = LogisticRegression(solver='lbfgs', max_iter=400, class_weight='balanced')
classifier.fit(X_train, y_train)

# evaluate on the test data
pred = classifier.predict(X_test)
print('accuracy score: ', accuracy_score(y_test, pred))
print('precision score: ', precision_score(y_test, pred, average='micro'))
print('recall score: ', recall_score(y_test, pred, average='micro'))
print('f1 score: ', f1_score(y_test, pred, average='micro'))
probs = classifier.predict_proba(X_test)
print('log loss: ', log_loss(y_test, probs))
```

```
    accuracy score:  0.7420643729189789
    precision score:  0.7420643729189789
    recall score:  0.7420643729189789
    f1 score:  0.7420643729189789
    log loss:  0.6202404241029336
```

Neural Networks

```
#importing pandas
import pandas as pd
#reading in the csv
df = pd.read_csv('/kaggle/input/fifa-world-cup-2022-tweets/fifa_world_cup_2022_tweets.csv', header=0, usecols=[4,5], encoding='lat
#printing number of rows and columns in our dataframe
print('rows and columns:', df.shape)
#show the head of our dataframe
print(df.head())
```

```
rows and columns: (22524, 2)
                                               Tweet Sentiment
0  What are we drinking today @TucanTribe \n@MadB...   neutral
1  Amazing @CanadaSoccerEN  #WorldCup2022 launch ...  positive
2  Worth reading while watching #WorldCup2022 htt...  positive
3  Golden Maknae shinning bright\n\nhttps://t.co/...  positive
4  If the BBC cares so much about human rights, h...  negative
```

```
# text preprocessing
from nltk.corpus import stopwords
from sklearn.feature_extraction.text import TfidfVectorizer

stopwords = set(stopwords.words('english'))
vectorizer = TfidfVectorizer(stop_words=stopwords, binary=True)
```

```
# define X and y
X = vectorizer.fit_transform(df.Tweet)
y = df.Sentiment
```

```
# divide into train/test
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, train_size=0.8, random_state=1234)
```

```
#training on the train data

#I modified the max_iter and hidden_layer_sizes until I stopped getting an error
#about reaching iteration limit

from sklearn.neural_network import MLPClassifier
classifier = MLPClassifier(solver='lbfgs', alpha=1e-5, max_iter = 1000,
                   hidden_layer_sizes=(30, 2), random_state=1)
classifier.fit(X_train, y_train)
```

```
MLPClassifier(alpha=1e-05, hidden_layer_sizes=(30, 2), max_iter=1000,
              random_state=1, solver='lbfgs')
```

```
#evaluating on the test data
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score, recall_score, f1_score
pred = classifier.predict(X_test)
print('accuracy score: ', accuracy_score(y_test, pred))
print('precision score: ', precision_score(y_test, pred, average='micro'))
print('recall score: ', recall_score(y_test, pred, average='micro'))
print('f1 score: ', f1_score(y_test, pred, average='micro'))
```

```
accuracy score:  0.3749167591564928
precision score:  0.3749167591564928
recall score:  0.3749167591564928
f1 score:  0.3749167591564928
```

Write up your analysis of the performance of various approaches

The best performing of the three for my chosen dataset was the Logistic Regression. Second was Naive Bayes and third was Neural Networks. The Naive Bayes scores were fairly close to the Logistic Regression scores. The Neural Network scores were significantly lower than both of the other two algorithms. Perhaps if I gain more experience with using sklearn and using these different algorithms I could achieve higher accuracy scores, but for now these are the scores I have.