Text Classification using Machine Learning

Course: CS4395 Human Language Technologies

Name: Ashwin Somasundaram

NETID: AXS180183

```
import pandas as pd
```

```
df = pd.read_csv('https://raw.githubusercontent.com/ashwin-som/cs4372/main/finSentimentData.csv')
```

```
df
```

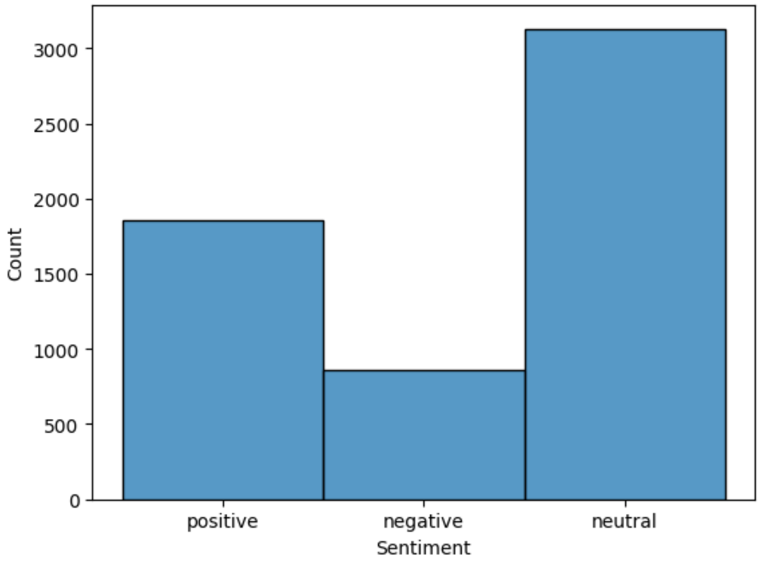
	Sentence	Sentiment
0	The GeoSolutions technology will leverage Bene...	positive
1	$ESI on lows, down $1.50 to $2.50 BK a real po...	negative
2	For the last quarter of 2010 , Componenta 's n...	positive
3	According to the Finnish-Russian Chamber of Co...	neutral
4	The Swedish buyout firm has sold its remaining...	neutral
...	...	...
5837	RISING costs have forced packaging producer Hu...	negative
5838	Nordic Walking was first used as a summer trai...	neutral
5839	According shipping company Viking Line , the E...	neutral
5840	In the building and home improvement trade , s...	neutral
5841	HELSINKI AFX - KCI Konecranes said it has won ...	positive

5842 rows × 2 columns

```
import seaborn as sns
import matplotlib.pyplot as plt
```

Distribution of Target Classes

```
sns.histplot(data=df,x='Sentiment')
plt.show()
```



```
x = df.Sentence
y = df.Sentiment
```

```
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)
```

```
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
from sklearn.feature_extraction.text import TfidfVectorizer
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
```

```
vectorizer = TfidfVectorizer()
```

```
x_train = vectorizer.fit_transform(x_train)
x_test = vectorizer.transform(x_test)
```

```
x_train.shape
```

```
(4673, 10161)
```

```
x_test.shape
```

```
(1169, 10161)
```

Naive Bayes Classifier

```
from sklearn.naive_bayes import MultinomialNB

nb_classifier = MultinomialNB()
```

```
nb_classifier.fit(x_train,y_train)
```

```
▸ MultinomialNB
```

```
nb_pred = nb_classifier.predict(x_test)
```

```
from sklearn.metrics import accuracy_score
```

```
print('accuracy score: ', accuracy_score(y_test, nb_pred))
```

```
accuracy score:  0.6398631308810949
```

```
from sklearn.metrics import classification_report
print(classification_report(y_test, nb_pred))
```

```
              precision    recall  f1-score   support

    negative       1.00      0.01      0.02       191
     neutral       0.63      0.98      0.77       620
    positive       0.70      0.38      0.49       358

    accuracy                           0.64      1169
   macro avg       0.77      0.46      0.43      1169
weighted avg       0.71      0.64      0.56      1169
```

Logistic Regression Classifier

```
from sklearn.linear_model import LogisticRegression
```

```
lr_classifier = LogisticRegression(multi_class='multinomial', solver='lbfgs',class_weight='balanced')
```

```
lr_classifier.fit(x_train,y_train)
```

```
▾                        LogisticRegression
LogisticRegression(class_weight='balanced', multi_class='multinomial')
```

```
lr_pred = lr_classifier.predict(x_test)
```

```
print('accuracy score: ', accuracy_score(y_test, lr_pred))
```

```
accuracy score:  0.6971770744225834
```

```
print(classification_report(y_test, lr_pred))
```

```
              precision    recall  f1-score   support

    negative       0.41      0.49      0.44       191
     neutral       0.78      0.74      0.76       620
    positive       0.74      0.73      0.73       358

    accuracy                           0.70      1169
   macro avg       0.64      0.65      0.65      1169
weighted avg       0.71      0.70      0.70      1169
```

Neural Network Classifier

```
from sklearn.neural_network import MLPClassifier
```

```
nn_classifier = MLPClassifier(solver='lbfgs', alpha=1e-5, hidden_layer_sizes=(15, 7), random_state=1)
```

```
nn_classifier.fit(x_train,y_train)
```

```
     /usr/local/lib/python3.9/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:541: ConvergenceWarning: lbfgs failec
     STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
```

```
nn_pred = nn_classifier.predict(x_test)
```

https://scikit-learn.org/stable/modules/preprocessing.html

```
print('accuracy score: ', accuracy_score(y_test, nn_pred))
```

```
     accuracy score:  0.6595380667236954
```

```
print(classification_report(y_test, nn_pred))
```

```
                   precision    recall  f1-score   support

         negative       0.27      0.23      0.24       191
          neutral       0.71      0.74      0.73       620
         positive       0.74      0.76      0.75       358

         accuracy                           0.66      1169
        macro avg       0.57      0.57      0.57      1169
     weighted avg       0.65      0.66      0.65      1169
```

Report

Data Set Summary:

There are a total of 5482 items in this dataset. Each item in the dataset consists of a sentence and its corresponding sentiment. The sentiment can be one of three categories: negative, neutral, and positive. Negative means that the sentence is a negative statement, positive means that the sentence is a positive statement, and neutral means that the sentence is not emotionally charged in any manner.

Goal of all 3 machine learning models:

After training, each model should be able to predict with a reasonable accuracy, the sentiment of a particular financial statement. For example, a statement such as "Bank XYZ has lost $7 million dollars in its lawsuit, and thus faces harsh setbacks in the financial market" would be expected to be classified as 'negative'.

Preprocessing

In terms of preprocessing, I will be dividing the dataset into a train and test split, with 20% of the data being in the test split. After splitting, the tranining set of sentences, will also be vectorized in order to enable the learning models to train on them. This will enable us to test our model once it has been trained, and thus scan for accuracy with respect to the actual results which might be expected.

Models Used

For the Naive Bayes, I utilized the MultinomialNB Classifier, and specified that this is a multiclass dataset for the Logistic Regresion and Neural Network Classifiers. For logistic Regression, I specified that each target class is balanced. I took the hidden layers for the Neural Network from the tutorial provided by Professor Mazidi in here github repository to maintain consistency and for the sake of being able to observe the performance of the model better.

Observations:

- The higest accuracy (~70%) was obtained by the Logistic Regression Classifier. This is followed by the MultiLayer Perceptron Neural Network Classifier and the Naive Bayes Classifier respectively.
- One potential reason for the neural network classifier not taking first place is the fact that it might have overfit to the training data and thus performs poorly against the test data. Upon changing the train test split, the neural network does produce marginally better results.
- It makes sense that the Naive bayes algorithm has a lower accuracy in comparision with logistic regression. This can be explained by the low bias and high variance in this dataset. This enables the Logistic Regression algorithm to be more accurate.
- While the neural network classifier, does indeed have better accuracy in comparision ot the Naive Bayes classifier, the time taken by the naive bayes classifier for running is a lot lesser and the difference in accuracy is extremely marginal. Thus, in a real world use case scenario, I would probably go with the NB Classifier.
- For each of the learning models, I printed the classification report which shows many other important details about the performance of each model such as the recal and precision for each respective target class. As expected, all three models did relatively well in classifying

the neutral sentences (high precision). This behavior is expected as the number of neutral statements far outweigh the number of positive and negative statements combined. The Neural Network performed poorly in classifying negative statements correctly, whereas the naive bayes performed exceptionally well in classifying the negative statements correctly.

- In terms of recall, the logistic regression classifier is more balanced than the other two models, and mostly has a recall above .50 which is highly desirable. The naive bayes classifier is the exact opposite of this in the fact that its recall is incredibly extreme, 0.02 for negatie and .98 for neutral statements.

Conclusion

In conclusion, after weighing the pros and cons of all three learning models, and their performances, I would choose the Logistics Regression Classifier as my final choice as a model to utilize in the real world. It is fast, accurate, and adequately balanced out between all performance metrics that are vital to the function that I am looking for, which is quick and accurate predictions.

✓ 0s completed at 2:28 PM ● ×