Text Classification

Alekhya Pinnamaneni

The data set used in this notebook contains 957 emails and their corresponding labels. Each email is either labeled/classified as spam or non-spam. The models created in this notebook should be able to predict whether each email in the test set is spam or non-spam.

In [1]:
```
# Read data into a pandas dataframe
import pandas as pd
df = pd.read_csv('emails.csv', encoding='latin-1', usecols=[1, 2])
```

In [2]:
```
# Clean up data columns
df.columns = ['text', 'spam']
df.spam = df.spam.astype('category').cat.codes
```

In [4]:
```
# Pre-processing
import nltk
from nltk.corpus import stopwords
nltk.download('stopwords')
from sklearn.feature_extraction.text import TfidfVectorizer
stopwords = list(stopwords.words('english'))
vectorizer = TfidfVectorizer(stop_words=stopwords)
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

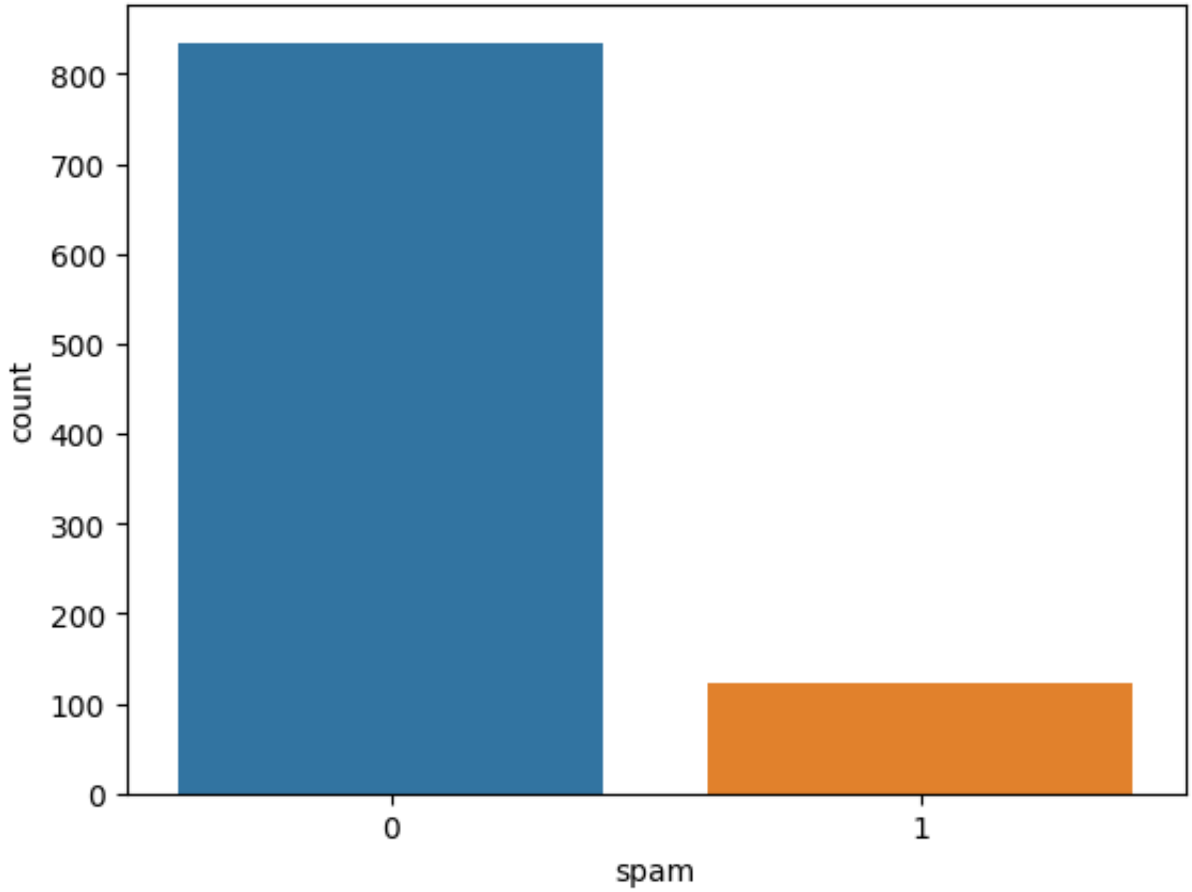
In [5]:
```
# Set up X and Y
X = df.text
y = df.spam
```

In [6]:
```
# Split data into train and test sets
import sklearn
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, tra
```

In [7]:
```
# Vectorize the text
X_train = vectorizer.fit_transform(X_train)
X_test = vectorizer.transform(X_test)
```

In [8]:
```
# Plot distribution of target classes
import seaborn as sb
sb.countplot(data=df, x='spam')
```

Out[8]: <Axes: xlabel='spam', ylabel='count'>



Naive Bayes

In [10]:
```
# Train the naive bayes model
from sklearn.naive_bayes import MultinomialNB
naive_bayes = MultinomialNB()
naive_bayes.fit(X_train, y_train)
```

Out[10]:
```
▾ MultinomialNB
MultinomialNB()
```

In [14]:
```
# Predict on the test data
from sklearn.metrics import accuracy_score, confusion_matrix, classification
pred = naive_bayes.predict(X_test)
```

In [17]:
```
# Print the classification metrics
print('accuracy score: ', accuracy_score(y_test, pred))
print('\nconfusion matrix: ', confusion_matrix(y_test, pred))
print('\n', classification_report(y_test, pred))
print('\nf1 score: ', f1_score(y_test, pred))
```

```
accuracy score:  0.9114583333333334

confusion matrix:  [[165   0]
 [ 17  10]]

              precision    recall  f1-score   support

           0       0.91      1.00      0.95       165
           1       1.00      0.37      0.54        27

    accuracy                           0.91       192
   macro avg       0.95      0.69      0.75       192
weighted avg       0.92      0.91      0.89       192


f1 score:  0.5405405405405406
```

Logistic Regression

In [18]:
```
# Train the logistic regression model
from sklearn.linear_model import LogisticRegression
log_reg = LogisticRegression(solver='lbfgs', class_weight='balanced')
log_reg.fit(X_train, y_train)
```

Out[18]:
```
▼ LogisticRegression
LogisticRegression(class_weight='balanced')
```

In [21]:
```
# Predict on the test data
from sklearn.metrics import accuracy_score, confusion_matrix, classification
pred2 = log_reg.predict(X_test)
```

In [22]:
```
# Print the classification metrics
print('accuracy score: ', accuracy_score(y_test, pred2))
print('\nconfusion matrix: ', confusion_matrix(y_test, pred2))
print('\n', classification_report(y_test, pred2))
print('\nf1 score: ', f1_score(y_test, pred2))
```

```
accuracy score:  0.9635416666666666

confusion matrix:  [[160   5]
 [  2  25]]

              precision    recall  f1-score   support

           0       0.99      0.97      0.98       165
           1       0.83      0.93      0.88        27

    accuracy                           0.96       192
   macro avg       0.91      0.95      0.93       192
weighted avg       0.97      0.96      0.96       192


f1 score:  0.8771929824561403
```

Neural Networks

In [47]:

```
# Train the neural network model
from sklearn.neural_network import MLPClassifier
nn = MLPClassifier(solver='lbfgs', alpha=1e-5,
                   hidden_layer_sizes=(15, 2), random_state=1)
nn.fit(X_train, y_train)
```

Out[47]:

```
▾ MLPClassifier
MLPClassifier(alpha=1e-05, hidden_layer_sizes=(15, 2), random_state
=1,
              solver='lbfgs')
```

In [48]:

```
# Predict on the test data
from sklearn.metrics import accuracy_score, confusion_matrix, classification
pred3 = nn.predict(X_test)
```

In [49]:

```
# Print the classification metrics
print('accuracy score: ', accuracy_score(y_test, pred3))
print('\nconfusion matrix: ', confusion_matrix(y_test, pred3))
print('\n', classification_report(y_test, pred3))
print('\nf1 score: ', f1_score(y_test, pred3))
```

```
accuracy score:  0.9791666666666666

confusion matrix:  [[163   2]
 [  2  25]]

              precision    recall  f1-score   support

           0       0.99      0.99      0.99       165
           1       0.93      0.93      0.93        27

    accuracy                           0.98       192
   macro avg       0.96      0.96      0.96       192
weighted avg       0.98      0.98      0.98       192


f1 score:  0.9259259259259259
```

Analysis

Out of the three models, the neural network model had the most accurate results on the test data. The naive Bayes model had the least accurate results, and the logistic regression model had the second most accurate results. A pattern was found in the results across all three models. All three models had more accurate results on the non-spam emails than on the spam emails.