Text Classification

for this assignmnt, I used the link https://www.kaggle.com/datasets/mfaisalqureshi/spam-email. this data contains ham and spam emails. the model should be able to predict if a given text is real or spam.

```
import pandas as pd
import io
from sklearn.model_selection import train_test_split

# Load the CSV file into a pandas dataframe
from google.colab import files

uploaded = files.upload()
```

```
 df = pd.read_csv(io.BytesIO(uploaded['spam.csv']))
```

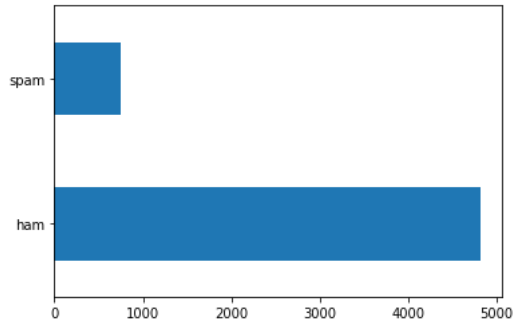
```
print(df)
```

```
     Category                                            Message
0         ham  Go until jurong point, crazy.. Available only ...
1         ham                      Ok lar... Joking wif u oni...
2        spam  Free entry in 2 a wkly comp to win FA Cup fina...
3         ham  U dun say so early hor... U c already then say...
4         ham  Nah I don't think he goes to usf, he lives aro...
...       ...                                                ...
5567     spam  This is the 2nd time we have tried 2 contact u...
5568      ham              Will ü b going to esplanade fr home?
5569      ham  Pity, * was in mood for that. So...any other s...
5570      ham  The guy did some bitching but I acted like i'd...
5571      ham                        Rofl. Its true to its name

[5572 rows x 2 columns]
```

```
#Graph target distribution
df['Category'].value_counts().plot(kind = 'barh')
```

```
<Axes: >
```




```
#divide into training and testing data
import numpy as np
np.random.seed(80085)
i = np.random.rand(len(df)) < 0.8
train = df[i]
test = df[~i]
```

```
#Print the train shape and test shape
print('Train Shape: ', train.shape)
print('Test Shape: ', test.shape)
```

```
Train Shape:  (4441, 2)
Test Shape:  (1131, 2)
```

```
# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(df['Message'], df['Category'], test_size=0.2, random_state=42)
```

Naïve Bayes classifier

```
from sklearn.feature extraction.text import CountVectorizer
```

```
# Create a CountVectorizer object to convert the messages into a matrix of token counts
vectorizer = CountVectorizer()


# Fit the vectorizer to the training data and transform the training and testing sets
X_train_counts = vectorizer.fit_transform(X_train)
X_test_counts = vectorizer.transform(X_test)


from sklearn.naive_bayes import MultinomialNB


# Create a Naïve Bayes classifier object and fit it to the training data
clf = MultinomialNB()
clf.fit(X_train_counts, y_train)


from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

# Make predictions on the testing data using the trained classifier
y_pred = clf.predict(X_test_counts)

# Calculate the accuracy, precision, recall, and F1-score of the classifier
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, pos_label='spam')
recall = recall_score(y_test, y_pred, pos_label='spam')
f1 = f1_score(y_test, y_pred, pos_label='spam')


# Print the evaluation metrics
print('Accuracy:', accuracy)
print('Precision:', precision)
print('Recall:', recall)
print('F1-score:', f1)
```

```
Accuracy: 0.9919282511210762
Precision: 1.0
Recall: 0.9395973154362416
F1-score: 0.9688581314878892
```

Logistic Regression

```
from sklearn.linear_model import LogisticRegression

# Create a Logistic Regression classifier object and fit it to the training data
clf = LogisticRegression()
clf.fit(X_train_counts, y_train)


from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

# Make predictions on the testing data using the trained classifier
y_pred = clf.predict(X_test_counts)

# Calculate the accuracy, precision, recall, and F1-score of the classifier
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, pos_label='spam')
recall = recall_score(y_test, y_pred, pos_label='spam')
f1 = f1_score(y_test, y_pred, pos_label='spam')

# Print the evaluation metrics
print('Accuracy:', accuracy)
print('Precision:', precision)
print('Recall:', recall)
print('F1-score:', f1)
```

```
Accuracy: 0.9865470852017937
Precision: 1.0
Recall: 0.8993288590604027
F1-score: 0.9469964664310955
```

Neural Networks using sklearn to predict

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

```
# Create a MLPClassifier object with a single hidden layer of 10 neurons and the ReLU activation function
clf = MLPClassifier(hidden_layer_sizes=(10,), activation='relu', max_iter=50, random_state=42)

# Train the classifier on the training data
clf.fit(X_train_counts, y_train)


from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

# Make predictions on the testing data using the trained classifier
y_pred = clf.predict(X_test_counts)

# Calculate the accuracy, precision, recall, and F1-score of the classifier
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, pos_label='spam')
recall = recall_score(y_test, y_pred, pos_label='spam')
f1 = f1_score(y_test, y_pred, pos_label='spam')

# Print the evaluation metrics
print('Accuracy:', accuracy)
print('Precision:', precision)
print('Recall:', recall)
print('F1-score:', f1)
```

```
Accuracy: 0.9883408071748879
Precision: 1.0
Recall: 0.912751677852349
F1-score: 0.9543859649122807
```

Analysis

From the evaluation metrics of the three models, we can conclude that all three models performed very well in predicting whether a message is spam or not. The Naive Bayes model achieved the highest accuracy of 99.19%, indicating that it correctly classified the majority of the messages in the dataset. It also achieved a perfect precision of 100%, meaning that it didn't misclassify any non-spam messages as spam. However, the recall of the Naive Bayes model was lower than that of the Logistic Regression and neural network classifiers, indicating that it may have missed some spam messages in the dataset.

The Logistic Regression model achieved a slightly lower accuracy of 98.65% compared to the Naive Bayes model, but a perfect precision of 100% like the Naive Bayes model. However, the recall of the Logistic Regression model was also slightly lower than the neural network classifier, indicating that it may have missed some spam messages in the dataset as well.

The neural network classifier achieved an accuracy of 98.83%, slightly higher than the Logistic Regression model, and also achieved perfect precision like the other models. It had a good recall and F1-score, slightly higher than the Logistic Regression model, indicating that it was able to correctly classify most of the spam messages in the dataset while still maintaining a low rate of false positives.

Overall, all three models achieved very high performance in predicting whether a message is spam or not, with slight differences in their strengths and weaknesses. The choice of which model to use would depend on the specific needs and requirements of the task at hand, as well as the available resources and constraints.

✓ 0s completed at 5:57 PM