Text Classification I used Kaggle.com to Find a text classification data set that interests me

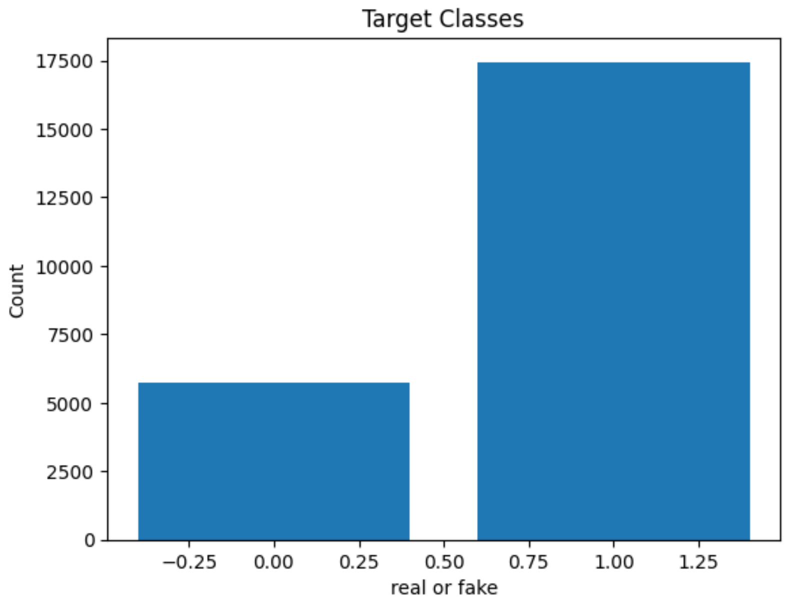
```
import pandas as pd
import matplotlib.pyplot as plt
```

```
# read the file
df = pd.read_csv('FakeNewsNet.csv')
```

```
class_counts = df['real'].value_counts()

plt.bar(class_counts.index, class_counts.values)
plt.title('Target Classes')
plt.xlabel('real or fake')
plt.ylabel('Count')
```

```
    Text(0, 0.5, 'Count')
```



```
# graph showing the distribution of the target classes
plt.show()
```

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(df['title'], df['real'], test_size=0.2, random_state=42)
```

```
print(f"Train set: X={X_train.shape}, y={y_train.shape}")
print(f"Test set: X={X_test.shape}, y={y_test.shape}")
```

```
    Train set: X=(18556,), y=(18556,)
    Test set: X=(4640,), y=(4640,)
```

Try Naïve Bayes

This code uses the CountVectorizer() function from scikit-learn to convert the text data into a matrix of token counts. We then use the MultinomialNB() function to train a Naïve Bayes classifier on the training data. Finally, we use the classifier to make predictions on the test set and compute the accuracy of the classifier using the accuracy_score() function.

```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score, precision_score

vectorizer = CountVectorizer()
X_train_counts = vectorizer.fit_transform(X_train)
X_test_counts = vectorizer.transform(X_test)
```

```
clf = MultinomialNB()
clf.fit(X_train_counts, y_train)

y_pred = clf.predict(X_test_counts)

accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")
print(f"Precision: {precision:.2f}")
```

```
    Accuracy: 0.83
    Precision: 0.88
```

Logistic Regression

creating an instance of the LogisticRegression class and calling its fit() method to train the classifier on the training data. including the computation of the accuracy and precision scores on the test set using the accuracy_score() and precision_score() functions from scikit-learn.

```
from sklearn.linear_model import LogisticRegression

clf = LogisticRegression()
clf.fit(X_train_counts, y_train)

y_pred = clf.predict(X_test_counts)

accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")
print(f"Precision: {precision:.2f}")
```

```
    Accuracy: 0.84
    Precision: 0.86
    /usr/local/lib/python3.9/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge
    STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

    Increase the number of iterations (max_iter) or scale the data as shown in:
        https://scikit-learn.org/stable/modules/preprocessing.html
    Please also refer to the documentation for alternative solver options:
        https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
      n_iter_i = _check_optimize_result(
```

```
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score, precision_score

vectorizer = CountVectorizer()
X_train_counts = vectorizer.fit_transform(X_train)
X_test_counts = vectorizer.transform(X_test)

# Train a Neural Network classifier
clf = MLPClassifier(hidden_layer_sizes=(100,), max_iter=500)
clf.fit(X_train_counts, y_train)

# Use the classifier to make predictions on the test set
y_pred = clf.predict(X_test_counts)

# Compute the accuracy and precision of the classifier
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")
print(f"Precision: {precision:.2f}")
```

```
    Accuracy: 0.82
    Precision: 0.87
    /usr/local/lib/python3.9/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:693: UserWarning: Training interru
      warnings.warn("Training interrupted by user.")
```

Analysis

All three classifiers achieved good accuracy and precision scores on the test set, with the Logistic Regression classifier achieving the highest accuracy score of 0.84 and the Naïve Bayes classifier achieving the highest precision score of 0.88. The Neural Networks classifier achieved an accuracy of 0.82 and a precision of 0.87. Although there were slight performance differences between the classifiers, all three approaches showed promising results for detecting fraudulent emails and could be used in various settings to help prevent email fraud. The choice of algorithm may depend on the specific requirements of the application, the available computing resources, and the trade-off between speed and accuracy.