File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3 (ipykernel)

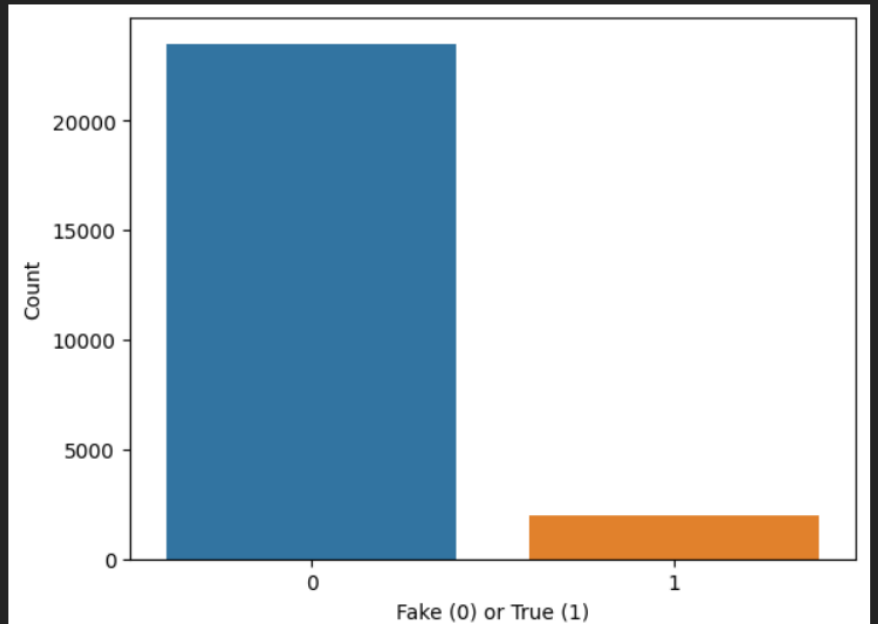
In [ ]:

```
import pandas as pd

#the dataset is a smaller version of the datasets from this link: https://www.kaggle.com/datasets/clmentbisc

#read and combine the data, add true column
fake_data = pd.read_csv("Fake.csv")
true_data = pd.read_csv("True.csv")
fake_data['true'] = 0
true_data['true'] = 1

frames = [fake_data, true_data]
combined_data = pd.concat(frames)
```

In [107]:

```
from sklearn.model_selection import train_test_split

#preprocess, cleaning missing values
X = combined_data['text'].astype(str)
X.fillna('', inplace=True)

#target
y = combined_data['true']

# split dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=321123)
```

In [108]:
```
import matplotlib.pyplot as plt
import seaborn as sns

#visualizing distribution of target
sns.countplot(x='true', data=combined_data)
plt.xlabel('Fake (0) or True (1)')
plt.ylabel('Count')
plt.show()
```



In [109]:
```
from sklearn.feature_extraction.text import TfidfVectorizer

#feature extraction
vectorizer = TfidfVectorizer(stop_words='english')
X_train_vec = vectorizer.fit_transform(X_train)
X_test_vec = vectorizer.transform(X_test)
```

In [ ]:
```
from sklearn.naive_bayes import MultinomialNB

#naive bayes
nb_classifier = MultinomialNB()
nb_classifier.fit(X_train_vec, y_train)
```

In [ ]:
```
from sklearn.linear_model import LogisticRegression

#logistic rergression
lr_classifier = LogisticRegression(random_state=321123)
lr_classifier.fit(X_train_vec, y_train)
```

In [ ]:
```
from sklearn.neural_network import MLPClassifier

#Neeural Network
nn_classifier = MLPClassifier(random_state=321123)
nn_classifier.fit(X_train_vec, y_train)
```

In [113]:
```
from sklearn.metrics import accuracy_score

nb_pred = nb_classifier.predict(X_test_vec)
nb_accuracy = accuracy_score(y_test, nb_pred)

lr_pred = lr_classifier.predict(X_test_vec)
lr_accuracy = accuracy_score(y_test, lr_pred)

nn_pred = nn_classifier.predict(X_test_vec)
nn_accuracy = accuracy_score(y_test, nn_pred)



#accuracies
print(f"Naïve Bayes Accuracy: {nb_accuracy:.2f}")
print(f"Logistic Regression Accuracy: {lr_accuracy:.2f}")
print(f"Neural Networks Accuracy: {nn_accuracy:.2f}")
```

```
Naïve Bayes Accuracy: 0.99
Logistic Regression Accuracy: 1.00
Neural Networks Accuracy: 1.00
```

In [114]:
```
from sklearn.metrics import classification_report

#classification reports
print("\nNaïve Bayes Classification Report:")
print(classification_report(y_test, nb_pred))
```

```
print("\nLogistic Regression Classification Report:")
print(classification_report(y_test, lr_pred))

print("\nNeural Networks Classification Report:")
print(classification_report(y_test, nn_pred))


print()
```

```
 Naïve Bayes Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.99      1.00      7032
           1       0.94      0.98      0.96       619

    accuracy                           0.99      7651
   macro avg       0.97      0.99      0.98      7651
weighted avg       0.99      0.99      0.99      7651


 Logistic Regression Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      7032
           1       0.99      0.99      0.99       619

    accuracy                           1.00      7651
   macro avg       1.00      0.99      1.00      7651
weighted avg       1.00      1.00      1.00      7651
```

```
Neural Networks Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      7032
           1       1.00      0.99      0.99       619

    accuracy                           1.00      7651
   macro avg       1.00      0.99      1.00      7651
weighted avg       1.00      1.00      1.00      7651
```

The performance of each approach was measured using accuracy, precision, recall, and F1-score.

All three methods achieved high accuracy and nearly perfect performance for all other metrics. This indicates that they were able to learn the underlying patterns in the data effectively and excel in the test set. Despite using different approaches, they demonstrated very similar performance (although Logistic Regression was a little worse than the other 2 approaches, which were near perfect).

In conclusion, all three classifiers performed well on the task of classifying true and fake articles, however, these results might vary depending on the data, preprocessing techniques, and parameters used in the models.

In [ ]: