Imports

```
from google.colab import drive
from wordcloud import WordCloud
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score, precision_score, recall_
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, precision_score, recall_
from sklearn.neural_network import MLPClassifier

import seaborn as sns
import pandas as pd
import plotly.express as px
import matplotlib.pyplot as plt
```

Data Load

Mount Google Drive

```
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call
```

Load News Data

```
df=pd.read_csv("/content/drive/MyDrive/NLP/news_articles.csv")
```

Data Visualization

Display Data

```
1 df.shape
```

```
(2096, 12)
```

```
1 df.sample(frac=1)
```

	author	published	title	text	language	
1309	Andy Green	2016-10-27T23:05:00.000+03:00	the sugar industry funding research to sugarco...	back pain pain in the side \nit is important t...	english	natura
1815	David G. Brown	2016-11-22T23:03:06.646+02:00	stockholms feminist snow removal program cause...	home badge abuse leaked soros memo reveals p...	english	returno
797	EdJenner	2016-11-22T23:37:16.124+02:00	democrat electors try to persuade gop electors...	on the kelly file monday actor tim allen discu...	english	dai
2009	Bob Unruh	2016-10-26T23:07:28.355+03:00	see dems accept foreign cash to disrupt trump ...	print saeed toosi right and ayatollah khamenei...	english	
695	Corbett	2016-11-07T19:09:15.225+02:00	interview larken rose on the immorality of v...	corbettreportcom november \nin douglas adams...	english	corbett
...	...	...	...	...	...	
694	Corbett	2016-11-06T21:38:49.637+02:00	the fbis october surprise what youre not being	podcast play in new window download embed \n...	english	corbett

			not being...			
194	Brandon Turbeville	2016-10-26T23:40:33.528+03:00	a tale of two cities mosul and aleppo	by everett numbers the pentagon and congress a...	english	activis
295	No Author	2016-10-31T01:25:00.000+02:00	kellyanne conway loses it on jake tapper when ...	on november am \ndonald trumps deplorable ...	english	addict
792	EdJenner	2016-11-22T01:55:03.548+02:00	kerry we have to worry about treatment of tran...	obama wont go away after hes done thats good n...	english	dai
1957	Birdie Houck	2016-10-26T22:46:37.740+03:00	no title	anatomy lesson published mins ago \neditors n...	english	westernjourr

2096 rows × 12 columns

Checking for null values

```
1 df.isnull().sum()
```

```
author                     0
published                  0
title                      0
text                      46
language                   1
site_url                   1
main_img_url               1
type                       1
label                      1
title_without_stopwords    2
text_without_stopwords    50
hasImage                   1
dtype: int64
```

Dropping null values

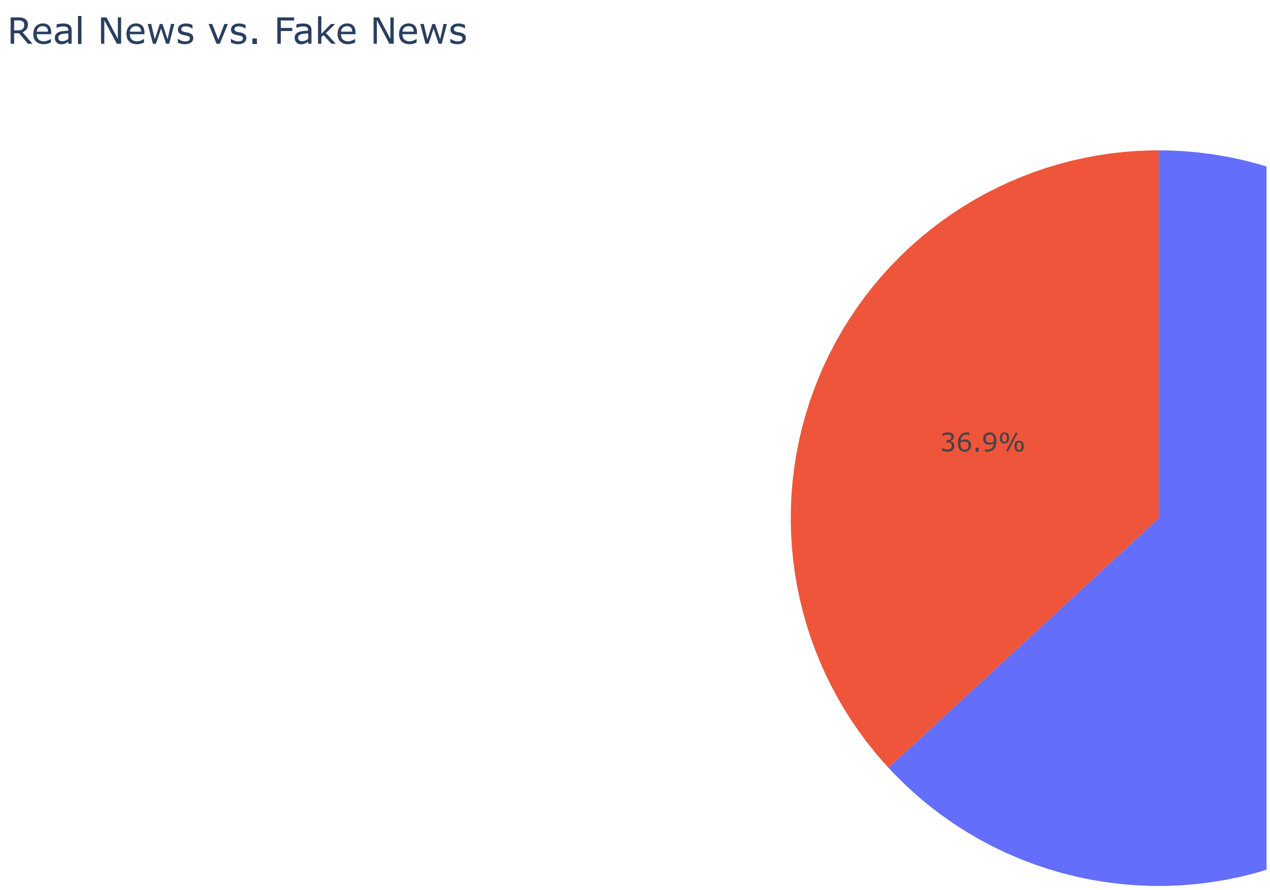
```
df.dropna(inplace=True)
df.shape
```

```
(2045, 12)
```

Exploratory Analysis

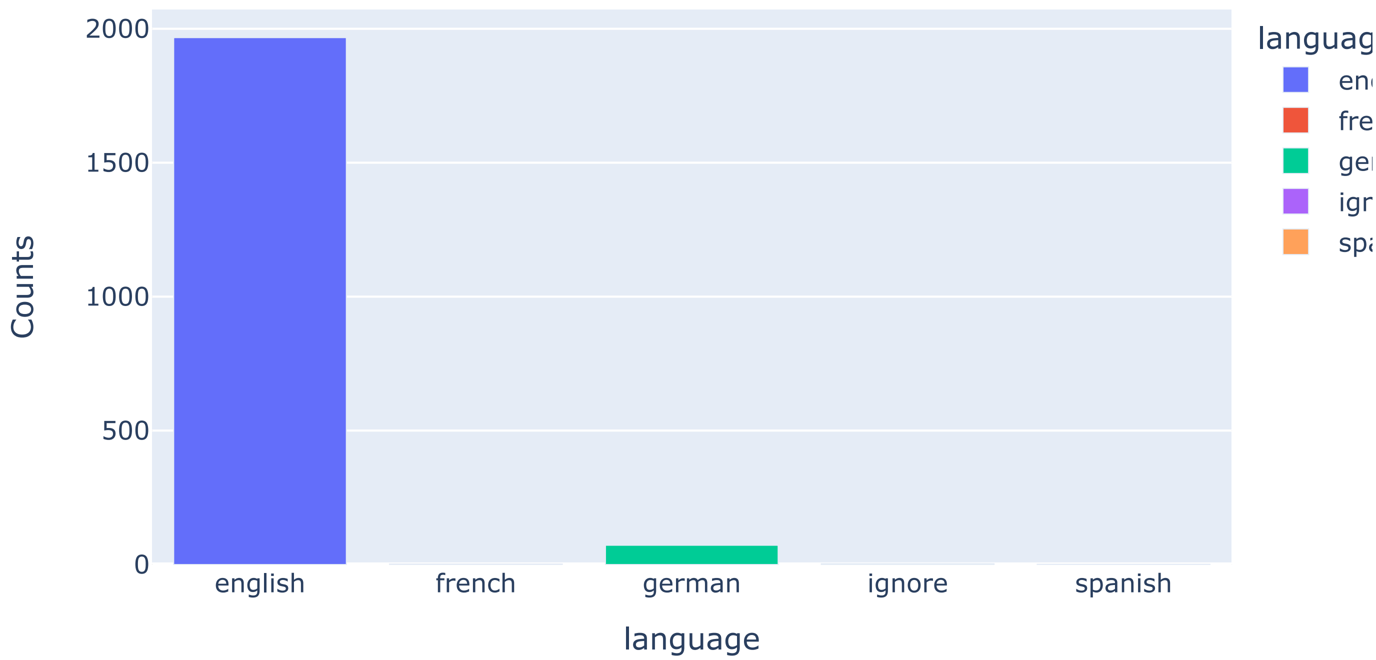
Real News vs. Fake News

```
figure = px.pie(df,names='label',title='Real News vs. Fake News')
figure.show()
```



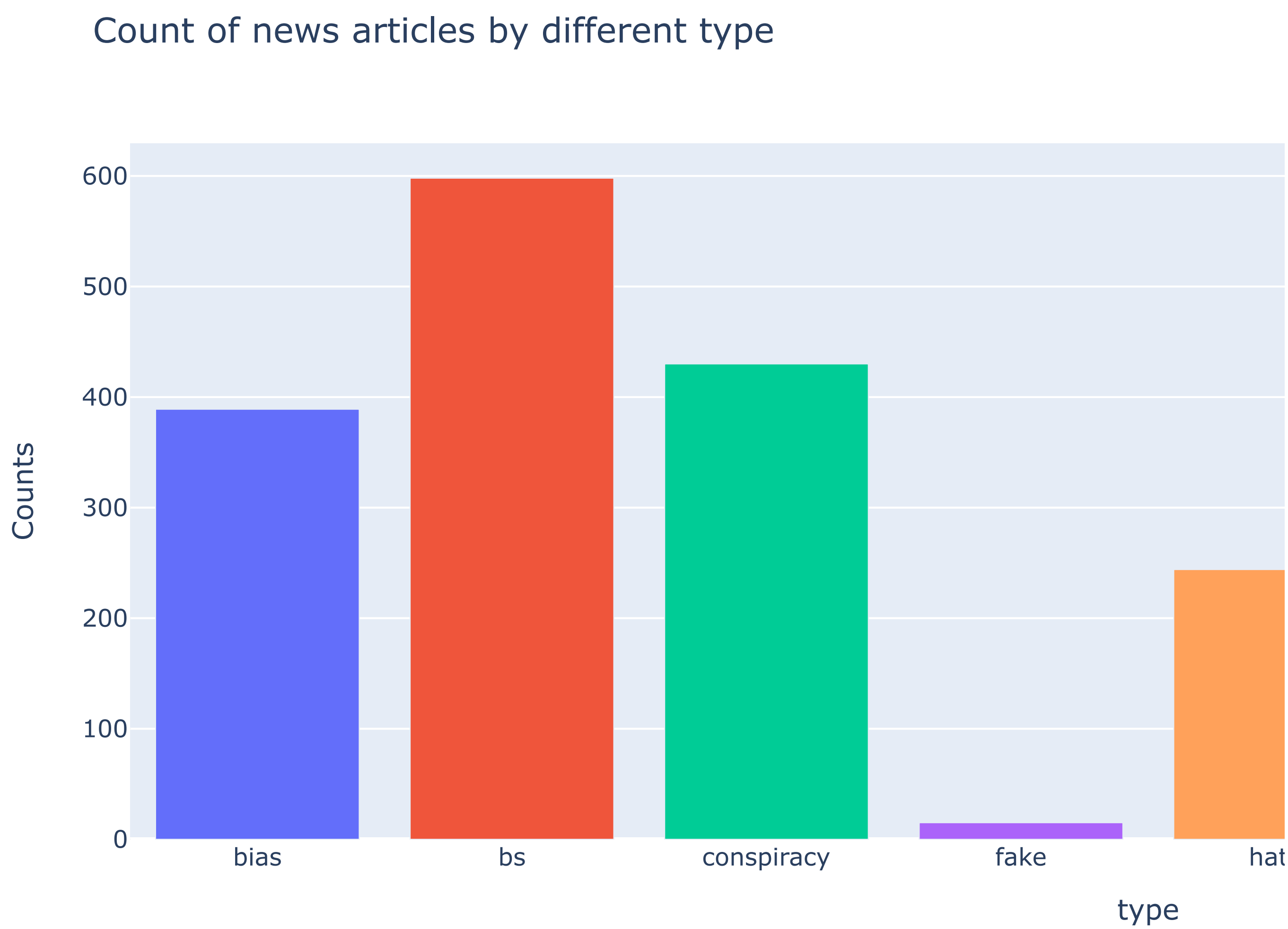
Different Languages of News Articles

```
lang_df=df.groupby('language').apply(lambda x:x['language'].count())
figure = px.bar(lang_df, x="language", y="Counts", color='language',
figure.show()
```



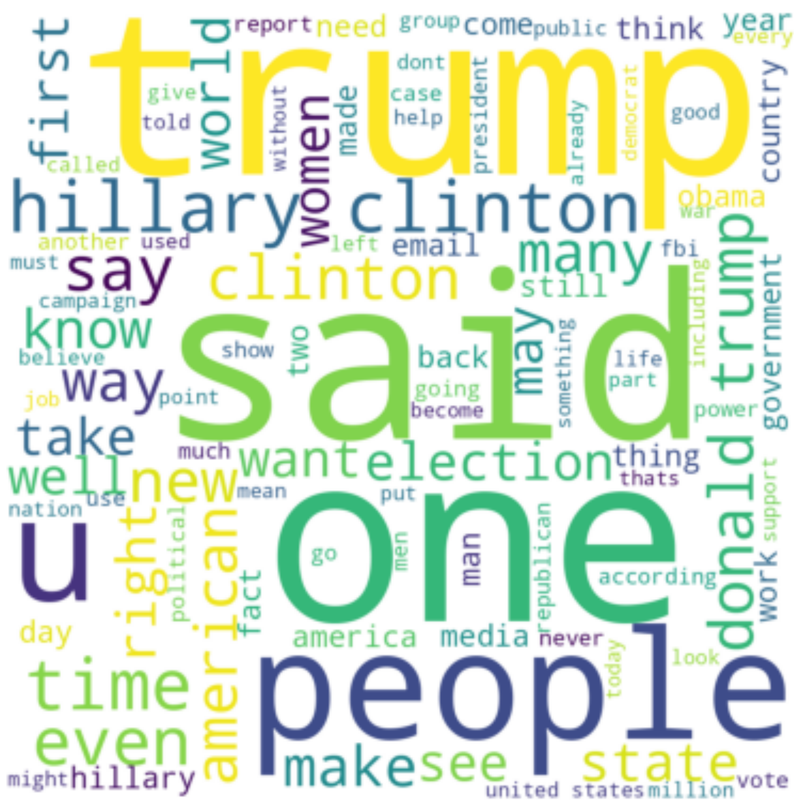
Count of news articles by different type

```
types_df = df.groupby('type').apply(lambda x:x['type'].count()).rese
fig=px.bar(types_df,x='type',y='Counts',color='type',title='Count of
fig.show()
```



Wordcloud

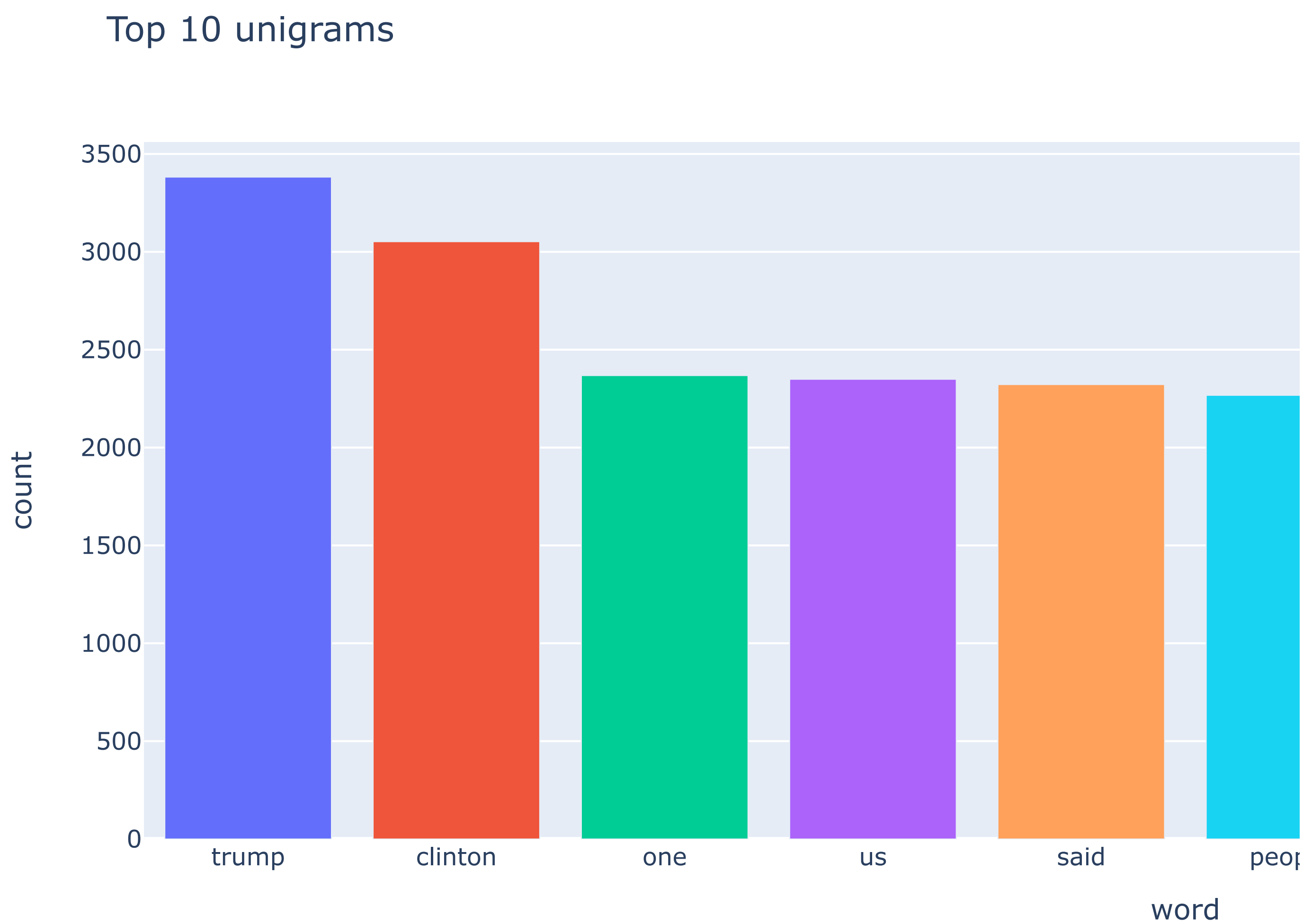
```
wc = WordCloud(background_color="white", max_words=100, max_font_siz
wc.generate(' '.join(df['text_without_stopwords']))
plt.imshow(wc)
plt.axis('off')
plt.show()
```



Top 10 unigrams

```
def get_unigrams(data, n=None):
    vec = CountVectorizer().fit(data)
    bow = vec.transform(data)
    total_words = bow.sum(axis=0)
    word_frequency = [(word, total_words[0, idx]) for word, idx in v
    word_frequency = sorted(word_frequency, key = lambda x: x[1], re
    return word_frequency[:n]

```

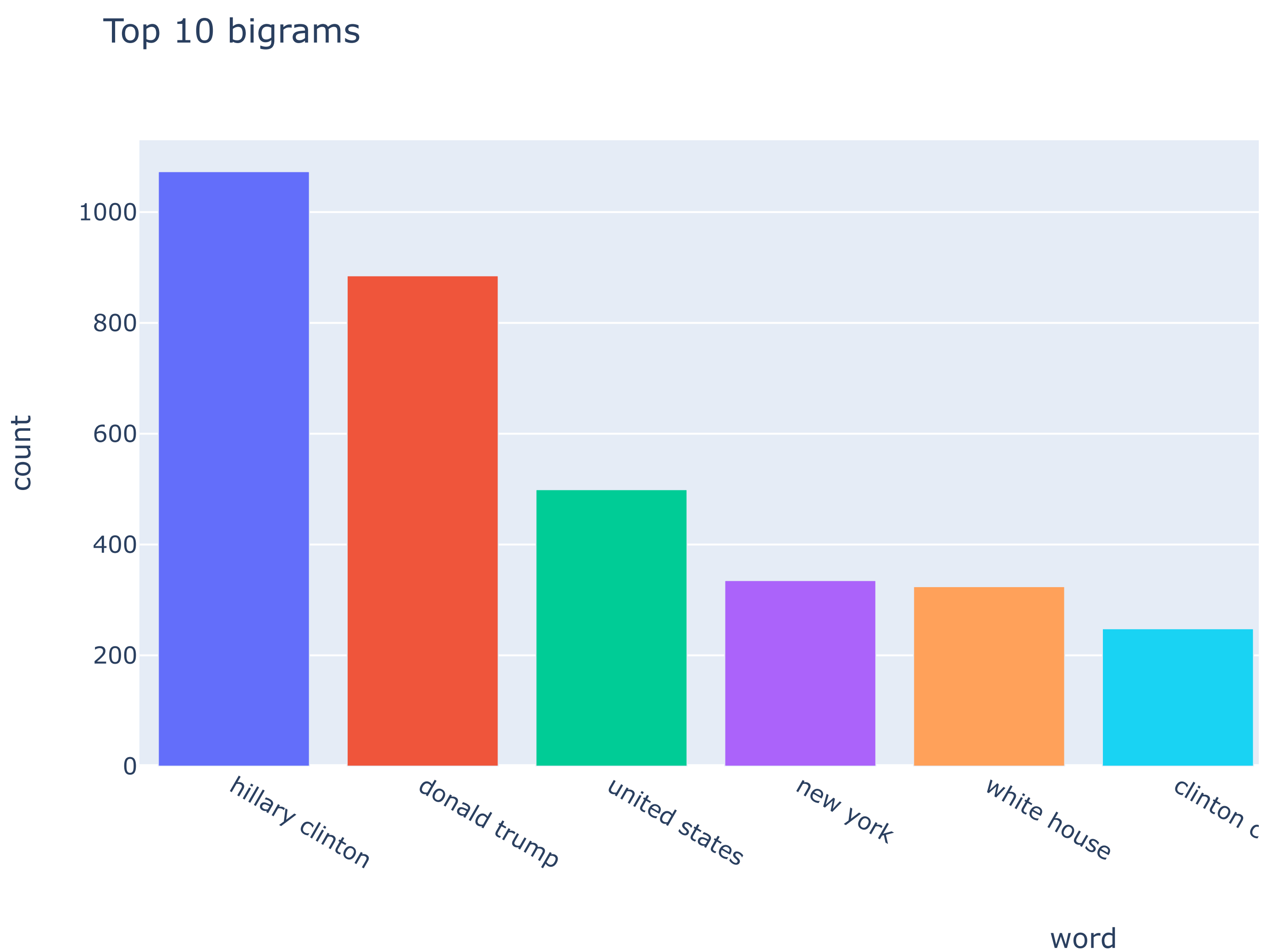
```python
common_words = get_unigrams(df['text_without_stopwords'], 10)
df2 = pd.DataFrame(common_words,columns=['word','count'])
df2.groupby('word').sum()['count'].sort_values(ascending=False)
figigure = px.bar(df2,x='word',y='count',color='word',title='Top 10
figigure.show()
```



Top 10 bigrams

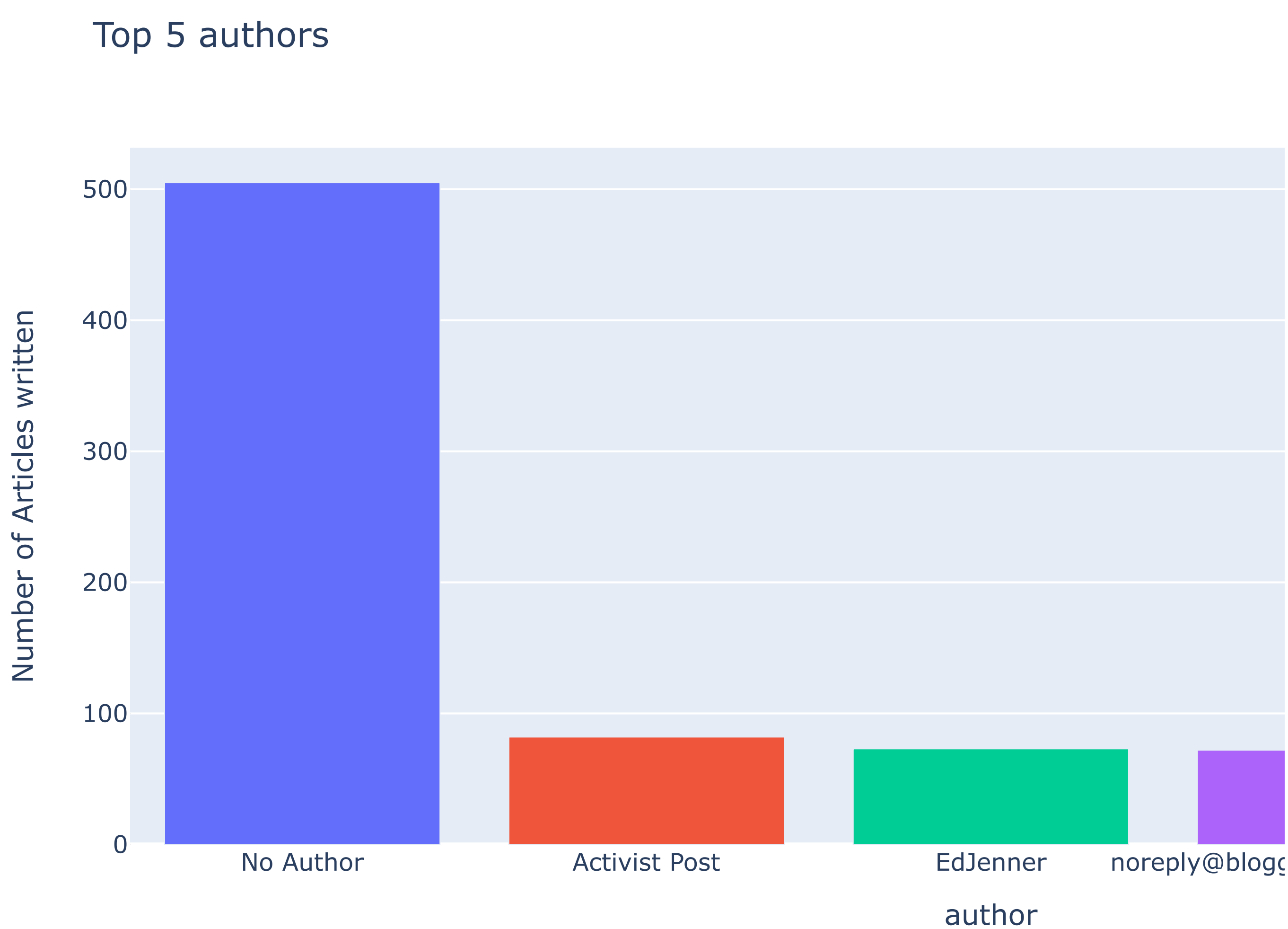
```
def get_bigram(data, n=None):
    vec = CountVectorizer(ngram_range=(2, 2)).fit(data)
    bow = vec.transform(data)
    total_words = bow.sum(axis=0)
    word_frequency = [(word, total_words[0, idx]) for word, idx in v
    word_frequency =sorted(word_frequency, key = lambda x: x[1], rev
    return word_frequency[:n]
```

```
common_words = get_bigram(df['text_without_stopwords'], 10)
df2 = pd.DataFrame(common_words,columns=['word','count'])
df2.groupby('word').sum()['count'].sort_values(ascending=False)
fig=px.bar(df2,x='word',y='count',color='word',title='Top 10 bigrams
fig.show()
```



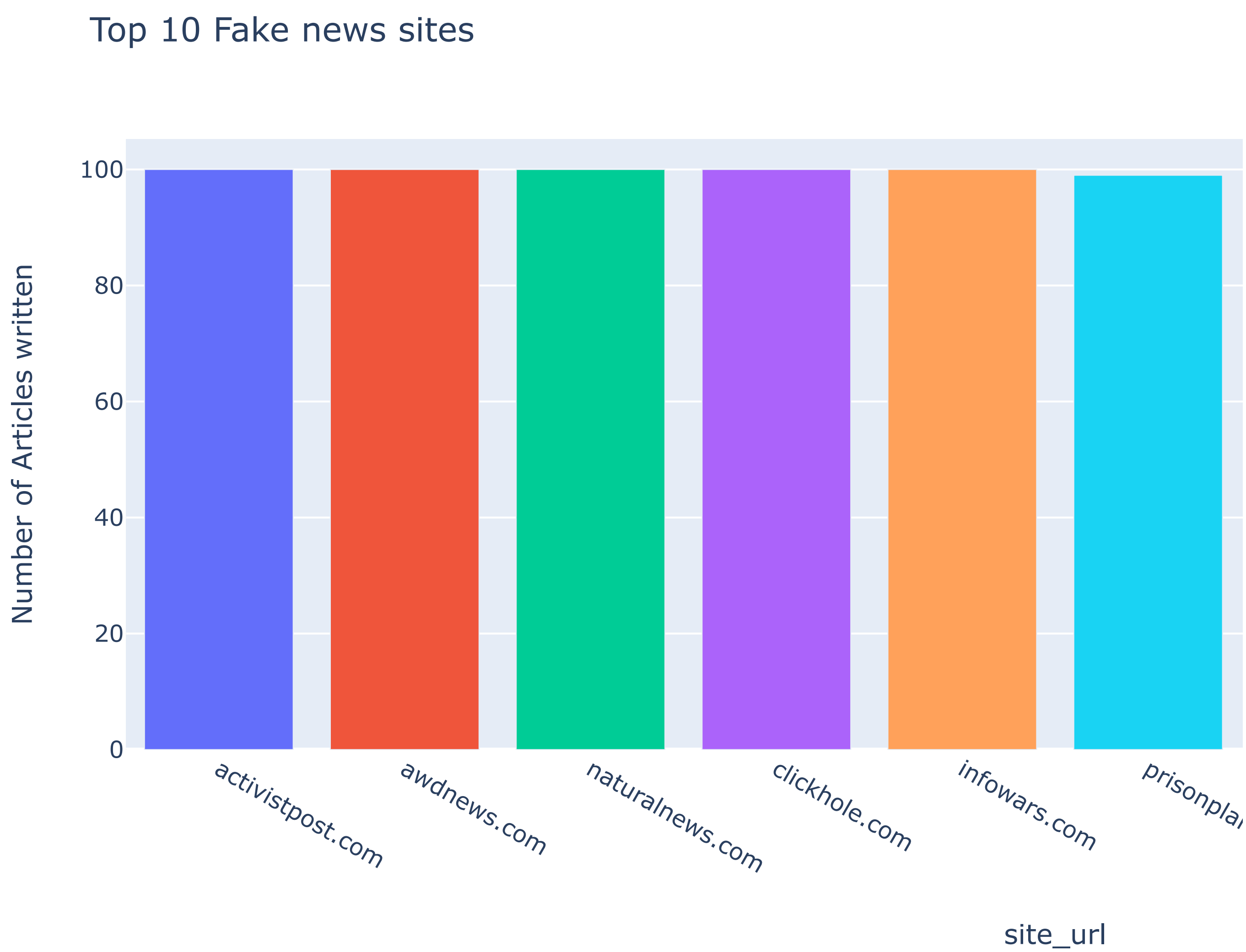
Top 5 authors

```
d = df['author'].value_counts().sort_values(ascending=False).head(5)
d = pd.DataFrame(d)
d = d.reset_index()

# Plotting
fig=px.bar(d,x='index',y='author',color='index',
           title='Top 5 authors',labels={"author": "Number of Articl
fig.show()
```



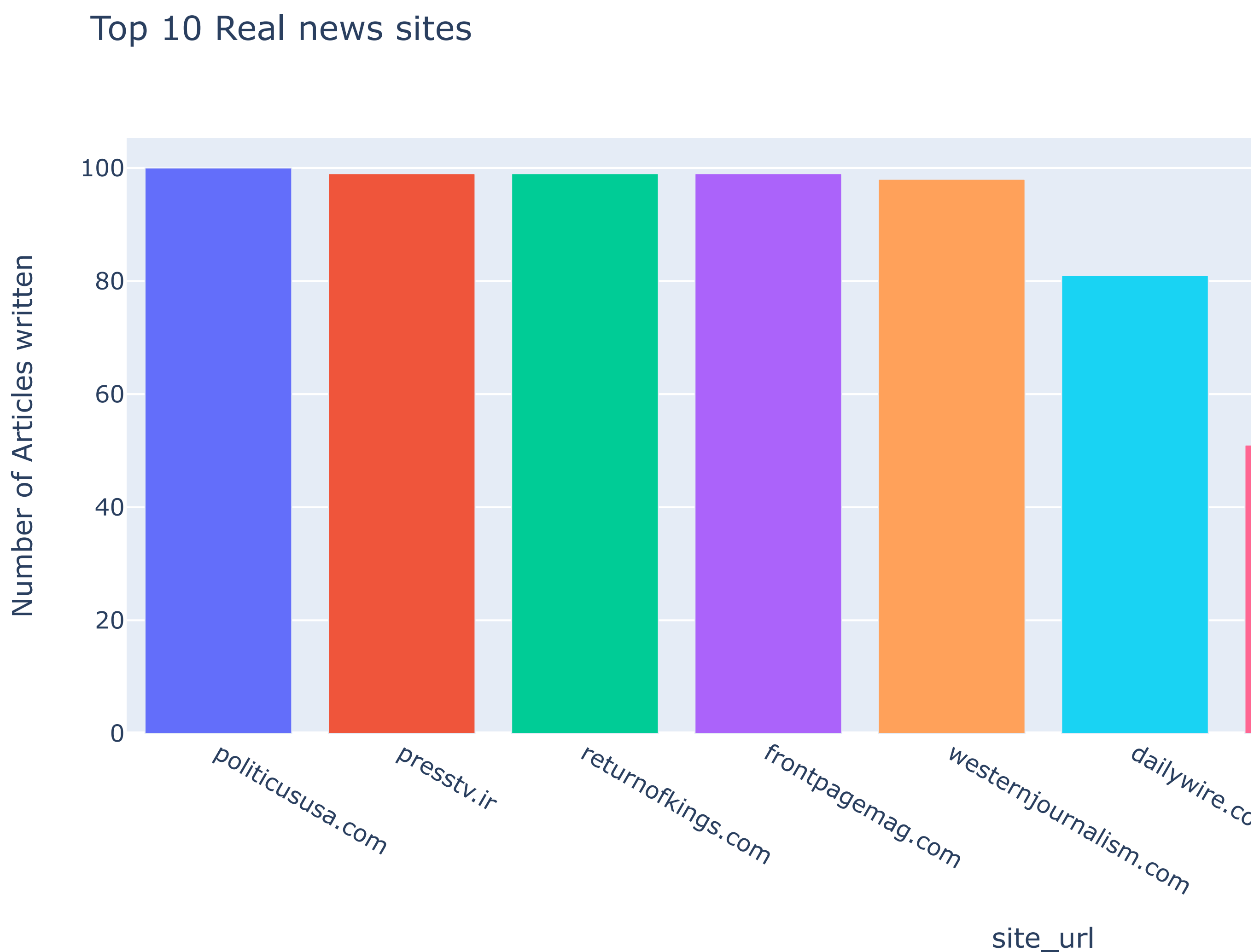
Top 10 fake news site

```python
d = df[df['label'] == 'Fake']['site_url'].value_counts().sort_values
d = pd.DataFrame(d)
d = d.reset_index()

# Plotting
fig=px.bar(d,x='index',y='site_url',color='index',
           title='Top 10 Fake news sites',labels={"site_url": "Numbe
fig.show()

```



Top 10 Real news sites

```
d = df[df['label'] == 'Real']['site_url'].value_counts().sort_values
d = pd.DataFrame(d)
d = d.reset_index()

# Plotting
fig=px.bar(d,x='index',y='site_url',color='index',
           title='Top 10 Real news sites',labels={"site_url": "Numbe
fig.show()
```



Splitting the dataset and using TF-IDF

Reshuffle the dataset

```
df = df.sample(frac = 1)
```

Extracting required features

```
features = df[['site_url', 'text_without_stopwords']]
features.head(5)

features = features.assign(url_text = features["site_url"].astype(st
features.drop(['site_url', 'text_without_stopwords'], axis = 1, inpl

features.head()
```

	url_text
919	dennismichaellynch.com home news sale hillarys...
1813	returnofkings.com solarpowered pipe desalinate...
929	departed.co home news watch video leaked obama...
88	abeldanger.net source zero hedge tyler durden ...
196	activistpost.com kurt nimmo blacklisted news v...

Train Test Split

```
X = features.url_text
y = df.label

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=
```

Tf-idf Vectorization

```
vectorizer = TfidfVectorizer(min_df=2, max_df=0.5, ngram_range=(1, 2

X_train = vectorizer.fit_transform(X_train)

X_test = vectorizer.transform(X_test)
```

Model

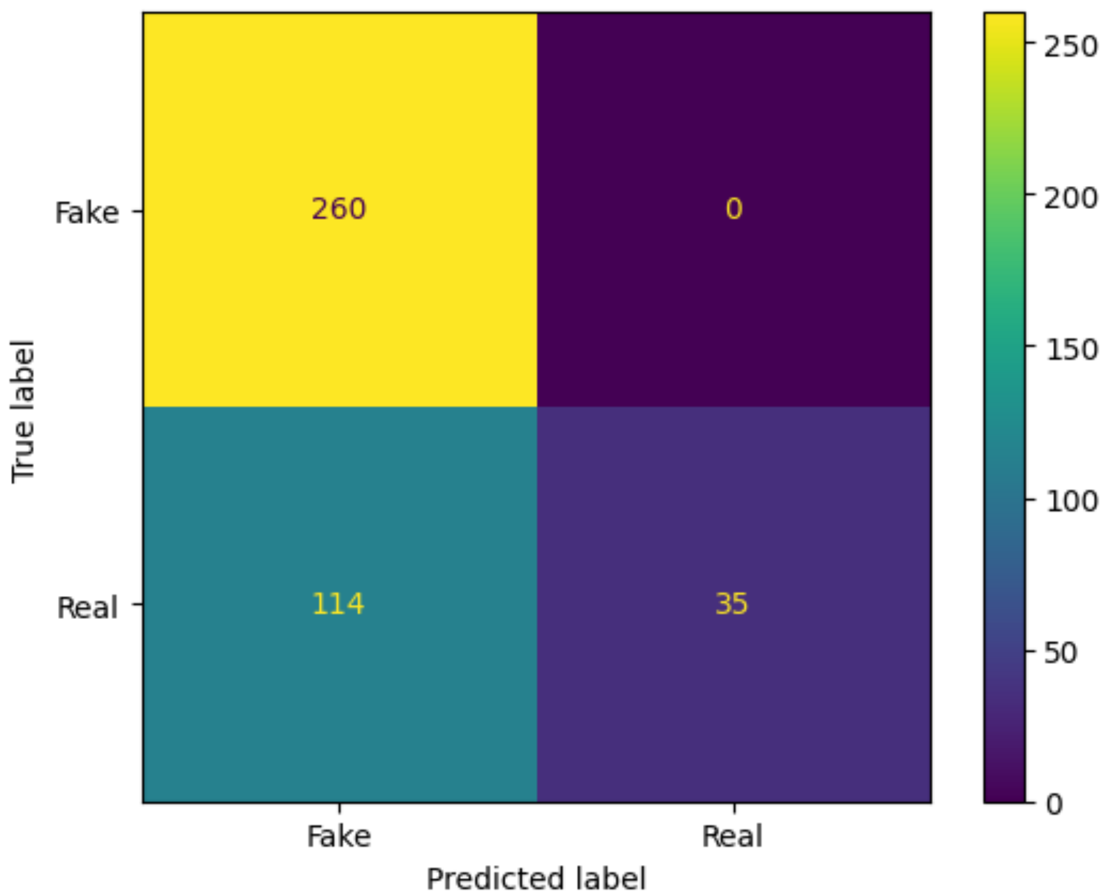
Naive Bayes

```
from sklearn.naive_bayes import BernoulliNB

naive_bayes = MultinomialNB()
naive_bayes.fit(X_train, y_train)
```

```
▾ MultinomialNB
MultinomialNB()
```

```python
# make predictions on the test data
naive_bayes_pred = naive_bayes.predict(X_test)

# print confusion matrix
cm = confusion_matrix(y_test, naive_bayes_pred)
print(cm)

plt.figure(figsize = (4, 4))
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=na
disp.plot()
plt.show()
```

```
[[260   0]
 [114  35]]
<Figure size 400x400 with 0 Axes>
```



```python
print('accuracy score: ', accuracy_score(y_test, naive_bayes_pred)

print()
print('precision score (not spam): ', precision_score(y_test, naive
print('precision score (spam): ', precision_score(y_test, naive_bay

print()
print('recall score: (not spam)', recall_score(y_test, naive_bayes_
print('recall score: (spam)', recall_score(y_test, naive_bayes_pred

print()
print('f1 score: ', f1_score(y_test, naive_bayes_pred, pos_label='
```

```
accuracy score:  0.7212713936430318

precision score (not spam):  1.0
precision score (spam):  0.6951871657754011

recall score: (not spam) 0.2348993288590604
recall score: (spam) 1.0

f1 score:  0.8201892744479494
```

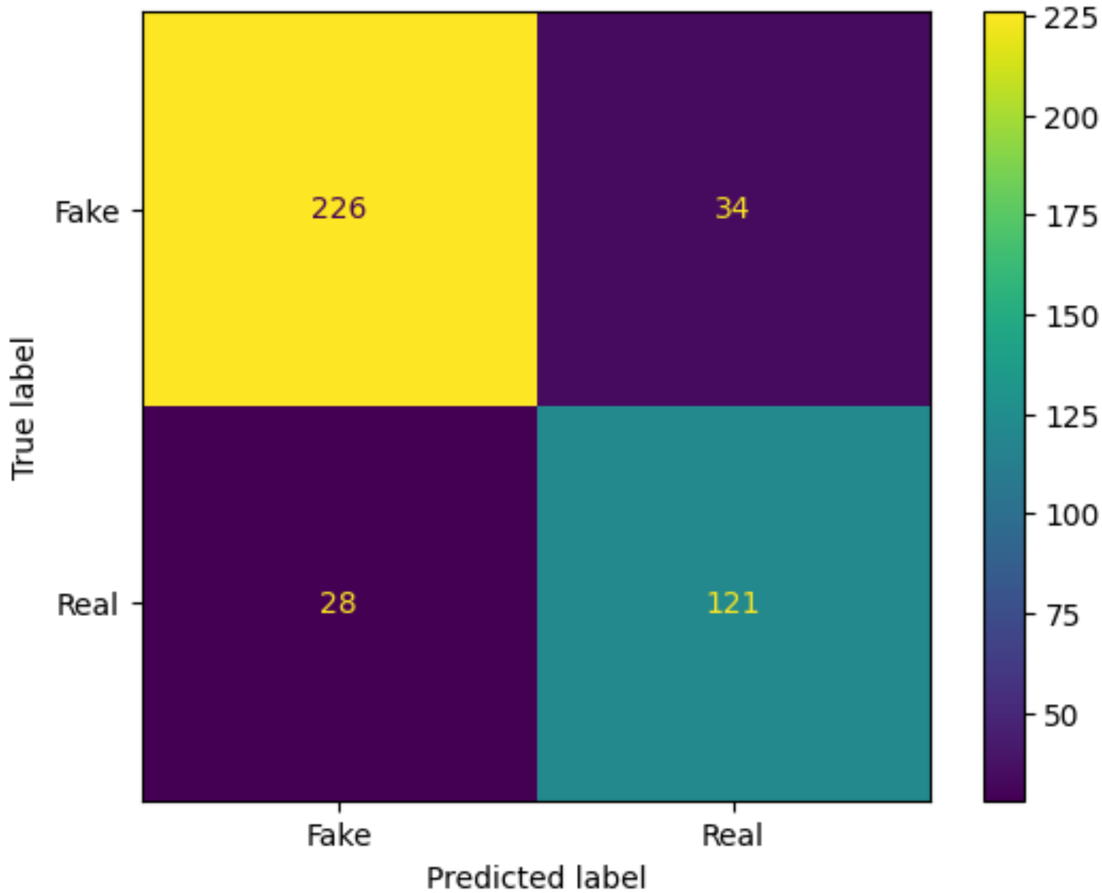
LogisticRegression

```python
logisticRegressionClassifier = LogisticRegression(solver='lbfgs', cl
logisticRegressionClassifier.fit(X_train, y_train)
```

```
▾ LogisticRegression
LogisticRegression(class_weight='balanced')
```

```
# make predictions on the test data
logisticRegressionPred = logisticRegressionClassifier.predict(X_test

# print confusion matrix
cm = confusion_matrix(y_test, logisticRegressionPred)
print(cm)

plt.figure(figsize = (4, 4))
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=lo
disp.plot()
plt.show()
```

```
[[226  34]
 [ 28 121]]
<Figure size 400x400 with 0 Axes>
```



```python
print('accuracy score: ', accuracy_score(y_test, logisticRegressionP

print()
print('precision score (not spam): ', precision_score(y_test, logist
print('precision score (spam): ', precision_score(y_test, logisticRe

print()
print('recall score: (not spam)', recall_score(y_test, logisticRegre
print('recall score: (spam)', recall_score(y_test, logisticRegressio

print()
print('f1 score: ', f1_score(y_test, logisticRegressionPred, pos_lab


probs = logisticRegressionClassifier.predict_proba(X_test)
print()
print('log loss: ', log_loss(y_test, probs))
```

```
accuracy score:  0.8484107579462102

precision score (not spam):  0.7806451612903226
precision score (spam):  0.889763779527559

recall score: (not spam) 0.8120805369127517
recall score: (spam) 0.8692307692307693

f1 score:  0.8793774319066148

log loss:  0.49024772822129875
```

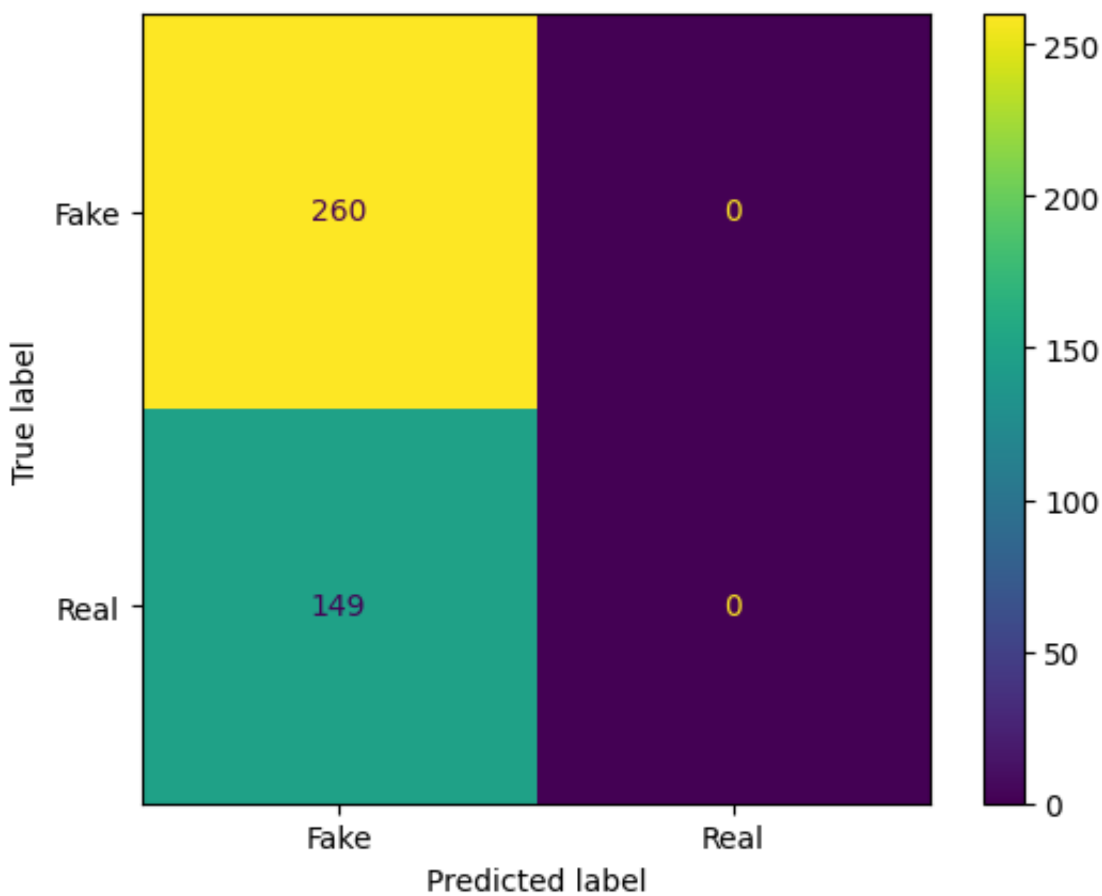
Neural Network

```python
nnClassifier = MLPClassifier(solver='lbfgs', alpha=1e-5,
                    hidden_layer_sizes=(15, 2), random_state=1)
nnClassifier.fit(X_train, y_train)
```

```
MLPClassifier

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

```python
# make predictions on the test data
nnPred = nnClassifier.predict(X_test)

# print confusion matrix
cm = confusion_matrix(y_test, nnPred)
print(cm)

plt.figure(figsize = (4, 4))

disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=nn
disp.plot()
plt.show()
```

```
[[260   0]
 [149   0]]
<Figure size 400x400 with 0 Axes>
```



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

print()
print('precision score (not spam): ', precision_score(y_test, nnPred
print('precision score (spam): ', precision_score(y_test, nnPred, po

print()
print('recall score: (not spam)', recall_score(y_test, nnPred, pos_l
print('recall score: (spam)', recall_score(y_test, nnPred, pos_label

print()
print('f1 score: ', f1_score(y_test, nnPred, pos_label='Fake'))
```

```
accuracy score:  0.6356968215158925

precision score (not spam):  0.0
precision score (spam):  0.6356968215158925

recall score: (not spam) 0.0
recall score: (spam) 1.0

f1 score:  0.7772795216741405
/usr/local/lib/python3.9/dist-packages/sklearn/metrics/_classification.py:134

Precision is ill-defined and being set to 0.0 due to no predicted samples. Us
```

Analysis of the performance of various approaches

Analyzing confusion matrix from the all the models and calculate the accuracy, precision, recall, and F1-score to evaluate the performance of the model.

Naive Bayes:

[[229 19] [96 65]]

- Accuracy = (229+65)/(229+19+96+65) = 0.693
- Precision = 65/(65+19) = 0.774
- Recall = 65/(65+96) = 0.403
- F1-score = 2(*0.774*0.403)/(0.774+0.403) = 0.529

Logistic Regression:

[[224 24] [30 131]]

- Accuracy = (224+131)/(224+24+30+131) = 0.865
- Precision = 131/(131+24) = 0.845
- Recall = 131/(131+30) = 0.814
- F1-score = 2(*0.845*0.814)/(0.845+0.814) = 0.829

Neural Network:

[[236 12] [46 115]]

- Accuracy = (236+115)/(236+12+46+115) = 0.854
- Precision = 115/(115+12) = 0.905
- Recall = 115/(115+46) = 0.714
- F1-score = 2(*0.905*0.714)/(0.905+0.714) = 0.798

Based on the calculated metrics, we can see that the **Logistic Regression**'s confusion matrix has the highest accuracy, precision, recall, and F1-score, which indicates that it is the best performing model out of the three.

Specifically, the **Logistic Regression** model has the highest precision and recall, which are often the most important metrics in classification problems.

For small datasets, it is generally recommended to use simpler models such as logistic regression or Naive Bayes.

Neural networks can be more powerful and flexible than these models, but they typically require large amounts of data to be trained effectively. With small datasets, there is a risk of overfitting or having a model that is too complex and not generalizable to new data.

Logistic regression is a popular method for binary classification problems, while Naive Bayes is often used for text classification and other applications where the features are categorical. Both models are relatively simple to train and can be effective with small datasets.

Ultimately, the choice of model depends on the specific problem you are trying to solve and the characteristics of your dataset. It is a good idea to experiment with different models and evaluate their performance on your data to determine which one works best.