```
!pip install -U scikit-learn
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/p
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.7/dist-
Requirement already satisfied: numpy>=1.14.6 in /usr/local/lib/python3.7/dist-package
Requirement already satisfied: scipy>=1.1.0 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.7/dist-packages
```

Uploading the CSV File

```
# upload csv file
from google.colab import files
uploaded = files.upload()
```

Browse... federalist.csv

federalist.csv(application/vnd.ms-excel) - 1100616 bytes, last modified: n/a - 100% done

```
Saving federalist.csv to federalist.csv
```

Converting the CSV file into pandas dataframe

```
import io
import pandas as pd

df = pd.read_csv(io.BytesIO(uploaded['federalist.csv']))
print(df.head())
print(df.author.value_counts())
```

```
     author                                               text
0  HAMILTON  FEDERALIST. No. 1 General Introduction For the...
1       JAY  FEDERALIST No. 2 Concerning Dangers from Forei...
2       JAY  FEDERALIST No. 3 The Same Subject Continued (C...
3       JAY  FEDERALIST No. 4 The Same Subject Continued (C...
4       JAY  FEDERALIST No. 5 The Same Subject Continued (C...
HAMILTON                49
MADISON                 15
HAMILTON OR MADISON     11
JAY                      5
HAMILTON AND MADISON     3
Name: author, dtype: int64
```

Importing NLTK and creating the vectorizer, removing stop words

```
import nltk
nltk.download('stopwords')
```

✓ 2s completed at 7:06 PM

```
     [nltk_data]   Unzipping corpora/stopwords.zip.
     True
```

```
from nltk.corpus import stopwords
from sklearn.feature_extraction.text import TfidfVectorizer

stopwords = set(stopwords.words('english'))
vectorizer = TfidfVectorizer(stop_words=stopwords)
```

Splitting into test and train sets and display the shape of each.

```
from sklearn.model_selection import train_test_split

X = df.text
y = df.author

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, train_size=0.8, ra

print("Train set shape:", X_train.shape)
print("Test set shape:", X_test.shape)
```

```
     Train set shape: (66,)
     Test set shape: (17,)
```

Vectorize the test and train sets

```
X_train = vectorizer.fit_transform(X_train)
X_test = vectorizer.transform(X_test)
```

Use the bernoulli naive bayes model to train the model and predict on the test set.

```
from sklearn.naive_bayes import BernoulliNB
from sklearn.metrics import accuracy_score

naive_bayes = BernoulliNB()
naive_bayes.fit(X_train, y_train)

pred = naive_bayes.predict(X_test)

print('Accuracy:', accuracy_score(y_test, pred))
```

```
     Accuracy: 0.5882352941176471
```

Try to get a better accuracy by limiting the amount of features and using bigrams.

```
vectorizer = TfidfVectorizer(stop_words=stopwords,max_features=1000,ngram_range=(1, 2))

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, train_size=0.8, ra

X_train = vectorizer.fit_transform(X_train)
X_test = vectorizer.transform(X_test)

naive_bayes.fit(X_train, y_train)

pred = naive_bayes.predict(X_test)

print('Accuracy:', accuracy_score(y_test, pred))
```

```
    Accuracy: 0.9411764705882353
```

Now use logistic regression on the same dataset to train the model and predict the test set.

```
from sklearn.linear_model import LogisticRegression

vectorizer = TfidfVectorizer(stop_words=stopwords,max_features=1000)

X_tf = vectorizer.fit_transform(X)

X_train, X_test, y_train, y_test = train_test_split(X_tf, y, test_size=0.2, train_size=0.8,


logistic_regression_default = LogisticRegression()
# Using different parameters to get better acurracy
logistic_regression =  LogisticRegression(multi_class='multinomial', solver='lbfgs', class_

logistic_regression_default.fit(X_train, y_train)
logistic_regression.fit(X_train,y_train)

pred_def = logistic_regression_default.predict(X_test)
pred = logistic_regression.predict(X_test)

print('Accuracy (default):', accuracy_score(y_test,pred_def))
print('Accuracy:', accuracy_score(y_test,pred))
```

```
    Accuracy (default): 0.5882352941176471
    Accuracy: 0.7647058823529411
```

Use MLPClassifier to train the model and predict the test set.

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

```
vectorizer = TfidfVectorizer(stop_words=stopwords,max_features=1000)

X_tf = vectorizer.fit_transform(X)

X_train, X_test, y_train, y_test = train_test_split(X_tf, y, test_size=0.2, train_size=0.8

classifier = MLPClassifier()
classifier.fit(X_train, y_train)

classifier_params = MLPClassifier(max_iter=1000, solver='lbfgs', hidden_layer_sizes=(30,20

classifier_params.fit(X_train, y_train)

pred = classifier.predict(X_test)
pred_params = classifier_params.predict(X_test)
print('Accuracy:', accuracy_score(y_test,pred))
print('Final accuracy (with params):', accuracy_score(y_test,pred_params))
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.
  ConvergenceWarning,
Accuracy: 0.7647058823529411
Final accuracy (with params): 0.8235294117647058
```