Name: Rahul Vijayvargiya
Student: 245784

Task:
Sentiment Labelled Sentences
Link for dataset:

[Kaggle Link to Dataset]

The collection contains sentences derived from the opinions of website users.
The dataset is used to learn models that recognise sentence sentiment between two classes – positive and negative. The dataset is designed for the classification task.

Sol.

Hello, let me tell you about my project, it's based on sentiments, we are using supervised learning technique called classification here,
with the help of sklearn Naive Bayes and Random Forest We gonna Classify which sentiment is good or which one is negative, we are using matplotlib, sklearn and pandas for our small project

Naive Bayes:

The Naive Bayes classification algorithm is a probabilistic classifier. It is based on probability models that incorporate strong independence assumptions.
The independence assumptions often do not have an impact on reality. Therefore they are considered naive.

Random Forest Classifier:

The random forest is a classification algorithm consisting of many decision trees. It uses bagging and feature randomness when building each individual tree to try to create an uncorrelated forest of trees whose prediction by committee is more accurate than that of any individual tree

We have a Dataset for reviews from yelp, amazon, imdb, we have to train our machine learning model to predict the sentiments of the sentences whether it is bad or good

Since it’s a supervised learning,

1.

[5]

	sentence	label	source
0	Wow... Loved this place.	1	yelp
1	Crust is not good.	0	yelp
2	Not tasty and the texture was just nasty.	0	yelp
3	Stopped by during the late May bank holiday of...	1	yelp
4	The selection on the menu was great and so wer...	1	yelp

As you can see the the dataframe of our data set, it has label, source, sentence
Where label tells us about 1 is positive and 0 stands for a bad and negative review

2.

```
#converting reviews cols into lower case

df['reviews_lowercase'] = df.loc[:,'reviews'].str.lower()
```

0.4s

As you can see we are converting our dataframe column review into lower case, we have to feed clean data into our model

3.

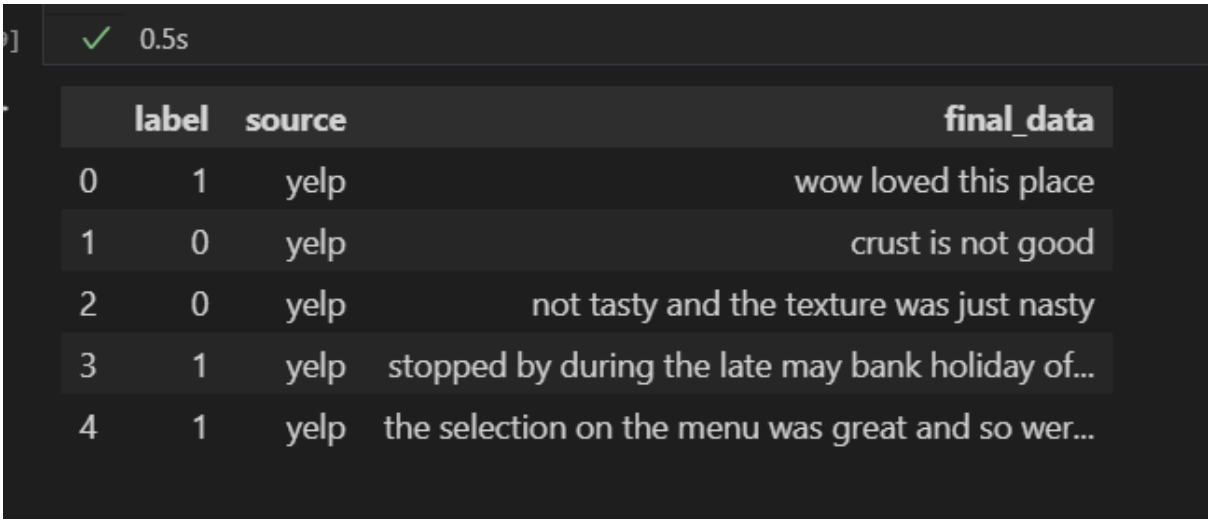
```
import string
punct = string.punctuation
print(punct)

def remove_puntuations(text):
    return text.translate(str.maketrans('','',punct))

df['final_data'] = df['reviews_lowercase'].apply(remove_puntuations)
```

✓ 0.7s

```
!"#$%&'()*+,-./:;<=>?@[\]^_`{|}~
```

As you can see this small function removing all the punctuation marks from the col, we just lower case, this punctuation mark doesn't add any value in the sentence, so we decide to remove them

4.

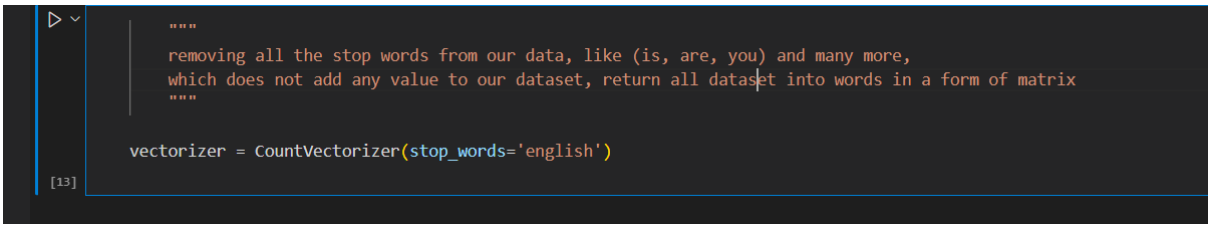


✓ 0.5s

	label	source	final_data
0	1	yelp	wow loved this place
1	0	yelp	crust is not good
2	0	yelp	not tasty and the texture was just nasty
3	1	yelp	stopped by during the late may bank holiday of...
4	1	yelp	the selection on the menu was great and so wer...

This is how our final data looks like, which we gonna feed to model but before we have to remove stop words as well from our data frame, a stop words doesn't add any value all to the sentence

5.



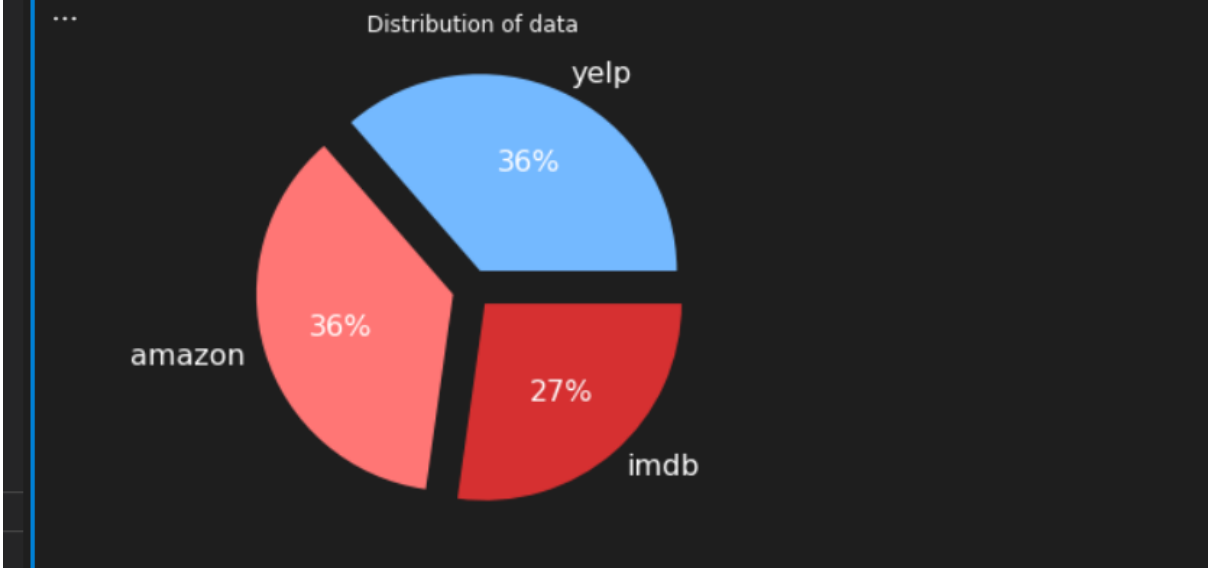
```
    """
    removing all the stop words from our data, like (is, are, you) and many more,
    which does not add any value to our dataset, return all dataset into words in a form of matrix
    """

vectorizer = CountVectorizer(stop_words='english')
```

[13]

From sklearn we are using count vectorizer to remove stop words such as top words in English are **"a", "the", "is", "are"** and etc

6. Pie plot to show distribution of dataset

```
plt.title("Distribution of data")
plt.show()
```
[10] ✓ 0.2s



As you can see the fair share of data imdb shares the lowest percentage

7. After removing the stop words and removing the punctuation, lowercase the dataframe,
   We gonna convert our data into a word vector using countvectorizer from sklearn

8.
```
(2748, 5118)
```
+ Code + Markdown

```
# Splitting data into training and test data set

x_train, x_test, y_train, y_test = train_test_split(all_features, df.label, test_size = 0.20, random_state=100)
```
[16]

```
# imported Multinomial Naive Bayes from Sk learn

classifier = MultinomialNB()
```
[17]

We are splitting data into training and test, and importing multinomial naive bayes model from sklearn
And fitting our training data into model

With the help of naive bayes classifier we predict

448 sentences classified correctly, 102 sentences classified incorrectly

9.

```
# The accuracy of the model

fraction_of_wrong = nr_incorrect / (nr_correct + nr_incorrect)

print(f'The (testing) accuracy of the model is {1-fraction_of_wrong:.2%}')
```

```
[22]
The (testing) accuracy of the model is 81.45%
```

And test data accuracy of the model is 81.45%

We also used random forest classifier, with the help of random forest classifier we predicted 436 correct and 114 incorrect sentences and
Test data accuracy of model is 79.27%

Experiments:

I have prepared a test case using naive bayes and it seems it predicted correctly

```
# test case using naive bayes

example = ['i hate you',
           'i love you',
          'i hate america',
          'i hate everyone',
          'product is bad',
          'not all bad',
          'boom boom hate',
          'this new movie ']


doc_to_word_mat = vectorizer.transform(example)
classifier.predict(doc_to_word_mat)
```

```
array([0, 1, 0, 0, 0, 0, 0, 1], dtype=int64)
```

2. Test Case using Random Forest Classifier

```
# test case using random forest

example = ['i hate you',
           'i love you',
           'i hate america',
           'i hate everyone',
           'product is bad',
           'not all bad',
           'boom boom hate']


doc_to_word_mat = vectorizer.transform(example)
rf.predict(doc_to_word_mat)
```

[33]

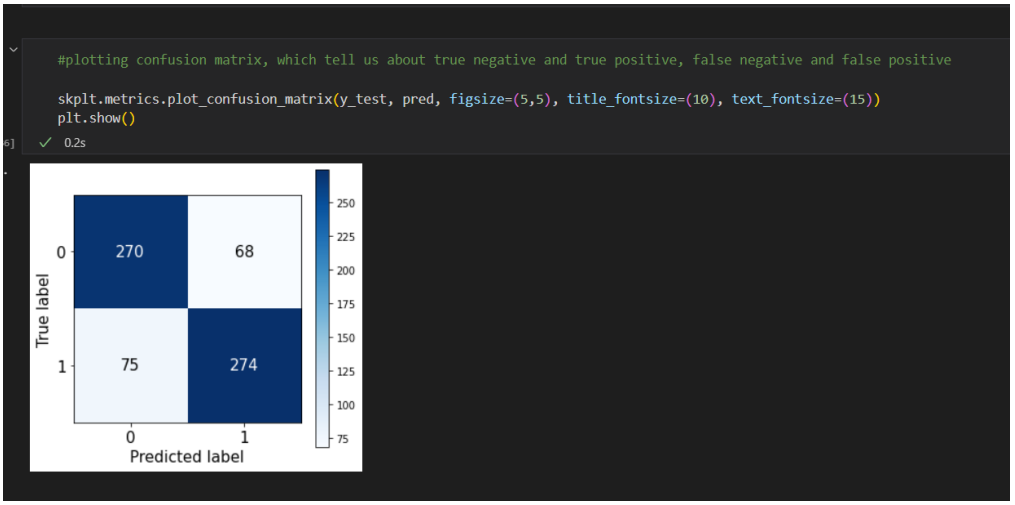
```
array([0, 1, 0, 0, 0, 0, 0], dtype=int64)
```

By adjustment in test size data and random state of data, we see that the accuracy of both model is decrease

```
# Splitting data into training and test data set

x_train, x_test, y_train, y_test = train_test_split(all_features, df.label, test_size = 0.25, random_state=75)
```

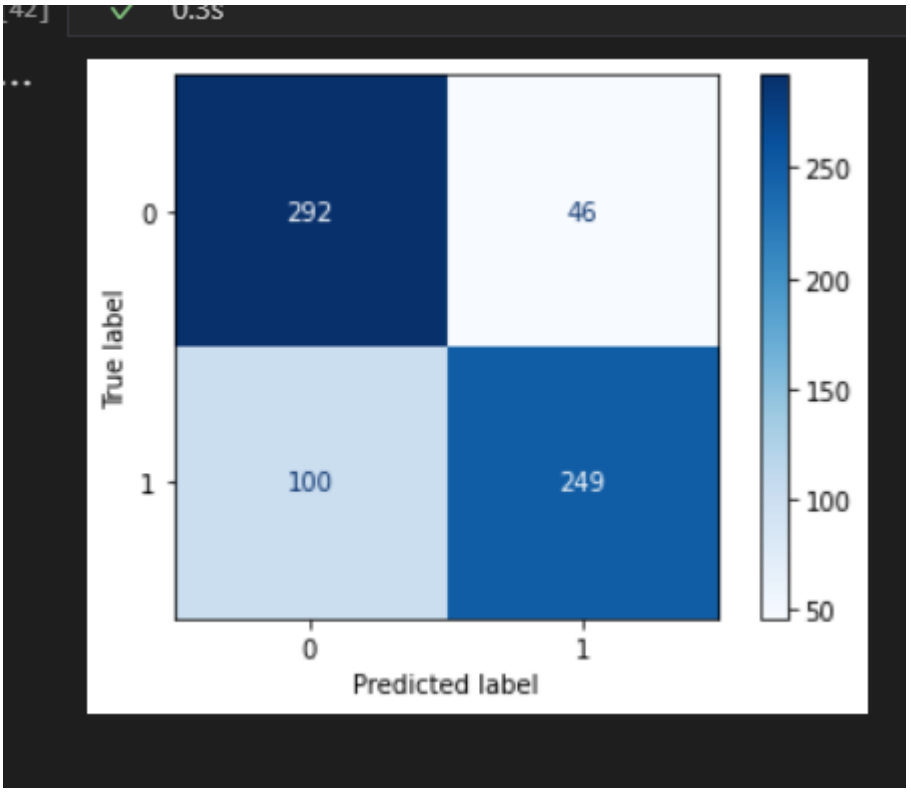
7] ✓ 0.1s

Naive bayes:



544 sentence classified correctly and rest all are incorrect

Random Forest Classifier:



541 sentences are classified correctly and rest are incorrect

Using TF-IDF Vectorizor:

```
# Splitting data into training and test data set

x_train, x_test, y_train, y_test = train_test_split(all_features, df.label, test_size = 0.20, random_state=100)
```

0.2s

As we see the performance of our model has increased compare to count vectorizer

Naive Bayes:

```
#checking score with sklearn naive bayes

print(f'The Accuracy Score on test data {classifier.score(x_test, y_test):.2%}')
```

[54] 0.1s

```
The Accuracy Score on test data 83.09%
```

+ Code + Markdown

]

Random Forest:
On random forest, compare to count vectorizer the accuracy is decrease

```
#The Accuracy Score on test data using random forest classifier

print(f'The Accuracy Score on test data using random forest classifier {rf.score(x_test, y_test):.2%}')
```

[58] 0.2s

```
The Accuracy Score on test data using random forest classifier 78.73%
```