```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
import numpy as np
import pandas as pd
```

```
data = pd.read_csv('/content/drive/MyDrive/Restaurant_Reviews.tsv', delimiter='\t', quoting=1)
```

```
data.shape
```

```
(1000, 2)
```

```
data.columns
```

```
Index(['Review', 'Liked'], dtype='object')
```

```
data.head()
```

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

```
data.info
```

```
<bound method DataFrame.info of                                                 Review  Liked
0                              Wow... Loved this place.      1
1                                    Crust is not good.      0
2             Not tasty and the texture was just nasty.      0
3    Stopped by during the late May bank holiday of...      1
4    The selection on the menu was great and so wer...      1
..                                                 ...    ...
995  I think food should have flavor and texture an...      0
996                            Appetite instantly gone.      0
997  Overall I was not impressed and would not go b...      0
998  The whole experience was underwhelming, and I ...      0
999  Then, as if I hadn't wasted enough of my life ...      0

[1000 rows x 2 columns]>
```

```
import nltk # natural language toolkit is a package from NLP
import re # Regular expressions
nltk.download('stopwords') # NLTK
from nltk.corpus import stopwords # a stop word is commonly used word that search engine is programmed to ignore
from nltk.stem.porter import PorterStemmer # mainly focuses on Data Mining and information retrieval .... it is used to remove the suffixes f
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

```
corpus = []
for i in range(0,1000):
  review = re.sub(pattern='[^a-zA-Z]', repl=' ', string=data['Review'][i]) # removes and replaces every non 'a-z' and 'A-Z' with ' '.
  review = review.lower() # to lower case
  review_words = review.split() # tokenizing review by words
  review_words = [word for word in review_words if not word in set(stopwords.words('english'))] # every non-stopword is added to the list ( i
  ps = PorterStemmer()
  review = [ps.stem(word) for word in review_words]
  review = ' '.join(review)
  corpus.append(review)
```

```
len(corpus)
```

```
1000
```

```
from sklearn.feature_extraction.text import CountVectorizer # cv is used to convert text to numerical data
cv = CountVectorizer(max_features=1500) # initialize cv to 1500(data size from corpus)
# in below fit_transform fits the vectorizer to the corpus (learns the vocabulary and tokenizes the documents) and then transforms the docume
# the DTM formed where each row represents a document, and each column represents a word from the vocabulary
X = cv.fit_transform(corpus).toarray() # corpus is converted into numerical data as an array format
Y = data.iloc[:, 1].values # extracting the likes(boolean) from data(origianl set tsv)



# here splitting data is done into two parts (training and testing) so that to train the model to training dataset and test its accuracy on u
from sklearn.model_selection import train_test_split # uses to split original data into training data & test data
X_train, X_test, Y_train, Y_test, = train_test_split(X,Y,test_size=0.20,random_state =0) # test_size for splitting purpose and random state s

# 80% - training dataset
# 20% - testing dataset


X_train.shape , X_test.shape, Y_train.shape, Y_test.shape
```

```
((800, 1500), (200, 1500), (800,), (200,))
```

```
# PREPROCESSING PART ENDS---
# MODEL TRAING----
# Multinomial Naive bayes - the algorithm is a probabilistic learning method that is mostly used in NLP. Its based on Bayes theorem and predi


# fitting Naive bayes to the training set
from sklearn.naive_bayes import MultinomialNB
# MNB is suitable for the classification purpose with discrete features( word counts for text classification)

classifier = MultinomialNB()
classifier.fit(X_train, Y_train) # fitting training data
```

```
▸ MultinomialNB
```

```
# Predicting test set results
Y_pred = classifier.predict(X_test)
Y_pred
```

```
array([0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0,
       0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0,
       1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0,
       1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0,
       0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1,
       0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1,
       1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1,
       0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1,
       0, 1])
```

```
# for accuracy , precision and recall
from sklearn.metrics import accuracy_score # it calculates the accuracy score for a set of predictedd values against true labels.

from sklearn.metrics import precision_score
from sklearn.metrics import recall_score

score_acc = accuracy_score(Y_test, Y_pred)
score_pre = precision_score(Y_test, Y_pred)
score_rec = recall_score(Y_test, Y_pred)

print("---Scores---")
print("Accuracy Score : {}% ".format(round(score_acc*100,2)))
print("Precision Score : {}% ".format(round(score_pre*100,2)))
print("Recall Score : {}% ".format(round(score_rec*100,2)))
```

```
---Scores---
Accuracy Score : 76.5%
Precision Score : 76.42%
Recall Score : 78.64%
```

```
# Making the Confusion Matrix
# A Confusion Matrix is a table that is used to define the performance of a classification algorithm. A Confusion Matrix visualizes and summa

from sklearn.metrics import confusion_matrix
# (test , predicted)
```

```
cm = confusion_matrix(Y_test, Y_pred)
# cm -> array -> [[true +ve, false +ve] , [false -ve, true -ve]
```

```
cm
```

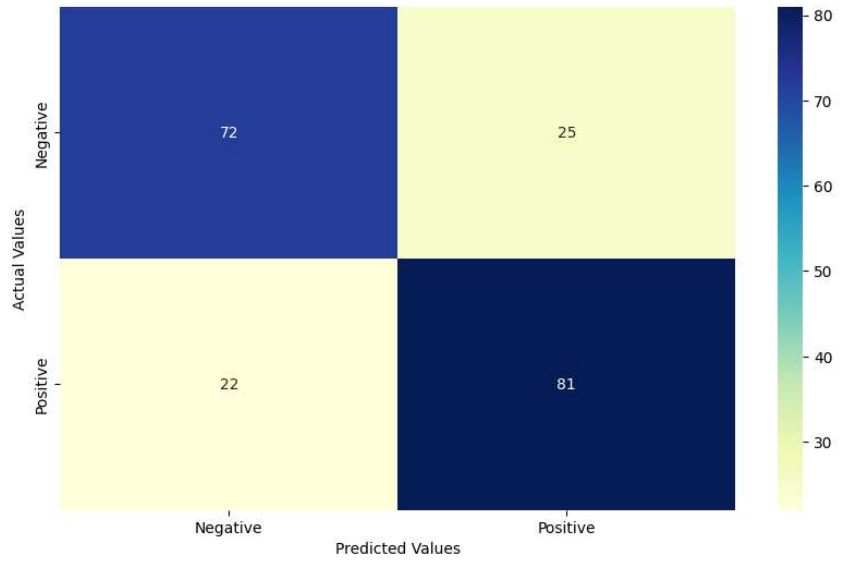
```
array([[72, 25],
       [22, 81]])
```

```
from ast import increment_lineno
# plotting confusion matrix

import matplotlib.pyplot as plt # a low level graph plotting library for visualizing purpose
import seaborn as sns # a visualizing library based on matplotlib
%matplotlib inline

plt.figure(figsize=(10,6))
sns.heatmap(cm, annot=True, cmap= 'YlGnBu' , xticklabels=['Negative','Positive'], yticklabels=['Negative','Positive'])
plt.xlabel("Predicted Values")
plt.ylabel("Actual Values")
```

```
Text(95.72222222222221, 0.5, 'Actual Values')
```




```
# Hyperparameter tuning the Naive Bayes Classifier -> to enhance the accuracy
best_accuracy = 0.0
alpha_val = 0.0
for i in np.arange(0.1,1.1,0.1):
  temp_classifier = MultinomialNB(alpha=i)
  temp_classifier.fit(X_train,Y_train)
  temp_Y_pred = temp_classifier.predict(X_test)
  score = accuracy_score(Y_test,temp_Y_pred)
  print("Accuracy Score for alpha={} is : {}%".format(round(i,1), round(score*100,2)))
  if score > best_accuracy:
    best_accuracy = score
    alpha_val = i
print('------------')
print('The Best Accuracy is {}% with alpha value as {}'.format(round(best_accuracy*100,2),round(alpha_val,1)))
```

```
Accuracy Score for alpha=0.1 is : 78.0%
Accuracy Score for alpha=0.2 is : 78.5%
Accuracy Score for alpha=0.3 is : 78.0%
Accuracy Score for alpha=0.4 is : 78.0%
Accuracy Score for alpha=0.5 is : 77.5%
Accuracy Score for alpha=0.6 is : 77.5%
Accuracy Score for alpha=0.7 is : 77.5%
Accuracy Score for alpha=0.8 is : 77.0%
Accuracy Score for alpha=0.9 is : 76.5%
Accuracy Score for alpha=1.0 is : 76.5%
------------
The Best Accuracy is 78.5% with alpha value as 0.2
```

```
classifier = MultinomialNB(alpha=0.2)
classifier.fit(X_train, Y_train)
```

```
▸ MultinomialNB
```

```
# PREDICTIONS

def predict_review(sample_review):
  sample_review = re.sub(pattern='[^a-zA-Z]', repl=' ', string=sample_review)
  sample_review = sample_review.lower()
  sample_review_words = sample_review.split()
  sample_review_words = [word for word in sample_review_words if not word in set(stopwords.words('english'))]
  ps = PorterStemmer()
  final_review = [ps.stem(word) for word in sample_review_words]
  final_review = ' '.join(final_review)

  temp = cv.transform([final_review]).toarray()
  return classifier.predict(temp)


sample_review = "My recent visit to the restaurant left me thoroughly disappointed. The confusing menu, filled with obscure descriptions, mad
if predict_review(sample_review):
  print('Positive review')
else:
  print('Negative review')
```

```
Negative review
```

```
sample_review = "My recent dining experience at the restaurant was nothing short of extraordinary. From the moment I stepped inside, I was en
if predict_review(sample_review):
  print('Positive review')
else:
  print('Negative review')
```

```
Positive review
```

Colab paid products - Cancel contracts here

✓ 0s completed at 3:51 AM