We train our model to understand some text and predict. Here case study is review of a restaurant.

Importing the libraries

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

import re
import nltk
```

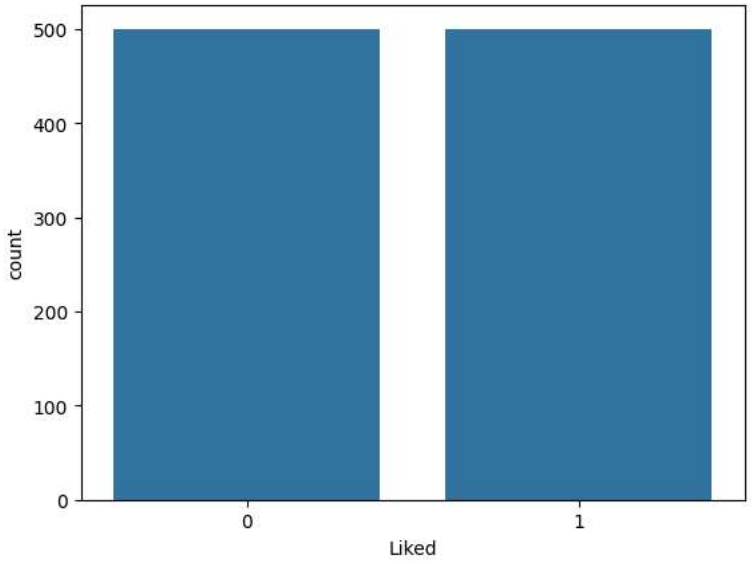
Importing the dataset

```
filepath='/content/Restaurant_Reviews.tsv'
df=pd.read_csv(filepath,sep='\t',quoting=3)
df
```

	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 × 2 columns

```
sns.countplot(x='Liked', data=df)
plt.show()
```



```
df['Liked'].value_counts()
```

```
1    500
0    500
Name: Liked, dtype: int64
```

Cleaning the texts

```
import re     # for cleaning etc
import nltk    # for downloading stopwords etc
nltk.download('stopwords')
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer

corpus=[]

for i in range(0,1000):
  review=re.sub('[^a-zA-Z^]',' ',df['Review'][i])
  review=review.lower()
  review=review.split()    # split each review into words,now its a list of words
  ps=PorterStemmer()
  all_stopwords=stopwords.words('english')
  all_stopwords.remove('not')
  review=[ps.stem(word) for word in review if not word in set(all_stopwords)]   # done stemming for every words except stopwords

  # join each words

  review=' '.join(review)
  corpus.append(review)
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
```

```
corpus[:5]
```

```
['wow love place',
 'crust not good',
 'not tasti textur nasti',
 'stop late may bank holiday rick steve recommend love',
 'select menu great price']
```

```
print(corpus)
```

```
['wow love place', 'crust not good', 'not tasti textur nasti', 'stop late may bank holiday rick steve recommend love', 'select menu grea
```

Creating bag of words model

```
from sklearn.feature_extraction.text import CountVectorizer
cv=CountVectorizer(max_features=1500)
# above we can give a parameter , since there are some words in the corpus like 'bank','textur' ...etc which is rarely repeating or not givi
# ideas of the review.so we can neglect it

X=cv.fit_transform(corpus).toarray()
Y=df['Liked']
```

```
len(X[0])
```

```
1500
```

```
Y
```

```
0      1
1      0
2      0
3      1
4      1
```

```
..
995    0
996    0
997    0
998    0
999    0
Name: Liked, Length: 1000, dtype: int64
```

```
X
```

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

Splitting the dataset into Training and Test dataset

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(X,Y,test_size=0.2,random_state=42)
```

```
print(f'Shape of x_train:{x_train.shape}')
print(f'Shape of x_test:{x_test.shape}')
print(f'Shape of y_train:{y_train.shape}')
print(f'Shape of y_test:{y_test.shape}')
```

```
Shape of x_train:(800, 1500)
Shape of x_test:(200, 1500)
Shape of y_train:(800,)
Shape of y_test:(200,)
```

training the Naive bayes model on the training dataset

```
from sklearn.naive_bayes import GaussianNB
model=GaussianNB()
```

```
model.fit(x_train,y_train)
```

```
GaussianNB
GaussianNB()
```

Predicting the Test set result

```
y_pred=model.predict(x_test)
y_pred
```

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

```
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report,ConfusionMatrixDisplay
```

Making the confusion matrix

```
model_acc=accuracy_score(y_pred,y_test)
model_acc
```

```
0.67
```

```
confusion_matrix=confusion_matrix(y_pred,y_test)
confusion_matrix
```

```
array([[48, 18],
       [48, 86]])
```