**SENTIMENTAL ANALYSIS**

* The SENTIMENTAL ANALYSIS is the process of analyzing the digital text to determine whether positive,negative or neutral.
* It is an approach to NLP(Natural language processing)



* It also goes beyond polarity to detect the specific feelings and emotions(angry,happy,sad,etc..).
* Depending on the customer reviews, we are gonna classify the data.

**CODE:**

import numpy as np

import pandas as pd

import nltk

import re

nltk.download('stopwords')

from nltk.corpus import stopwords

from nltk.stem.porter import PorterStemmer

import sklearn

from sklearn.feature\_extraction.text import CountVectorizer

from sklearn.model\_selection import train\_test\_split

from sklearn.naive\_bayes import MultinomialNB

from sklearn.metrics import accuracy\_score

* Import the libraries as numpy and pandas.
* The nltk is the natural language toolkit which imports natural language to the machine learning module.
* The stopwords are common words(is,are,etc..)
* The sklearn tools are used for machine learning,statistical modeling including classification,regression and clustering.
* The countvectorizer is used to transform a text to a vector form of tokens.
* The train and test split method is used to estimate the performance of machine learning which are used to make predictions on data not used to train the model.
* The MultinomialNB is a classifier to calculate the probability distribution of text data.

data = pd.read\_csv('/content/drive/MyDrive/Restaurant\_Reviews.tsv', delimiter='\t' , quoting=3)

* The csv file is set to be dataframe in pandas.

data.describe()

| **Liked** |  |
| --- | --- |
| **count** | 1000.00000 |
| **mean** | 0.50000 |
| **std** | 0.50025 |
| **min** | 0.00000 |
| **25%** | 0.00000 |
| **50%** | 0.50000 |
| **75%** | 1.00000 |
| **max** | 1.00000 |

* The data is described in the statistical methods.

corpus =[]

for i in range(0,1000):

review =re.sub(pattern='[^a-zA-Z]',repl=' ', string=data['Review'][i])

review = review.lower()

review\_words = review.split()

review\_words = [word for word in review\_words if not word in set(stopwords.words('english'))]

ps= PorterStemmer()

review =[ps.stem(word) for word in review\_words]

review = ' '.join(review)

corpus.append(review)

* Here the corpus is an empty list which stores the processed reviews.
* Using a loop iteration for 1000 reviews.
* The re.sub is to replace all characters in the review.
* The lower() converts the text into lower case.
* The review has been cleaned by splitting the review.
* Initializing the stemming by the porter stemmer , Stemming reduces the words in reviews like ‘running’ to ‘run’.
* Now join words back by string.
* And finally append in the corpus list.

from sklearn.feature\_extraction.text import CountVectorizer

cv = CountVectorizer(max\_features=1500)

X =cv.fit\_transform(corpus).toarray()

y = data.iloc[:,1].values

* Import the CountVectorizer by initializing with a parameter of max\_features= 1500.
* Now fix the corpus to CountVectorizer.
* Then extract the targeted variable by iloc[:,1]

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test =train\_test\_split(X,y,test\_size=0.20, random\_state=0)

* By using the train\_test\_split to split into the training and testing data.

y\_pred = classifier.predict(X\_test)

y\_pred

* This predicts the dependent variable .

from sklearn.metrics import accuracy\_score

from sklearn.metrics import precision\_score

from sklearn.metrics import recall\_score

score1 =accuracy\_score(y\_test,y\_pred)

score2 = accuracy\_score(y\_test,y\_pred)

score3 = recall\_score(y\_test,y\_pred)

print("---------SCORES--------")

print("Accuracy score is {}%".format(round(score1\*100,3)))

print("Precision score is {}%".format(round(score2\*100,3)))

print("recall score is {}%".format(round(score3\*100,3)))

* ---------SCORES--------
* Accuracy score is 76.5%
* Precision score is 76.5%
* recall score is 78.641%

from sklearn.metrics import confusion\_matrix

cm=confusion\_matrix(y\_test, y\_pred)

cm

* Its keeps the predictions into the confused matrix

array([[72, 25],

[22, 81]])

import matplotlib.pyplot as plt

import seaborn as sns

%matplotlib inline

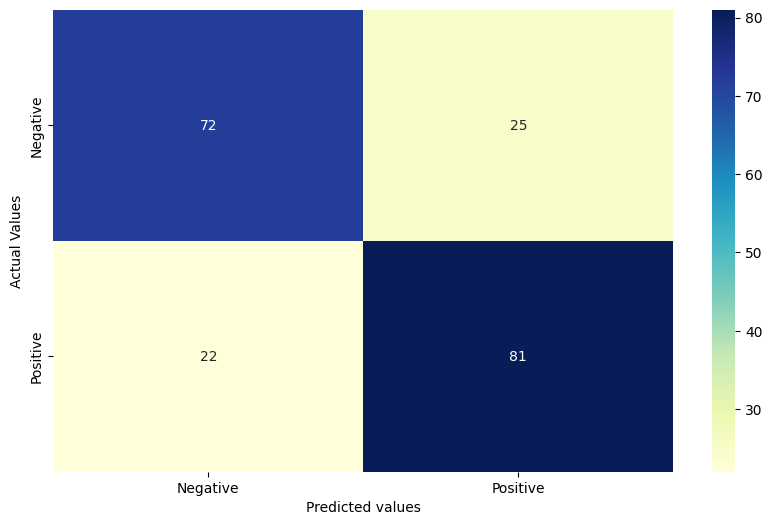
plt.figure(figsize =(10,6))

sns.heatmap(cm, annot=True, cmap="YlGnBu", xticklabels=['Negative','Positive'],yticklabels=['Negative','Positive'])

plt.xlabel('Predicted values')

plt.ylabel('Actual Values')

* It's plots in the graph of predicted values and actual values



from ssl import ALERT\_DESCRIPTION\_HANDSHAKE\_FAILURE

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\*100,3)))

if score>best\_accuracy:

best\_accuracy=score

alpha\_val =i

print('----------------------------------------------------')

print("The Best Accuracy Score 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 Score is 78.5% with alpha value as 0.

sample\_review ='The food is really bad.'

if predict\_sentiment(sample\_review):

print("Positive review")

else:

print("Negative review")

* It finally gives the output of the Negative review.

**CONCLUSION:** The SENTIMENTAL ANALYSIS works on the opinion of the customer feedback.