**Table of content**

|  |  |
| --- | --- |
| Topics | Page No. |
| Abstract | 2 |
| Introduction | 2 |
| Software and Libraries | 2-3 |
| Dataset Analysis | 4 |
| Machine Learning Models | 4 |
| Metrics | 5 |
| Algorithm | 6-7 |
| Results | 8-10 |
| Conclusion | 10 |
| Code | 11-17 |

**Table of figures/graphs**

|  |  |
| --- | --- |
| **Figure Name** | **Page NO** |
| Plotting of variance(Eigen) values for PCA | 7 |
| Output Plot Confusion Matrix | 8-9 |

**Abstract**

The reviews can be anything which are related to the food of the restaurant, staff of the restaurant and also overall

review of the restaurant. The model uses Support Vector Machine (SVM) , K-NN, SVM with PCA algorithm for classifying the reviews. The classified reviews are helpful for the restaurant to analyze their shortcomings in different areas and improve the quality of food and service in the restaurant. The reviews are stored on the cloud and can be accessed by the admin.

**Chapters**

1. **Introduction :**

In today’s world technology and automation in every sector is rapidly increasing. People rely more on mobile devices for almost every task in their day to day lives. Restaurant Business is a sector which has a very

large scope in automation and use of technology. At such times waiting for the waiter to take orders, delivering food, lengthy queues, etc. can be very displeasing for the customers of the restaurant. To overcome these problems a concept of automated restaurants using a system which uses LCD displays, mobile/tablet devices and a system for the chef to interact with customers is proposed. Using Machine Learning and Data Science predictions are made based on the reviews and other data of the customer can help make the dining experience better as well as it will help the restaurant to manage and make the restaurant grow.

In a restaurant while placing order, the customer has to ask the waiter whether a particular food item is available or not and after that he/she has to give the order. As well as several times it happens that customers

have to wait for the waiter to come to their table which is sometimes frustrating. Storing the statistical data of the restaurant is a very tedious task. There is need of managing the data of

inventory, customer orders and reviews, staff, payroll.

1. **Software and Libraries:**

This project was designed using the Python coding language on a Jupyter Notebook editor platform and PyCharm IDE. Python is a high-level programming language which is developed using C programming language. The vast collection of open source libraries that support Python especially in linear algebra computations makes the development and use of many ML models simplified and user friendly. The libraries used in our project are as follows:

* 1. **Pandas :**

The pandas package is the most important tool at the disposal of Data Scientists and Analysts working in Python today. The powerful machine learning and glamorous visualization tools may get all the attention, but pandas is the backbone of most data projects. Through pandas, you get acquainted with your data by cleaning, transforming, and analysing it.

* 1. **NumPy:**

NumPy is a module for Python. It is an extension module for Python, mostly written in C. This makes sure that the precompiled mathematical and numerical functions and functionalities of Numpy guarantee great execution speed. NumPy enriches the programming language Python with powerful data structures, implementing multi-dimensional arrays and matrices. These data structures guarantee efficient calculations with matrices and arrays.

* 1. **Sklearn (Scikit-learn):**

Scikit-learn provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python. It is an open source software machine learning library which consists of various regression, classification and clustering algorithms.

* 1. **Matplotlib:**

Matplotlib is a plotting library which consists of the NumPy extension. It is used to visualize the vectors using 2 dimensional or 3 dimensional plots and analyze the data in more effective way.

* 1. **NLTK:**

The NLTK module is a huge toolkit designed to help you with the entire Natural Language Processing (NLP) approach. NLTK will provide you with everything from splitting paragraphs to sentences, splitting words, identifying the part of speech, highlighting themes, and even helping your machine understand what the text is about.

* 1. **re:**

Regular expressions (called REs, or regexes, or regex patterns) are essentially a tiny, highly specialized programming language embedded inside Python and made available through the [re](https://docs.python.org/3/library/re.html#module-re) module. Using this little language, you specify the rules for the set of possible strings that you want to match; this set might contain English sentences, or e-mail addresses, or TeX commands, or anything you like. You can also use REs to modify a string or to split it apart in various ways

1. **Dataset Analysis:**

**3.1 Normalisation:**

Normalization generally refers to a series of related tasks meant to put all text on a level playing field: converting all text to the same case (upper or lower), removing punctuation, converting numbers to their word equivalents, and so on. Normalization puts all words on equal footing, and allows processing to proceed uniformly.

* 1. **Tokenization:**

Tokenization is a step which splits longer strings of text into smaller pieces, or tokens. Larger chunks of text can be tokenized into sentences, sentences can be tokenized into words, etc.

**3.3 Principal Component Analysis (PCA):**

PCA is a decomposition technique used reduce the number of features of the dataset. It attempts to create dimensions sequentially based o the amount of information the feature captures.

**Machine Learning Models:**

**4.1 Support Vector Machine**

* 1. **K-Nearest Neighbours(K-NN)**

K-NN is a method of classifying objects used for both classification and regression techniques. It is a simple algorithm that stores all available cases and classifies new cases based on a similarity measure. A peculiarity of k\_NN algorithm is that it is sensitive to the local structure of the data.

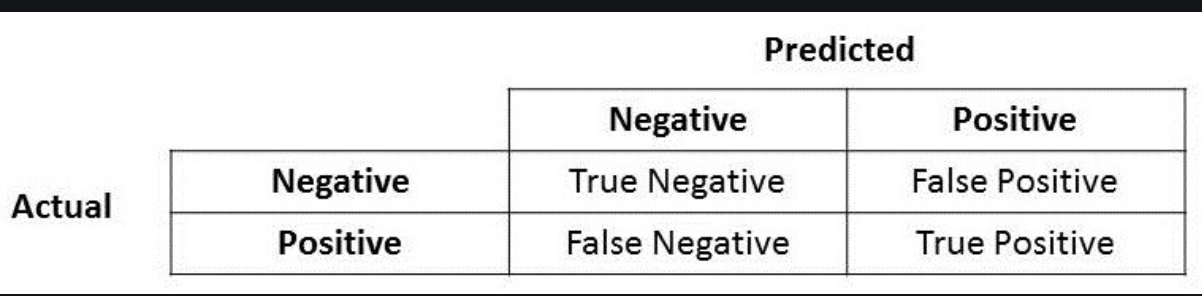
1. **Metrics:**

The metrics used to evaluate the models are :

* 1. **Accuracy Score:**

Accuracy is one metric for evaluating classification models. Informally, **accuracy** is the fraction of predictions our model got right. Formally, accuracy has the following definition:

Accuracy=Number of correct predictions/ Total number of predictions



* 1. **Confusion Matrix:**

A confusion matrix is a table that is often used to describe the performance of a classification model (or “classifier”) on a set of test data for which the true values are known. It allows the visualization of the performance of an algorithm.

1. **Algorithm:**

**General:**

* Import all the necessary libraries.
* Import the dataset which is in the TSV format into pandas data frame.
* Removing the quotes
* specify the boundary between separate, independent regions in plain text or other data streams. (Delimiting)
* Creating Bag of words

**SVM:**

* Importing the dataset
* Implementing General Algorithm
* Splitting the dataset into training and test samples
* Classifying the predictors and target
* Initializing Support Vector Machine and fitting the training data
* Predicting the classes for test set
* Comparing the actual classes and predictions
* Calculating the accuracy of the predictions

**K-nn**:

* Importing the dataset
* Implementing general algorithm
* Splitting the dataset into training and test samples
* Fitting the dataset
* Calculating the accuracy of the predictions

**SVM with PCA:**

* Importing the dataset
* Implementing General Algorithm
* Importing PCA
* Finding the best fit for n\_estimators
* Refitting Input x using after PCA
* Splitting the dataset into training and test samples
* Classifying the predictors and target
* Initializing Support Vector Machine and fitting the training data
* Predicting the classes for test set
* Comparing the actual classes and predictions
* Calculating the accuracy of the predictions

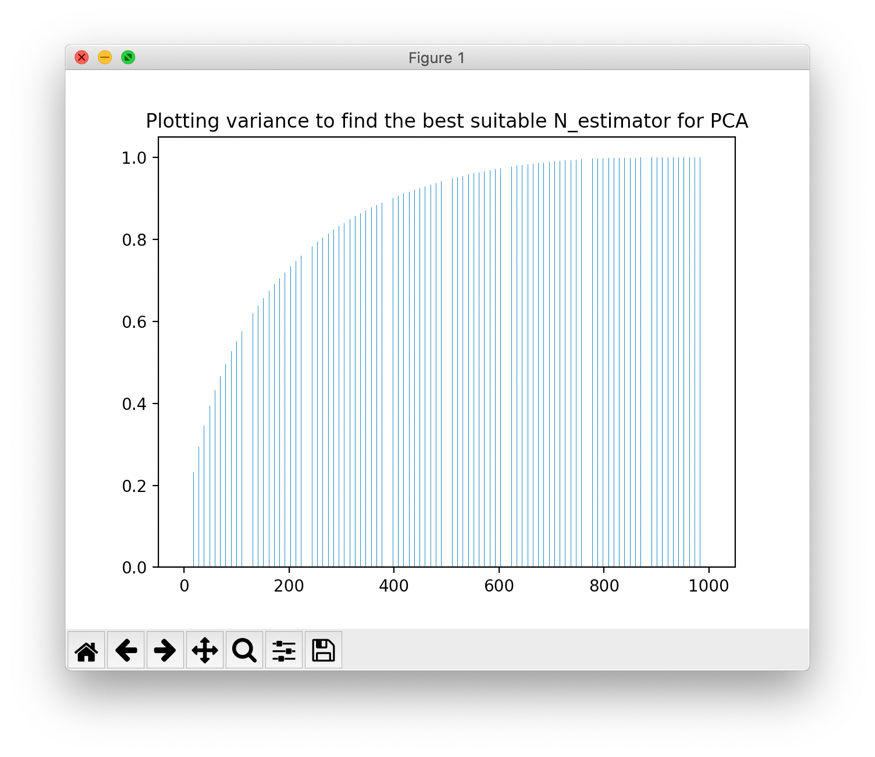


Fig: BarGraph to identify the best n\_estimator value for PCA

1. **Results:**

The Support Vector Machine(SVM) model performed to give an accuracy score of 86% and the Confusion Matrix is as below.

A screenshot of a cell phone

Description automatically generated

Fig 1 : Confusion Matrix of SVM

The K\_Nearest\_Neighbors (K-nn) model performed to give an accuracy score of 92.80% and the ConfussionMatrix is shown below.

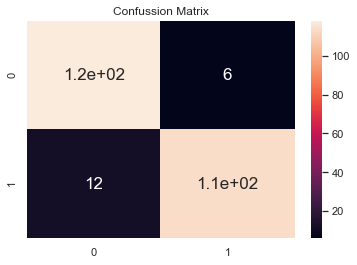


Fig 2: Confusion Matrix of K-nn

The Support Vector Machine(SVM) model with PCA performed to give an accuracy score of 86% and the Confusion Matrix is as below.

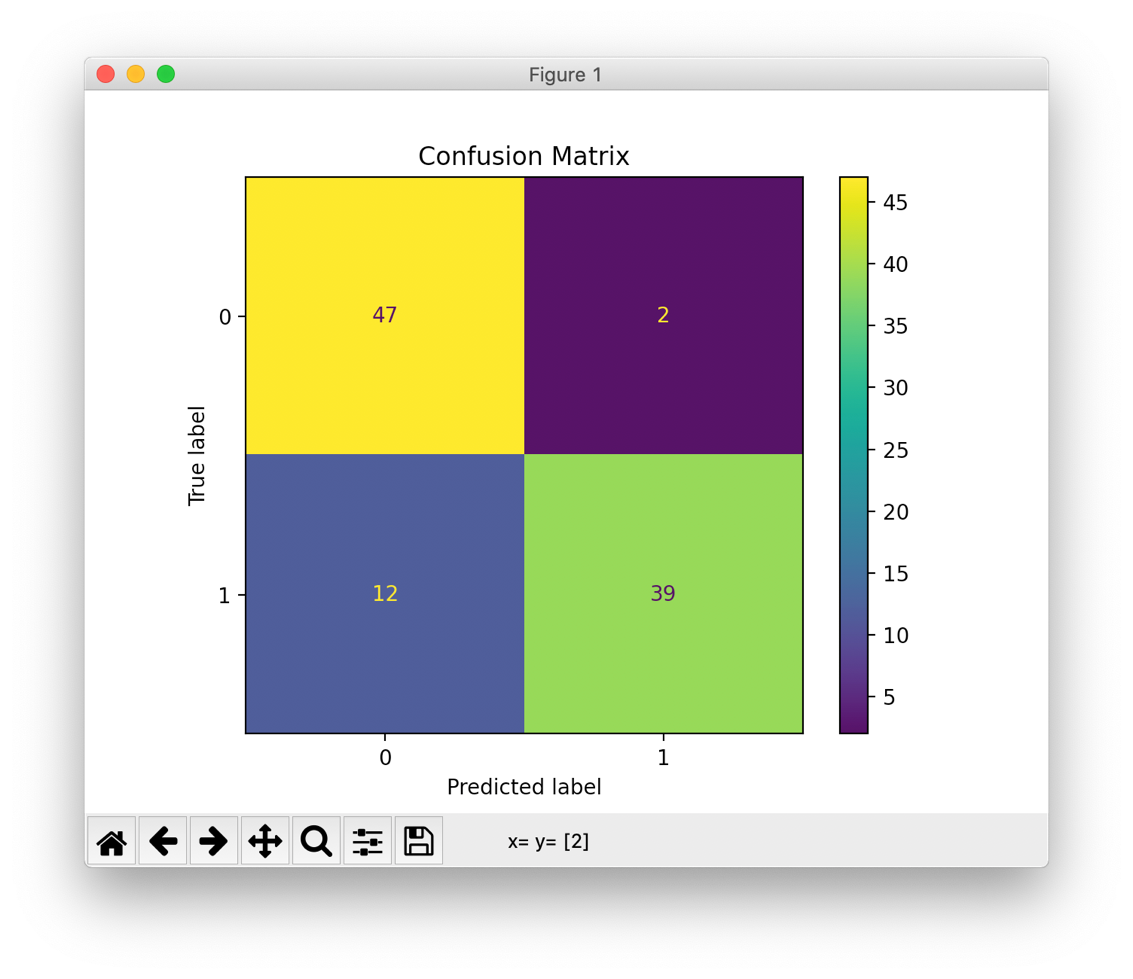


Fig : Confusion Matrix of SVM with PCA

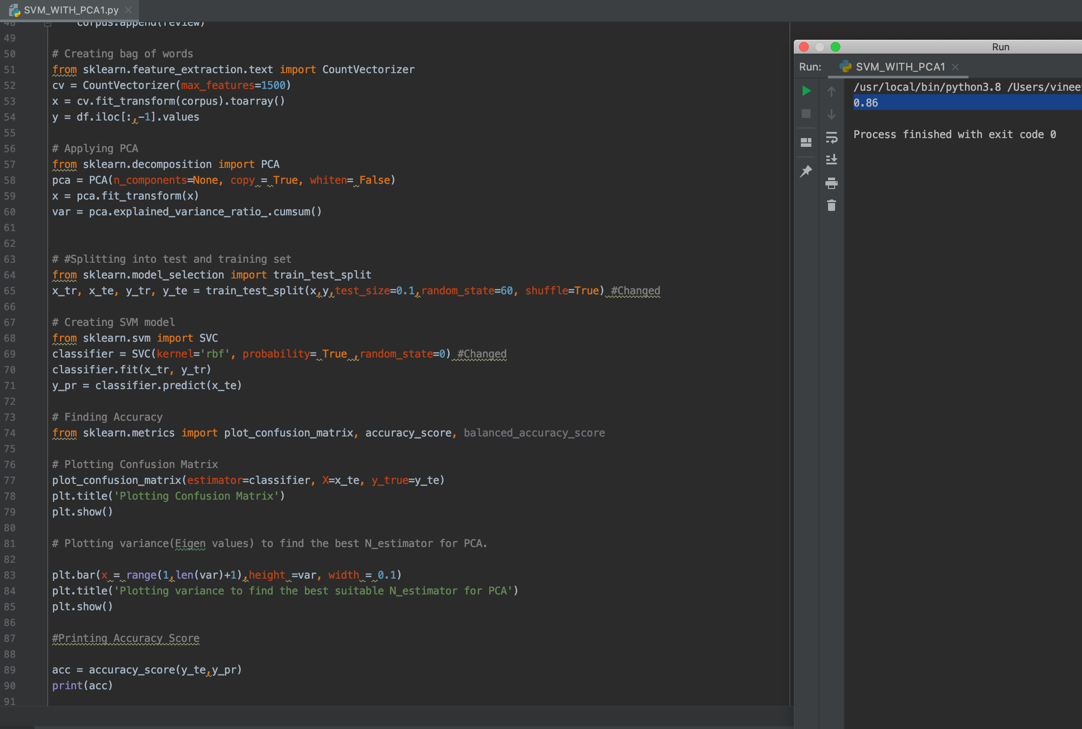


Fig: Accuracy Output of SVM with PCA model

**Conclusion**

Thus, in this project an efficient and user-friendly method is proposed which will provide automated systems in the restaurant and solve problems faced by the restaurants using technologies like Android, Web Development and Machine Learning. Interactive User Interfaces for the customers and restaurant staff will be

provided and customers can order food directly through the module without interacting with waiter. Using Machine Learning Models prediction of the food preferred by the customers and also information necessary for the restaurant to grow in business through customer reviews and other data. The system saves a lot of time of the customer as well as the restaurant staff and helps the restaurant in many ways

**Complete code**

**SVM:**

**import numpy as np**

**import pandas as pd**

**import matplotlib.pyplot as plt**

**import nltk**

**import re**

**from collections import Counter**

**from nltk.tokenize import RegexpTokenizer**

**from nltk.corpus import stopwords**

**stop\_words=set(stopwords.words('english'))**

**from nltk import word\_tokenize**

**from nltk.stem.wordnet import WordNetLemmatizer**

**lem = WordNetLemmatizer()**

**from nltk.stem.porter import PorterStemmer**

**stem = PorterStemmer()**

**df = pd.read\_csv('nlp.tsv', delimiter='\t', quoting=3)**

**new\_dict = {"n't":"not", 'sooooo': 'so', 'soooooo': 'so', 'cant': 'cannot','andddd':'and', 'honeslty' : 'honestly', 'ohhh': 'oh','serivce': 'service','flavourful' : 'flavorful', 'Veggitarian' : 'vegetarian', 'delicioso': 'delicious','outta' : 'out of', 'transcendant': 'transcendent', 'connisseur': 'connoisseur' , 'absolutley': 'absolutely','gooodd':'good', 'Im': 'I am', 'cavier' : 'caviar', 'dont':'do not', 'vinegrette': 'vinaigrette ','perpared': 'prepared', 'fo':'of', 'accomodate': 'accommodate', 'lil': 'little', 'thats': 'that is','definately': 'definitely','restaraunt': 'restaurant', 'satifying': 'satisfying', 'pissd': 'pissed', 'devine' :'divine', 'temp': 'temperature','definately':'definitely',"ya'all":"you all", 'disapppointment': 'disappointment' }**

**corpus = []**

**for i in range(0,1000):**

**review = df['Review'][i]**

**review = re.sub('[^a-zA-Z!]', ' ', review)**

**review = review.lower()**

**words = review.split()**

**new\_word= []**

**for word in words:**

**if word in new\_dict:**

**word = new\_dict[word]**

**new\_word.append(word)**

**new\_text = " ".join(new\_word)**

**tokens = word\_tokenize(new\_text)**

**all\_stopwords = stopwords.words('english')**

**all\_stopwords.remove('not')**

**all\_stopwords.remove('no')**

**all\_stopwords.remove('but')**

**all\_stopwords.remove('won')**

**all\_stopwords.append('really')**

**all\_stopwords.append('come')**

**all\_stopwords.append('get')**

**filtered\_words = [lem.lemmatize(w, "v") for w in tokens if w not in all\_stopwords]**

**review = ' '.join(filtered\_words)**

**corpus.append(review)**

**# Creating a bag of words**

**from sklearn.feature\_extraction.text import CountVectorizer**

**cv = CountVectorizer(max\_features=1500)**

**x = cv.fit\_transform(corpus).toarray()**

**y = df.iloc[:,-1].values**

**# #Splitting the data into testing and training data set**

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

**x\_tr, x\_te, y\_tr, y\_te = train\_test\_split(x,y,test\_size=0.1,random\_state=60, shuffle=True)**

**# Creating SVM model**

**from sklearn.svm import SVC**

**classifier = SVC(kernel='rbf', probability= True ,random\_state=0)**

**classifier.fit(x\_tr, y\_tr)**

**y\_pr = classifier.predict(x\_te)**

**# Finding Accuracy**

**from sklearn.metrics import plot\_confusion\_matrix, accuracy\_score, balanced\_accuracy\_score**

**plot\_confusion\_matrix(estimator=classifier, X=x\_te, y\_true=y\_te)**

**acc = accuracy\_score(y\_te,y\_pr)**

**plt.title('Plotting Confusion Matrix')**

**plt.show()**

**print(acc)**

**KNN**

import numpy as np

import pandas as pd

# Import dataset

dataset = pd.read\_csv('Restaurant\_Reviews.tsv', delimiter = '\t')

import re

import nltk

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]', ' ', dataset['Review'][i])

review = review.lower()

review = review.split()

ps = PorterStemmer()

review = [ps.stem(word) for word in review

if not word in set(stopwords.words('english'))]

review = ' '.join(review)

corpus.append(review)

from sklearn.feature\_extraction.text import CountVectorizer

cv = CountVectorizer(max\_features = 1500)

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

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

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

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.25)

# Predicting the Test set results

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

# Fitting Random Forest Classification to the Training set

from sklearn.ensemble import RandomForestClassifier

# n\_estimators can be said as number of

model = RandomForestClassifier(n\_estimators = 501,

criterion = 'entropy')

model.fit(X\_train, y\_train)

from sklearn.metrics import plot\_confusion\_matrix,confusion\_matrix,accuracy\_score

Confusion\_Matrix = confusion\_matrix(y\_test,y\_pred)

Accuracy\_Score = accuracy\_score(y\_test,y\_pred)

# plotting and printing accuracy

import seaborn as sns

import matplotlib.pyplot as plt

sns.set()

df\_cm = pd.DataFrame(Confusion\_Matrix, range(2),

range(2))

sns.set(font\_scale=1)#for label size

sns.heatmap(df\_cm, annot=True,annot\_kws={"size": 17})# font size

print("Accuracy Score is :", Accuracy\_Score\*100)

plt.title('Confussion Matrix')

plt.show()

**SVM\_with\_PCA:**

import numpy as np  
import pandas as pd  
import matplotlib.pyplot as plt  
import nltk  
import re  
from collections import Counter  
from nltk.tokenize import RegexpTokenizer  
from nltk.corpus import stopwords  
stop\_words=set(stopwords.words('english'))  
from nltk import word\_tokenize  
from nltk.stem.wordnet import WordNetLemmatizer  
lem = WordNetLemmatizer()  
from nltk.stem.porter import PorterStemmer  
stem = PorterStemmer()  
  
  
df = pd.read\_csv('Restaurant\_Reviews.tsv', delimiter='\t', quoting=3)  
  
new\_dict = {"n't":"not", 'sooooo': 'so', 'soooooo': 'so', 'cant': 'cannot','andddd':'and', 'honeslty' : 'honestly', 'ohhh': 'oh','serivce': 'service','flavourful' : 'flavorful', 'Veggitarian' : 'vegetarian', 'delicioso': 'delicious','outta' : 'out of', 'transcendant': 'transcendent', 'connisseur': 'connoisseur' , 'absolutley': 'absolutely','gooodd':'good', 'Im': 'I am', 'cavier' : 'caviar', 'dont':'do not', 'vinegrette': 'vinaigrette ','perpared': 'prepared', 'fo':'of', 'accomodate': 'accommodate', 'lil': 'little', 'thats': 'that is','definately': 'definitely','restaraunt': 'restaurant', 'satifying': 'satisfying', 'pissd': 'pissed', 'devine' :'divine', 'temp': 'temperature','definately':'definitely',"ya'all":"you all", 'disapppointment': 'disappointment' }  
corpus = []  
  
for i in range(0,1000):  
 review = df['Review'][i]  
 review = re.sub('[^a-zA-Z!]', ' ', review)  
 review = review.lower()  
 words = review.split()  
 # print(words)  
 new\_word= []  
 for word in words:  
 #print(word)  
 if word in new\_dict:  
 word = new\_dict[word]  
 new\_word.append(word) # a new list of standard words  
 new\_text = " ".join(new\_word)  
 tokens = word\_tokenize(new\_text)  
 all\_stopwords = stopwords.words('english')  
 all\_stopwords.remove('not')  
 all\_stopwords.remove('no')  
 all\_stopwords.remove('but')  
 all\_stopwords.remove('won')  
  
 all\_stopwords.append('really')  
 all\_stopwords.append('come')  
 all\_stopwords.append('get')  
  
 filtered\_words = [lem.lemmatize(w, "v") for w in tokens if w not in all\_stopwords]  
 review = ' '.join(filtered\_words)  
 corpus.append(review)  
  
# Creating bag of words  
from sklearn.feature\_extraction.text import CountVectorizer  
cv = CountVectorizer(max\_features=1500)  
x = cv.fit\_transform(corpus).toarray()  
y = df.iloc[:,-1].values  
  
# Applying PCA  
from sklearn.decomposition import PCA  
pca = PCA(n\_components=None, copy = True, whiten= False)  
x = pca.fit\_transform(x)  
var = pca.explained\_variance\_ratio\_.cumsum()  
  
  
# #Splitting into test and training set  
from sklearn.model\_selection import train\_test\_split  
x\_tr, x\_te, y\_tr, y\_te = train\_test\_split(x,y,test\_size=0.1,random\_state=60, shuffle=True) #Changed  
  
# Creating SVM model  
from sklearn.svm import SVC  
classifier = SVC(kernel='rbf', probability= True ,random\_state=0) #Changed  
classifier.fit(x\_tr, y\_tr)  
y\_pr = classifier.predict(x\_te)  
  
# Finding Accuracy  
from sklearn.metrics import plot\_confusion\_matrix, accuracy\_score, balanced\_accuracy\_score  
  
# Plotting Confusion Matrix  
plot\_confusion\_matrix(estimator=classifier, X=x\_te, y\_true=y\_te)  
plt.title('Plotting Confusion Matrix')  
plt.show()  
  
# Plotting variance(Eigen values) to find the best N\_estimator for PCA.  
  
plt.bar(x = range(1,len(var)+1),height =var, width = 0.1)  
plt.title('Plotting variance to find the best suitable N\_estimator for PCA')  
plt.show()  
  
#Printing Accuracy Score  
  
acc = accuracy\_score(y\_te,y\_pr)  
print(acc)