**PROJECT TITLE:-**

**PREDICTION OF QUALITY OF WINE USING MACHINE LEARNING ALGORITHMS**

**PREPARED BY-**SHASHI NANDAN PRASAD

**INSTITUTE-**NIT JAMSHEDPUR

**BRANCH**-PRODUCTION AND INDUSTRIAL ENGINEERING

**ROLL NO**-2017UGPI015

**CONTENTS**

1. [Introduction 2](#_TOC_250007)
2. Problem Statement 3
3. [Xgboost Classifier 3](#_TOC_250006)
4. [Linear Discriminant Classifier(LDA) and Logistic Regression. 4](#_TOC_250005)
5. [Kernel SVM….. 5](#_TOC_250004)
6. K-Nearest neighbor classifier… 7
7. [Naive bayes classifier… 8](#_TOC_250002)
8. [Decision tree classifier 9](#_TOC_250001)

9.[Random Forest classifier……………………………..................10](#_TOC_250000)

[10.Machine learning model……………………………………….11](#_TOC_250000)

[11.Result and Conclusion………………………………………....12](#_TOC_250000)

[12.Annexure…... 12](#_TOC_250000)

**1.INTRODUCTION**

The aim of this project is to predict the quality of wine on a scale of 3-8 which is the target variable i.e ouputs.The dataset used here is of “Wine Quality Data set” which is picked from UCI Machine Repository.In this dataset, input variables are fixed acidity,volatile acidity,citric acid,residual sugar,chlorides,free sulphates,alcohol.

The ouput variable is quality wine which is scale from 3-8 i.e[ 3,4,5,6,7,8].Higher value represents better quality of wine.

In this paper,I am going to explain the steps I had taken for predicting quality of wine using different Classification techniques and among them which is the best and optimum classification technique for this prediction of quality of wine.

Classification techniques used here are as follows:-

* Xgboost Classifier
* Linear Discriminant Analysis (LDA)
* Logistic Regression using LDA
* Kernel SVM
* K-Nearest Neighbors
* Naïve Bayes Classifier
* Decision Tree Classifier
* Random Forest Classifier

I also used Feature selection Technique to predict the top 5 features related to quality of wine.

**2.)PROBLEM STATEMENT**

Our objective is to predict the quality of wine using various Machine Learning Algorithms by applying various classification Techniques.

**3.)XGBOOST CLASSIFIER**

eXtreme Gradient Boosting or [XGBoost](https://xgboost.readthedocs.io/en/latest/) is a library of gradient boosting algorithms optimized for modern data science problems and tools. It leverages the techniques mentioned with boosting and comes wrapped in an easy to use library. Some of the major benefits of XGBoost are that its highly scalable/parallelizable, quick to execute, and typically outperforms other algorithms.

**4.)LINEAR DISCRIMINANT ANALYSIS(LDA) and LOGISTIC REGRESSION**

**Linear discriminant analysis** (**LDA**), **normal discriminant analysis** (**NDA**), or **discriminant function analysis** is a generalization of **Fisher's linear discriminant**, a method used in [statistics](https://en.wikipedia.org/wiki/Statistics), [pattern recognition](https://en.wikipedia.org/wiki/Pattern_recognition), and [machine learning](https://en.wikipedia.org/wiki/Machine_learning) to find a [linear combination](https://en.wikipedia.org/wiki/Linear_combination) of [features](https://en.wikipedia.org/wiki/Features_(pattern_recognition)) that characterizes or separates two or more classes of objects or events. The resulting combination may be used as a [linear classifier](https://en.wikipedia.org/wiki/Linear_classifier), or, more commonly, for [dimensionality reduction](https://en.wikipedia.org/wiki/Dimensionality_reduction) before later [classification](https://en.wikipedia.org/wiki/Statistical_classification).

**Assumptions:-**

* [Multivariate normality](https://en.wikipedia.org/wiki/Multivariate_normal_distribution): Independent variables are normal for each level of the grouping variable.
* Homogeneity of variance/covariance ([homoscedasticity](https://en.wikipedia.org/wiki/Homoscedasticity)): Variances among group variables are the same across levels of predictors. Can be tested with [Box's M](https://en.wikipedia.org/wiki/Box%27s_M_test) statistic.[[9]](https://en.wikipedia.org/wiki/Linear_discriminant_analysis#cite_note-green-9) It has been suggested, however, that linear discriminant analysis be used when covariances are equal, and that [quadratic discriminant analysis](https://en.wikipedia.org/wiki/Quadratic_classifier#Quadratic_discriminant_analysis) may be used when covariances are not equal.
* [Multicollinearity](https://en.wikipedia.org/wiki/Multicollinearity): Predictive power can decrease with an increased correlation between predictor variables.
* [Independence](https://en.wikipedia.org/wiki/Statistical_independence): Participants are assumed to be randomly sampled, and a participant's score on one variable is assumed to be independent of scores on that variable for all other participants.

In [statistics](https://en.wikipedia.org/wiki/Statistics), the **logistic model** (or **logit model**) is used to model the probability of a certain class or event existing such as pass/fail, win/lose, alive/dead or healthy/sick. This can be extended to model several classes of events such as determining whether an image contains a cat, dog, lion, etc. Each object being detected in the image would be assigned a probability between 0 and 1 and the sum adding to one.

Logistic regression is a [statistical model](https://en.wikipedia.org/wiki/Statistical_model) that in its basic form uses a [logistic function](https://en.wikipedia.org/wiki/Logistic_function) to model a [binary](https://en.wikipedia.org/wiki/Binary_variable) [dependent variable](https://en.wikipedia.org/wiki/Dependent_variable), although many more complex [extensions](https://en.wikipedia.org/wiki/Logistic_regression#Extensions) exist. In [regression analysis](https://en.wikipedia.org/wiki/Regression_analysis), **logistic regression**[[1]](https://en.wikipedia.org/wiki/Logistic_regression" \l "cite_note-1) (or **logit regression**) is [estimating](https://en.wikipedia.org/wiki/Estimation_theory) the parameters of a logistic model (a form of [binary regression](https://en.wikipedia.org/wiki/Binary_regression)). Mathematically, a binary logistic model has a dependent variable with two possible values, such as pass/fail which is represented by an [indicator variable](https://en.wikipedia.org/wiki/Indicator_variable), where the two values are labeled "0" and "1".

**5.)KERNEL SVM**

In [machine learning](https://en.wikipedia.org/wiki/Machine_learning), **kernel methods** are a class of algorithms for [pattern analysis](https://en.wikipedia.org/wiki/Pattern_analysis), whose best known member is the [support vector machine](https://en.wikipedia.org/wiki/Support_vector_machine) (SVM). The general task of pattern analysis is to find and study general types of relations (for example [clusters](https://en.wikipedia.org/wiki/Cluster_analysis), [rankings](https://en.wikipedia.org/wiki/Ranking), [principal components](https://en.wikipedia.org/wiki/Principal_components), [correlations](https://en.wikipedia.org/wiki/Correlation), [classifications](https://en.wikipedia.org/wiki/Statistical_classification)) in datasets. For many algorithms that solve these tasks, the data in raw representation have to be explicitly transformed into [feature vector](https://en.wikipedia.org/wiki/Feature_vector) representations via a user-specified *feature map*: in contrast, kernel methods require only a user-specified *kernel*, i.e., a [similarity function](https://en.wikipedia.org/wiki/Similarity_function) over pairs of data points in raw representation.

Kernel methods owe their name to the use of [kernel functions](https://en.wikipedia.org/wiki/Positive-definite_kernel), which enable them to operate in a high-dimensional, *implicit* [feature space](https://en.wikipedia.org/wiki/Feature_space) without ever computing the coordinates of the data in that space, but rather by simply computing the [inner products](https://en.wikipedia.org/wiki/Inner_product) between the [images](https://en.wikipedia.org/wiki/Image_(mathematics)) of all pairs of data in the feature space. This operation is often computationally cheaper than the explicit computation of the coordinates. This approach is called the "kernel trick".[[1]](https://en.wikipedia.org/wiki/Kernel_method#cite_note-1) Kernel functions have been introduced for sequence data, [graphs](https://en.wikipedia.org/wiki/Graph_kernel), text, images, as well as vectors.

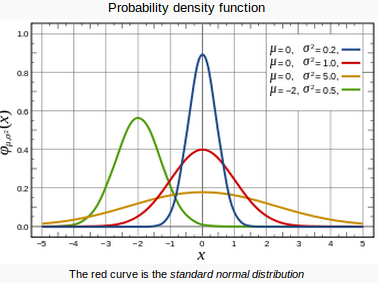
**MATHEMATICAL EXPRESSION :-**

http://2.bp.blogspot.com/-vXcuTGyGnuo/UyUxxDCBgpI/AAAAAAAAAOY/R_q5AXD98Tw/s1600/Screenshot+from+2014-03-16+01:08:02.png

**X-Distance from landmark**

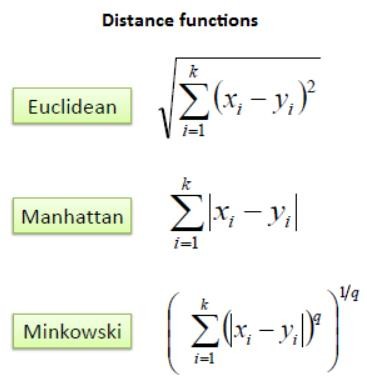
**X’-Landmark**

**sigma** is, as usually defined in a Gaussian Distribution, is standard deviation.  It determines the width for Gaussian distribution, as shown in the following:-



**6.)K-NEAREST NEIGHBORS CLASSIFIER**

In [pattern recognition](https://en.wikipedia.org/wiki/Pattern_recognition), the ***k*-nearest neighbors algorithm** (***k*-NN**) is a [non-parametric](https://en.wikipedia.org/wiki/Non-parametric_statistics) method used for [classification](https://en.wikipedia.org/wiki/Statistical_classification) and [regression](https://en.wikipedia.org/wiki/Regression_analysis).[[1]](https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm#cite_note-1) In both cases, the input consists of the *k* closest training examples in the [feature space](https://en.wikipedia.org/wiki/Feature_space). The output depends on whether *k*-NN is used for classification or regression

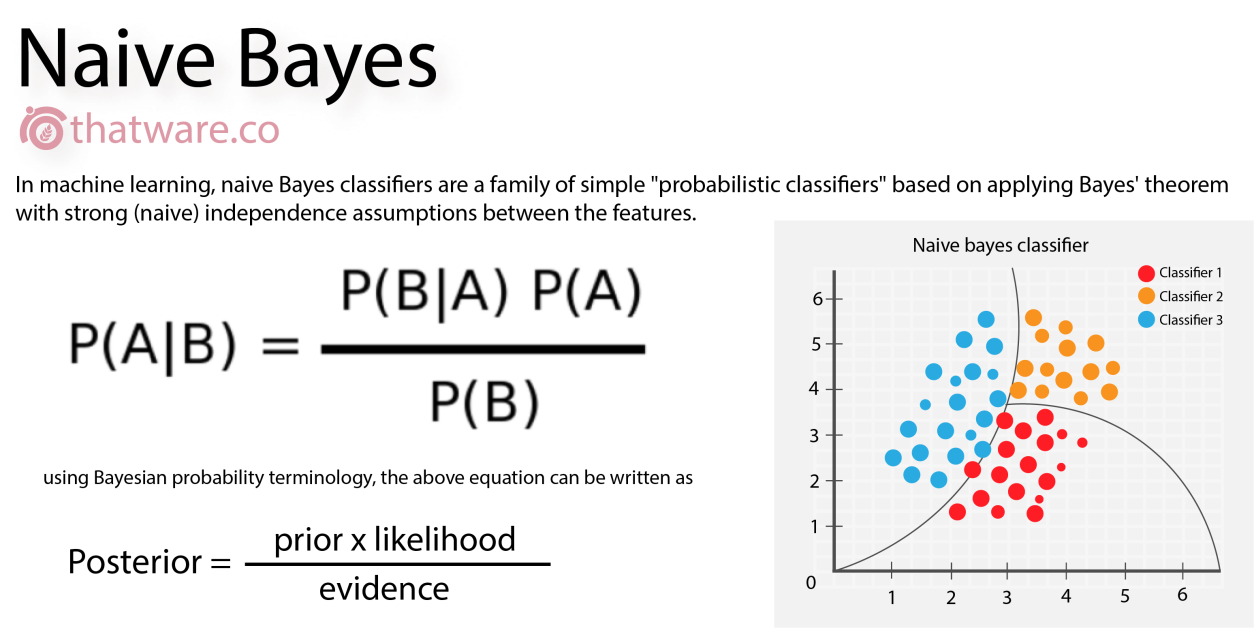


Choosing the optimal value for K is best done by first inspecting the data. In general, a large K value is more precise as it reduces the overall noise; however, the compromise is that the distinct boundaries within the feature space are blurred.

* In *k-NN classification*, the output is a class membership. An object is classified by a plurality vote of its neighbors, with the object being assigned to the class most common among its *k* nearest neighbors (*k* is a positive [integer](https://en.wikipedia.org/wiki/Integer), typically small). If *k* = 1, then the object is simply assigned to the class of that single nearest neighbor.
* In *k-NN regression*, the output is the property value for the object. This value is the average of the values of *k* nearest neighbors

**7.)NAiVE BAYES CLASSIFIER**

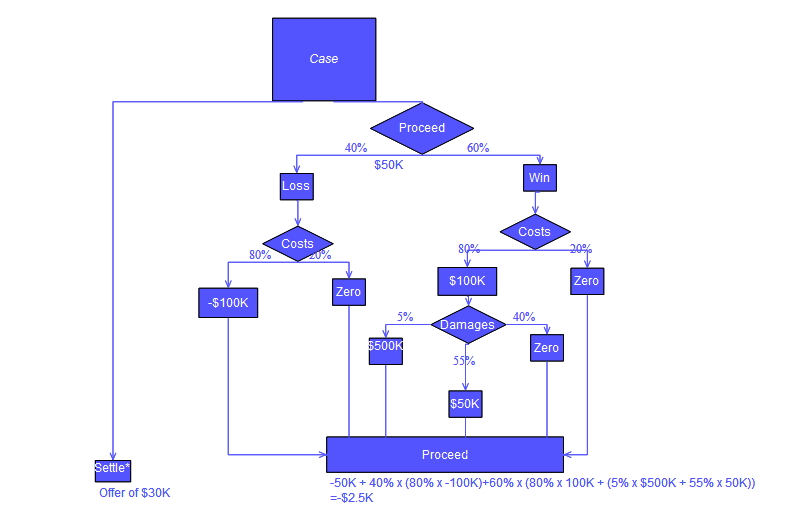
In [machine learning](https://en.wikipedia.org/wiki/Machine_learning), **naïve Bayes classifiers** are a family of simple "[probabilistic classifiers](https://en.wikipedia.org/wiki/Probabilistic_classification)" based on applying [Bayes' theorem](https://en.wikipedia.org/wiki/Bayes%27_theorem) with strong (naïve) [independence](https://en.wikipedia.org/wiki/Statistical_independence) assumptions between the features. They are among the simplest [Bayesian network](https://en.wikipedia.org/wiki/Bayesian_network) models.[[1]](https://en.wikipedia.org/wiki/Naive_Bayes_classifier#cite_note-1) But they could be coupled with [Kernel density estimation](https://en.wikipedia.org/wiki/Kernel_density_estimation) and achieve higher accuracy levels



**8.)DECISION TREE CLASSIFIER**

A decision tree is a [decision support](https://en.wikipedia.org/wiki/Decision_support_system) tool that uses a [tree-like](https://en.wikipedia.org/wiki/Tree_(graph_theory)) [model](https://en.wikipedia.org/wiki/Causal_model) of decisions and their possible consequences, including [chance](https://en.wikipedia.org/wiki/Probability) event outcomes, resource costs, and [utility](https://en.wikipedia.org/wiki/Utility). It is one way to display an [algorithm](https://en.wikipedia.org/wiki/Algorithm) that only contains conditional control statements.

Decision trees are commonly used in [operations research](https://en.wikipedia.org/wiki/Operations_research), specifically in [decision analysis](https://en.wikipedia.org/wiki/Decision_analysis), to help identify a strategy most likely to reach a [goal](https://en.wikipedia.org/wiki/Goal), but are also a popular tool in [machine learning](https://en.wikipedia.org/wiki/Decision_tree_learning).



* Information regarding the dependent variable is done by splitting the dataset of dependent variable using “**Information Entropy**”.
* If at further splitting,it can’t get more information ,there it stops.
* Algorithm finds the optimal split of dataset into these leaves called as “**Terminal Leaves**”.
* We take average of each points of each terminal leaves
* It is a Non-linear,non-continuous classification model
* Suitable for 2D graph Visualisation.

**9.)RANDOM FOREST CLASSIFIER**

**Random forests** or **random decision forests** are an [ensemble learning](https://en.wikipedia.org/wiki/Ensemble_learning) method for [classification](https://en.wikipedia.org/wiki/Statistical_classification), [regression](https://en.wikipedia.org/wiki/Regression_analysis) and other tasks that operate by constructing a multitude of [decision trees](https://en.wikipedia.org/wiki/Decision_tree_learning) at training time and outputting the class that is the [mode](https://en.wikipedia.org/wiki/Mode_(statistics)) of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of [overfitting](https://en.wikipedia.org/wiki/Overfitting) to their [training set](https://en.wikipedia.org/wiki/Test_set).

**ENSEMBLE LEARNING:-**

* It refers to collection of Machine Learning algorithms and form a specific ML algorithm termed as “Ensemble Learning”
* n\_estimators parameter help to detect overfitting and help the machine to predict better optimum result.

**10.)MACHINE LEARNING MODEL**

For designing a model following steps were taken:-

a) **Dataset Source(Kaggle):-**

<https://www.kaggle.com/uciml/red-wine-quality-cortez-et-al-2009>

b.) **Data Preparation:** The data was combined in a single csv file for further processing. Our aim was to predict the “**quality**” of wine, so it became our dependent variable. A correlation matrix was checked to find the relation among the parameters. The following variables were considered for model making-

* Fixed acidity
* Volatile acidity
* Citric acid
* Residual sugar
* Chlorides
* Free suphur dioxide
* Total sulphur dioxide
* Density
* PH
* Sulphate
* Alcohol

**c)Choosing a model:** Different models were trained on the train set and its overall accuracy i.e mean\_accuracy was predicted by using “**K-FOLD CROSS VALIDATION”** technique:-

**a)Xgboost Classifier**

**Mean\_Accuracy=0.6568845462713387**

**b)Logistic Regression using LDA**

**Mean\_Accuracy=0.584428346810422**

**c)Kernel SVM**

**Mean\_Accuracy=0.5969564240790657**

**d)K-nearest neighbor classifier**

**Mean\_Accuracy=0.5343384995507637**

**e)Naïve Bayes Classifier**

**Mean\_Accuracy=0.5889038634321653**

**f)Decision tree classifier**

**Mean\_Accuracy=0.5844901168014376**

**g)Random Forest classifier**

**Mean\_Accuracy=0.5978829739442947**

**11.)RESULT AND CONCLUSION**

The best fit is 65.68% for **Xgboost Classifier,** which is quite good as compared to other classifier. Hence, This model can be utilized for the prediction of quality of wine.

**12.)Annexure**

import numpy as np

import matplotlib.pyplot as plt

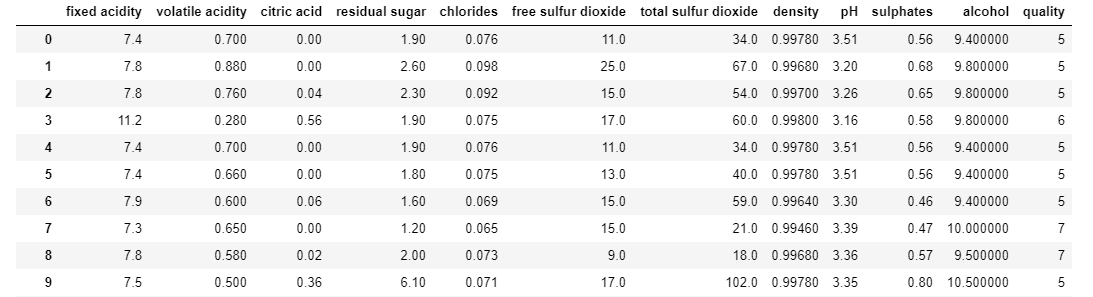
import pandas as pd

pd.set\_option('display.max\_rows', None)

pd.set\_option('display.max\_columns', None)

df=pd.read\_csv('../input/datasets\_4458\_8204\_winequality-red.csv')

df



#Checking of missing values

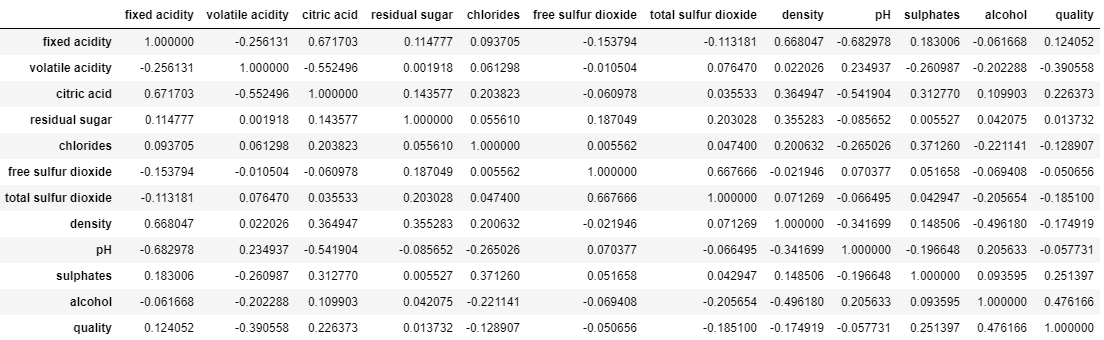
df.isnull().sum()



# Correlation Matrix

corr\_matrix = df.corr()

corr\_matrix

****

# Feature Selection

from sklearn.feature\_selection import SelectKBest

from sklearn.feature\_selection import f\_classif

x = df.iloc[:,0:-1] #independent columns

y = df.iloc[:,-1] #target column

#apply SelectKBest class to extract top 5 best features

bestfeatures = SelectKBest(score\_func=f\_classif, k=5)

fit = bestfeatures.fit(x,y)

dfscores = pd.DataFrame(fit.scores\_)

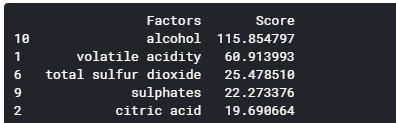
dfcolumns = pd.DataFrame(x.columns)

#concat two dataframes for better visualization

featureScores = pd.concat([dfcolumns,dfscores],axis=1)

featureScores.columns = ['Factors','Score'] #naming the dataframe columns

print(featureScores.nlargest(5,'Score')) #print 5 best features



x=df[["alcohol","volatile acidity","sulphates","citric acid","total sulfur dioxide"]]

y=df[["quality"]]

#splitting the dataset into training set and test set

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

x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y,test\_size=0.3,random\_state=0)

#**Xgboost classifier**

#fitting xgb to the training set

from xgboost import XGBClassifier

classifier=XGBClassifier()

classifier.fit(x\_train,y\_train)

#predicting the test set results

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

#applying the k-for cross validation

#best way to find accuracy of model

from sklearn.model\_selection import cross\_val\_score

accuracy=cross\_val\_score(estimator=classifier,X=x\_train,y=y\_train,cv=7)

accuracy.mean()

#Linear Discriminant Analysis (LDA)

#feature scaling

from sklearn.preprocessing import StandardScaler

sc\_x=StandardScaler()

x\_train=sc\_x.fit\_transform(x\_train)

x\_test=sc\_x.transform(x\_test)

#applying the LDA

from sklearn.discriminant\_analysis import LinearDiscriminantAnalysis as LDA

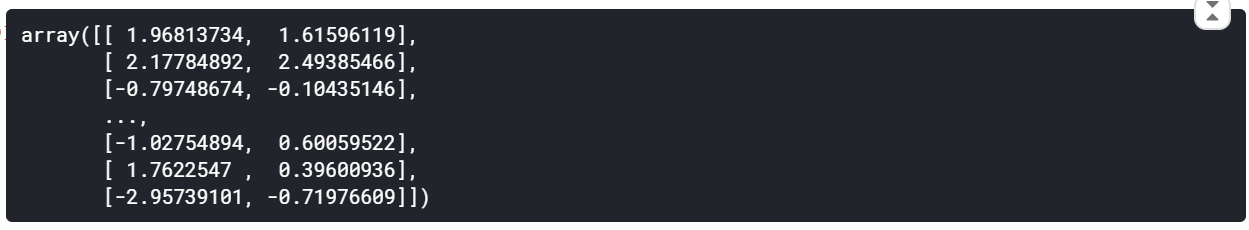
lda=LDA(n\_components=2)

x\_train=lda.fit\_transform(x\_train,y\_train)

x\_test=lda.transform(x\_test)

x\_train

#two new extracted features that will separate most of the classes of the target variable



#APPLYING LOGISTIC REGRESSION ON EXTRACTED FEATURES

#fitting the logistic regression to the training set

from sklearn.linear\_model import LogisticRegression

classifier=LogisticRegression(random\_state=0)

classifier.fit(x\_train,y\_train)

#predicting the test set results

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

#applying the k-for cross validation

#best way to find accuracy of model

from sklearn.model\_selection import cross\_val\_score

accuracy=cross\_val\_score(estimator=classifier,X=x\_train,y=y\_train,cv=7)

accuracy.mean()

#**KERNEL SVM**

#fitting the kernel svm to the training dataset

from sklearn.svm import SVC

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

classifier.fit(x\_train,y\_train)

#applying grid\_search to find optimal model and optimal parameters

from sklearn.model\_selection import GridSearchCV

parameters=[{'C':[1,10,100,1000],'kernel':['linear']},

{'C':[1,10,100,1000],'kernel':['rbf'],'gamma':[0.5,0.1,0.01,0.001,0.0001]}]

grid\_search=GridSearchCV(estimator=classifier,

param\_grid=parameters,

scoring='accuracy',

cv=7,

n\_jobs=-1)

grid\_search=grid\_search.fit(x\_train,y\_train)

best\_parameter=grid\_search.best\_params\_

best\_accuracy=grid\_search.best\_score\_

best\_parameter

E:\SCREENSHOTS_1\Screenshot (7).png

#fitting the kernel svm to the training dataset

from sklearn.svm import SVC

classifier=SVC(kernel='rbf',C=100,gamma=0.5)

classifier.fit(x\_train,y\_train)

#predicting the test set results

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

#applying the k-for cross validation

#best way to find accuracy of model

from sklearn.model\_selection import cross\_val\_score

accuracy=cross\_val\_score(estimator=classifier,X=x\_train,y=y\_train,cv=7)

accuracy.mean()

#**K-NEAREST NEIGHBOR CLASSIFIER**

#fitting the k nearest neighbour to the training set

from sklearn.neighbors import KNeighborsClassifier

classifier=KNeighborsClassifier(n\_neighbors=5,metric= 'minkowski',p=2)

classifier.fit(x\_train,y\_train)

#predicting the test set results

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

#applying the k-for cross validation

#best way to find accuracy of model

from sklearn.model\_selection import cross\_val\_score

accuracy=cross\_val\_score(estimator=classifier,X=x\_train,y=y\_train,cv=7)

accuracy.mean()

#**NAIVE BAYES CLASSIFIER**

#fitting the naive bayes to the training dataset

from sklearn.naive\_bayes import GaussianNB

classifier=GaussianNB()

classifier.fit(x\_train,y\_train)

#predicting the test set results

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

#applying the k-for cross validation

#best way to find accuracy of model

from sklearn.model\_selection import cross\_val\_score

accuracy=cross\_val\_score(estimator=classifier,X=x\_train,y=y\_train,cv=7)

accuracy.mean()

#**DECISION TREE CLASSIFIER**

#fitting the decision tree classifier to the training set

from sklearn.tree import DecisionTreeClassifier

classifier=DecisionTreeClassifier(criterion='entropy',random\_state=0)

classifier.fit(x\_train,y\_train)

#predicting the test set results

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

#applying the k-for cross validation

#best way to find accuracy of model

from sklearn.model\_selection import cross\_val\_score

accuracy=cross\_val\_score(estimator=classifier,X=x\_train,y=y\_train,cv=7)

accuracy.mean()

#**RANDOM FOREST CLASSIFIER**

from sklearn.ensemble import RandomForestClassifier

classifier=RandomForestClassifier(n\_estimators=100,criterion='entropy',random\_state=0)

classifier.fit(x\_train,y\_train)

#predicting the test set results

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

#best way to find accuracy of model

from sklearn.model\_selection import cross\_val\_score

accuracy=cross\_val\_score(estimator=classifier,X=x\_train,y=y\_train,cv=7)

accuracy.mean()

00fbcksasdadba

{\displaystyle p(C\_{k}\mid \mathbf {x} )={\frac {p(C\_{k})\ p(\mathbf {x} \mid C\_{k})}{p(\mathbf {x} )}}\,}