**DATA MINING: HOME LOAN PREDICTION**

A

Mini Project Report By

M.AARTHI 1602-19-733-309

D.SUPRIYA 1602-19-733-312

**BACHELOR OF ENGINEERING**

IN

**COMPUTER SCIENCE & ENGINEERING**



**Department of Computer Science & Engineering Vasavi College of Engineering (Autonomous) (Affiliated to Osmania University)**

**Ibrahimbagh, Hyderabad-31 2022**

**ACKNOWLEDGMENT**

We take this opportunity with pride and enormous gratitude, to express the deeply embedded feeling and gratefulness to our respectable guide Mrs. Sunitha Reddy, Department of Computer Science and Engineering. Whose guidance was unforgettable and innovative ideas as well as his constructive suggestions has made the presentation of my thesis a grand success. We are thankful to Dr. T. Adi Lakshmi, Head of Department (CSE), Vasavi College of Engineering for their help during our course work. Finally, at last but not least express our heart full thanks to the management of our college, Vasavi College of Engineering for providing the necessary arrangements and support to complete our project successively.

**CONTENTS**

1. [ABSTRACT 2](#_bookmark0)
2. [INTRODUCTION 3](#_bookmark1)
3. [LITERATURE REVIEW 4](#_bookmark2)
   1. [Data Mining and Classification 4](#_bookmark3)
4. [METHODOLOGY 6](#_bookmark4)

### [LINEAR REGRESSION](#_bookmark5)

### [Linear Regression Equation](#_bookmark6)

### [Linear Regression Formula](#_bookmark7)

### [Simple Linear Regression](#_bookmark8)

### [Least Square Regression Line or Linear Regression Line](#_bookmark9)

### [Properties of Linear Regression](#_bookmark10)

### [Regression Coefficient](#_bookmark11)

### [KNN: K-Nearest Neighbor 10](#_bookmark12)

### [Working of KNN Algorithm](#_bookmark13)

### [Example](#_bookmark14)

### [Pros](#_bookmark15)

### Cons

### [Applications of KNN](#_bookmark16)

### [Decision Tree 13](#_bookmark17)

### [Construction of Decision Tree](#_bookmark18)

### [Decision Tree Representation](#_bookmark19)

### [PROPOSED WORK 15](#_bookmark20)

### [Overall description and various classification](#_bookmark21)

### [Confusion Matrix](#_bookmark22)

### [IMPLEMENTATION 20](#_bookmark23)

### [OUTPUTS 29](#_bookmark24)

### [CONCLUSION 40](#_bookmark25)

### [REFERENCES 41](#_bookmark25)

# 1 ABSTRACT

This paper presents a study about HOME LOAN PREDICTION based on data mining methods to discover an effective way to predict loan. The objective of this paper is to compare and identify an accurate model to predict the Loan eligibility based on various customer records. Four data mining models are applied in this paper, i.e.Naive Bayes classifier, Decision tree. Furthermore, feature space is highly discussed in this paper due to its high influence on the efficiency and effectiveness of the learning process. To test the influence of feature space reduction, a hybrid between principal component analysis (PCA) and related data mining models is proposed, which applies a principle component analysis method to reduce the feature space. To evaluate the performance of these models, two widely used test data sets are used.

# 2 INTRODUCTION

### Dream Housing Finance company deals in all home loans. They have presence across all urban, semi urban and rural areas. Customer first apply for home loan after that company validates the customer eligibility for loan.

### The company wants to automate the loan eligibility process (real time) based on customer detail provided while filling online application form. These details are Gender, Marital Status, Education, Number of Dependents, Income, Loan Amount, Credit History and others. To automate this process, they have given a problem to identify the customers segments, those are eligible for loan amount so that they can specifically target these customers. Here they have provided a partial data set.

# 3 LITERATURE REVIEW

Classification and clustering are two widely used methods in data mining. Clustering methods aim to extract information from a data set to discover groups or clusters and describe the data set itself.

Classification, also known as supervised learning in machine learning, aims to classify unknown situations based on learning existing patterns and categories from the data set and subsequently predict future situations.

The training set, which is used to build the classifying structure, and the test set, which tends to assess the classifier, are commonly mentioned in classification tasks. This research mainly focuses on the classification and prediction of breast cancer.

Related works regarding supervised learning techniques and their application in breast cancer diagnosis are reviewed in this section.

## 3.1 Data Mining and Classification

There are numerous data mining methods which can be used for data handling tasks according to different goals and purposes.

In general, two types of data mining methods are most frequently mentioned, verification-oriented and discovery oriented. Verification-oriented methods are used for verifying and evaluating system hypotheses, such as goodness of fit test, tests of hypotheses, analysis of variance (ANOVA), etc. Discovery oriented methods try to find unknown patterns, which includes prediction methods and description methods. Description oriented methods include clustering, data summarization, data visualization, etc.

Prediction oriented methods include regression, and classification [12]. Regression methods, such as linear regression, regression trees, multivariate adaptive regression splines, etc., are applied to solve continuous response tasks [13-14]. For categorical

response data, classification is applied, such as decision tree, support vector machine (SVM), artificial neural networks (ANN), Naïve Bayes classifier, etc.

Similar to the manner in which humans learn from past experiences, computer systems learn from data sets. Theoretically, a classification model generally includes three parts:

1) a random vector x from an unknown distribution pattern, which is used as an input vector for a learning machine, 2) a response vector y related to input vector x, which performs like a decision vector, and 3) a learning machine that implements a certain rule to be trained from the input vector and decision vector.

# 4 METHODOLOGY

## LINEAR REGRESSION

Linear regression strives to show the relationship between two variables by applying a linear equation to observed data. One variable is supposed to be an independent variable, and the other is to be a dependent variable.

For example, a modeler might want to relate the weights of individuals to their heights using a linear regression model.

Before attempting to fit a linear model to observed data, a modeler should first determine whether or not there is a relationship between the variables of interest. This does not necessarily imply that one variable *causes* the other (for example, higher SAT scores do not *cause* higher college grades), but that there is some significant association between the two variables. A [scatterplot](http://www.stat.yale.edu/Courses/1997-98/101/scatter.htm) can be a helpful tool in determining the strength of the relationship between two variables.

If there appears to be no association between the proposed explanatory and dependent variables (i.e., the scatterplot does not indicate any increasing or decreasing trends), then fitting a linear regression model to the data probably will not provide a useful model. A valuable numerical measure of association between two variables is the [correlation](http://www.stat.yale.edu/Courses/1997-98/101/correl.htm) [coefficient,](http://www.stat.yale.edu/Courses/1997-98/101/correl.htm) which is a value between -1 and 1 indicating the strength of the association of the observed data for the two variables.

#### Linear Regression Equation

The measure of the extent of the relationship between two variables is shown by the **correlation coefficient**. The range of this coefficient lies between -1 to +1. This coefficient shows the strength of the association of the observed data for two variables.

A linear regression line equation is written in the form of:

#### Y = a + bX

where X is the independent variable and plotted along the x-axis Y is the dependent variable and plotted along the y-axis

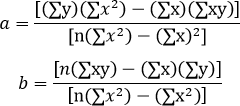
The slope of the line is b, and a is the intercept (the value of y when x = 0).

#### Linear Regression Formula

Linear regression shows the linear relationship between two variables. The equation of linear regression is similar to the slope formula what we have learned before in earlier classes such as [linear equations in two variables.](https://byjus.com/maths/linear-equations-in-two-variables/) It is given by;

Y= a + bX

Now, here we need to find the value of the slope of the line, b, plotted in scatter plot and the intercept, a.



#### Simple Linear Regression

The very most straightforward case of a single scalar predictor variable x and a single scalar response variable y is known as simple linear regression. The equation for this regression is represented by;

y=a+bx

The expansion to multiple and vector-valued predictor variables is known as [multiple](https://byjus.com/maths/multiple-regression/) [linear regression,](https://byjus.com/maths/multiple-regression/) also known as multivariable linear regression. The equation for this regression is represented by;

Y = a+bX

#### Least Square Regression Line or Linear Regression Line

The most popular method to fit a regression line in the XY plot is the method of least- squares. This process determines the best-fitting line for the noted data by reducing the sum of the squares of the vertical deviations from each data point to the line.

If a point rests on the fitted line accurately, then its perpendicular deviation is 0. Because the variations are first squared, then added, their positive and negative values will not be cancelled.



*Figure 1 Linear Regression*

Linear regression determines the straight line, called the least-squares regression line or LSRL, that best expresses observations in a [bivariate analysis](https://byjus.com/maths/bivariate-analysis/) of data set.

Suppose Y is a dependent variable, and X is an independent variable, then the population regression line is given by;

Y = B0+B1X

Where

B0 is a constant

B1 is the regression coefficient

If a random sample of observations is given, then the regression line is expressed by; ŷ = b0 + b1x

where b0 is a constant, b1 is the regression coefficient, x is the independent variable, and ŷ is the predicted value of the dependent variable.

#### Properties of Linear Regression

For the regression line where the regression parameters b0 and b1 are defined, the properties are given as:

* The line reduces the sum of squared differences between observed values and predicted values.
* The regression line passes through the mean of X and Y variable values
* The regression constant (b0) is equal to y-intercept the linear regression
* The regression coefficient (b1) is the slope of the regression line whichis equal to the average change in the dependent variable (Y) for a unit change in the independent variable (X).

#### Regression Coefficient

In the linear regression line, we have seen the equation is given by; Y = B0+B1X

Where

B0 is a constant

B1 is the regression coefficient

Now, let us see the formula to find the value of the regression coefficient. B1 = b1 = Σ [ (xi – x)(yi – y) ] / Σ [ (xi – x)2]

Where xi and yi are the observed data sets. And x and y are the mean value.

#### KNN: K-Nearest Neighbor

K-nearest neighbors (KNN) algorithm is a type of supervised ML algorithm which can be used for both classification as well as regression predictive problems. However, it is mainly used for classification predictive problems in industry. The following two properties would define KNN well −

* **Lazy learning algorithm** − KNN is a lazy learning algorithm because it does not have a specialized training phase and uses all the data for training while classification.
* **Non-parametric learning algorithm** − KNN is also a non-parametric learning algorithm because it doesn’t assume anything about the underlying data.

#### Working of KNN Algorithm

K-nearest neighbors (KNN) algorithm uses ‘feature similarity’ to predict the values of new datapoints which further means that the new data point will be assigned a value based on how closely it matches the points in the training set. We can understand its working with the help of following steps −

**Step 1** − For implementing any algorithm, we need dataset. So during the first stepof KNN, we must load the training as well as test data.

**Step 2** − Next, we need to choose the value of K i.e. the nearest data points. K can be any integer.

**Step 3** − For each point in the test data do the following −

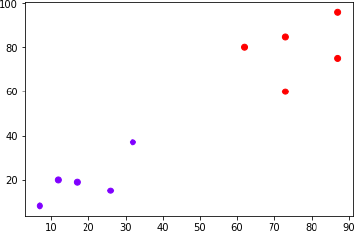
* **3.1** − Calculate the distance between test data and each row of training data with the help of any of the method namely: Euclidean, Manhattan or Hamming distance. The most commonly used method to calculate distance isEuclidean.
* **3.2** − Now, based on the distance value, sort them in ascending order.
* **3.3** − Next, it will choose the top K rows from the sorted array.
* **3.4** − Now, it will assign a class to the test point based on most frequent class of these rows.

**Step 4** − End

#### Example

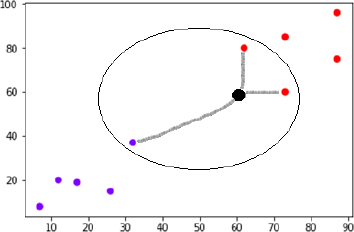
The following is an example to understand the concept of K and working of KNN algorithm −

Suppose we have a dataset which can be plotted as follows −



*Figure 2 KNN- Plotting*

Now, we need to classify new data point with black dot (at point 60,60) into blue or red class. We are assuming K = 3 i.e. it would find three nearest data points. It is shown in the next diagram −



*Figure 3 KNN- Finding Nearest Neighbors*

We can see in the above diagram the three nearest neighbors of the data point with black dot. Among those three, two of them lies in Red class hence the black dot will also be assigned in red class.

#### Pros

* It is very simple algorithm to understand and interpret.
* It is very useful for nonlinear data because there is no assumption about data in this algorithm.
* It is a versatile algorithm as we can use it for classification as well as regression.
* It has relatively high accuracy but there are much better supervisedlearning models than KNN.

#### Cons

* It is computationally a bit expensive algorithm because it stores all the training data.
* High memory storage required as compared to other supervisedlearning algorithms.
* Prediction is slow in case of big N.
* It is very sensitive to the scale of data as well as irrelevant features.

#### Applications of KNN

The following are some of the areas in which KNN can be applied successfully −

#### Banking System

KNN can be used in banking system to predict weather an individual is fit for loan approval? Does that individual have the characteristics similar to the defaulters one?

#### Calculating Credit Ratings

KNN algorithms can be used to find an individual’s credit rating by comparing with the persons having similar traits.

* 1. **Decision Tree**

Decision tree is the most powerful and popular tool for classification and prediction. A Decision tree is a flowchart like tree structure, where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node(terminal node) holds a class label.

#### Construction of Decision Tree

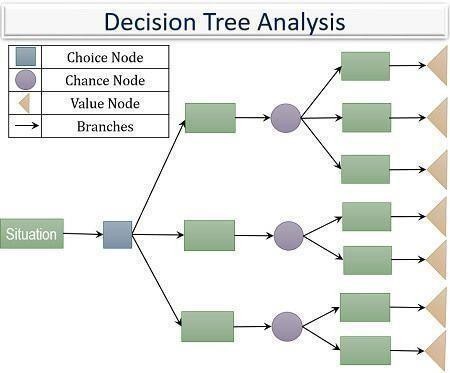
A tree can be *“learned”* by splitting the source set into subsets based on an attribute value test.

This process is repeated on each derived subset in a recursive manner called *recursive partitioning*.

The recursion is completed when the subset at a node all has the same value of the target variable, or when splitting no longer adds value to the predictions. The construction of decision tree classifier does not require any domain knowledge or parameter setting, and therefore is appropriate for exploratory knowledge discovery.

Decision trees can handle high dimensional data. In general decision tree classifier has good accuracy. Decision tree induction is a typical inductive approach to learn knowledge on classification.

#### Decision Tree Representation



*Figure 4 Decision Tree Representation*

Decision trees classify instances by sorting them down the tree from the root to some leaf node, which provides the classification of the instance. An instance is classified by starting at the root node of the tree, testing the attribute specified by this node, then moving down the tree branch corresponding to the value of the attribute as shown in the above figure. This process is then repeated for the subtree rooted at the new node.

**5 PROPOSED WORK**

Processing of data is essential to sort the dataset suitable for several classifiers in order to perform classification. Data in the real world is dirty; it can be cleaned by removing the incorrect and nosy data which is then integrated from multiple sources, known as meta data that removes duplicates and redundant data.

Data pre-processing can be achieved by various subsequent means such as reduction of data, data cleaning, data integration, data transformation and data discretization. The data set comprises of the following attributes such as Evenness of Cell Size, Evenness of Cell Shape, Clump Thickness, Marginal Adhesion, Single Epithelial Cell Size, Bare Nuclei, Bland Chromatin, Normal Nucleoli, and Mitoses. Data set, Wisconsin breast cancer is chosen from the UCI Machine Learning Repository.

It is essential to have a suitable Data Mining tool in order to carry out classification, prediction and extracting rules on the data available.10-fold cross validation is applied on all the dataset using different classifiers. The obtained results are calculated by evaluating the performance of the models using various classifiers for breast cancer using measurements like Accuracy, RMSE, Specificity, Sensitivity, Time taken to build the model, F-measure, ROC curve area and Kappa Statistics.

* 1. **Overall description and various classification**

methods used In this Paper we have selected the open source software WEKA, proficiently works with limited data. Few tasks offered by data mining like pre- processing of data, Classification, Clustering, Association and Visualization can be performed. Data set is fed in the form of Comma Separated Values-CSV format.

In general WEKA is used in medical analysis for preliminary collection of data. A decomposition method is applied to discover information from the Breast Cancer dataset. The extracted knowledge is used for prediction purposes. The framework

illustrated in Fig summarizes the suggested framework for Predicting Breast Cancer using Logistic Regression and Multiclass Classifiers.

Experimentations are carried out on the Home loan dataset taken from the UCI machine Learning Repository, fed to various classifiers like Simple Logistic-regression method, IBK, K-star, MLP, Random Forest, Decision table, Decision Tree.

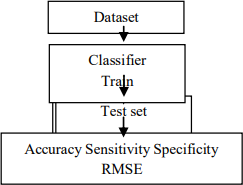
The obtained results are calculated and evaluated in terms of measures like Accuracy, RMSE, Specificity, Sensitivity, Time taken to build the model, F-measure, ROC curve area and Kappa Statistics.

Fig Frame work representing proposed approach 4.2 Methods used Logistic Regression: Multinomial logistic regression is used to build a trained model for predicting with a ridge estimator.

IBK: It is a K-nearest neighbor’s classifier and calculates distance weighting to select appropriate value of K based on cross validation. K-Star: K\*, an instance-based classifier. As determined by some similarity function classifies test instances based upon the model obtained after training instances similar to it.

MLP: A neural network, each entity accomplishes a biased weighted sum of inputs to them and passes this activation level through a transfer function to generate output. Logistic and hyperbolic tangent sigmoid functions are the most common activation functions in MLP. Random Forest: Random trees are formed in randomly and resembles like a forest and class is predicted from that.

Decision Table: Class is predicted from a simple decision table constructed by majority classifiers. Decision Trees: J48 algorithm is employed to construct the decision tree, starting from the root of the tree and proceeding down to its leaf nodes. Class label for a test item is obtained from a decision tree by starting at the root of the tree and moving through it until a leaf node, which provides the classification of the instance.



*Figure 5 Decision Tree Flow*

PART: A partial C4.5 decision tree is built in each repetition and makes the "best" leaf into a rule Class for generating a PART decision list by using separate-and-conquer strategy.

Multi-class Classifier: Multi-class datasets with 2-class classifiers are handled by a meta classifier. REP Tree: A decision tree is built by means of information gain and prunes it using reduced-error pruning. Datasets are huge in dimensions to solve actual domain classification problems.

Knowledge discovery consumes major part of time in order to mine the knowledge from those databases by the standard Machine Learning Techniques [2]. In this paper the hidden knowledge to predict patient’s suffering from Breast Cancer is extractedby analyzing the results obtained and are evaluated by considering various parameters and are explained in detail here

* + 1. The percentage of test instances that are correctly classified on a given test set is determined as the accuracy of a classifier.
    2. Sensitivity and Specificity is calculated from Confusion Matrix obtained in the model.
    3. Sensitivity is the proposition of the positive instances that are correctly identified

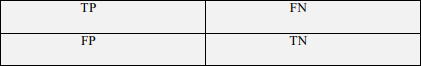
TP = (TP/ TP +FN) \* 100.

* + 1. Specificity is the proposition of the negative instances that are correctly identified

TN = (TN/TN + FP) \* 100.

* + 1. RMSE Root Mean Square Error is a measure of the difference between values predicted by a model and the values actually observed.
    2. F-measure, ROC curve area and Kappa Statistics are also calculated using Confusion Matrix with the help of WEKA tool.
  1. **Confusion Matrix**

A comparison is drawn between the actual class labels and the predicted class labels based on the class labels by the classifiers. The following describes the case when we deal with two-class classification problem [33]. The generated confusion matrix is 2 \* 2 matrixes.



*Figure 6 Confusion Matrix*

**6 IMPLEMENTATION**

import pandas as pd import numpy as np

import matplotlib.pyplot as plt

%matplotlib inline

data = pd.read\_csv("/content/train.csv") data.head()

data.info()

# checking missing values in each column of train dataset data.apply(lambda x: sum(x.isnull()),axis=0)

data['Gender'].value\_counts() data.Gender = data.Gender.fillna('Male') data['Married'].value\_counts() data.Married = data.Married.fillna('Yes')

data.Dependents = data.Dependents.fillna('0') data.Self\_Employed = data.Self\_Employed.fillna('No')

data.LoanAmount = data.LoanAmount.fillna(data.LoanAmount.mean()) data['Loan\_Amount\_Term'].value\_counts()

data.Loan\_Amount\_Term = data.Loan\_Amount\_Term.fillna(360.0) data['Credit\_History'].value\_counts()

data.Credit\_History = data.Credit\_History.fillna(1.0) data.apply(lambda x: sum(x.isnull()),axis=0)

# Splitting traing data

X = data.iloc[:, 1: 12].values y = data.iloc[:, 12].values

# Splitting the dataset into the 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 = 1/3, random\_state = 0) # Encoding categorical data

# Encoding the Independent Variable

from sklearn.preprocessing import LabelEncoder labelencoder\_X = LabelEncoder()

for i in range(0, 5):

X\_train[:,i] = labelencoder\_X.fit\_transform(X\_train[:,i]) X\_train[:,10] = labelencoder\_X.fit\_transform(X\_train[:,10])

# Encoding the Dependent Variable labelencoder\_y = LabelEncoder()

y\_train = labelencoder\_y.fit\_transform(y\_train)

# Encoding categorical data

# Encoding the Independent Variable

from sklearn.preprocessing import LabelEncoder, OneHotEncoder labelencoder\_X = LabelEncoder()

for i in range(0, 5):

X\_test[:,i] = labelencoder\_X.fit\_transform(X\_test[:,i]) X\_test[:,10] = labelencoder\_X.fit\_transform(X\_test[:,10]) # Encoding the Dependent Variable

labelencoder\_y = LabelEncoder()

y\_test = labelencoder\_y.fit\_transform(y\_test)

# Feature Scaling

from sklearn.preprocessing import StandardScaler sc = StandardScaler()

X\_train = sc.fit\_transform(X\_train) X\_test = sc.fit\_transform(X\_test)

Applying PCA

**# Applying PCA**

from sklearn.decomposition import PCA pca = PCA(n\_components = 2)

X\_train = pca.fit\_transform(X\_train) X\_test = pca.fit\_transform(X\_test)

explained\_variance = pca.explained\_variance\_ratio\_

Classification Algorithms

Logistic Regression

# Fitting 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)

# Measuring Accuracy from sklearn import metrics

print('The accuracy of Logistic Regression is: ', metrics.accuracy\_score(y\_pred, y\_test))

# Making the Confusion Matrix

from sklearn.metrics import confusion\_matrix cm = confusion\_matrix(y\_test, y\_pred)

# Visualising the Training set results

from matplotlib.colors import ListedColormap X\_set, y\_set = X\_train, y\_train

X1, X2 = np.meshgrid(np.arange(start = X\_set[:, 0].min() - 1, stop = X\_set[:, 0].max() + 1, step = 0.01), np.arange(start = X\_set[:, 1].min() - 1, stop = X\_set[:, 1].max() + 1, step = 0.01))

plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape), alpha = 0.75, cmap = ListedColormap(('pink', 'lightgreen')))

plt.xlim(X1.min(), X1.max())

plt.ylim(X2.min(), X2.max())

for i, j in enumerate(np.unique(y\_set)): plt.scatter(X\_set[y\_set == j, 0], X\_set[y\_set == j, 1],

c = ListedColormap(('red', 'green'))(i), label = j) plt.title('Logistic Regression (Training set)') plt.xlabel('PC1')

plt.ylabel('PC2') plt.legend() plt.show()

# Visualising the Test set results

from matplotlib.colors import ListedColormap X\_set, y\_set = X\_test, y\_test

X1, X2 = np.meshgrid(np.arange(start = X\_set[:, 0].min() - 1, stop = X\_set[:, 0].max() + 1, step = 0.01), np.arange(start = X\_set[:, 1].min() - 1, stop = X\_set[:, 1].max() + 1, step = 0.01))

plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape), alpha = 0.75, cmap = ListedColormap(('pink', 'lightgreen')))

plt.xlim(X1.min(), X1.max())

plt.ylim(X2.min(), X2.max())

for i, j in enumerate(np.unique(y\_set)): plt.scatter(X\_set[y\_set == j, 0], X\_set[y\_set == j, 1],

c = ListedColormap(('red', 'green'))(i), label = j)

plt.title('Logistic Regression (Test set)') plt.xlabel('PC1')

plt.ylabel('PC2') plt.legend() plt.show()

K-NN

# Fitting K-NN 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) # Measuring Accuracy

from sklearn import metrics

print('The accuracy of KNN is: ', metrics.accuracy\_score(y\_pred, y\_test))

# Making confusion matrix

from sklearn.metrics import confusion\_matrix print(confusion\_matrix(y\_test, y\_pred))

# Visualising the Training set results

from matplotlib.colors import ListedColormap X\_set, y\_set = X\_train, y\_train

X1, X2 = np.meshgrid(np.arange(start = X\_set[:, 0].min() - 1, stop = X\_set[:, 0].max() + 1, step = 0.01), np.arange(start = X\_set[:, 1].min() - 1, stop = X\_set[:, 1].max() + 1, step = 0.01))

plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape), alpha = 0.75, cmap = ListedColormap(('pink', 'lightgreen')))

plt.xlim(X1.min(), X1.max())

plt.ylim(X2.min(), X2.max())

for i, j in enumerate(np.unique(y\_set)): plt.scatter(X\_set[y\_set == j, 0], X\_set[y\_set == j, 1],

c = ListedColormap(('red', 'green'))(i), label = j) plt.title('KNN (Training set)')

plt.xlabel('PC1') plt.ylabel('PC2') plt.legend() plt.show()

# Visualising the Test set results

from matplotlib.colors import ListedColormap X\_set, y\_set = X\_test, y\_test

X1, X2 = np.meshgrid(np.arange(start = X\_set[:, 0].min() - 1, stop = X\_set[:, 0].max() + 1, step = 0.01), np.arange(start = X\_set[:, 1].min() - 1, stop = X\_set[:, 1].max() + 1, step = 0.01))

plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape), alpha = 0.75, cmap = ListedColormap(('pink', 'lightgreen')))

plt.xlim(X1.min(), X1.max())

plt.ylim(X2.min(), X2.max())

for i, j in enumerate(np.unique(y\_set)): plt.scatter(X\_set[y\_set == j, 0], X\_set[y\_set == j, 1],

c = ListedColormap(('red', 'green'))(i), label = j) plt.title('KNN (Testing set)')

plt.xlabel('PC1') plt.ylabel('PC2') plt.legend() plt.show()

SVM

# Fitting SVM to the Training set from sklearn.svm import SVC

classifier = SVC(kernel = 'linear', random\_state = 0) classifier.fit(X\_train, y\_train)

# Predicting the Test set results y\_pred = classifier.predict(X\_test) # Measuring Accuracy

from sklearn import metrics

print('The accuracy of SVM is: ', metrics.accuracy\_score(y\_pred, y\_test)) # Making confusion matrix

from sklearn.metrics import confusion\_matrix print(confusion\_matrix(y\_test, y\_pred))

# Visualising the Training set results

from matplotlib.colors import ListedColormap X\_set, y\_set = X\_train, y\_train

X1, X2 = np.meshgrid(np.arange(start = X\_set[:, 0].min() - 1, stop = X\_set[:, 0].max() + 1, step = 0.01), np.arange(start = X\_set[:, 1].min() - 1, stop = X\_set[:, 1].max() + 1, step = 0.01))

plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape), alpha = 0.75, cmap = ListedColormap(('pink', 'lightgreen')))

plt.xlim(X1.min(), X1.max())

plt.ylim(X2.min(), X2.max())

for i, j in enumerate(np.unique(y\_set)): plt.scatter(X\_set[y\_set == j, 0], X\_set[y\_set == j, 1],

c = ListedColormap(('red', 'green'))(i), label = j) plt.title('SVM (Training set)')

plt.xlabel('PC1') plt.ylabel('PC2') plt.legend() plt.show()

# Visualising the Test set results

from matplotlib.colors import ListedColormap X\_set, y\_set = X\_test, y\_test

X1, X2 = np.meshgrid(np.arange(start = X\_set[:, 0].min() - 1, stop = X\_set[:, 0].max() + 1, step = 0.01), np.arange(start = X\_set[:, 1].min() - 1, stop = X\_set[:, 1].max() + 1, step = 0.01))

plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape), alpha = 0.75, cmap = ListedColormap(('pink', 'lightgreen')))

plt.xlim(X1.min(), X1.max())

plt.ylim(X2.min(), X2.max())

for i, j in enumerate(np.unique(y\_set)): plt.scatter(X\_set[y\_set == j, 0], X\_set[y\_set == j, 1],

c = ListedColormap(('red', 'green'))(i), label = j) plt.title('SVM (Test set)')

plt.xlabel('PC1') plt.ylabel('PC2') plt.legend() plt.show()

Naive Bayes

# Fitting Naive Bayes to the Training set 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) y\_pred

# Measuring Accuracy from sklearn import metrics

print('The accuracy of Naive Bayes is: ', metrics.accuracy\_score(y\_pred, y\_test))

# Making confusion matrix

from sklearn.metrics import confusion\_matrix print(confusion\_matrix(y\_test, y\_pred))

# Visualising the Training set results

from matplotlib.colors import ListedColormap X\_set, y\_set = X\_train, y\_train

X1, X2 = np.meshgrid(np.arange(start = X\_set[:, 0].min() - 1, stop = X\_set[:, 0].max() + 1, step = 0.01), np.arange(start = X\_set[:, 1].min() - 1, stop = X\_set[:, 1].max() + 1, step = 0.01))

plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape), alpha = 0.75, cmap = ListedColormap(('pink', 'lightgreen')))

plt.xlim(X1.min(), X1.max())

plt.ylim(X2.min(), X2.max())

for i, j in enumerate(np.unique(y\_set)): plt.scatter(X\_set[y\_set == j, 0], X\_set[y\_set == j, 1],

c = ListedColormap(('red', 'green'))(i), label = j) plt.title('Naive Bayes (Training set)')

plt.xlabel('PC1') plt.ylabel('PC2') plt.legend() plt.show()

# Visualising the Test set results

from matplotlib.colors import ListedColormap X\_set, y\_set = X\_test, y\_test

X1, X2 = np.meshgrid(np.arange(start = X\_set[:, 0].min() - 1, stop = X\_set[:, 0].max() + 1, step = 0.01), np.arange(start = X\_set[:, 1].min() - 1, stop = X\_set[:, 1].max() + 1, step = 0.01))

plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape), alpha = 0.75, cmap = ListedColormap(('pink', 'lightgreen')))

plt.xlim(X1.min(), X1.max())

plt.ylim(X2.min(), X2.max())

for i, j in enumerate(np.unique(y\_set)): plt.scatter(X\_set[y\_set == j, 0], X\_set[y\_set == j, 1],

c = ListedColormap(('red', 'green'))(i), label = j) plt.title('Naive Bayes (Test set)')

plt.xlabel('PC1') plt.ylabel('PC2') plt.legend() plt.show()

Decision Tree Classification

# Fitting Decision Tree Classification 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) y\_pred

# Measuring Accuracy from sklearn import metrics

print('The accuracy of Decision Tree Classifier is: ', metrics.accuracy\_score(y\_pred, y\_test)) # Making confusion matrix

from sklearn.metrics import confusion\_matrix print(confusion\_matrix(y\_test, y\_pred))

# Visualising the Training set results

from matplotlib.colors import ListedColormap X\_set, y\_set = X\_train, y\_train

X1, X2 = np.meshgrid(np.arange(start = X\_set[:, 0].min() - 1, stop = X\_set[:, 0].max() + 1, step = 0.01), np.arange(start = X\_set[:, 1].min() - 1, stop = X\_set[:, 1].max() + 1, step = 0.01))

plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape), alpha = 0.75, cmap = ListedColormap(('pink', 'lightgreen')))

plt.xlim(X1.min(), X1.max())

plt.ylim(X2.min(), X2.max())

for i, j in enumerate(np.unique(y\_set)): plt.scatter(X\_set[y\_set == j, 0], X\_set[y\_set == j, 1],

c = ListedColormap(('red', 'green'))(i), label = j) plt.title('Decision Tree Classifier (Training set)') plt.xlabel('PC1')

plt.ylabel('PC2') plt.legend() plt.show()

# Visualising the Test set results

from matplotlib.colors import ListedColormap X\_set, y\_set = X\_test, y\_test

X1, X2 = np.meshgrid(np.arange(start = X\_set[:, 0].min() - 1, stop = X\_set[:, 0].max() + 1, step = 0.01), np.arange(start = X\_set[:, 1].min() - 1, stop = X\_set[:, 1].max() + 1, step = 0.01))

plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape), alpha = 0.75, cmap = ListedColormap(('pink', 'lightgreen')))

plt.xlim(X1.min(), X1.max())

plt.ylim(X2.min(), X2.max())

for i, j in enumerate(np.unique(y\_set)): plt.scatter(X\_set[y\_set == j, 0], X\_set[y\_set == j, 1],

c = ListedColormap(('red', 'green'))(i), label = j) plt.title('Decision Tree Classifier (Test set)') plt.xlabel('PC1')

plt.ylabel('PC2') plt.legend() plt.show()

Random Forest Classification

# Fitting Random Forest Classification to the Training set from sklearn.ensemble import RandomForestClassifier

classifier = RandomForestClassifier(n\_estimators = 10, criterion = 'entropy', random\_state = 0) classifier.fit(X\_train, y\_train)

# Predicting the Test set results y\_pred = classifier.predict(X\_test) y\_pred

# Measuring Accuracy from sklearn import metrics

print('The accuracy of Random Forest Classification is: ', metrics.accuracy\_score(y\_pred, y\_test)) # Making confusion matrix

from sklearn.metrics import confusion\_matrix print(confusion\_matrix(y\_test, y\_pred))

# Visualising the Training set results

from matplotlib.colors import ListedColormap X\_set, y\_set = X\_train, y\_train

X1, X2 = np.meshgrid(np.arange(start = X\_set[:, 0].min() - 1, stop = X\_set[:, 0].max() + 1, step = 0.01),

np.arange(start = X\_set[:, 1].min() - 1, stop = X\_set[:, 1].max() + 1, step = 0.01)) plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),

alpha = 0.75, cmap = ListedColormap(('pink', 'lightgreen'))) plt.xlim(X1.min(), X1.max())

plt.ylim(X2.min(), X2.max())

for i, j in enumerate(np.unique(y\_set)): plt.scatter(X\_set[y\_set == j, 0], X\_set[y\_set == j, 1],

c = ListedColormap(('red', 'green'))(i), label = j) plt.title('Random Forest Classification (Training set)') plt.xlabel('PC1')

plt.ylabel('PC2') plt.legend() plt.show()

# Visualising the Test set results

from matplotlib.colors import ListedColormap X\_set, y\_set = X\_test, y\_test

X1, X2 = np.meshgrid(np.arange(start = X\_set[:, 0].min() - 1, stop = X\_set[:, 0].max() + 1, step = 0.01), np.arange(start = X\_set[:, 1].min() - 1, stop = X\_set[:, 1].max() + 1, step = 0.01))

plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape), alpha = 0.75, cmap = ListedColormap(('pink', 'lightgreen')))

plt.xlim(X1.min(), X1.max())

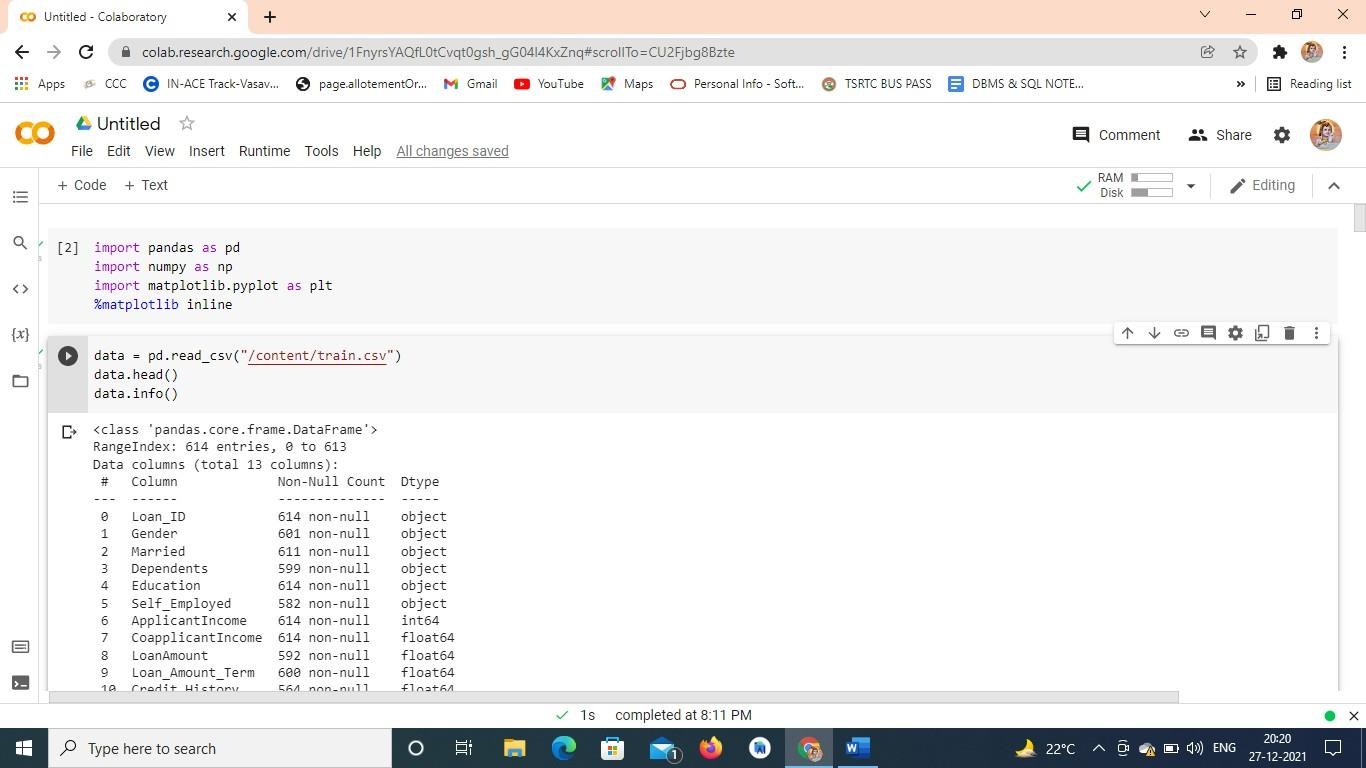
plt.ylim(X2.min(), X2.max())

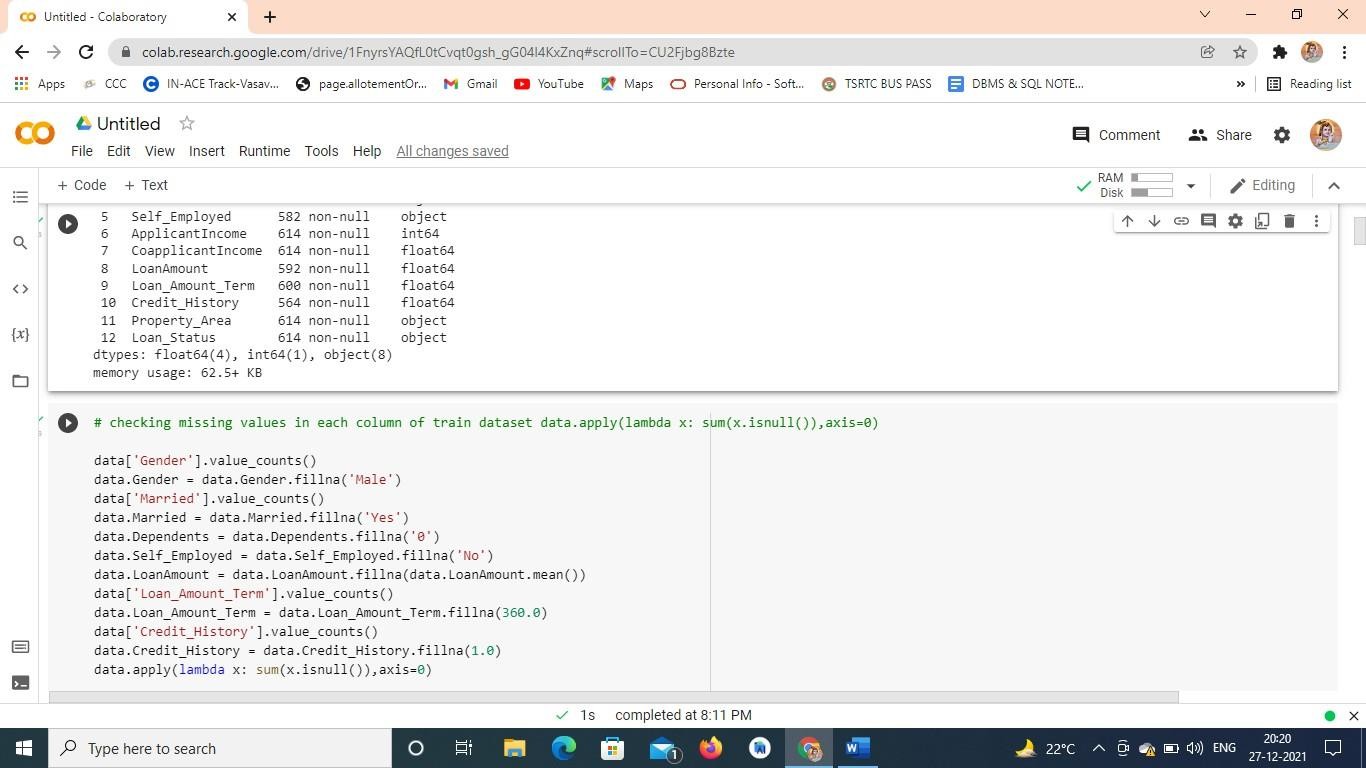
for i, j in enumerate(np.unique(y\_set)): plt.scatter(X\_set[y\_set == j, 0], X\_set[y\_set == j, 1],

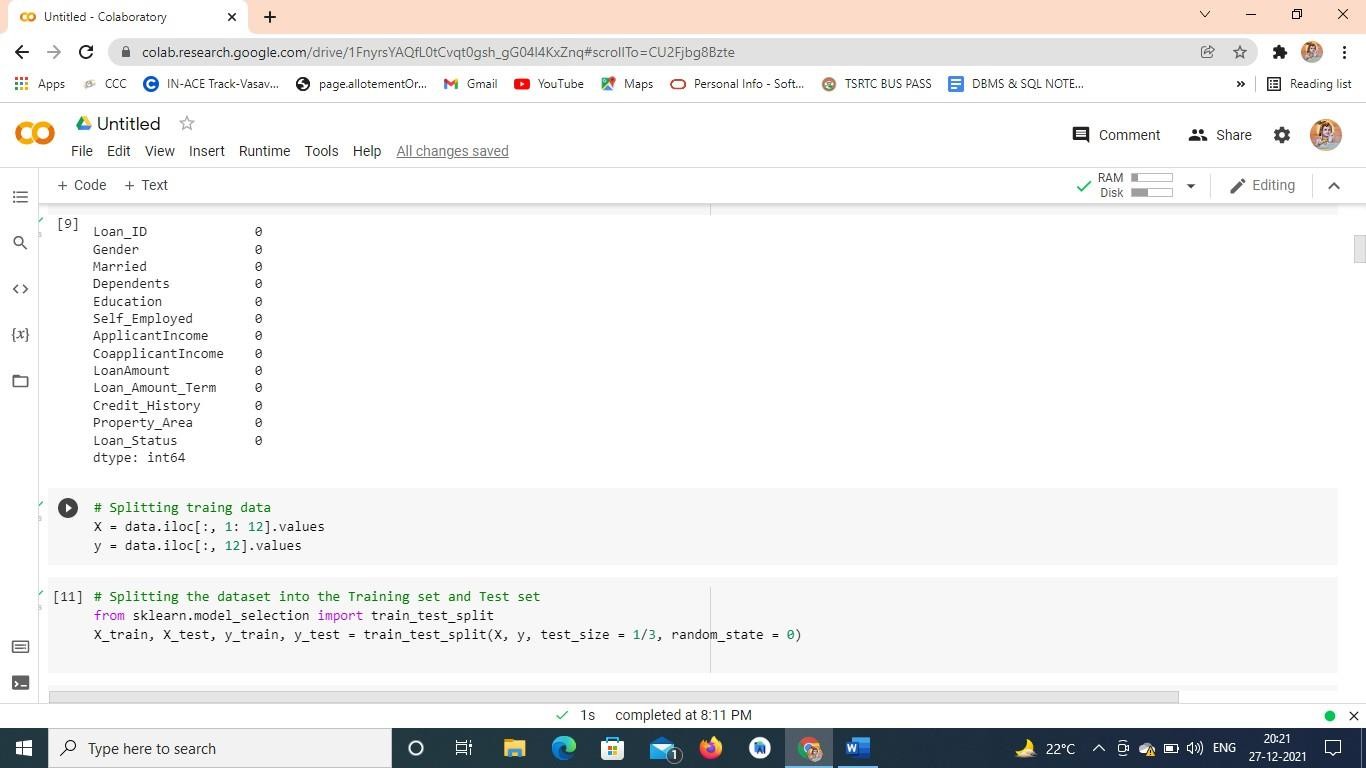
c = ListedColormap(('red', 'green'))(i), label = j) plt.title('Random Forest Classification (Test set)') plt.xlabel('PC1')

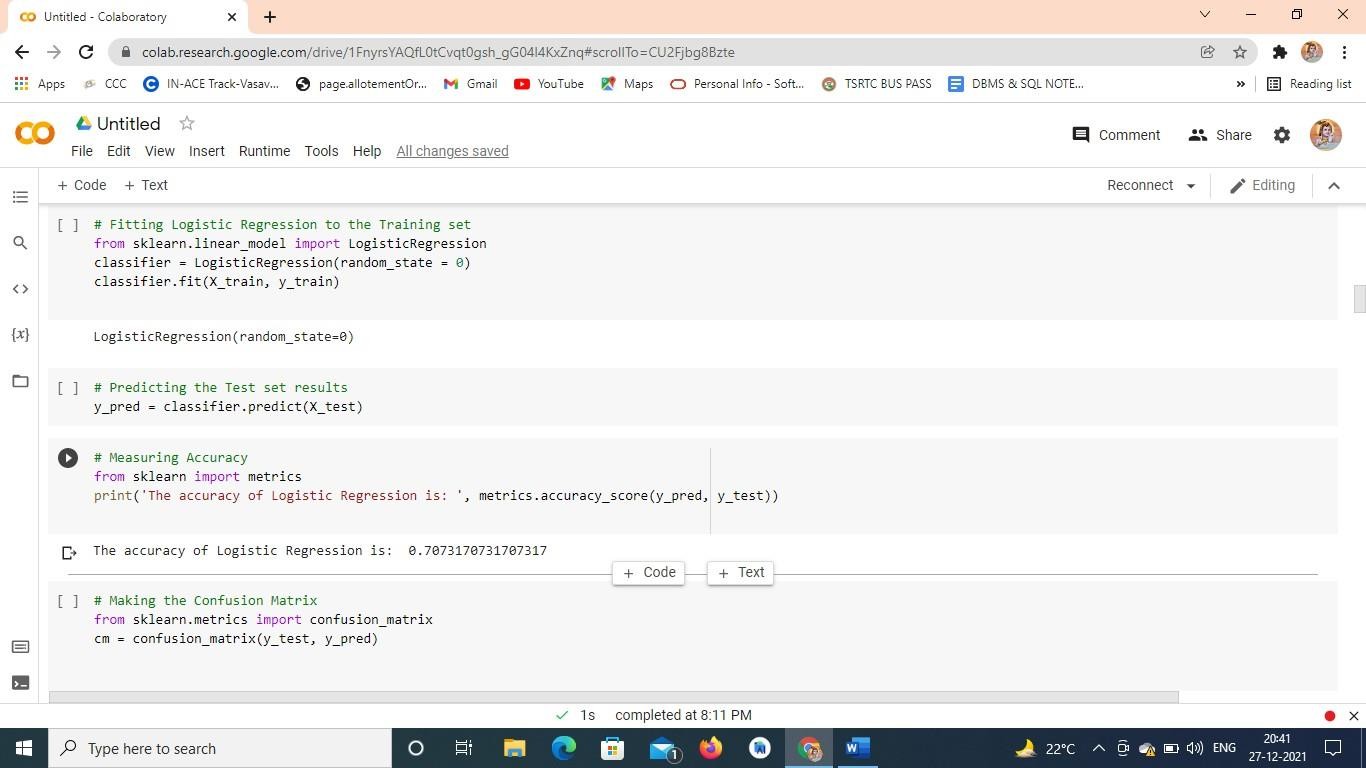
plt.ylabel('PC2') plt.legend() plt.show()

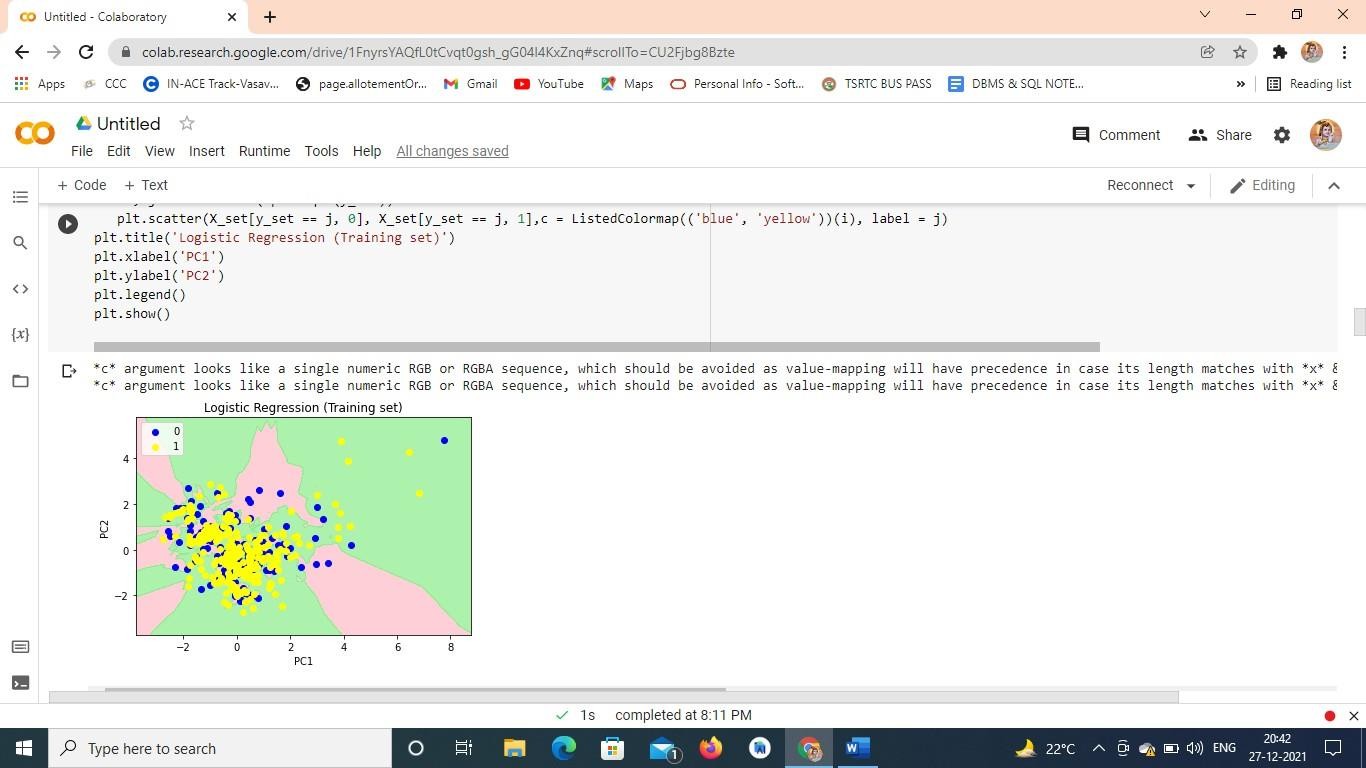
**7. OUTPUTS**

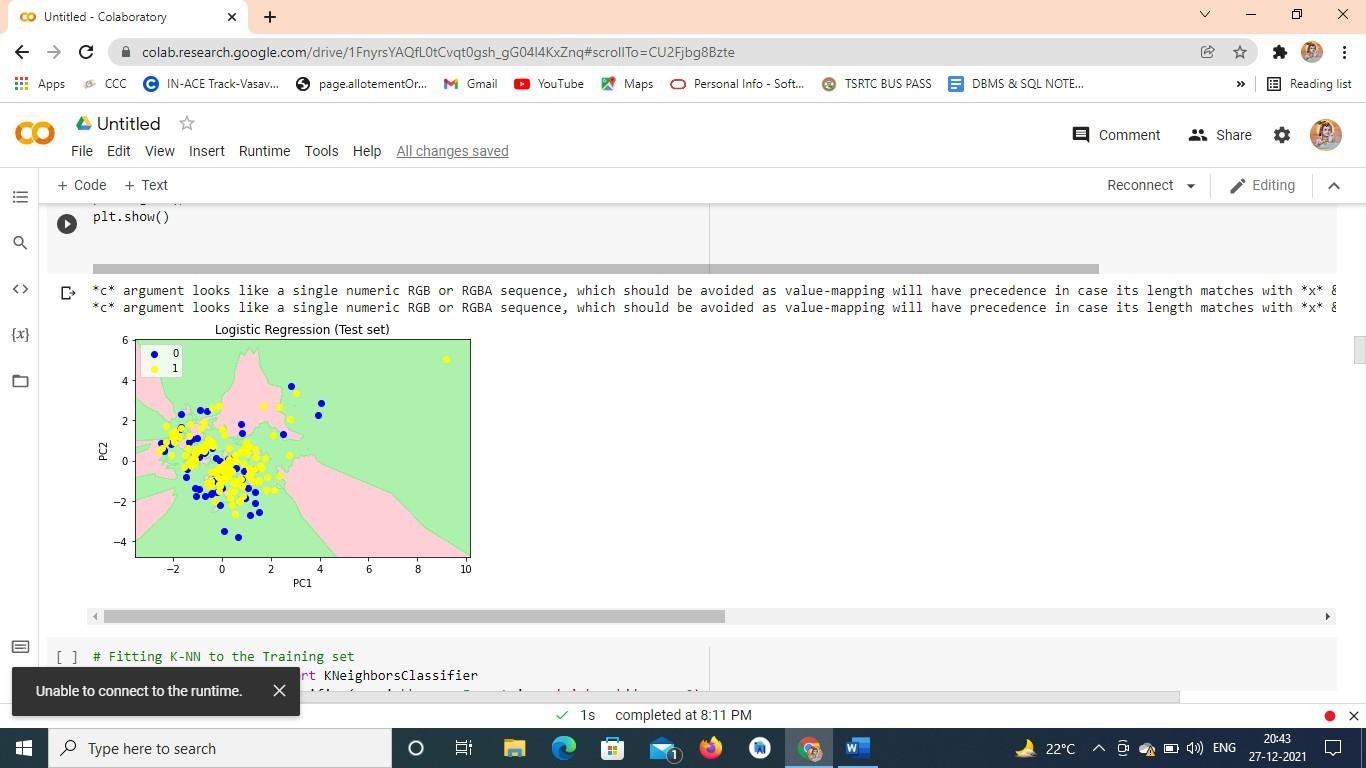


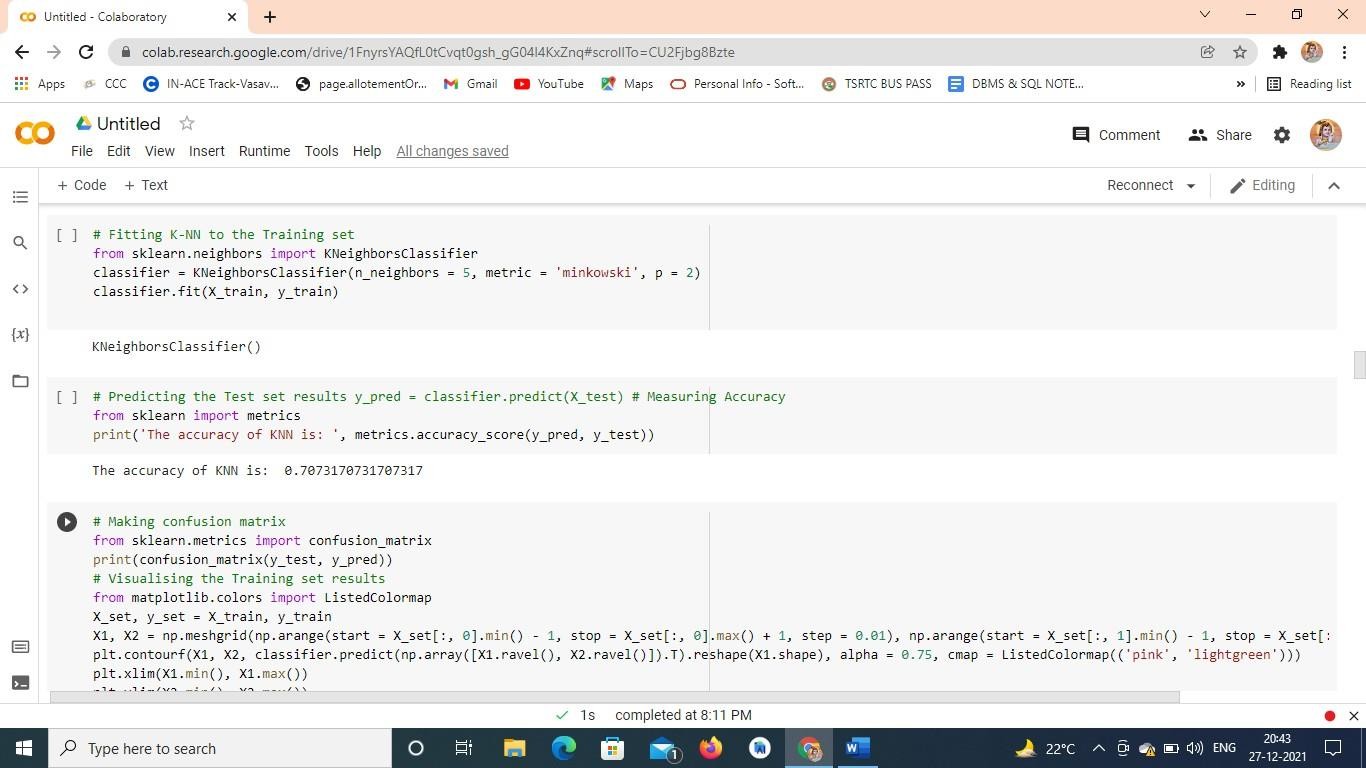


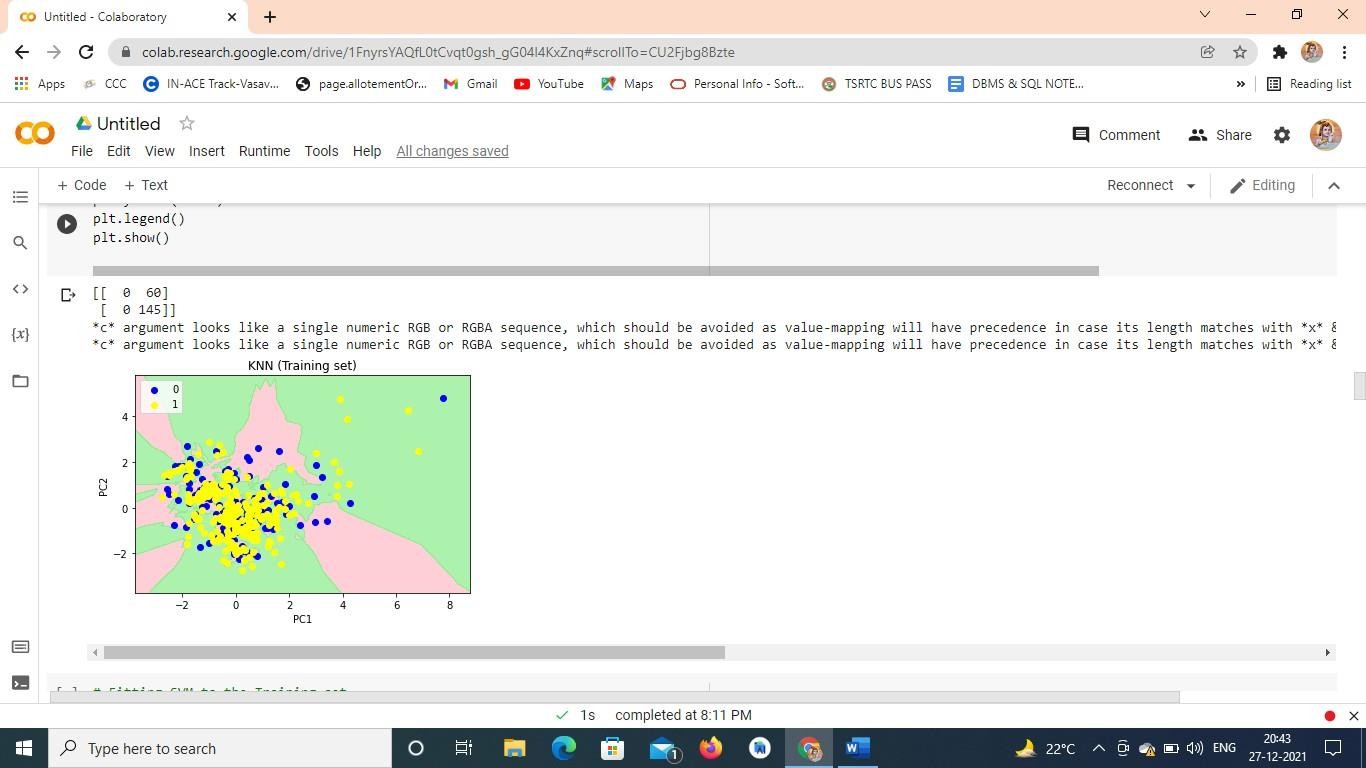


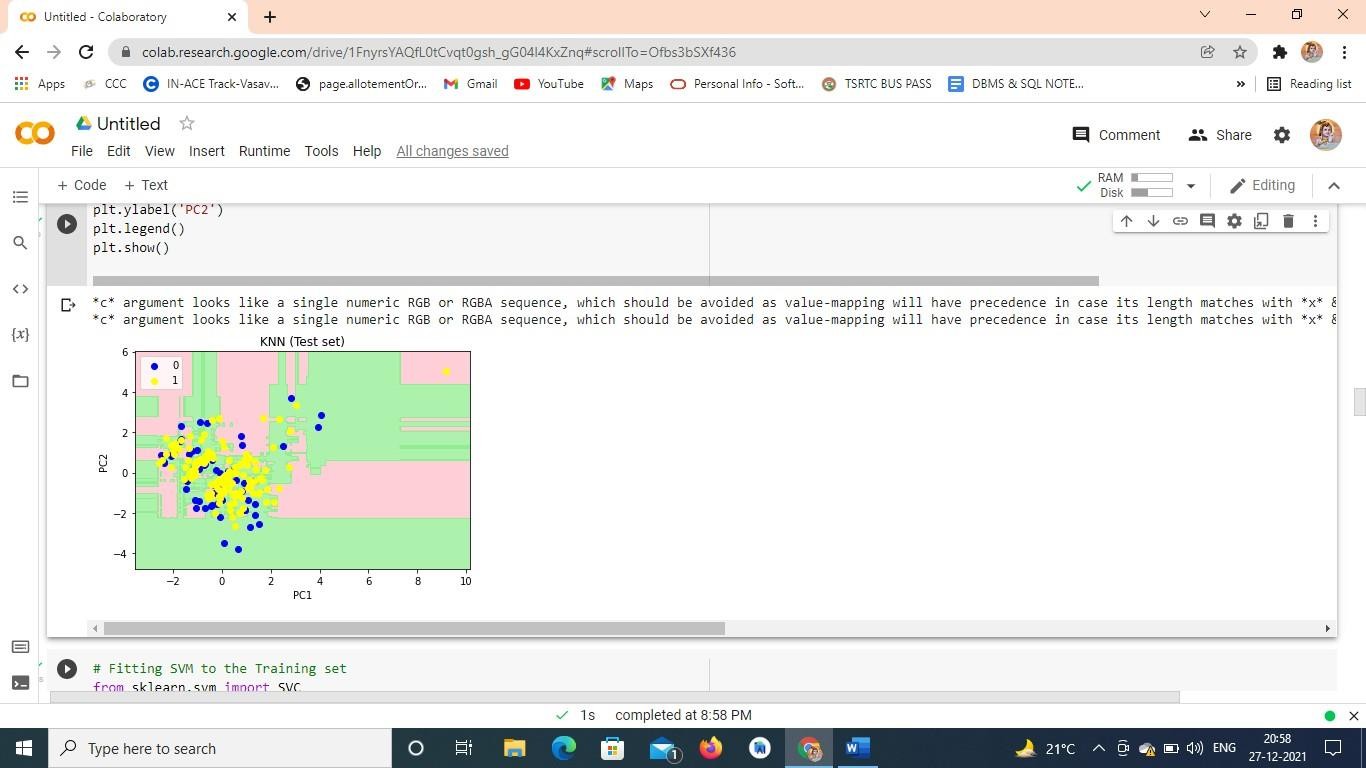


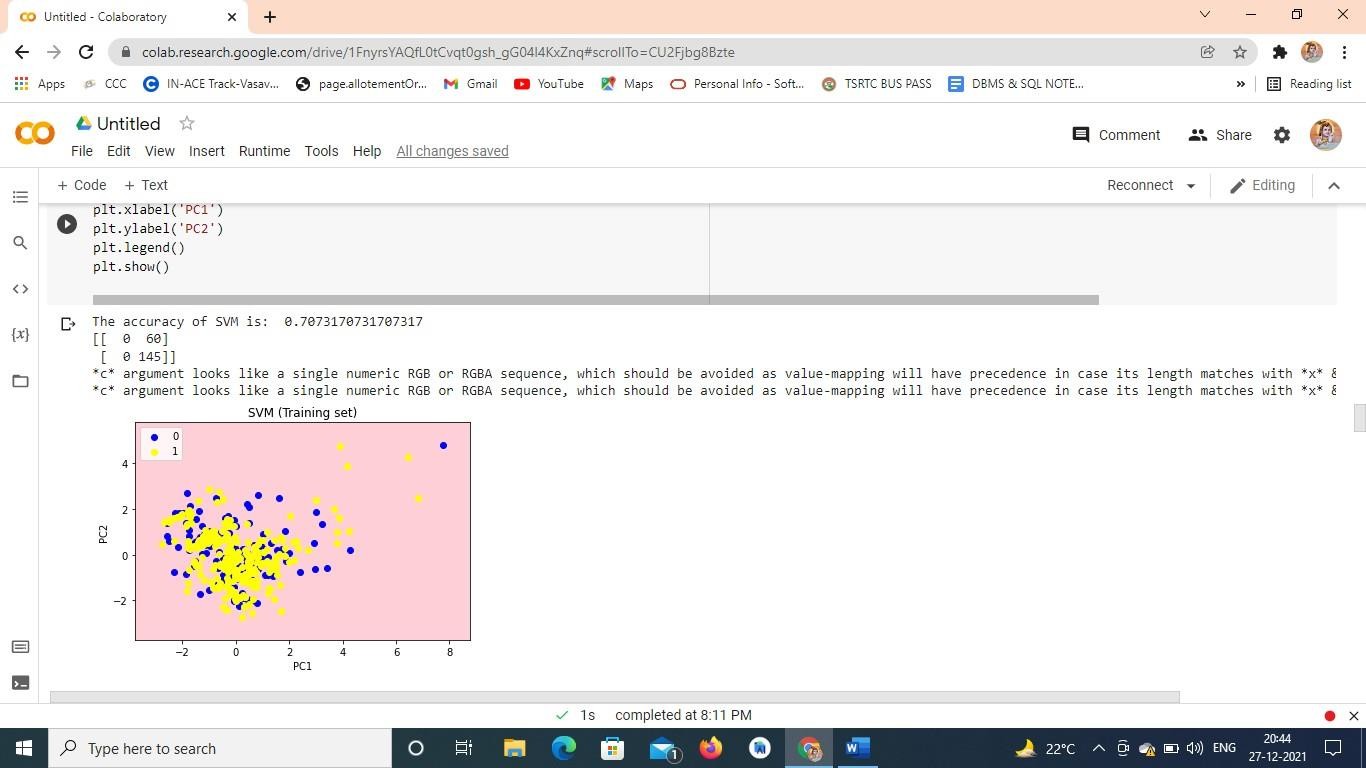


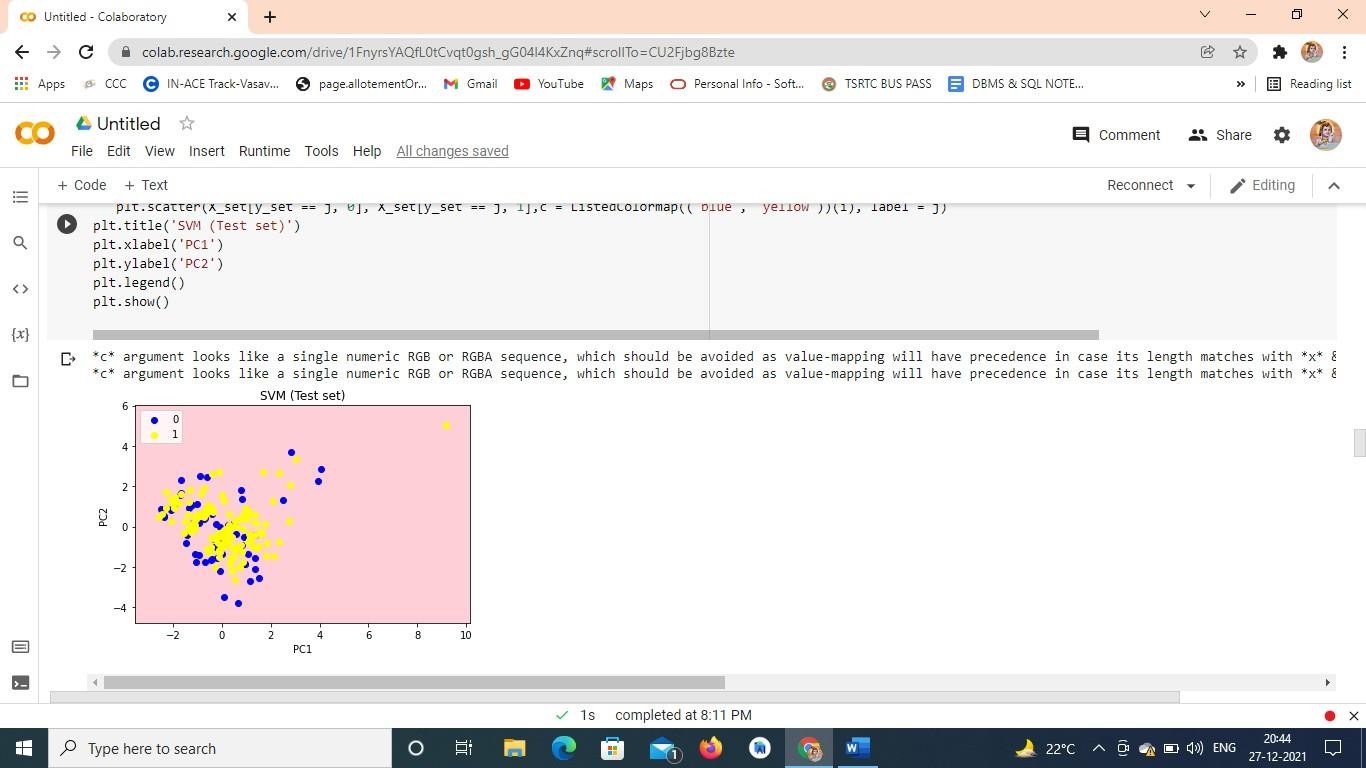


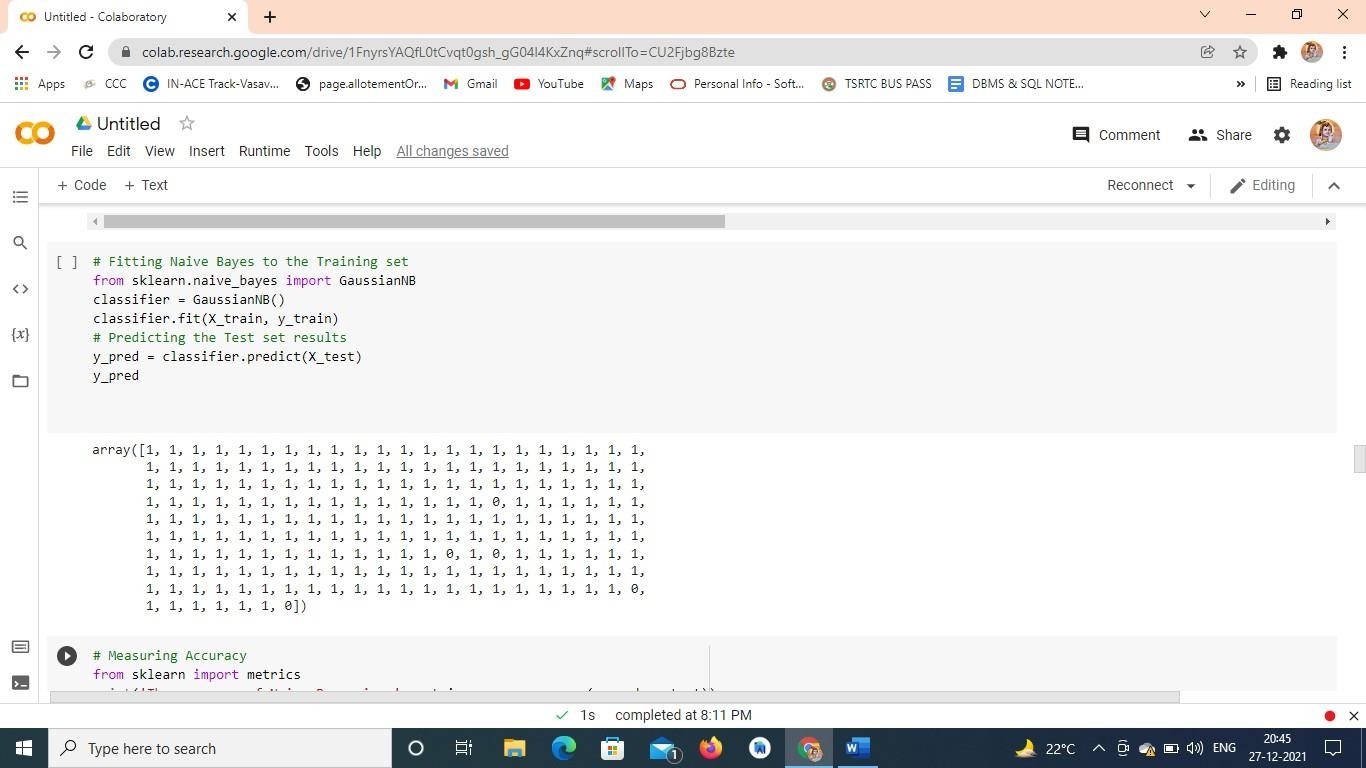


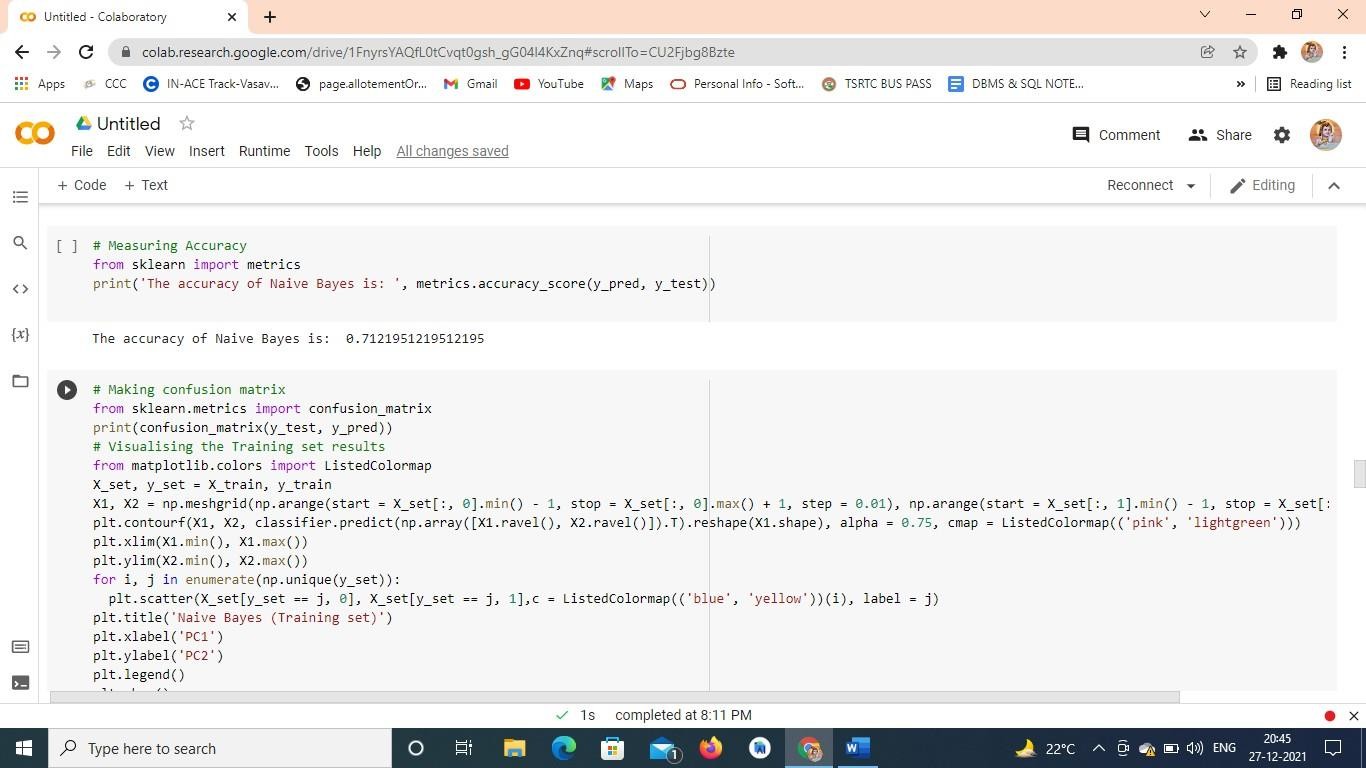


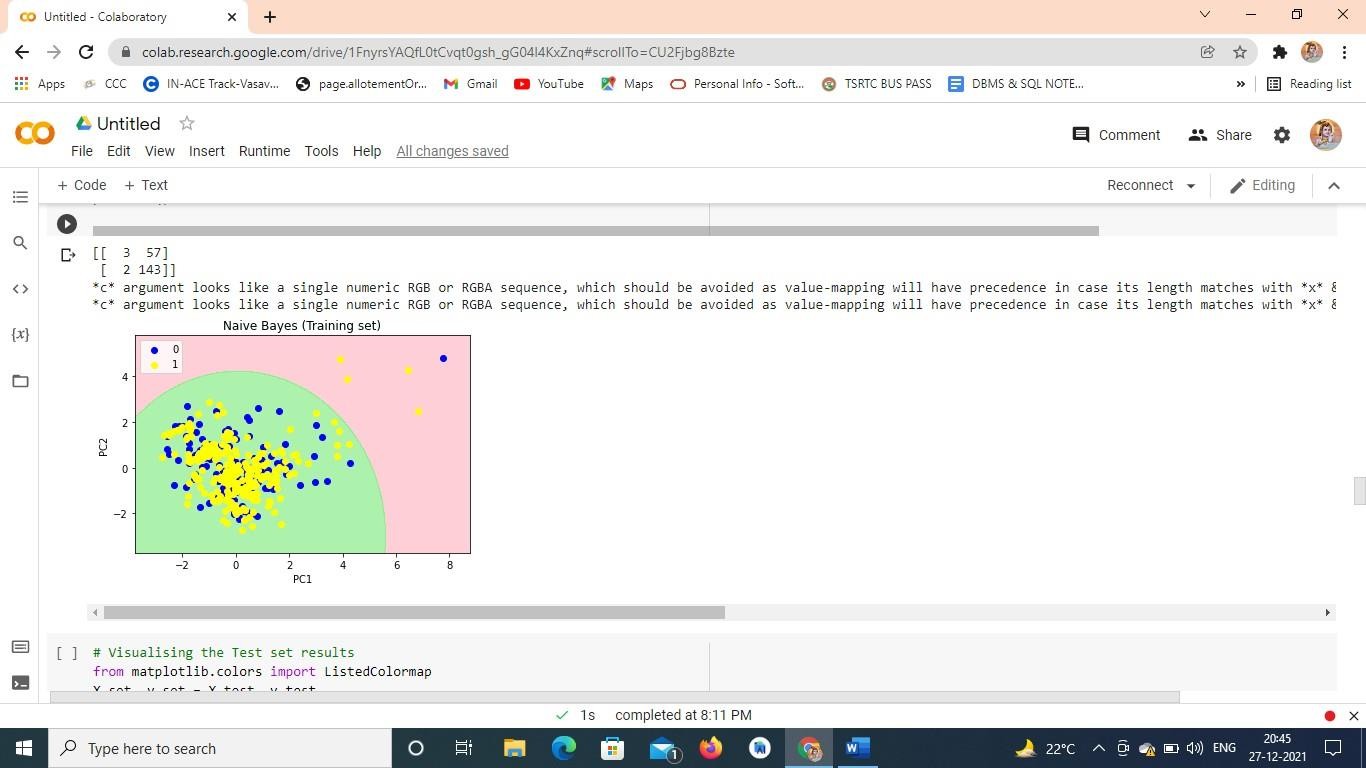


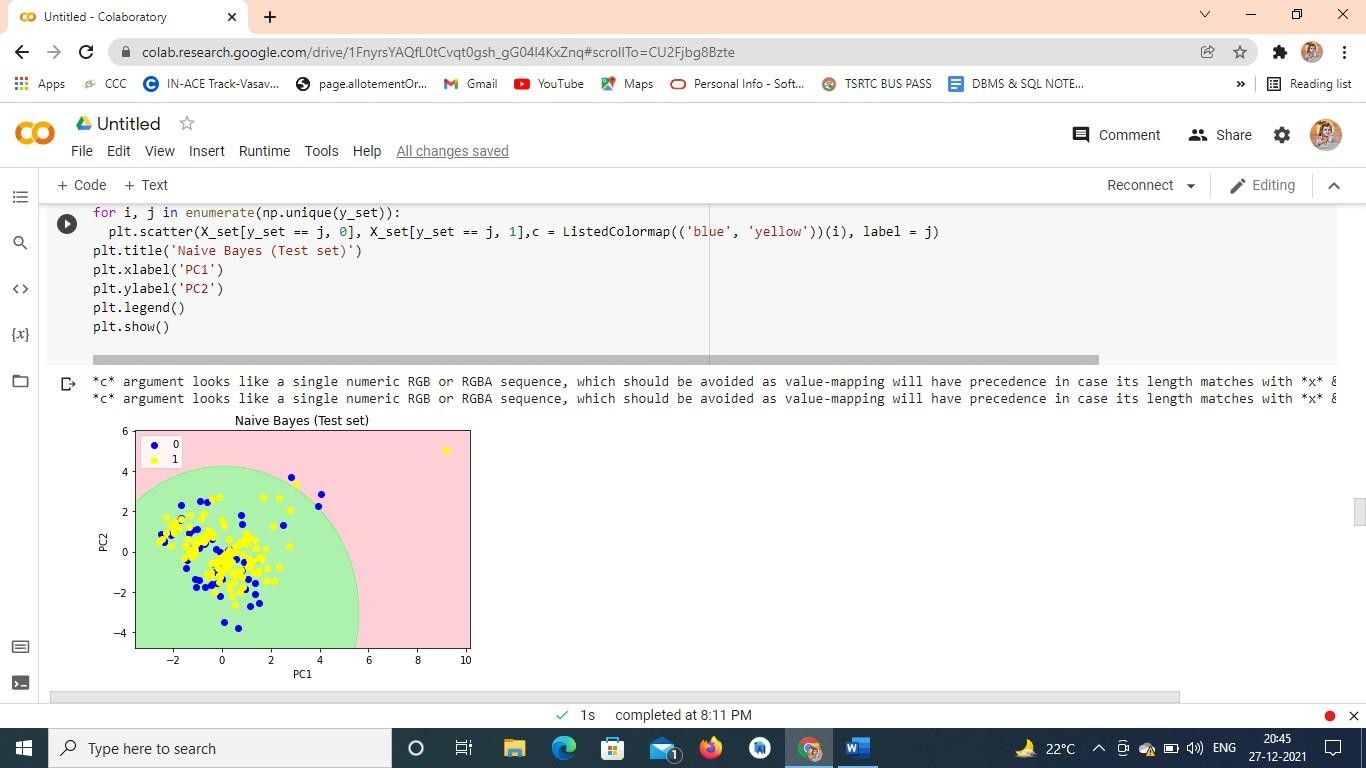


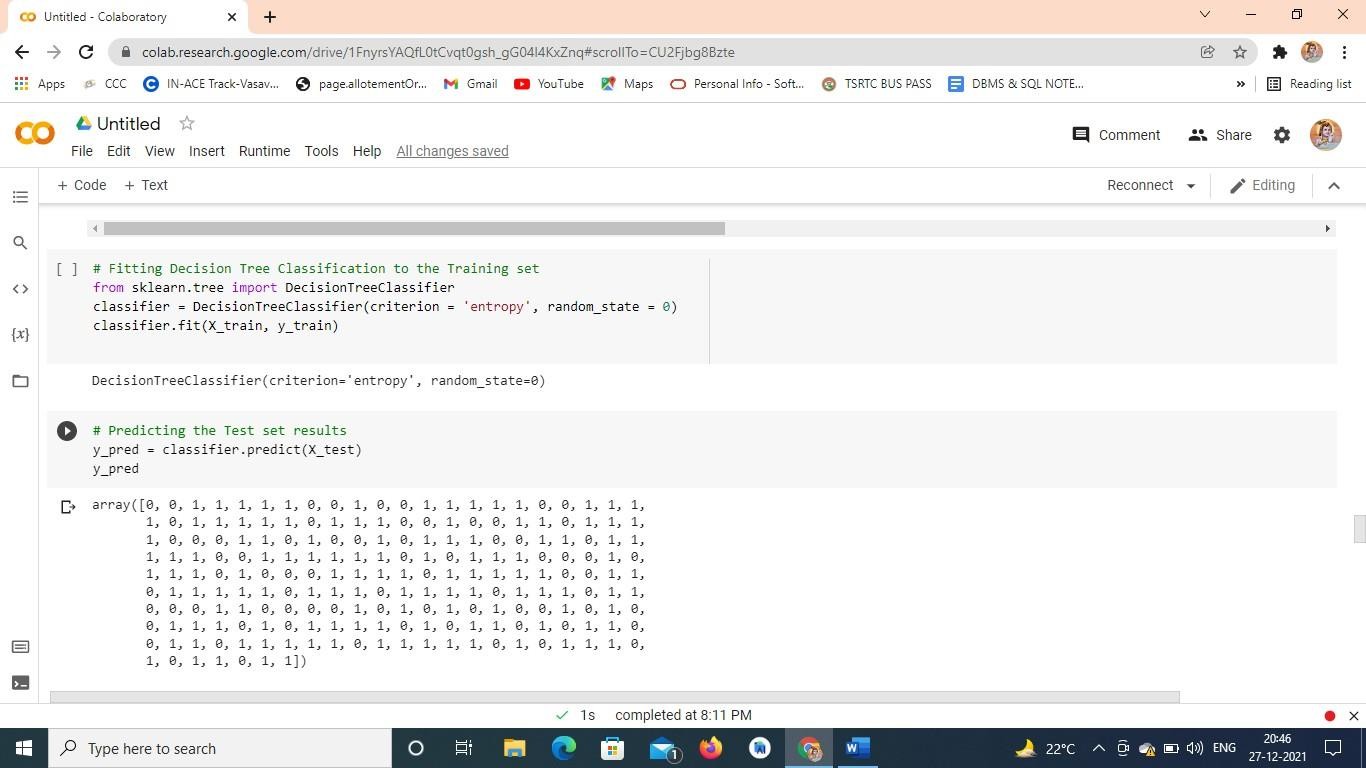


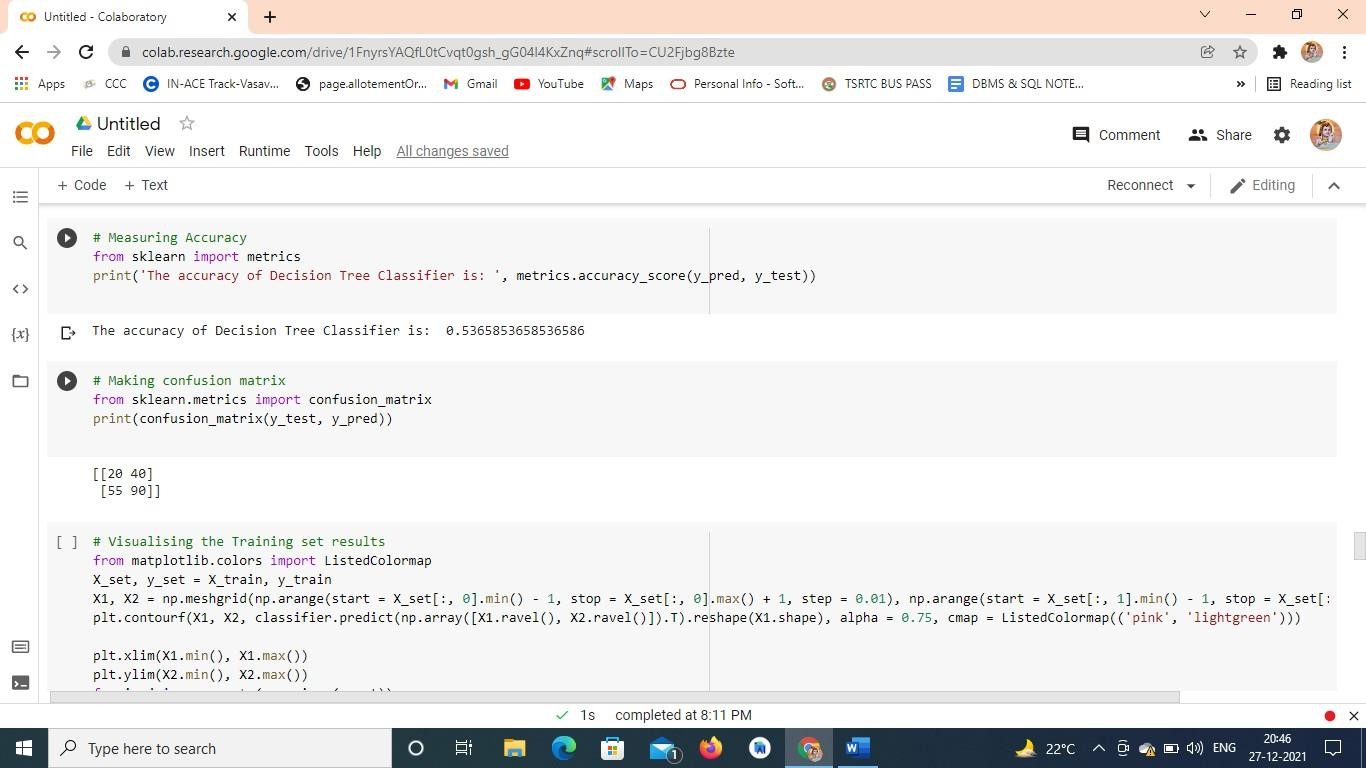


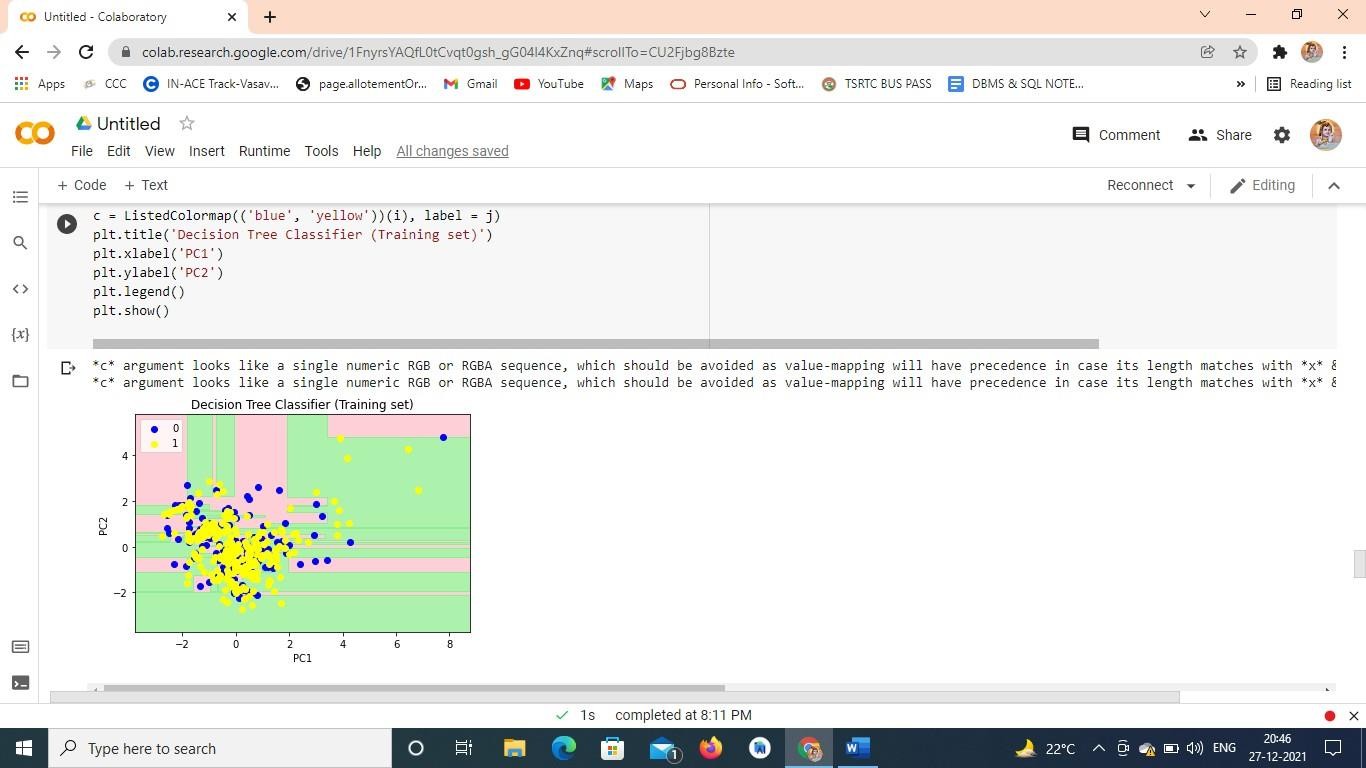


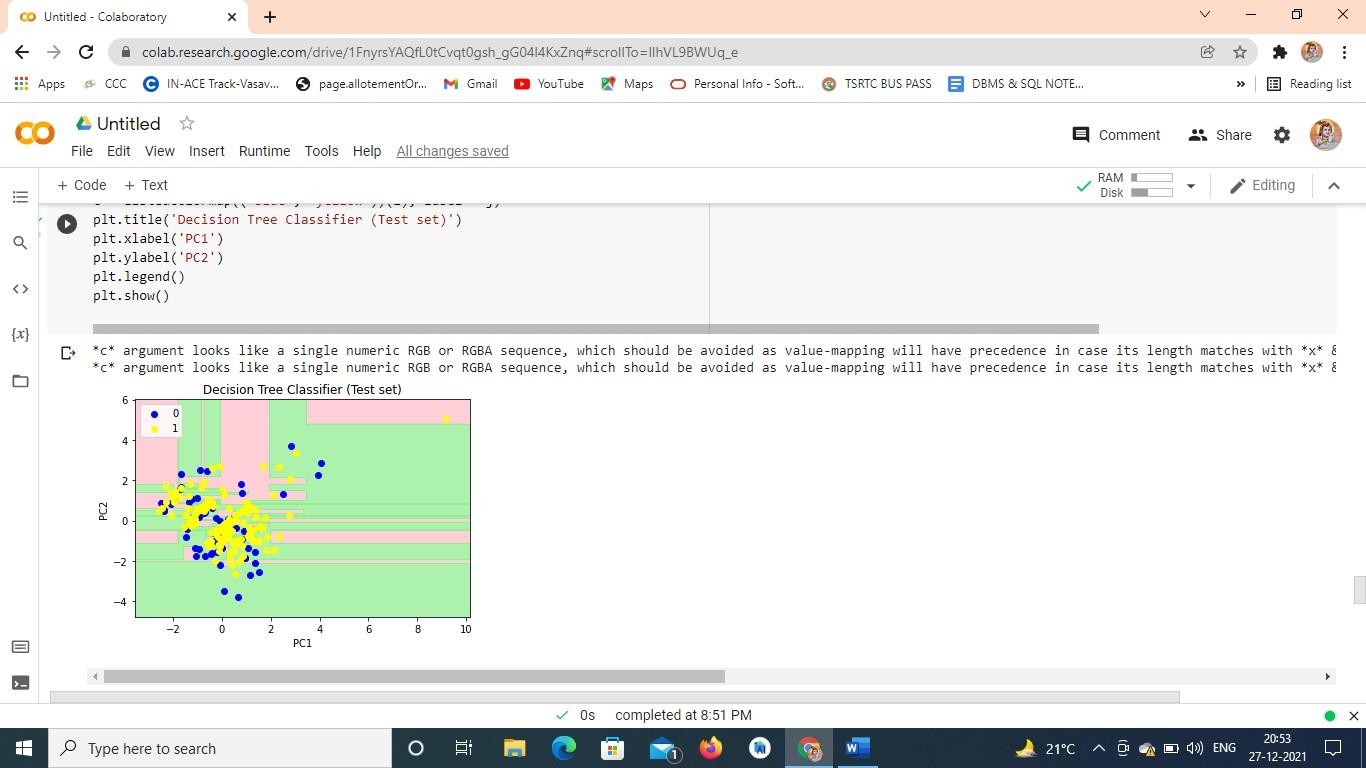


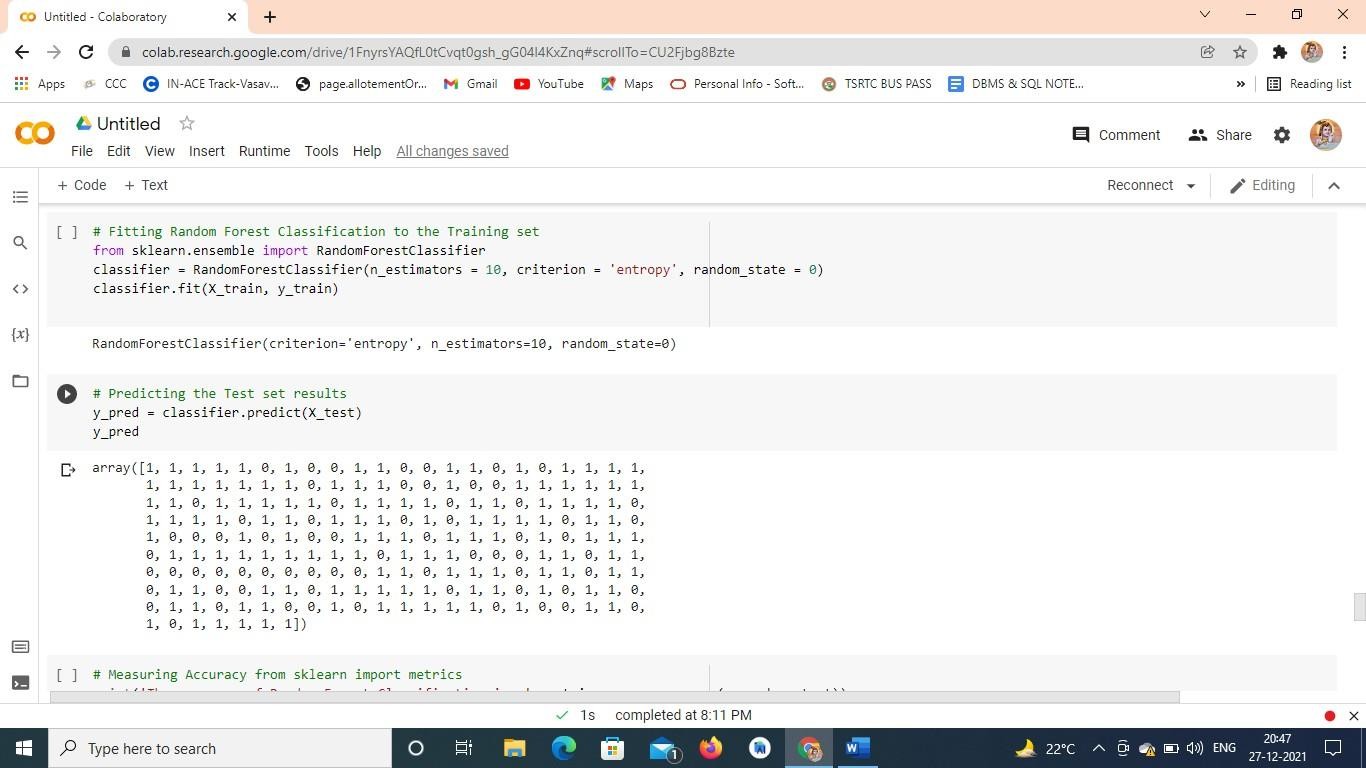




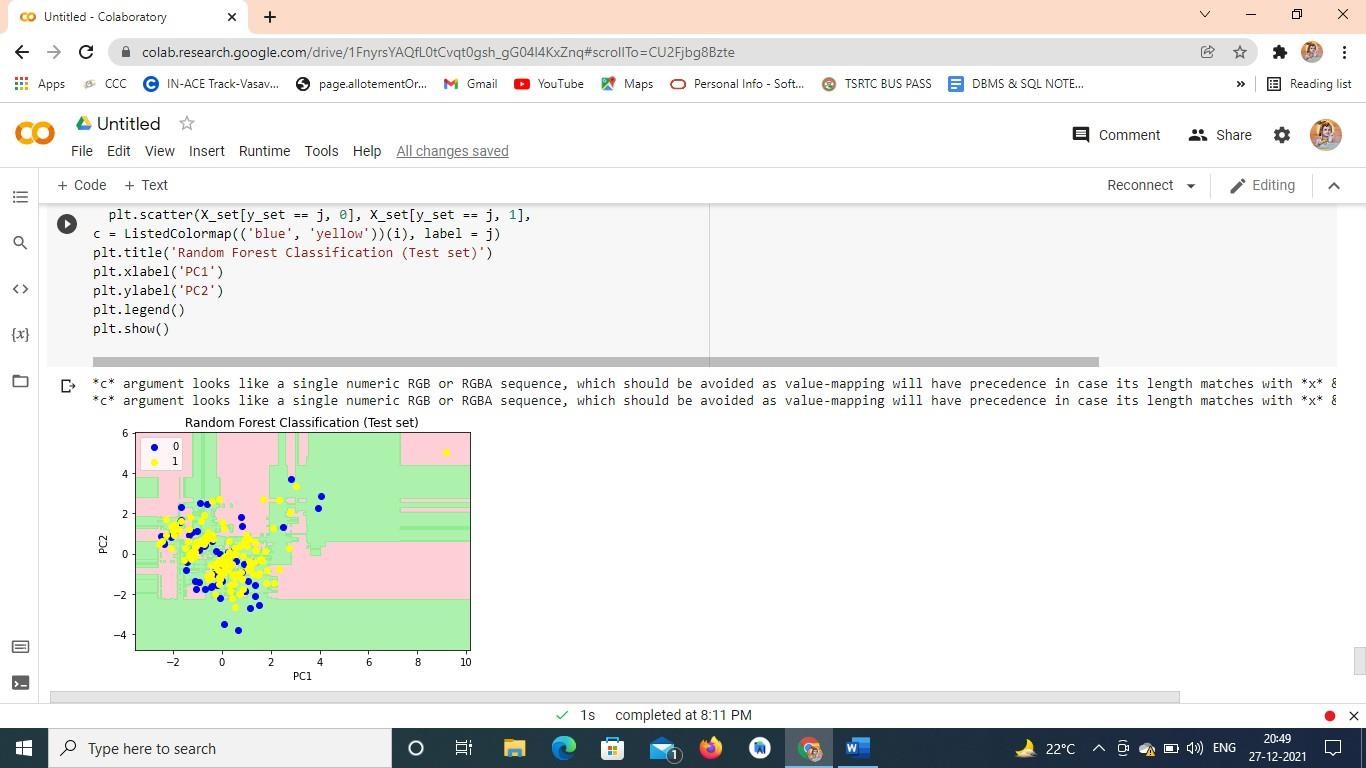












**8 CONCLUSION**

On comparing accuracy of each classification

The accuracy of Logistic Regression is: 70.73 % The accuracy of KNN is: 62.92 %

The accuracy of SVM is: 70.73 %

The accuracy of Naive Bayes is: 71.21 %

The accuracy of Decision Tree Classifier is: 53.63 %

The accuracy of Random Forest Classification is: 58.53 % For loan prediction Naïve Bayes classifier is best.

***REFERENCES***

1. *Senturk ZK, Kara R. Breast Cancer Diagnosis via Data mining: Performance Analysis Of Seven Different Algorithms. Computer Science & Engineering: An International Journal (CSEIJ); 4:35–46 (2014).*
2. *Rajesh K, Anand S. Analysis of SEER Dataset for Breast Cancer Diagnosis using C4.5 Classification Algorithm. International Journal of Advanced Research in Computer and Communication Engineering. 1:72–77 (2012).*
3. *Gupta S, Kumar D, Sharma A. Data Mining Classification Techniques Applied For Breast Cancer Diagnosis and Prognosis. Indian Journal of Computer Science and Engineering. 2 (2011).*
4. *Kumar R, Verma R. Classification Algorithms for Data Mining: A Survey. International Journal of Innovations in Engineering and Technology (IJIET) 1:7–14 (2012).*
5. *Kesavaraj G, Sukumaran S. A Study on Classification Techniques in Data Mining. 1 4th ICCCNT (2012).*
6. *Soundarya M, Balakrishnan R. Survey on Classification Techniques in Data mining. International Journal of Advanced Research in Computer and Communication Engineering Vol.3:7550–7552 (2014).*
7. *Li J, Wong L. Rule-Based Data Mining Methods for Classification Problems in Biomedical Domains; 15th European Conference on Machine Learning (ECML) (2004).*
8. *Kumar D, Beniwal S. Genetic Algorithm and Programming Based Classification: A Survey. Journal of Theoretical and Applied Information Technology. 54:48–58 (2013).*
9. *Mansuri AM, Verma M, Laxkar P. A Survey of Classifier Designing Using Genetic Programming and Genetic Operators. International Journal of Engineering Research and Reviews (IJERR) Vol. 2:16–22 (2017)*