Course Title: Applied Machine Learning

Course Code: CSE4008

Lab Activity-1

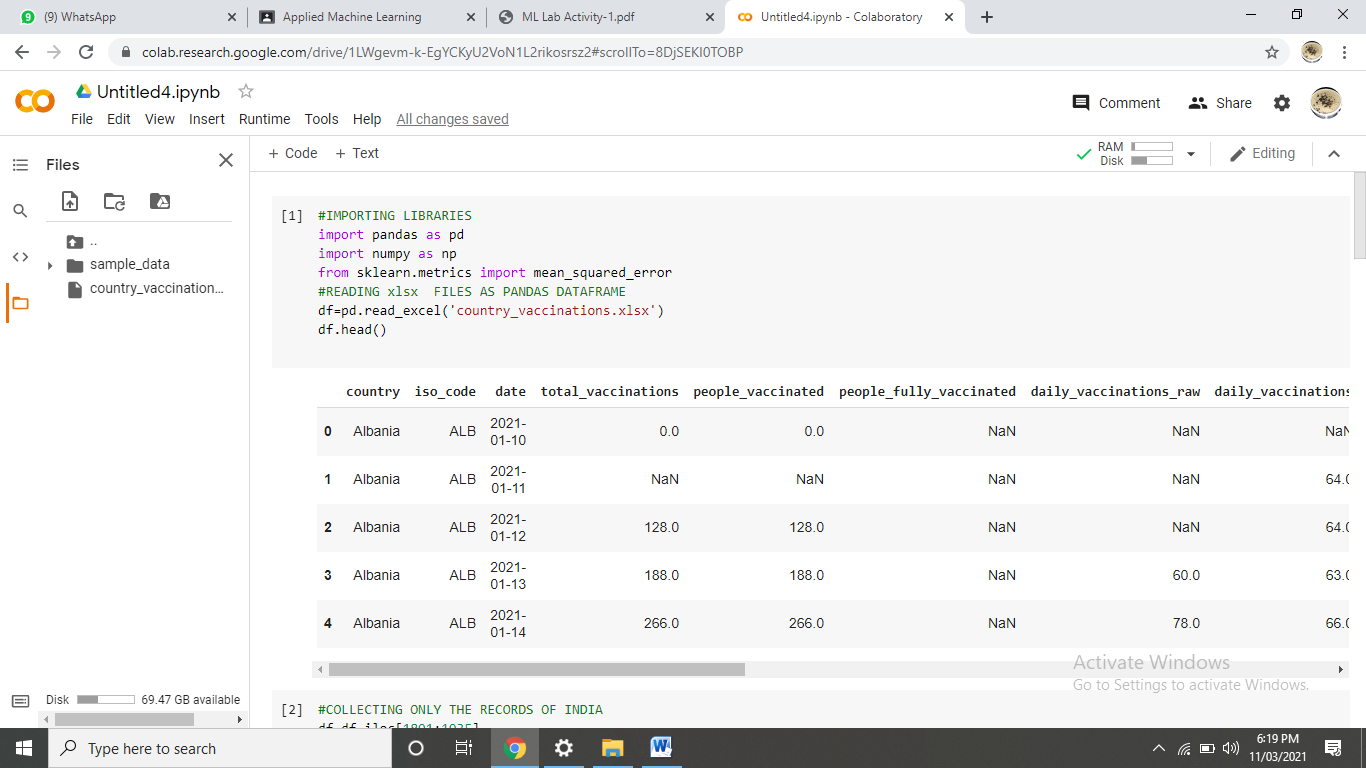
Faculty Name: Dr. Virendra Singh Kushwah

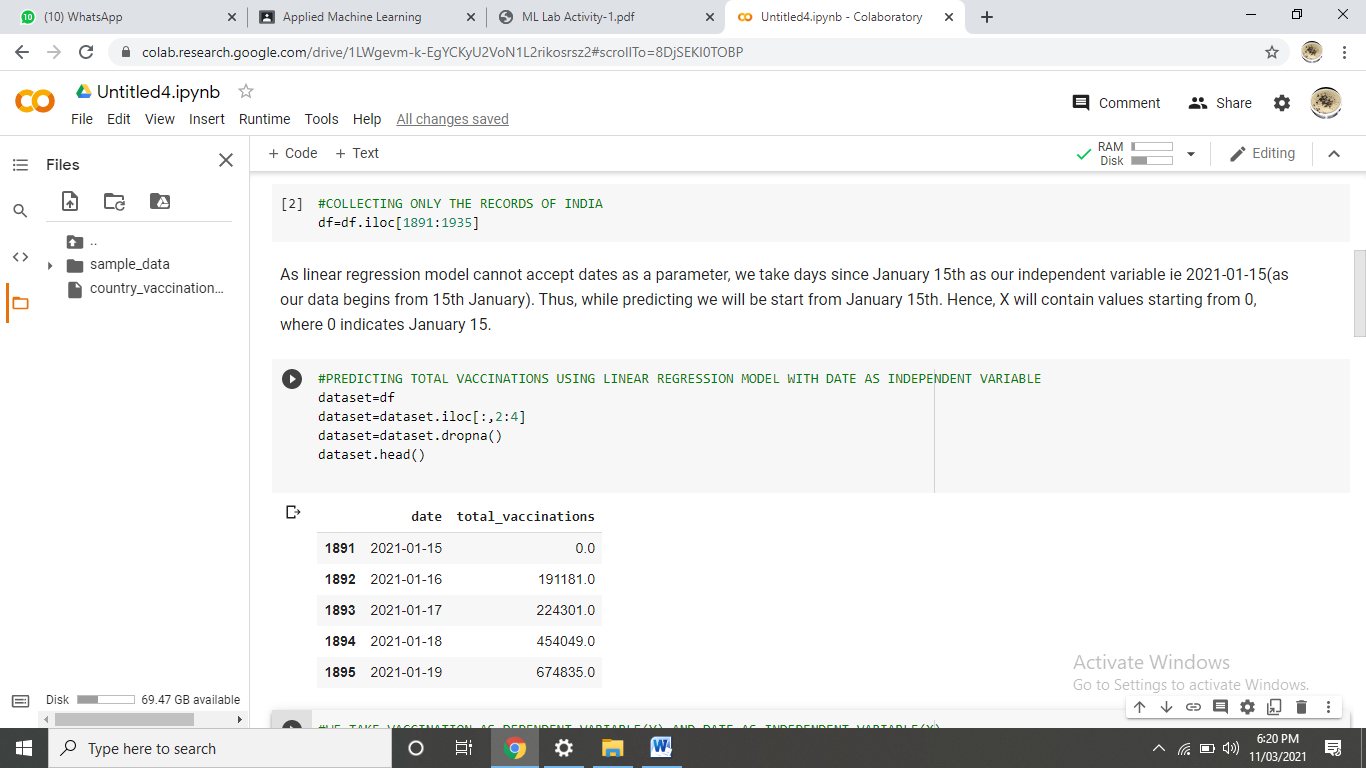
Student Name: Prachi Bhatt

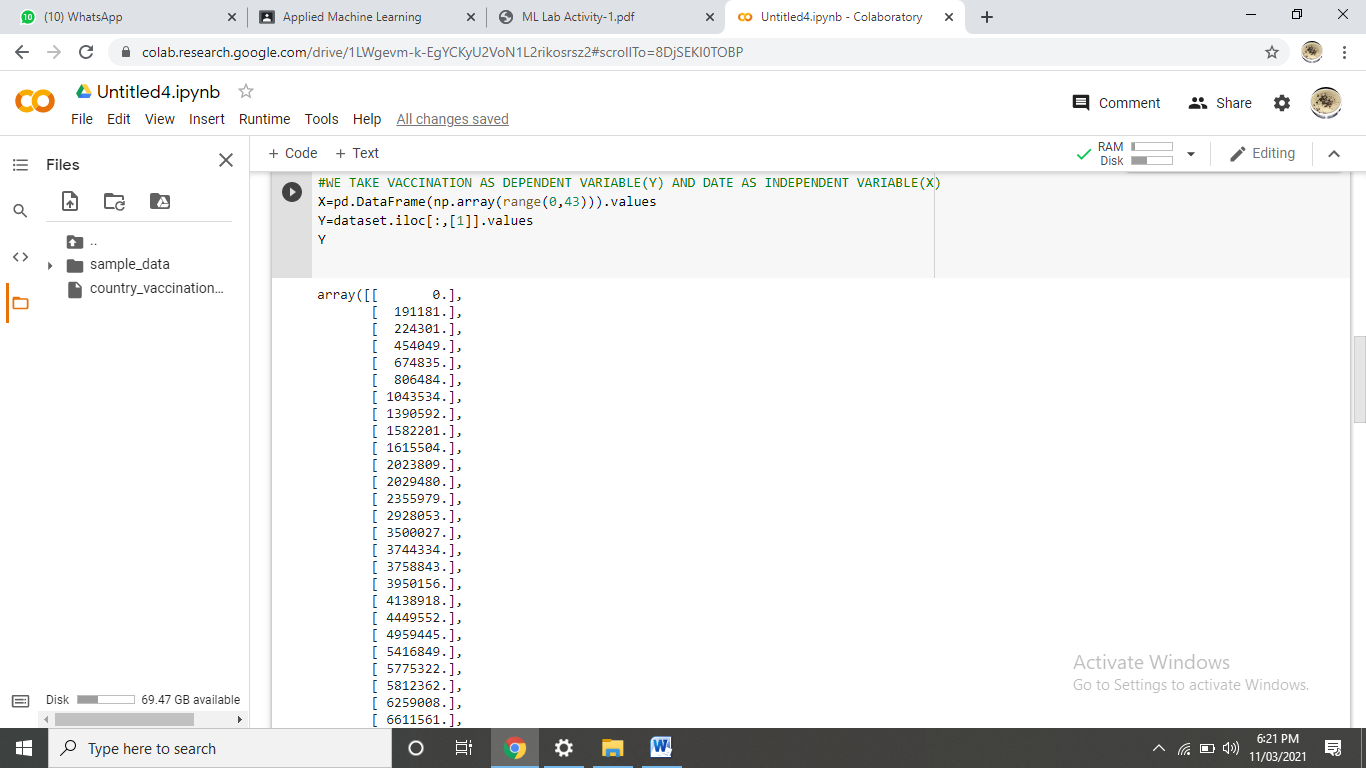
Reg.No.: 19BAI10111

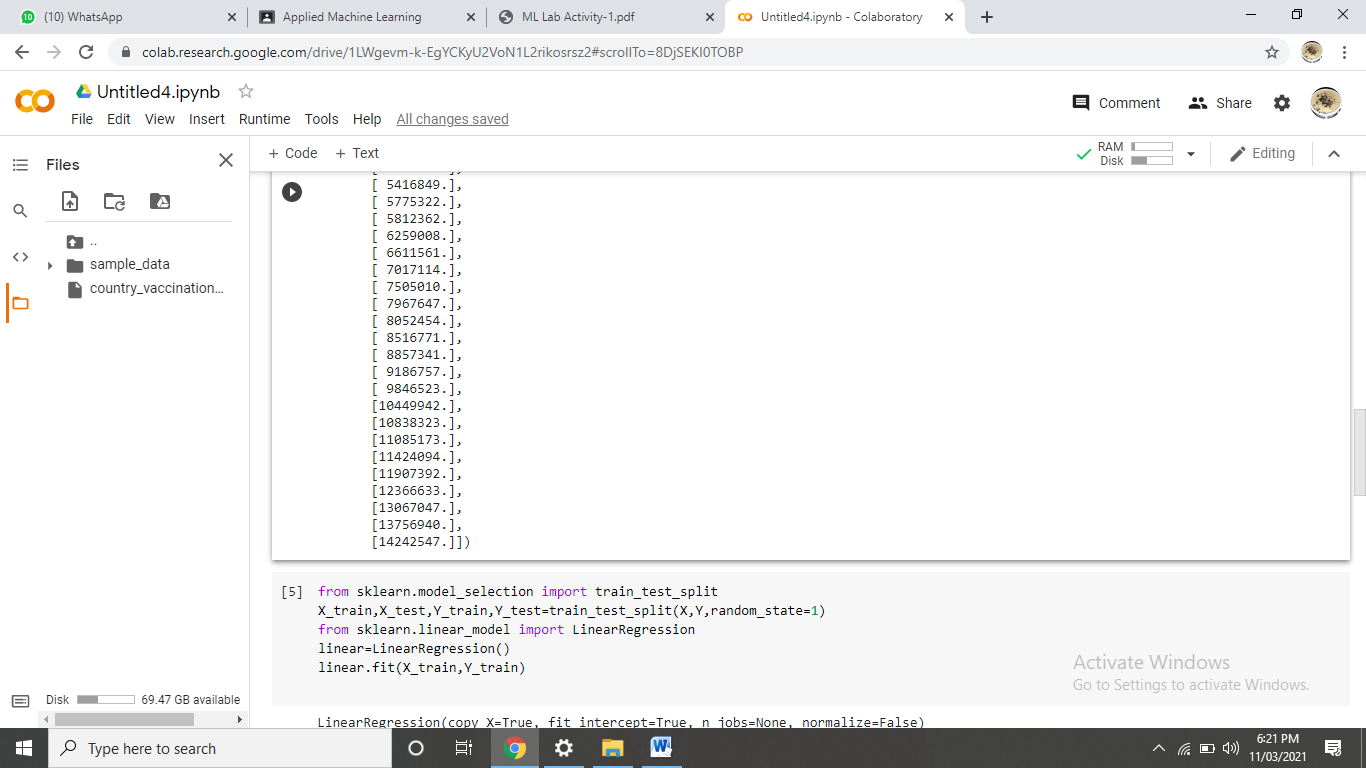
Screen Shots of programs:

**Practical: 1**

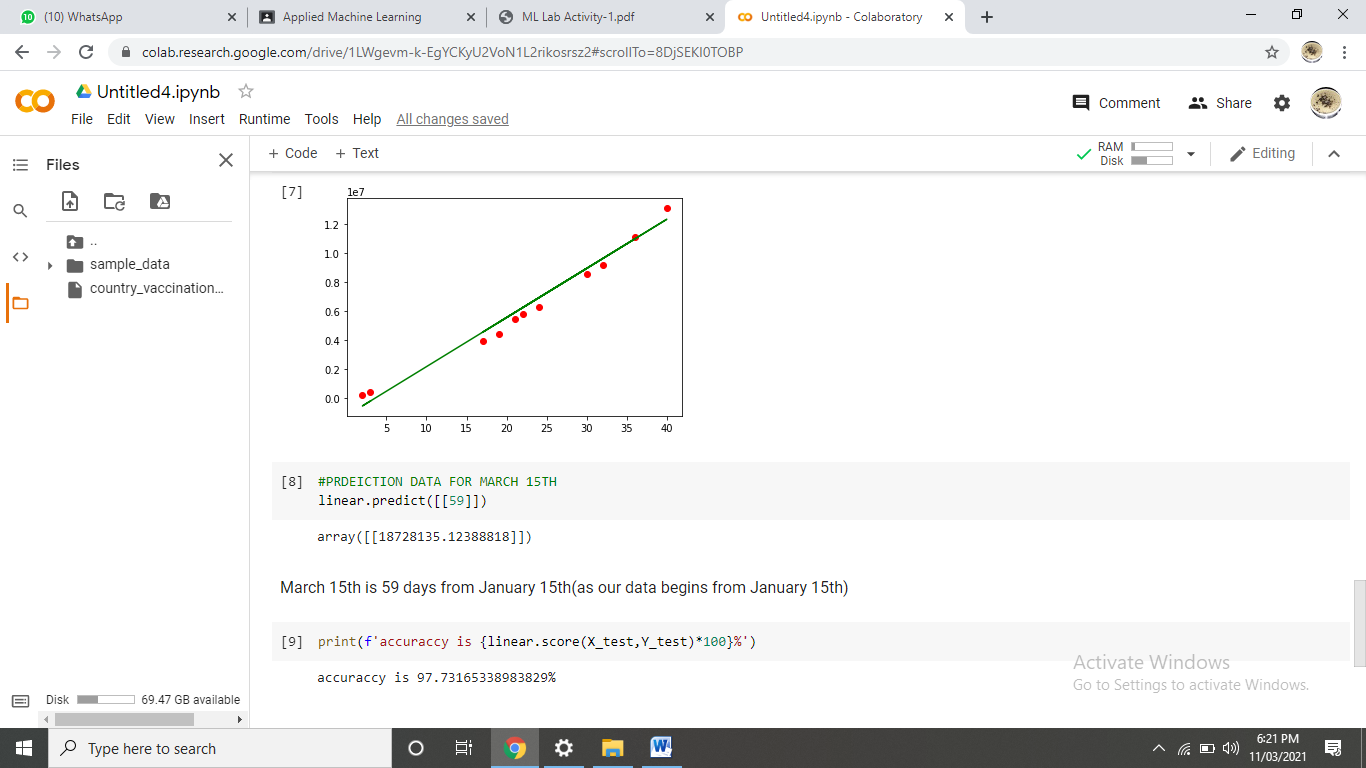




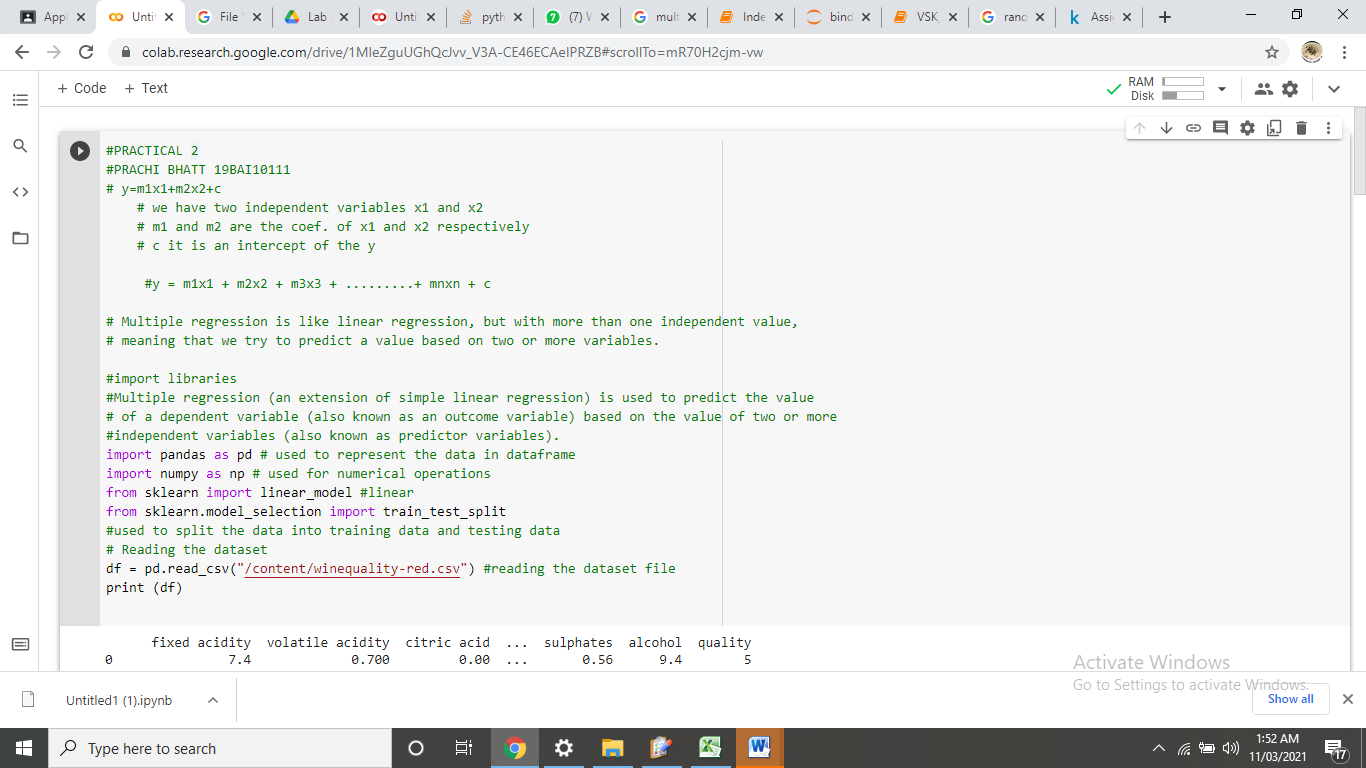


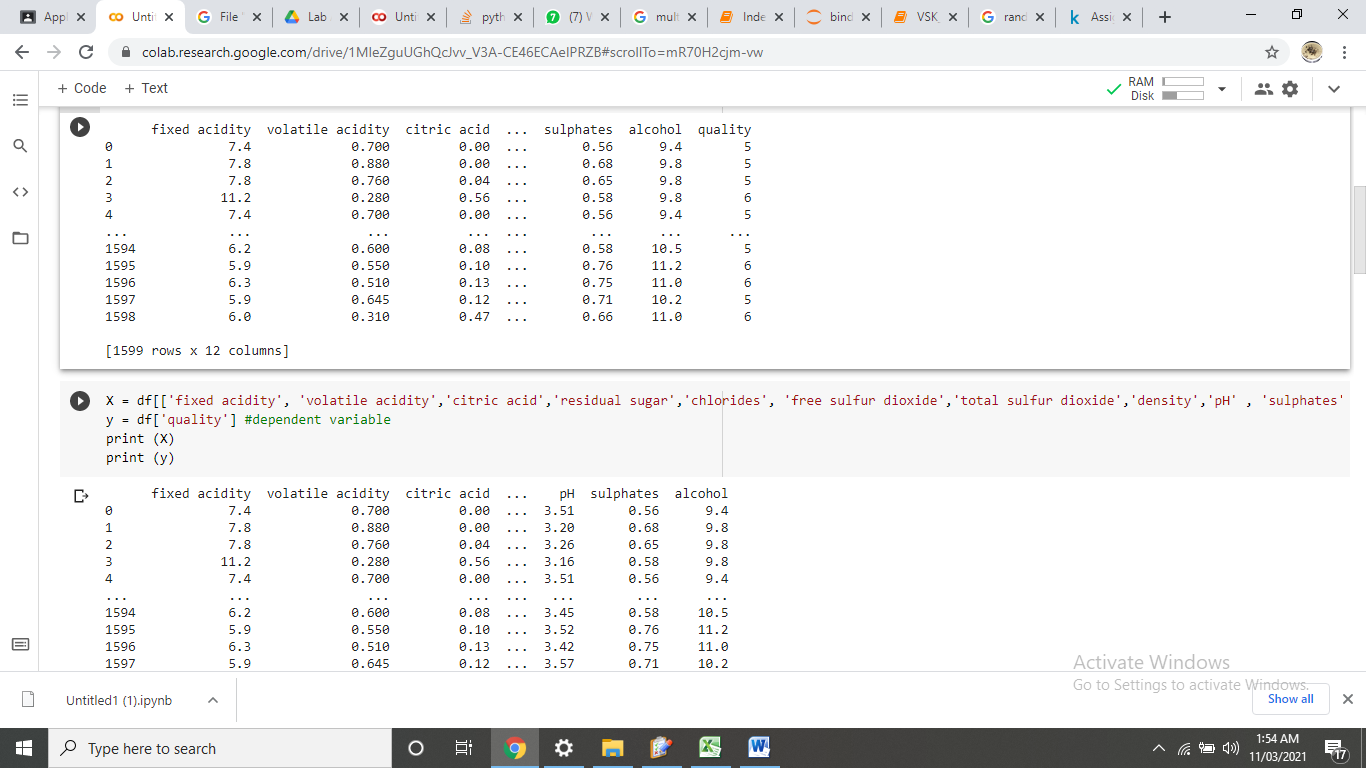


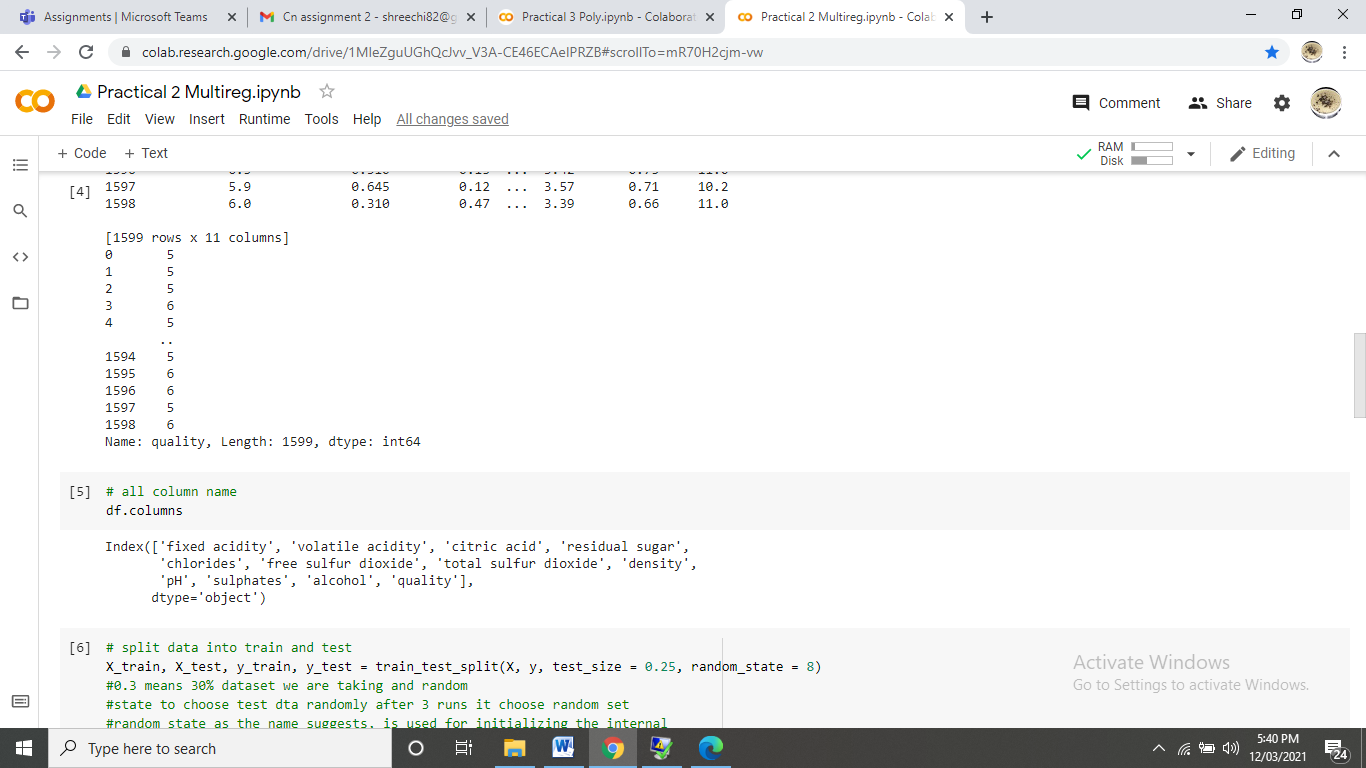


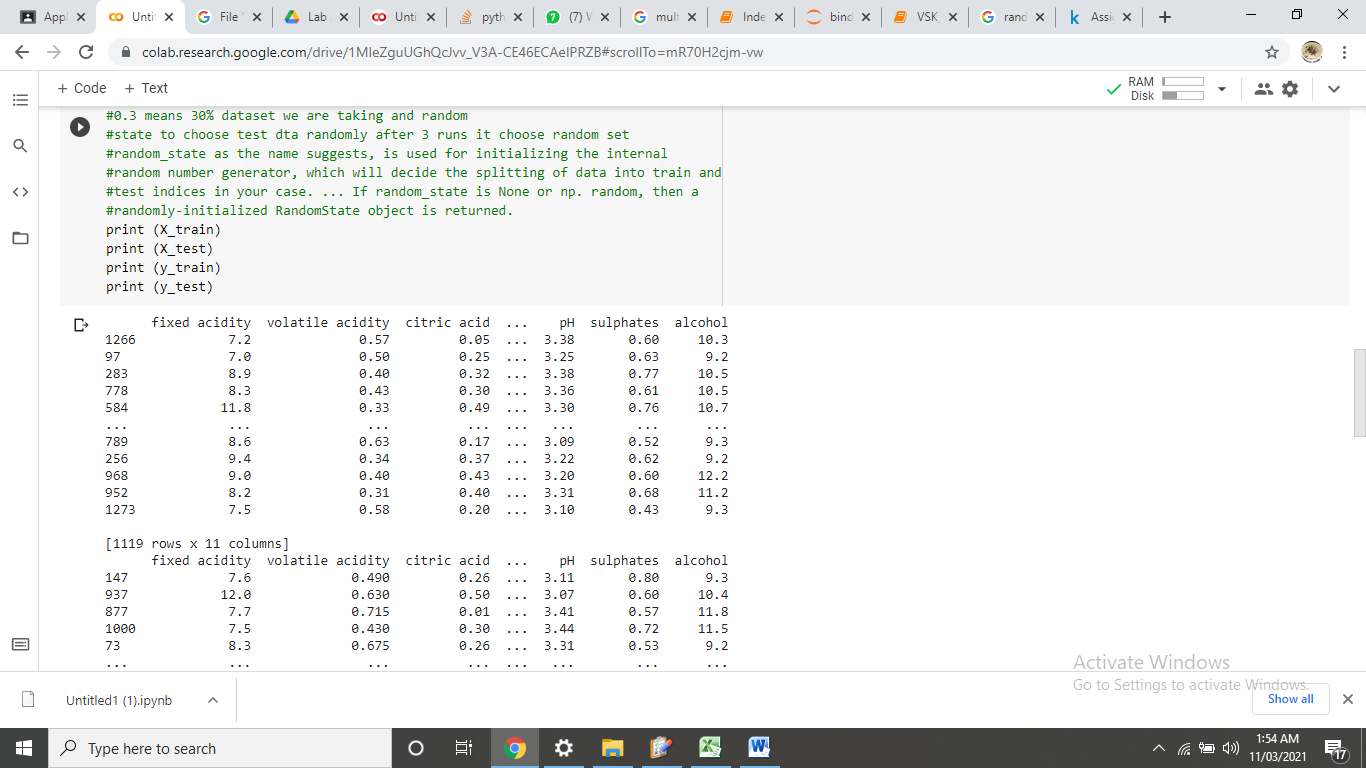


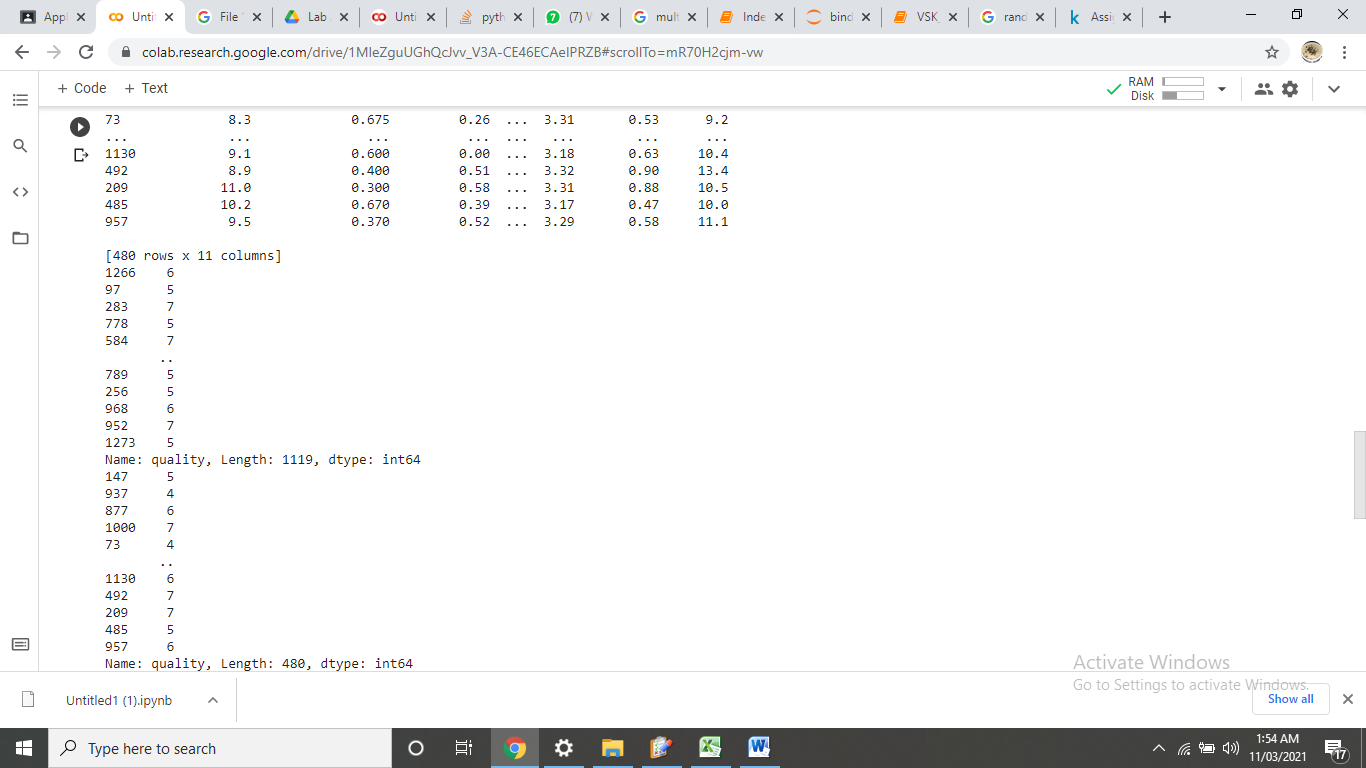
**Practical: 2**

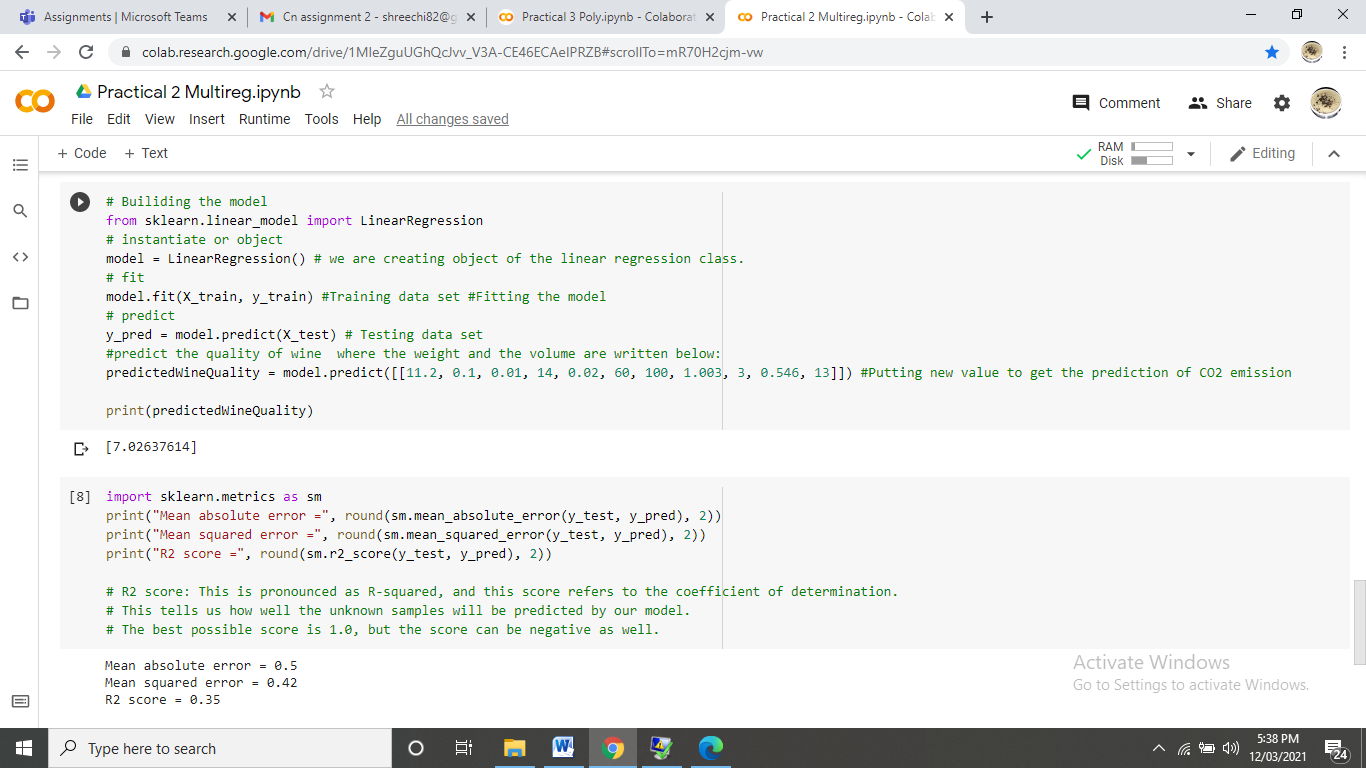
**



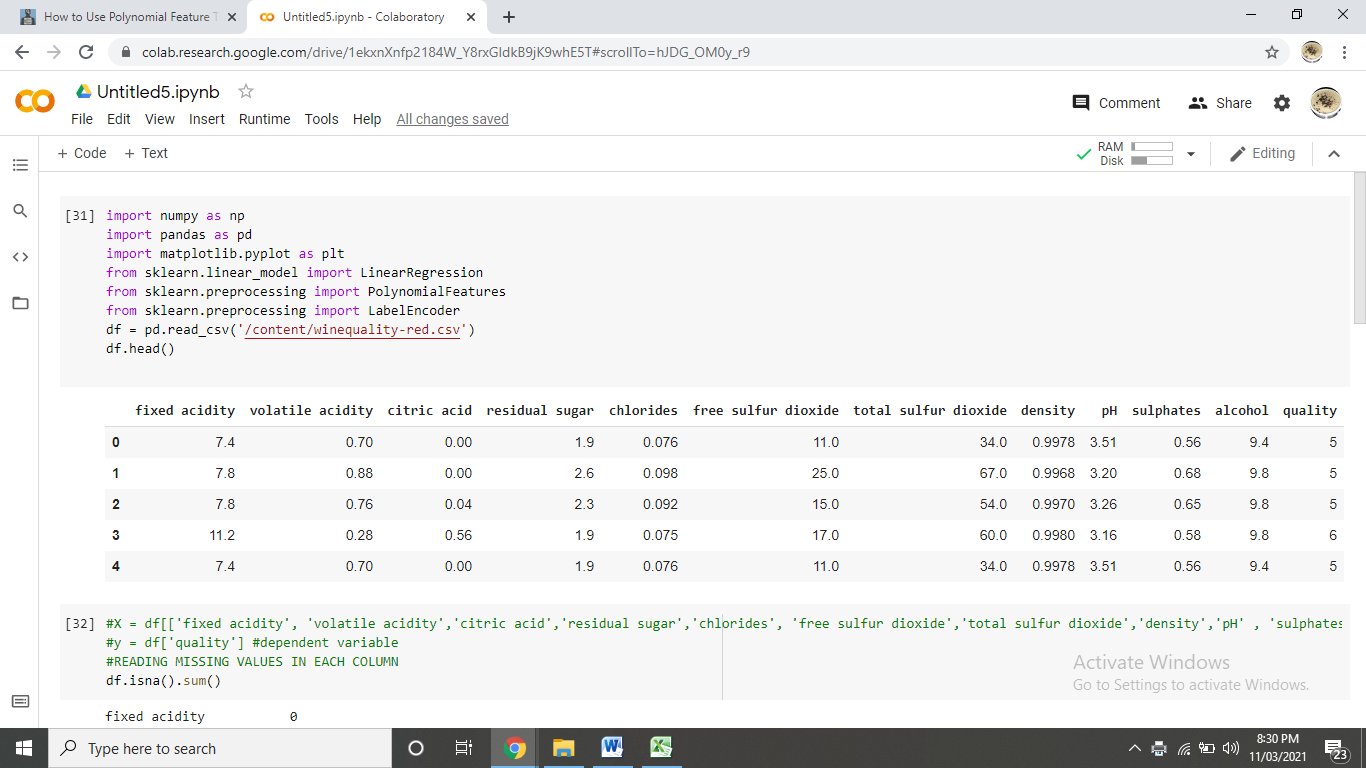


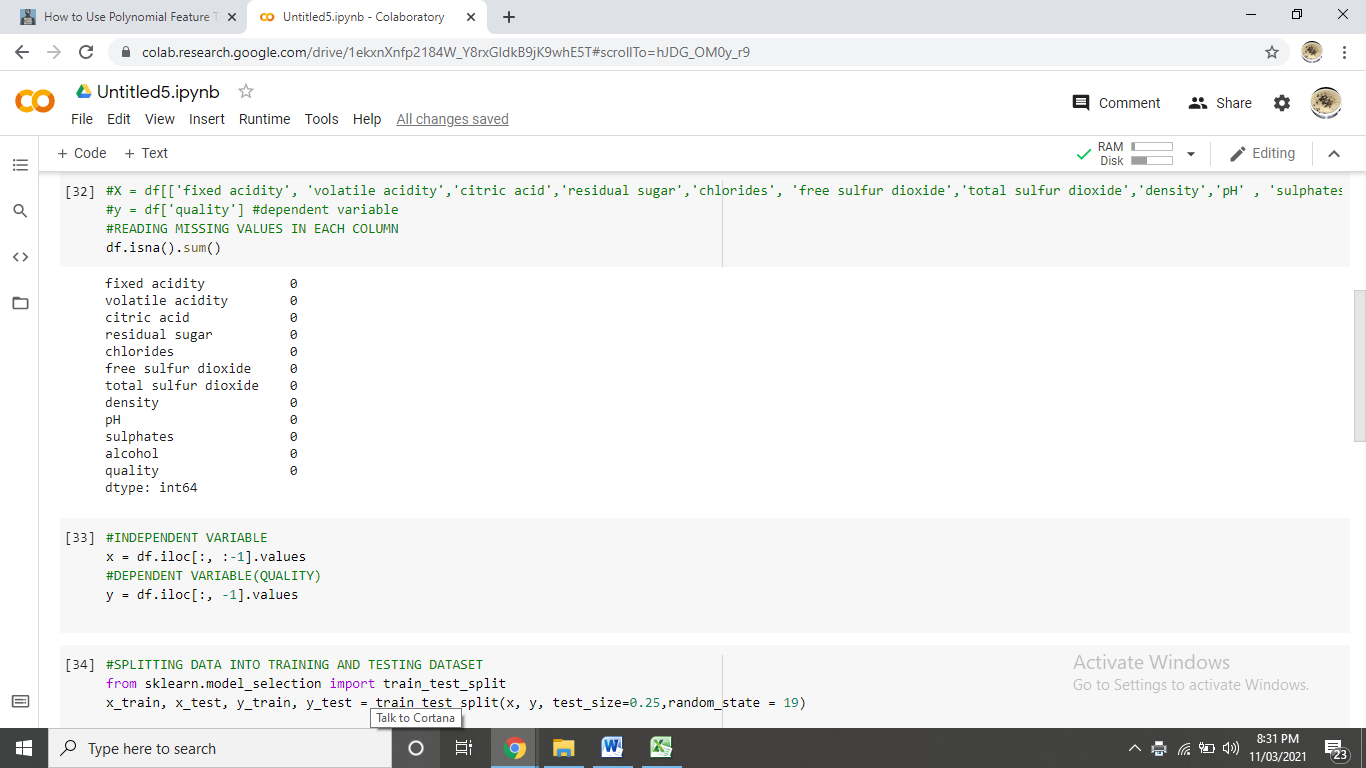


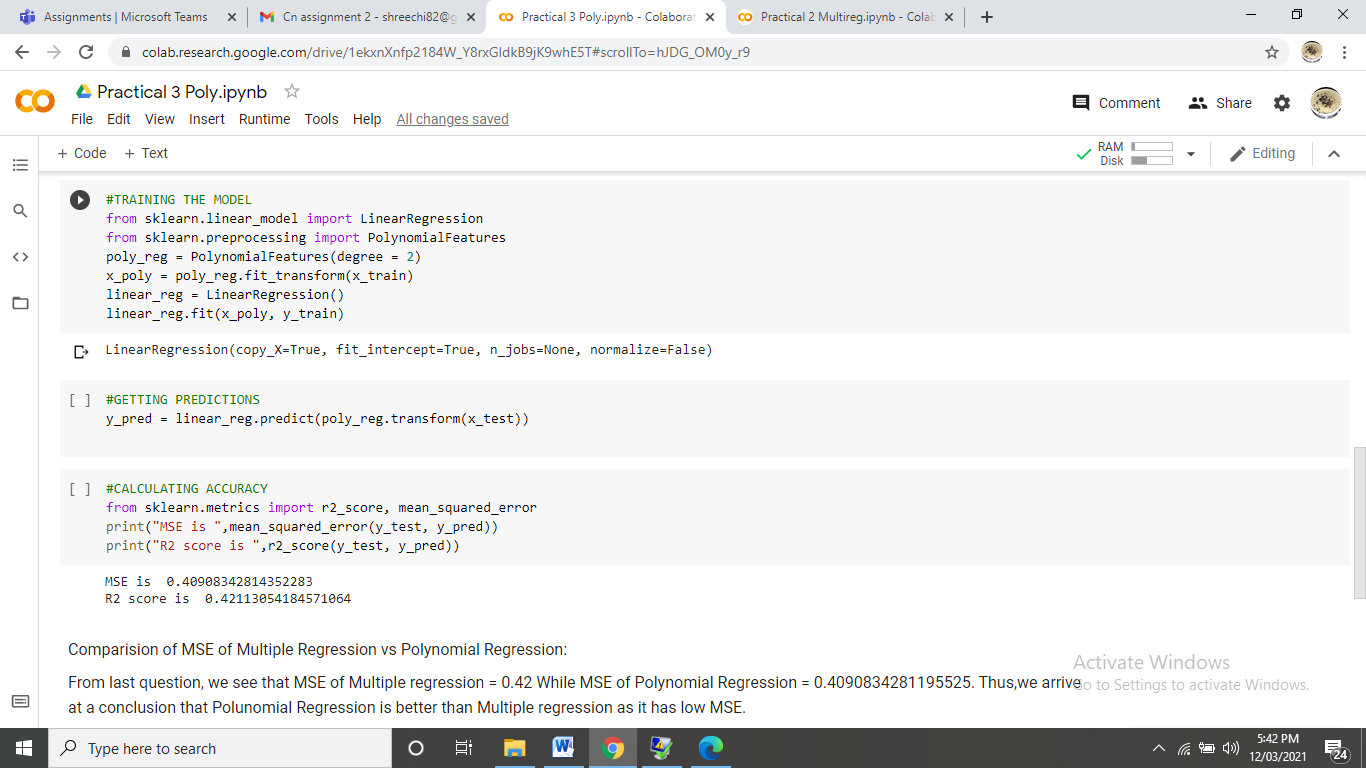




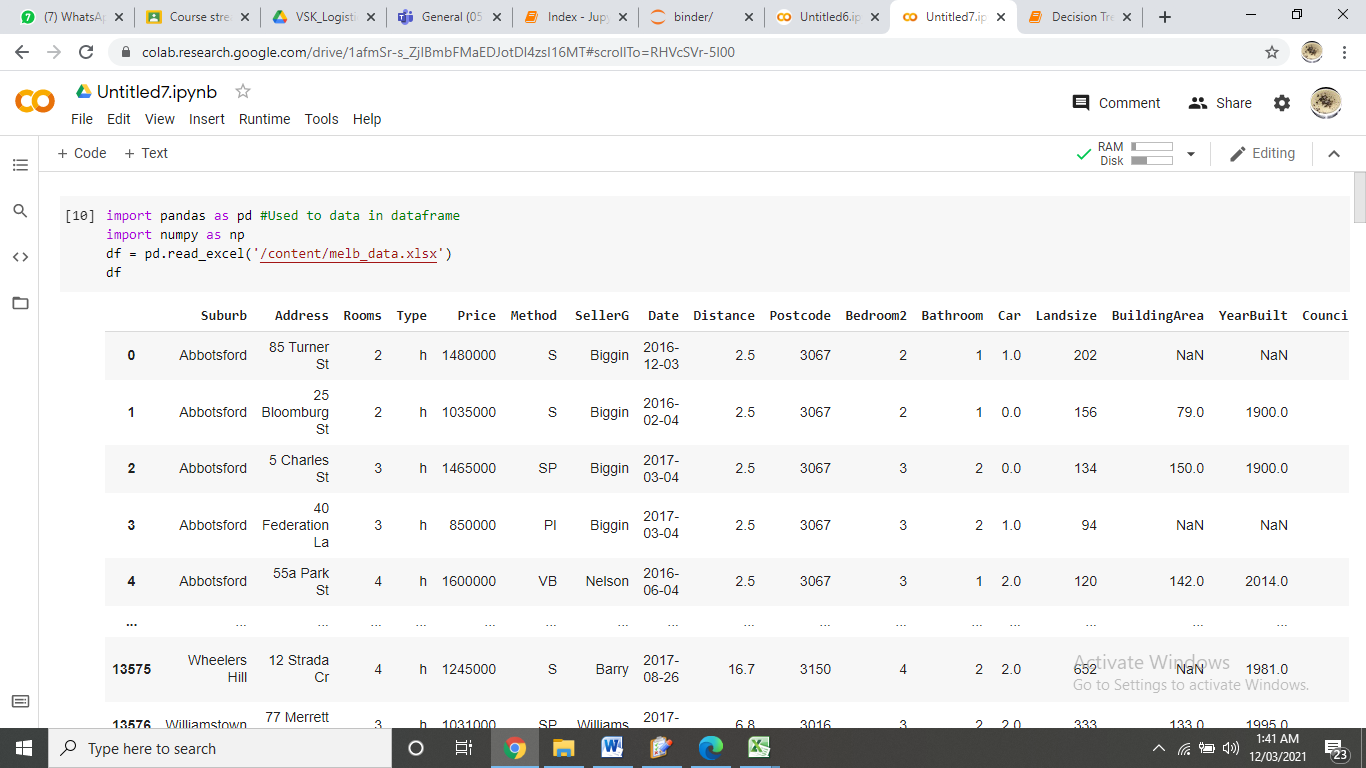
Practical: 3

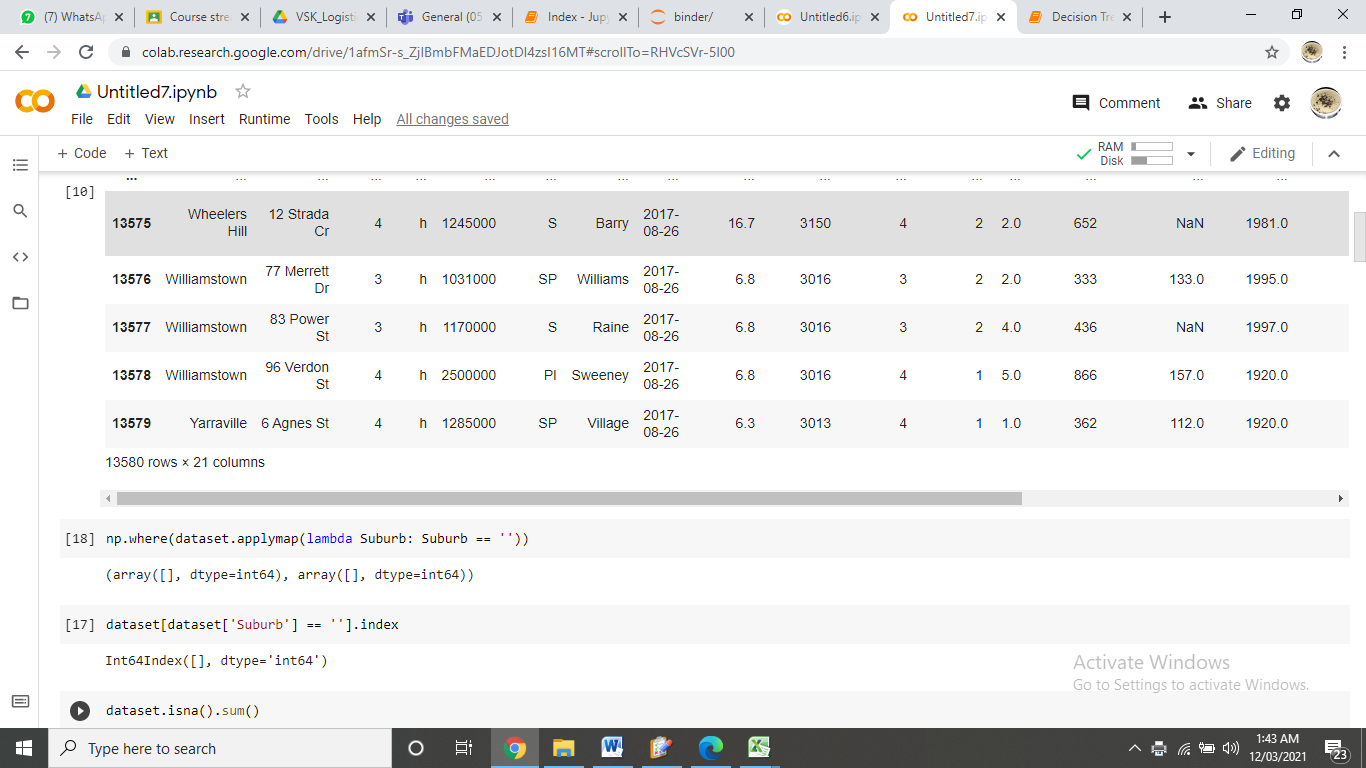


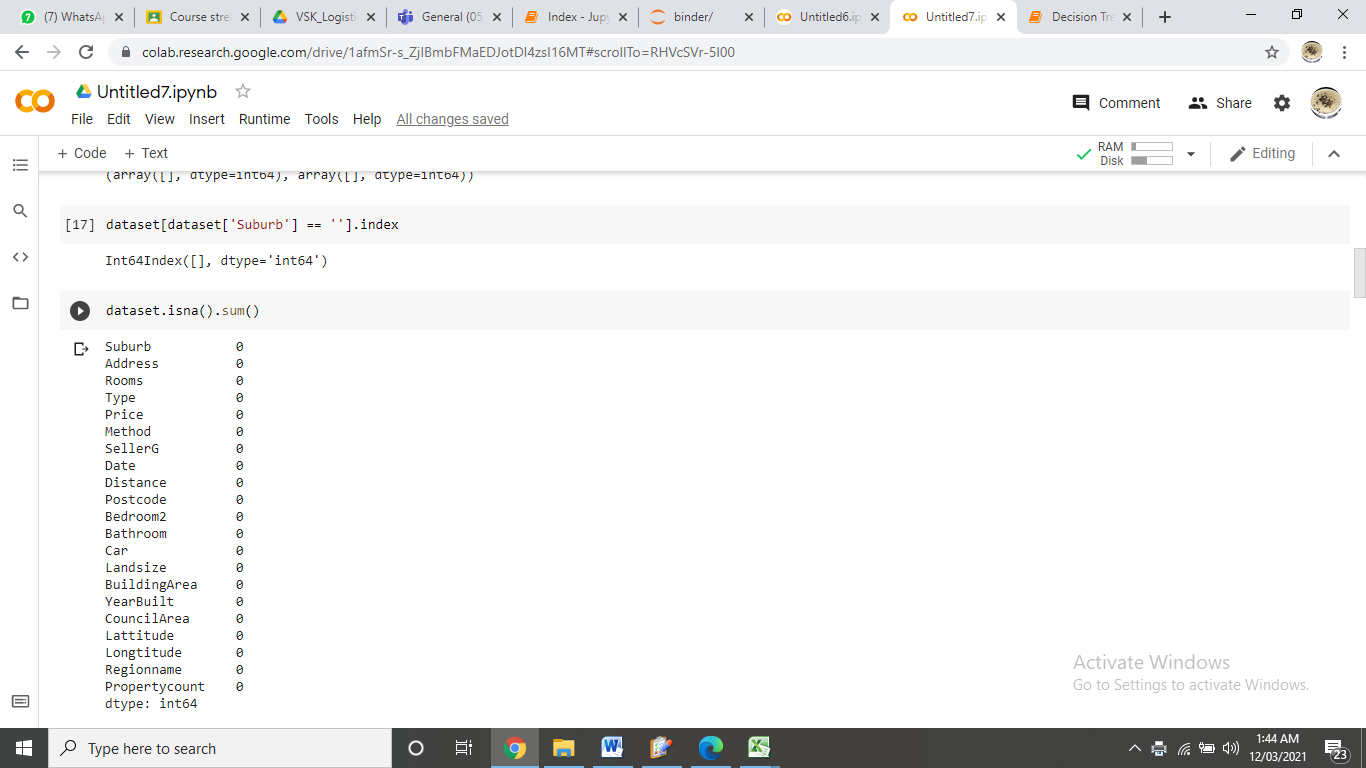


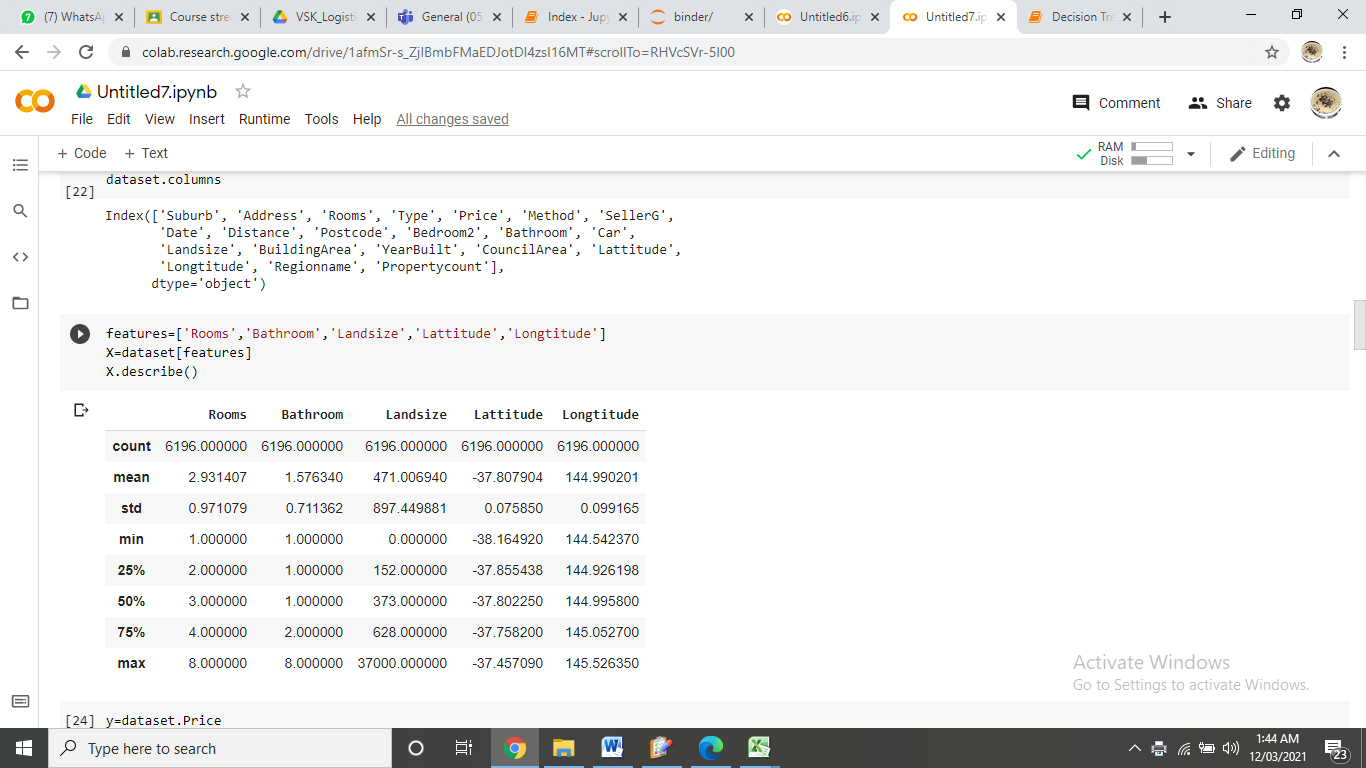


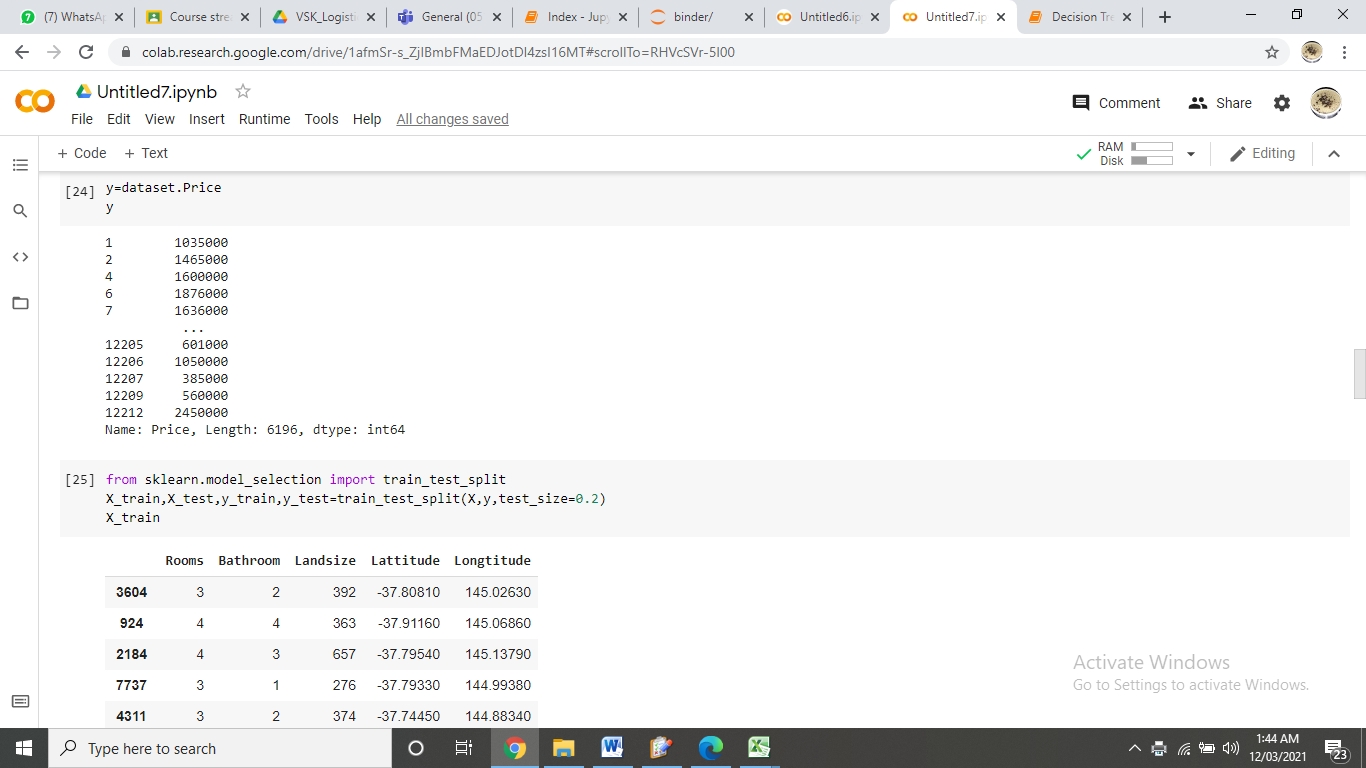
Practical: 4

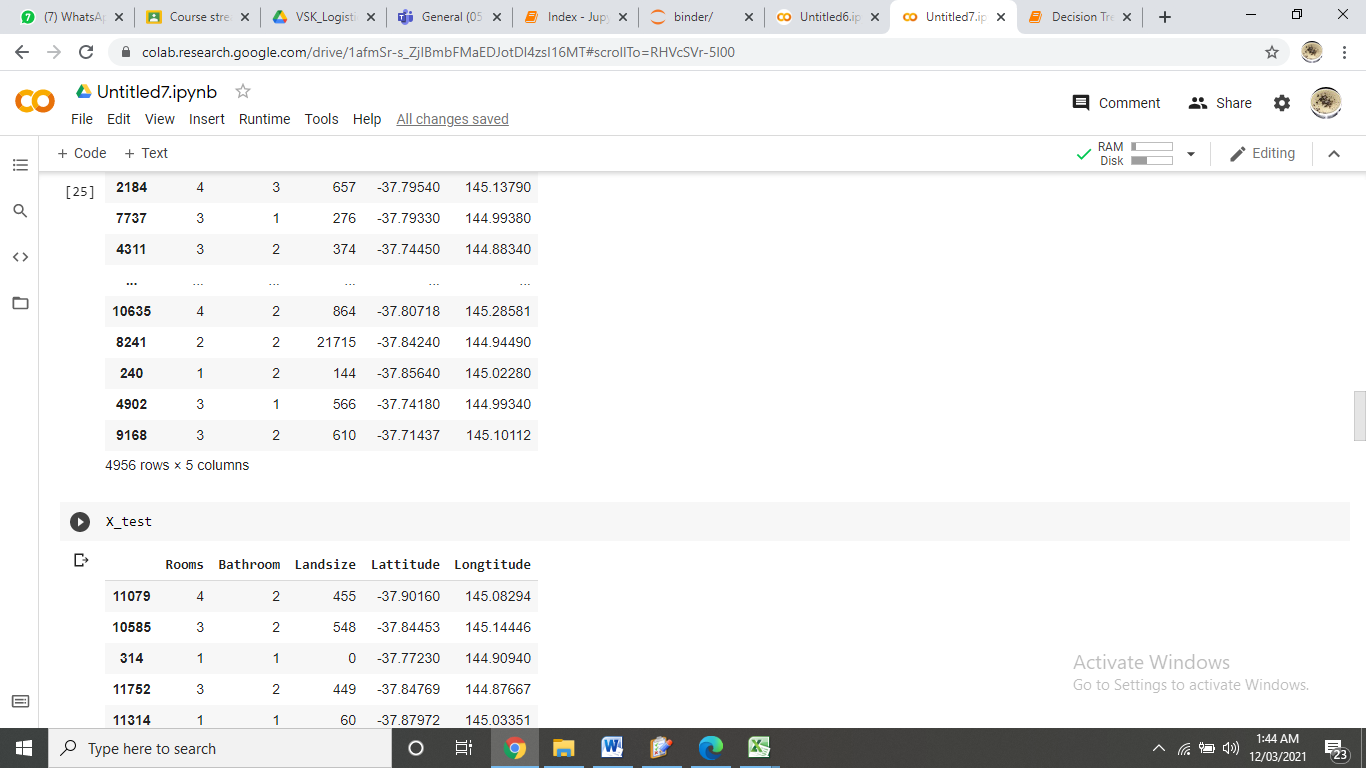


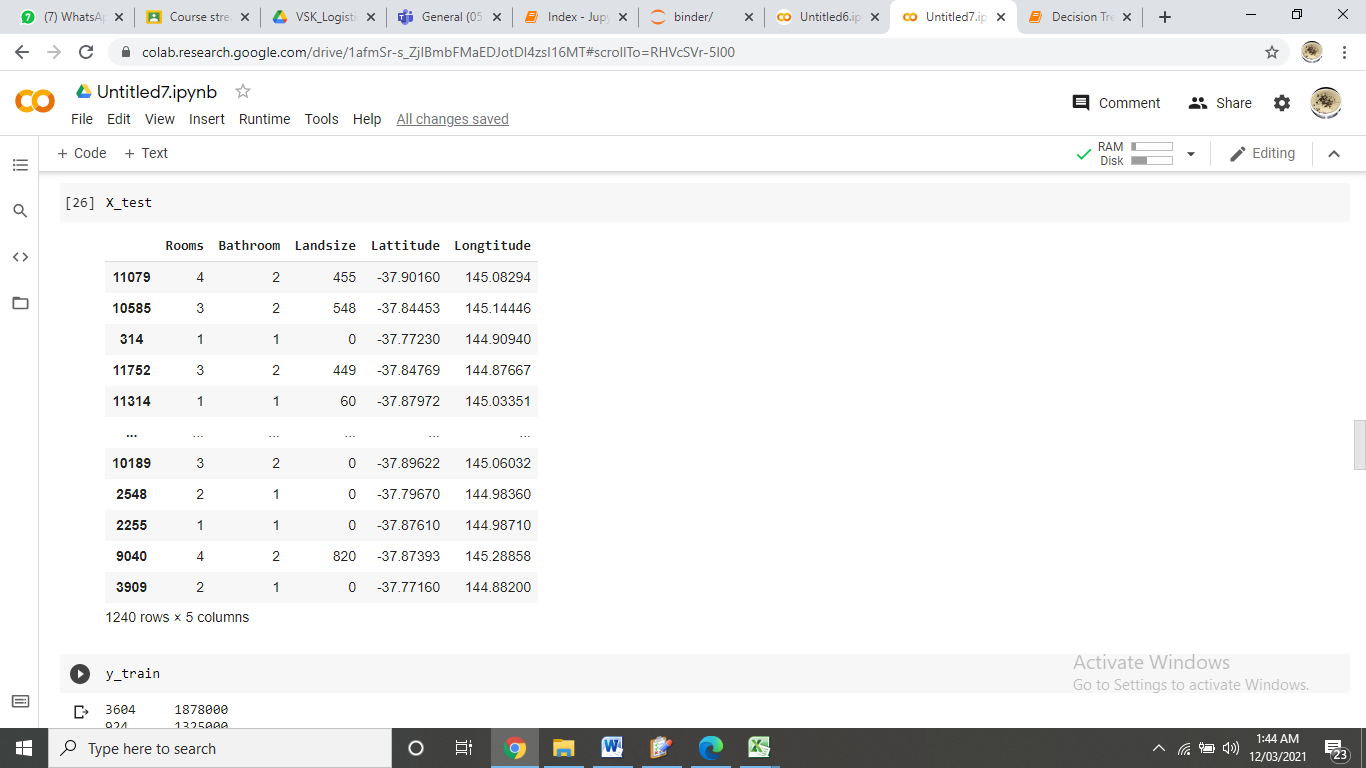


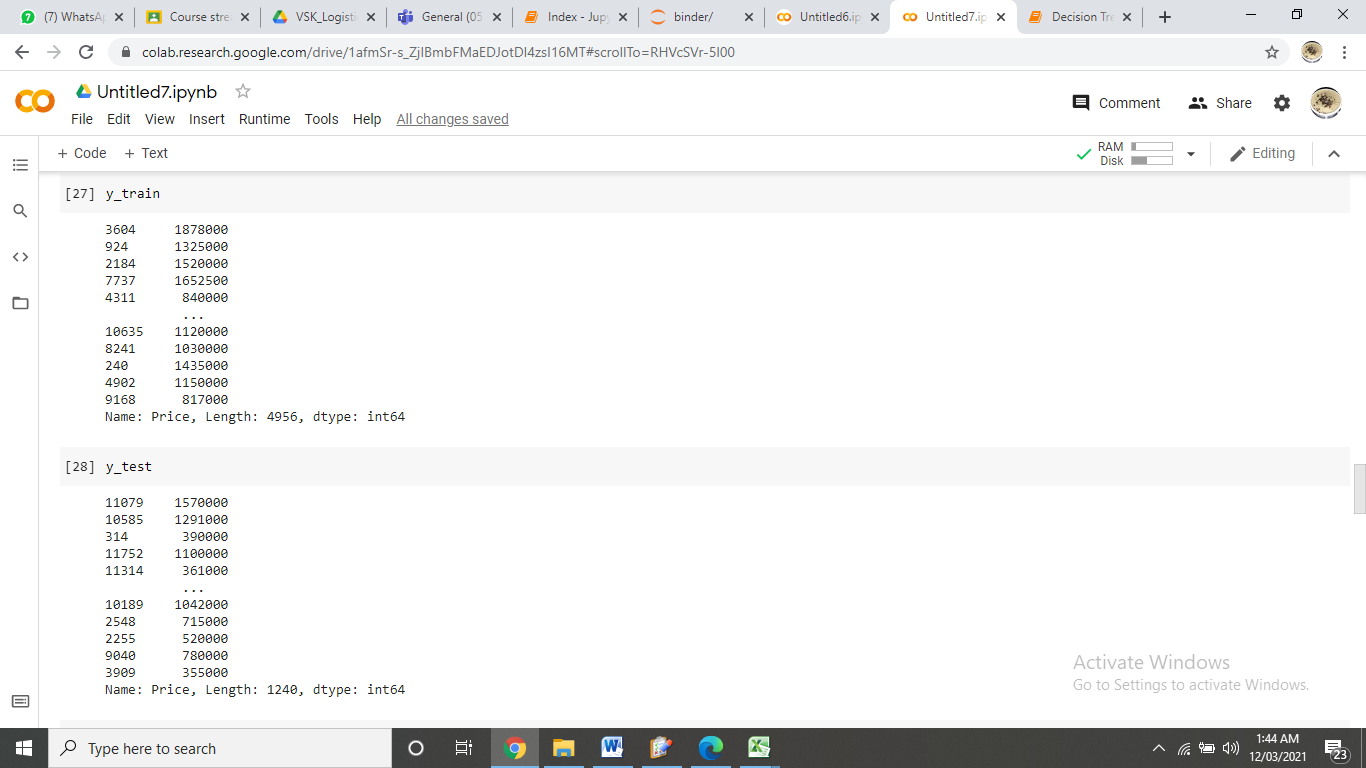


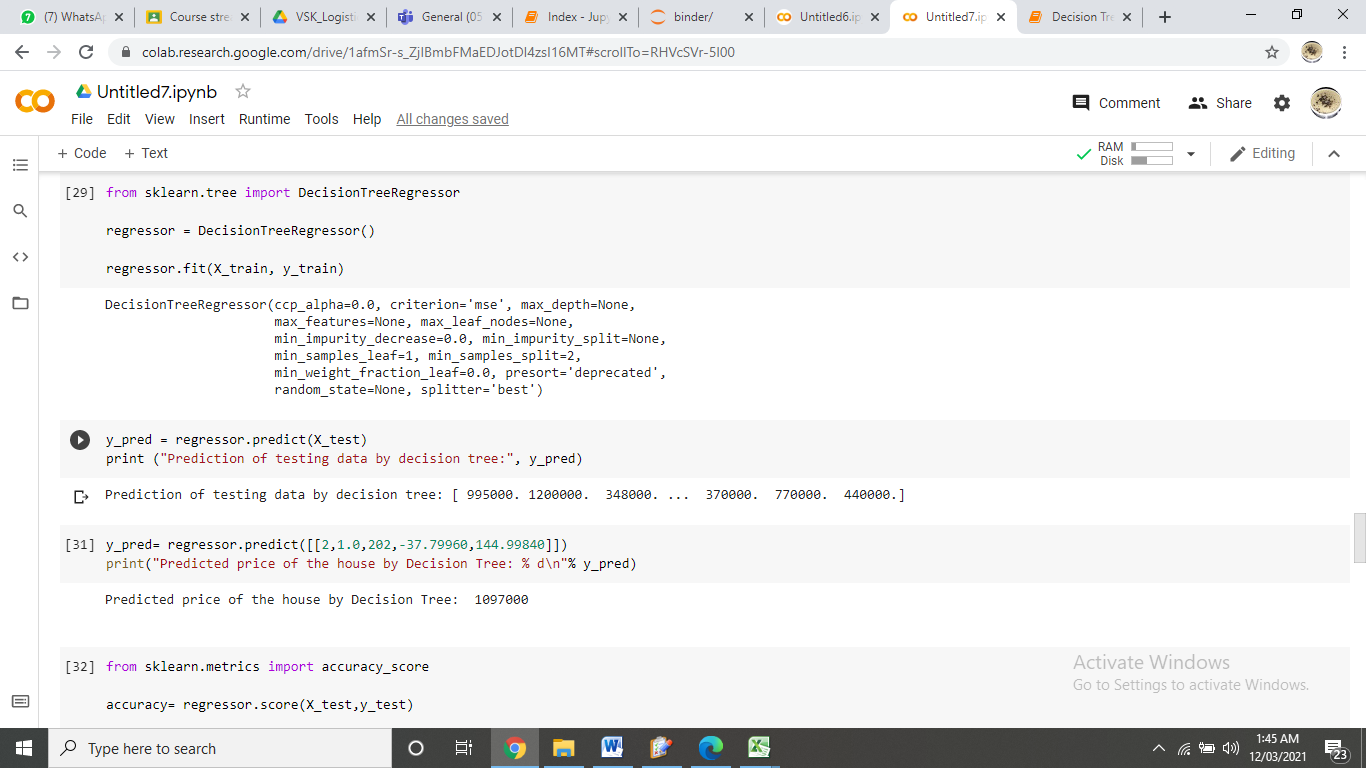


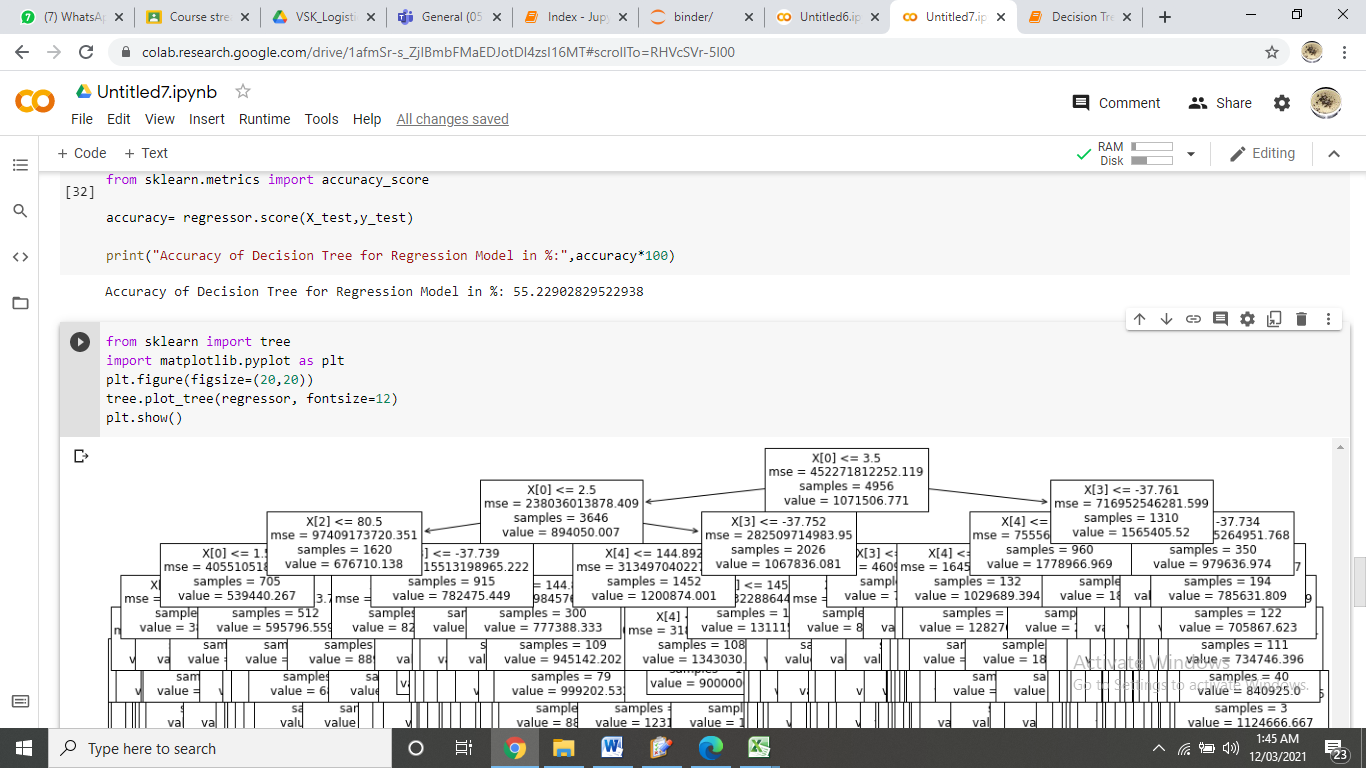


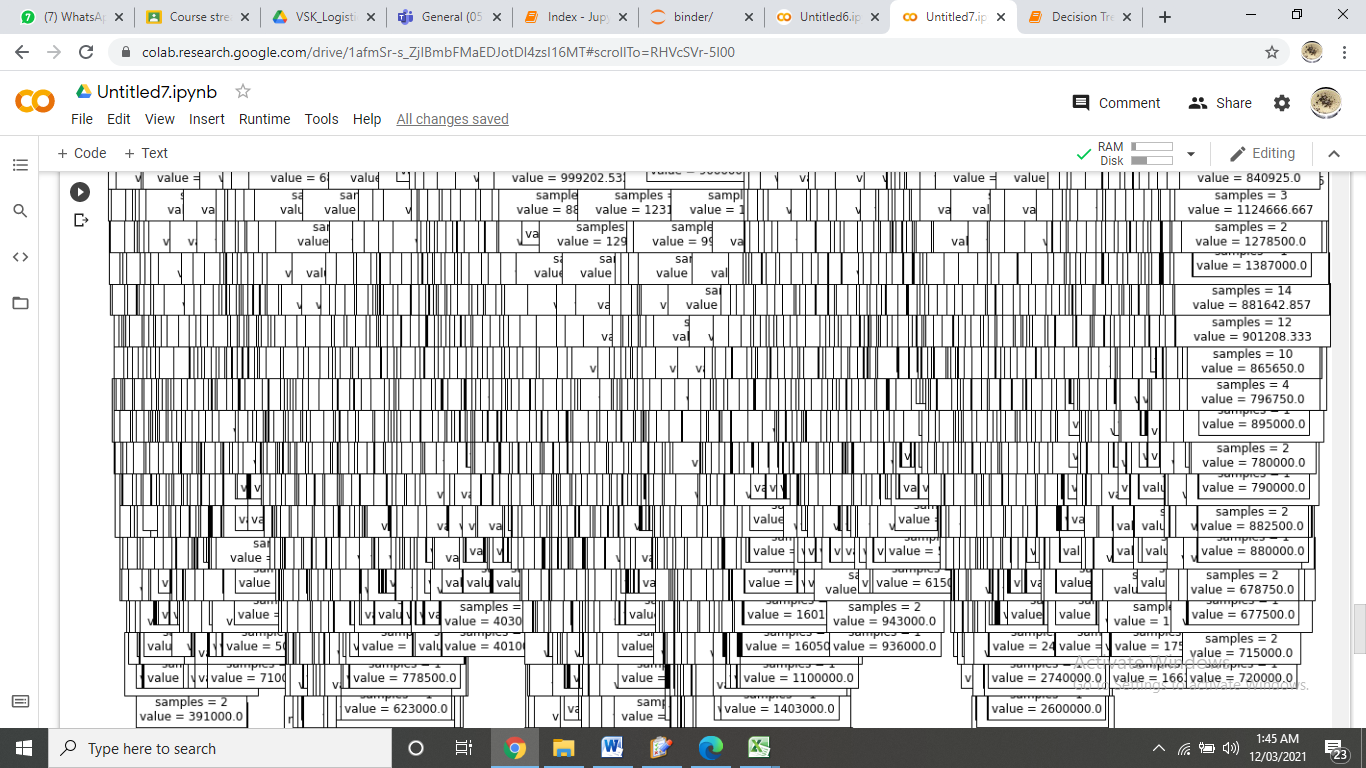


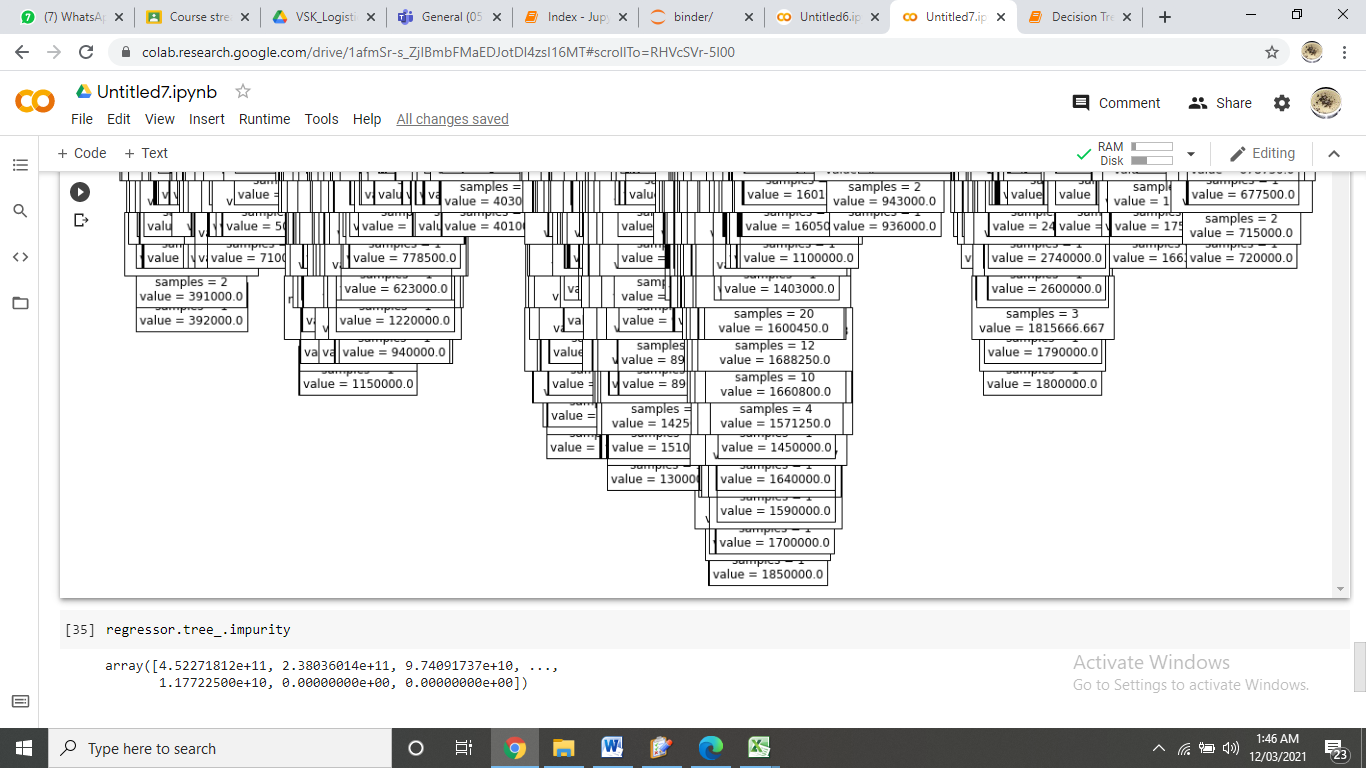




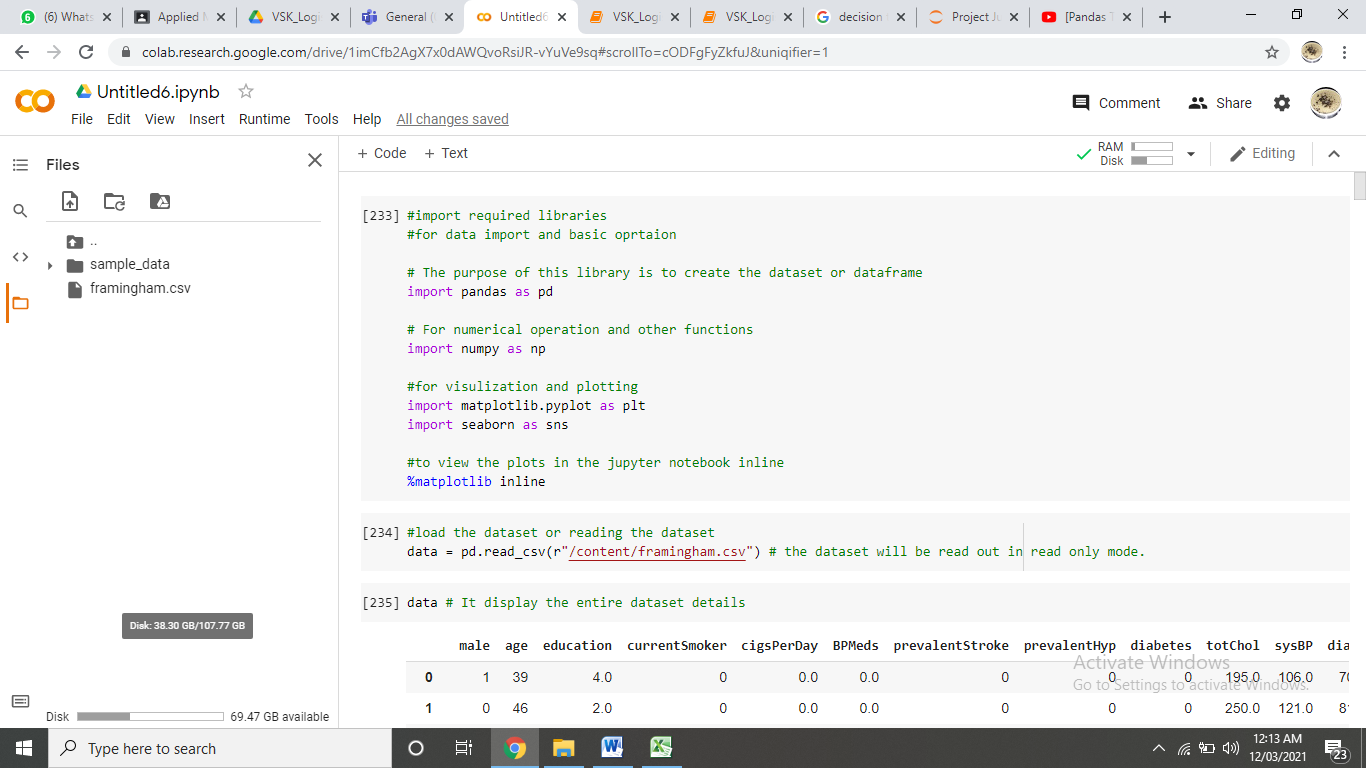


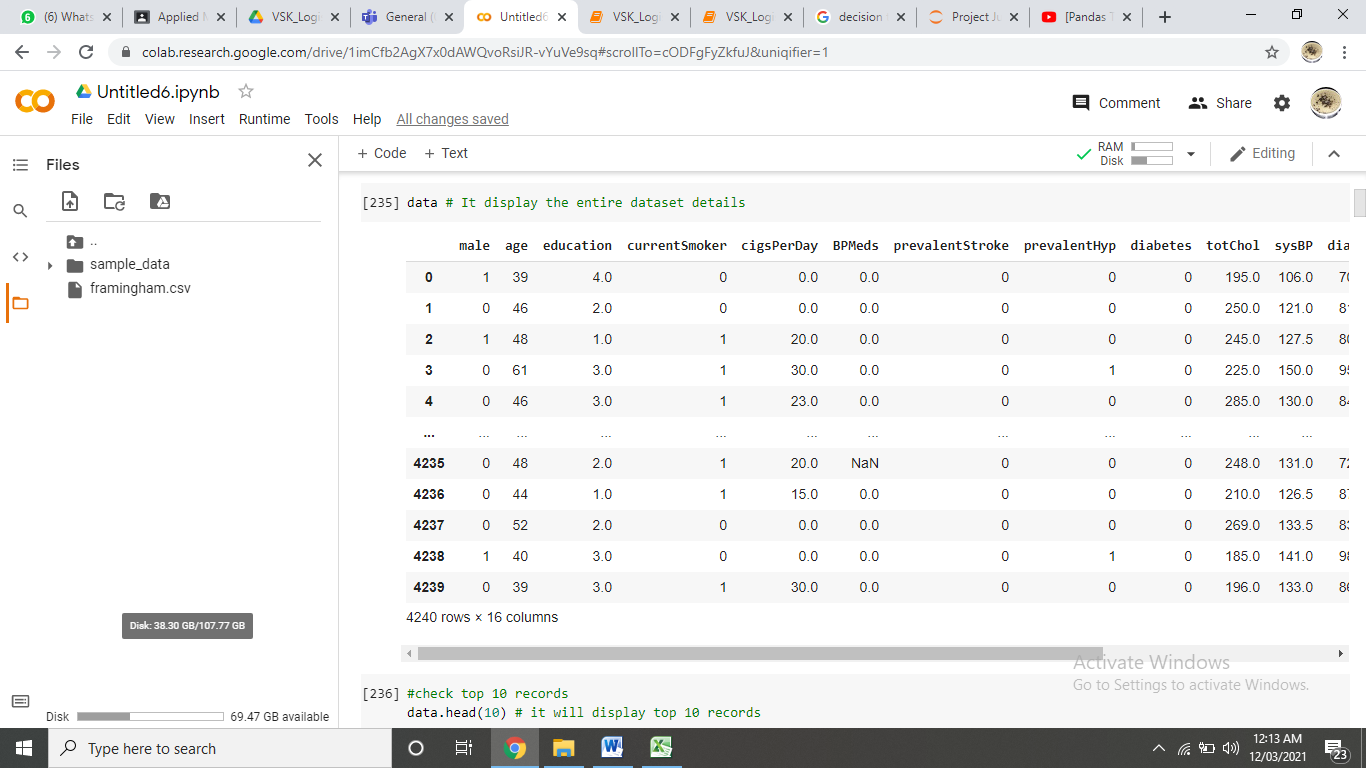


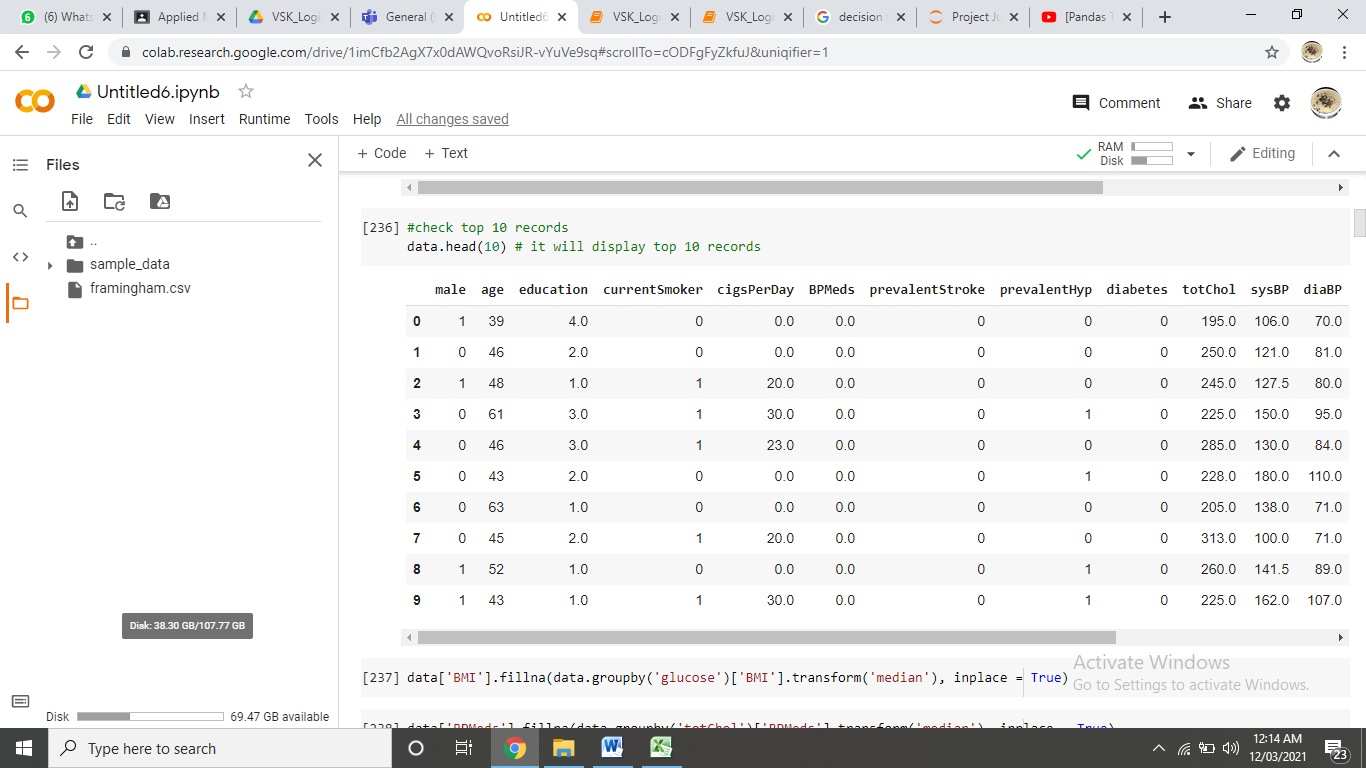


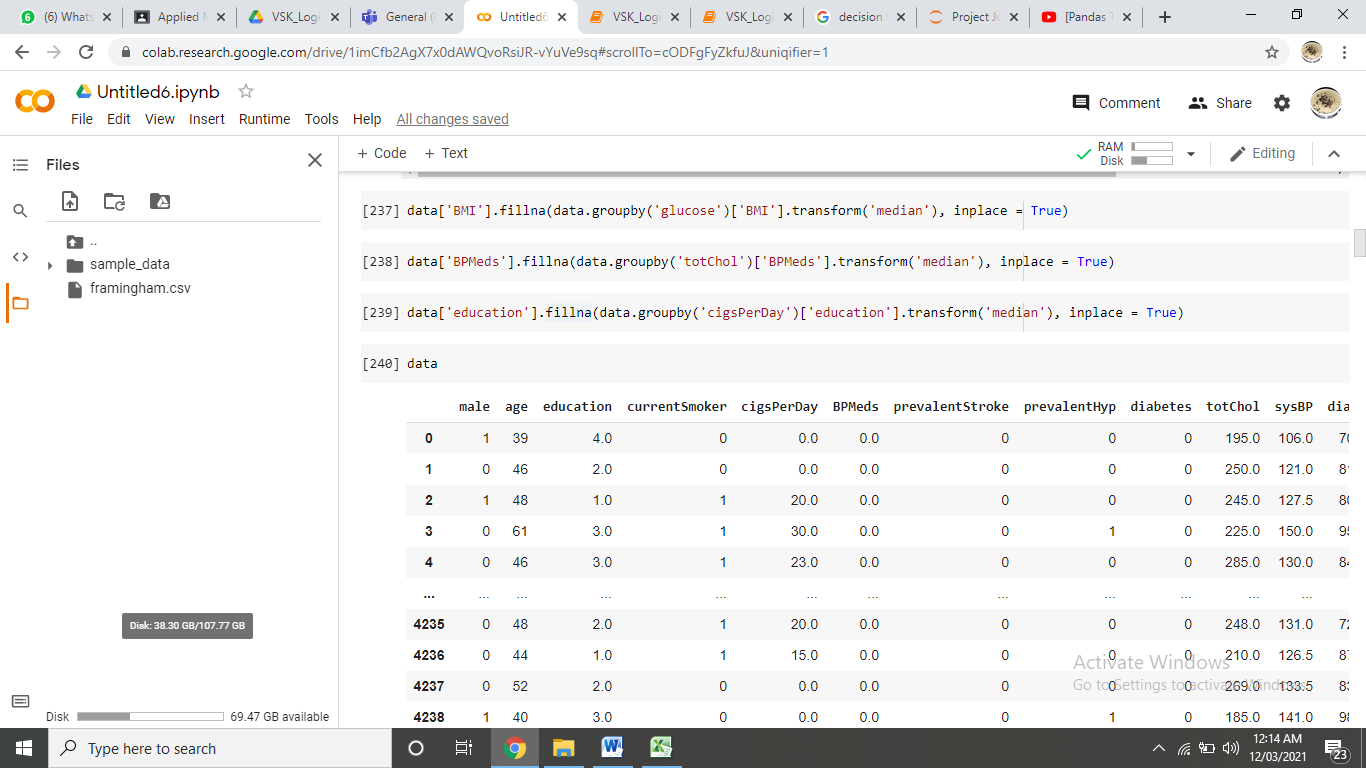


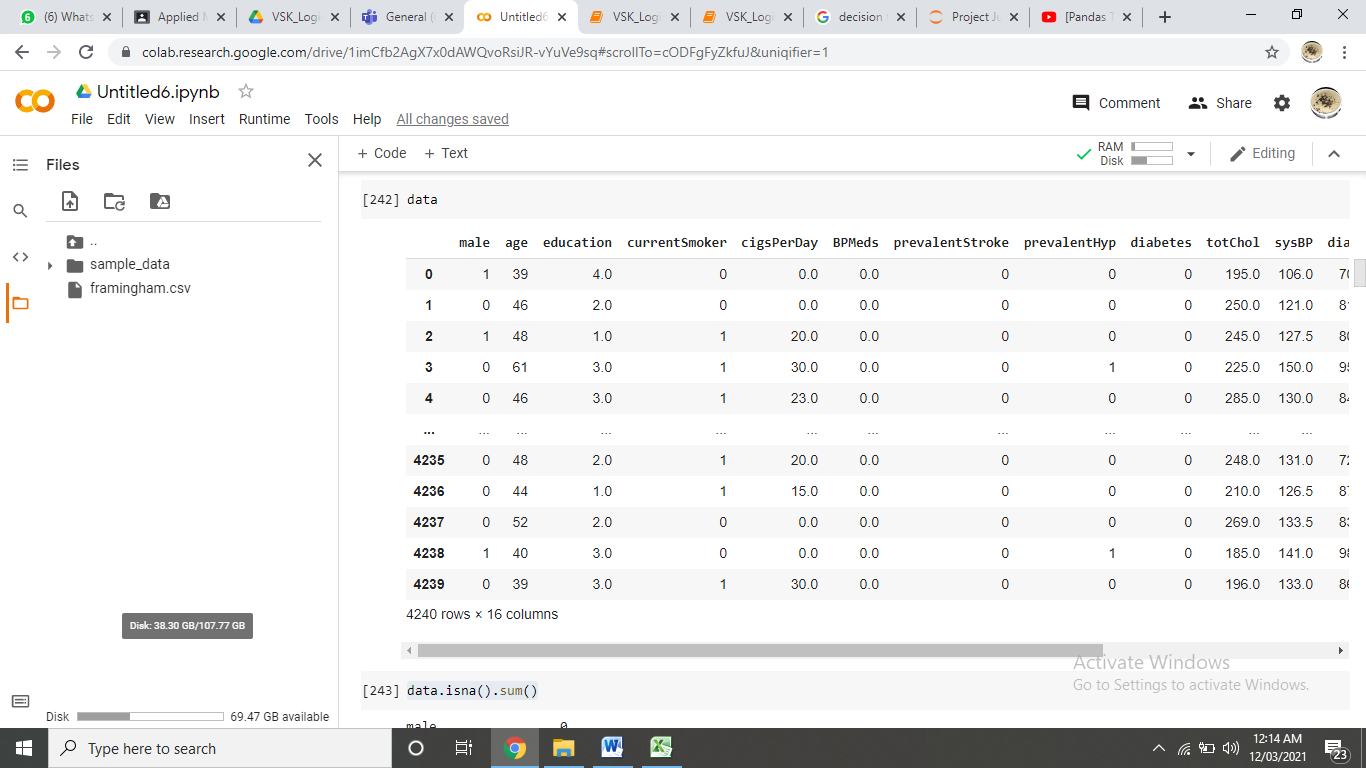
Practical: 5

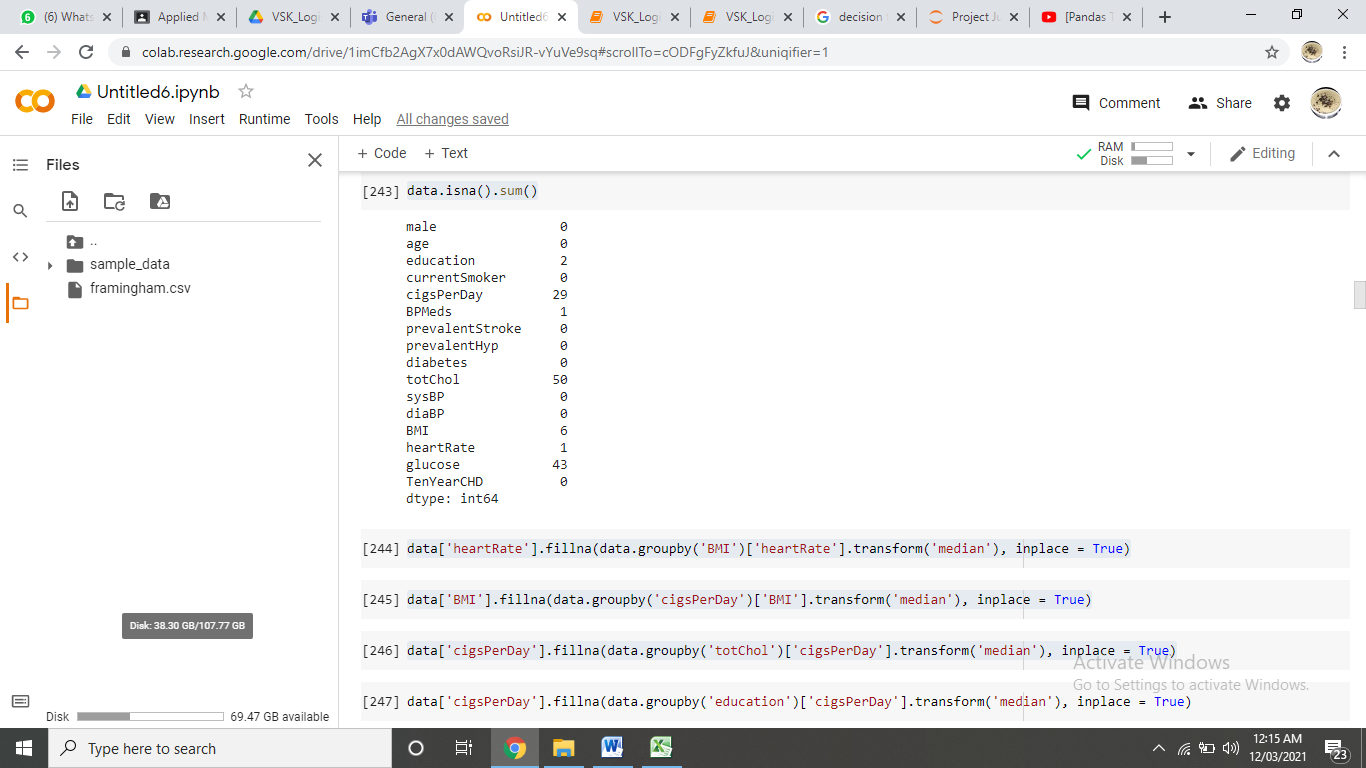


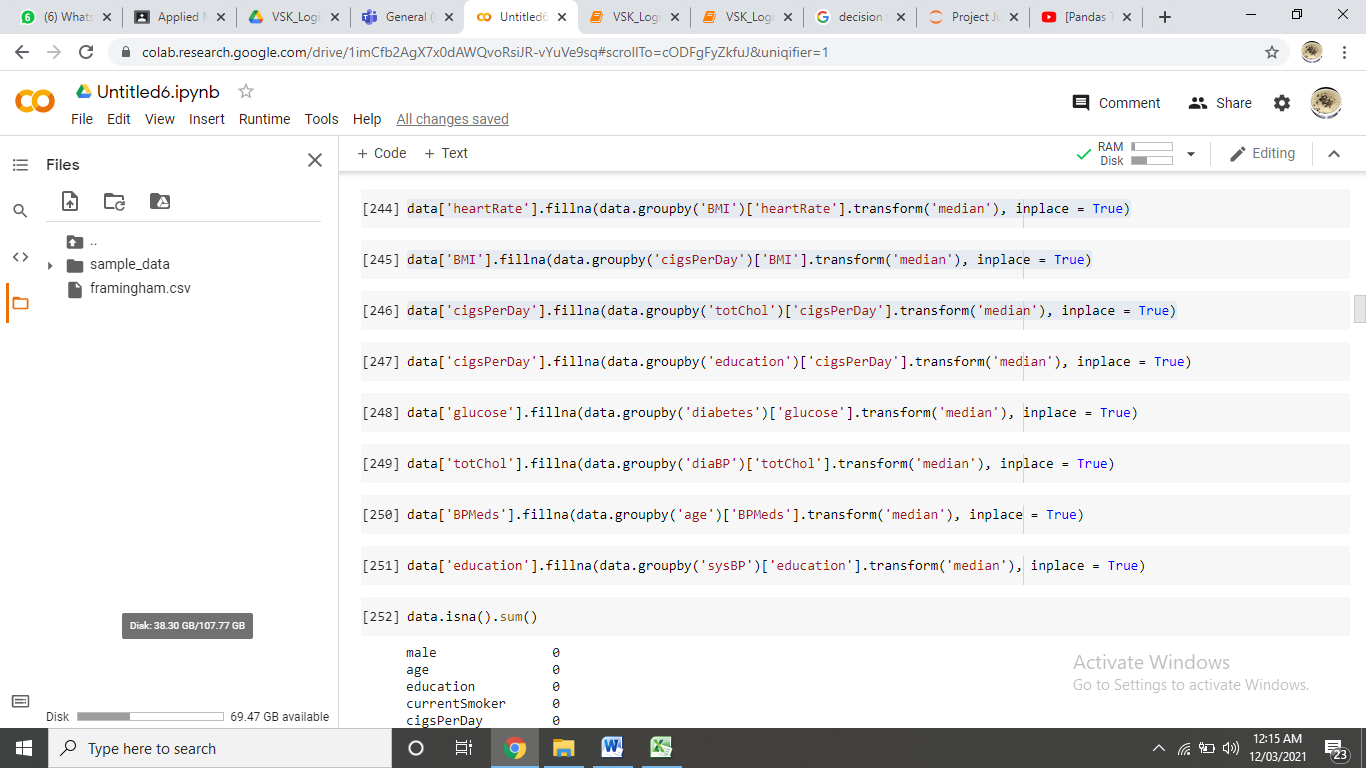


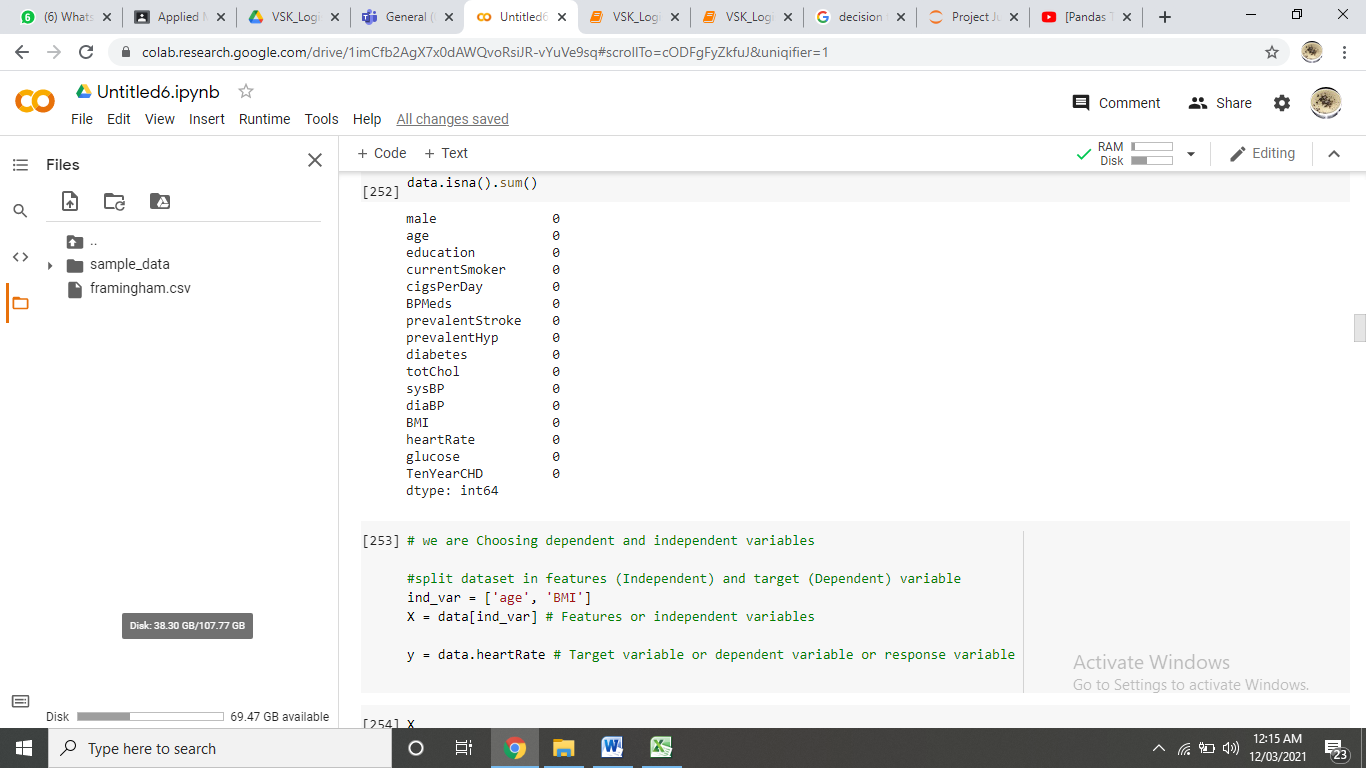


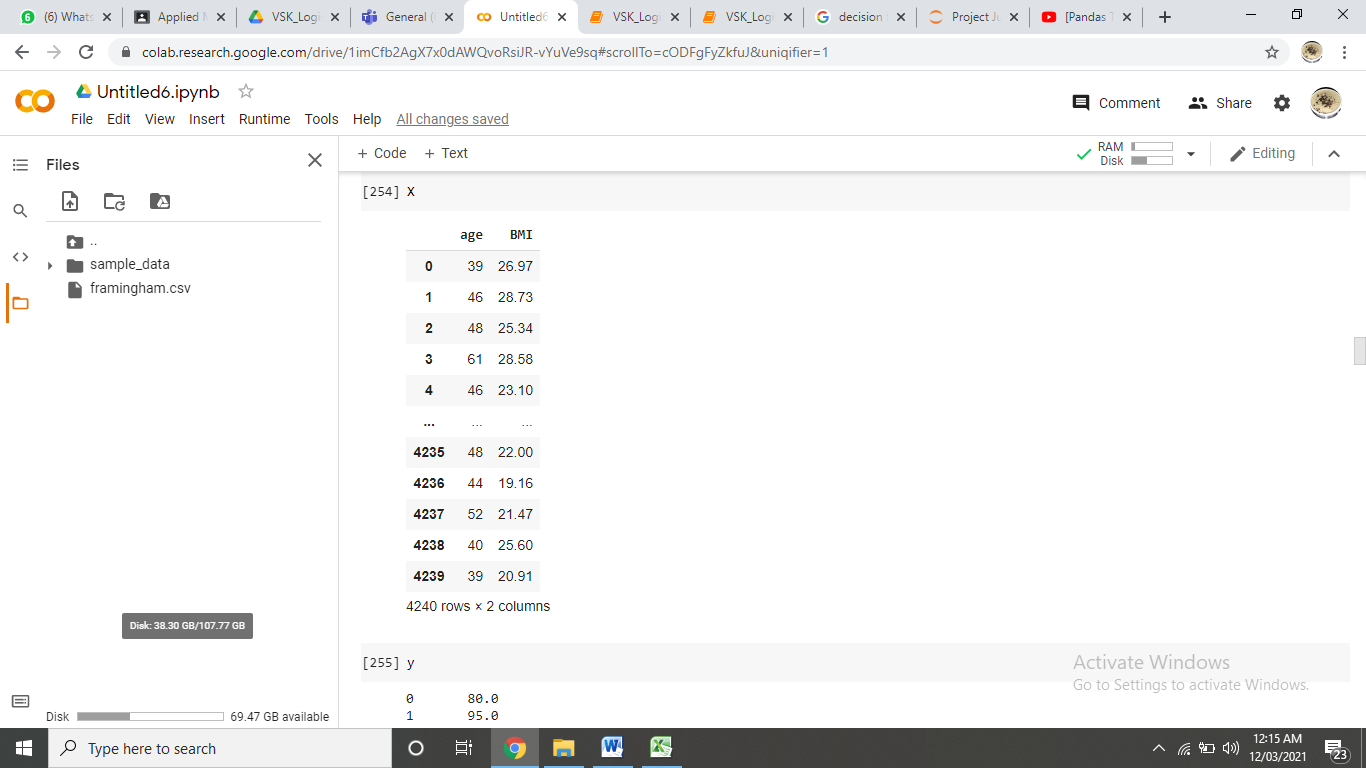


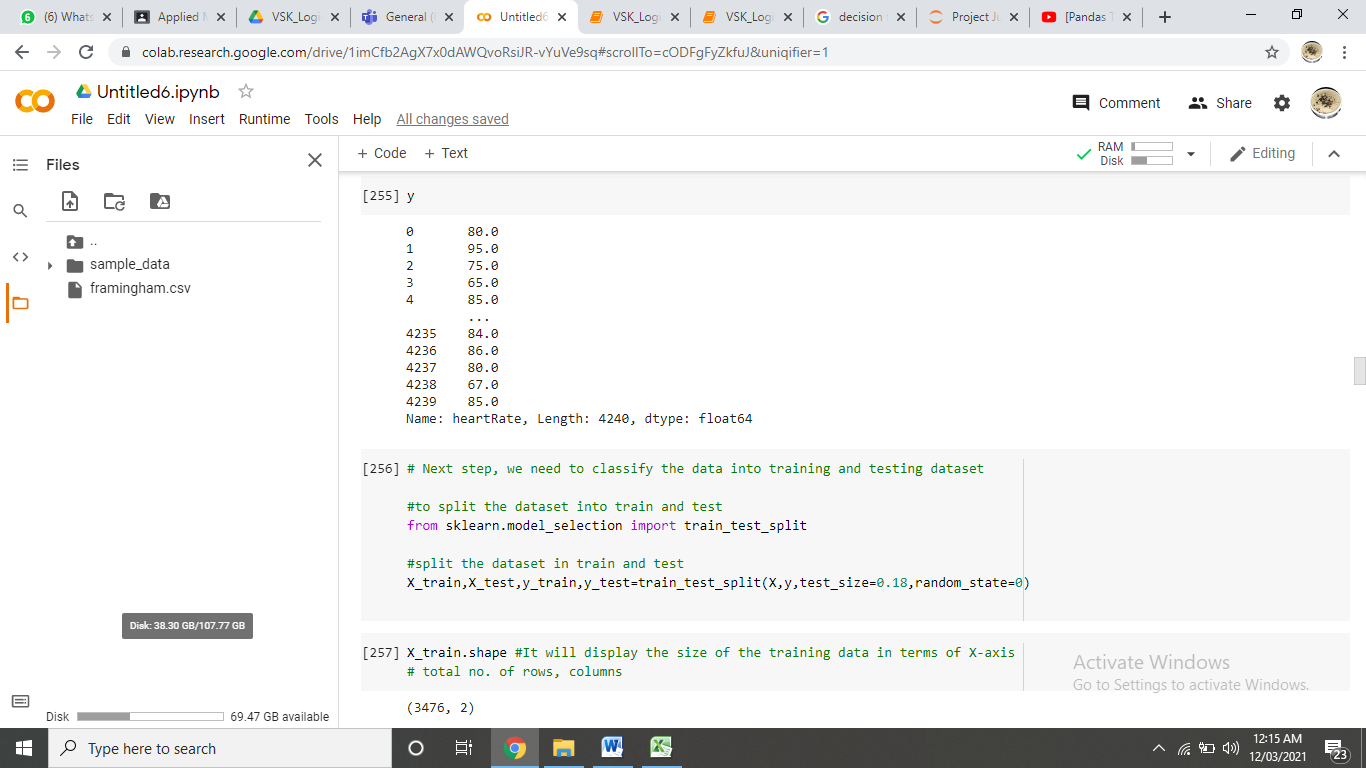


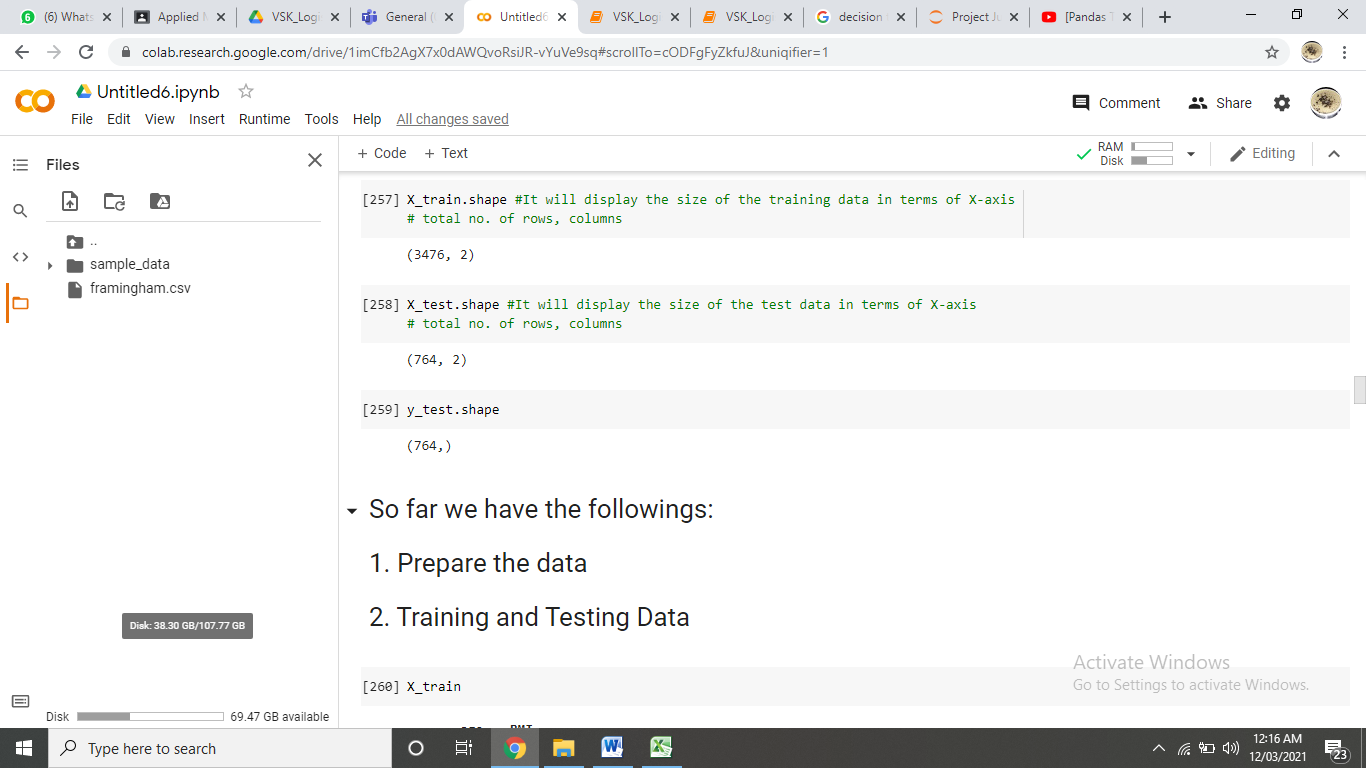


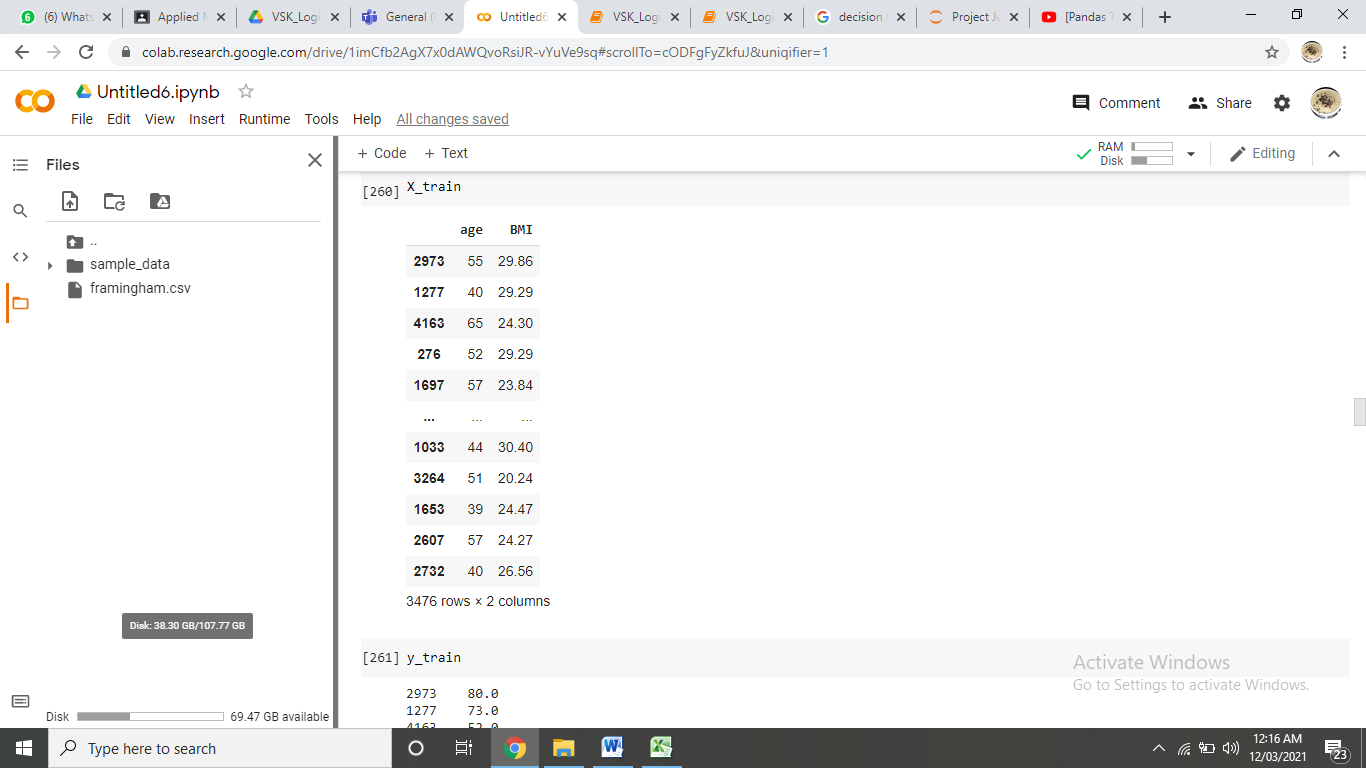


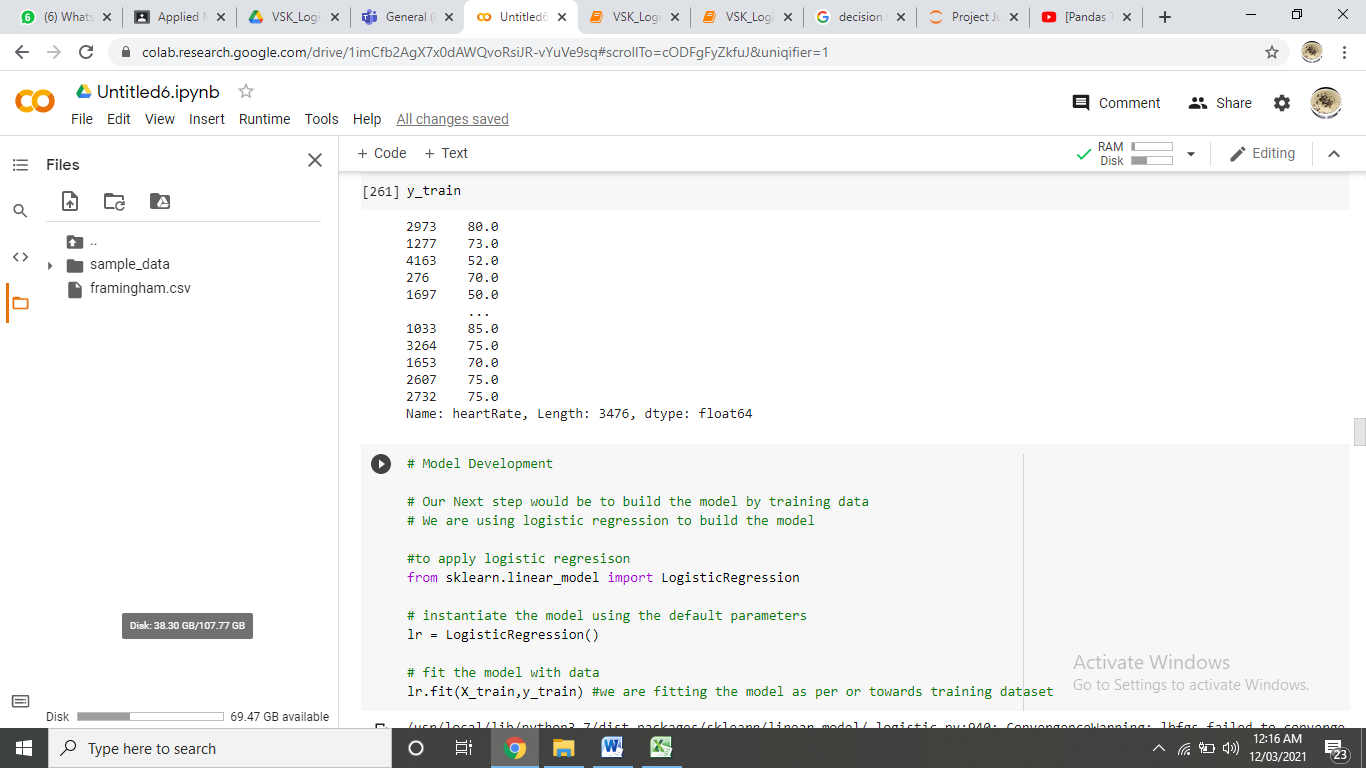


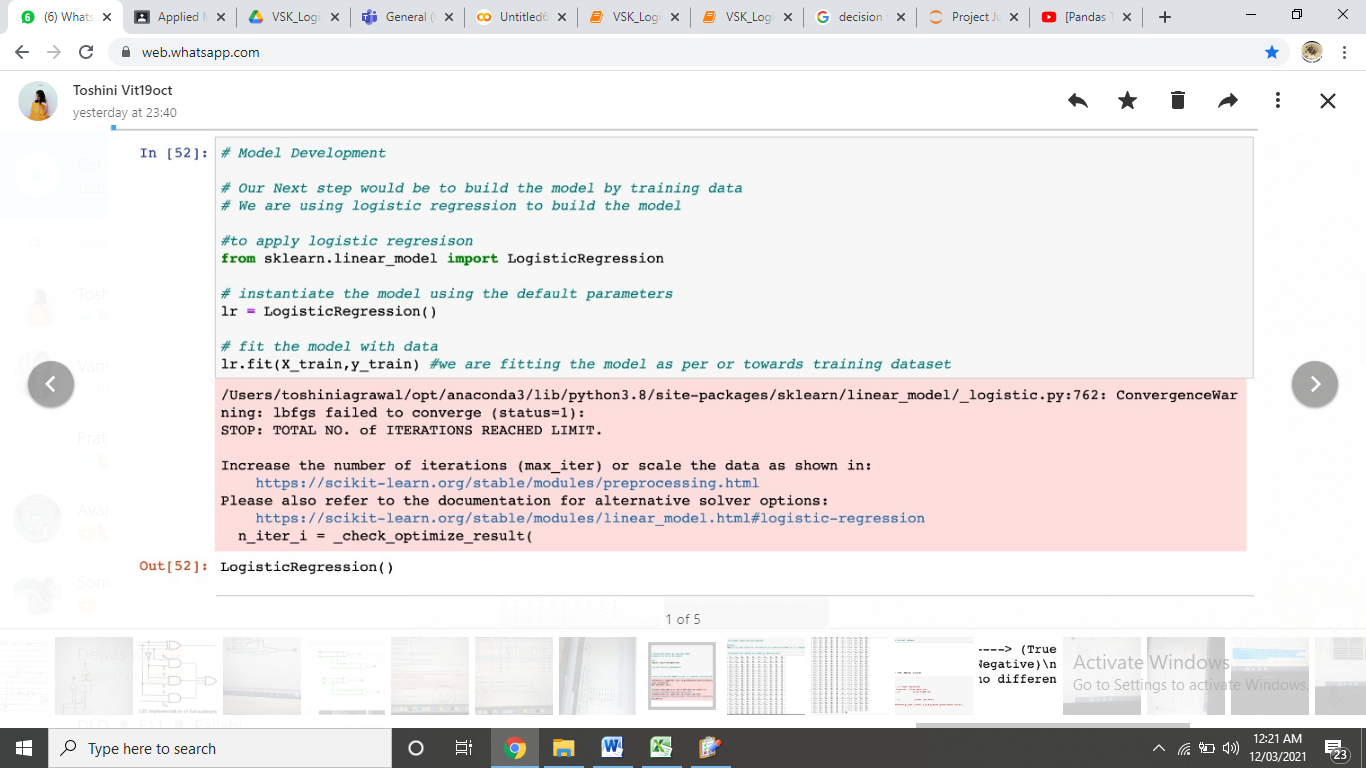


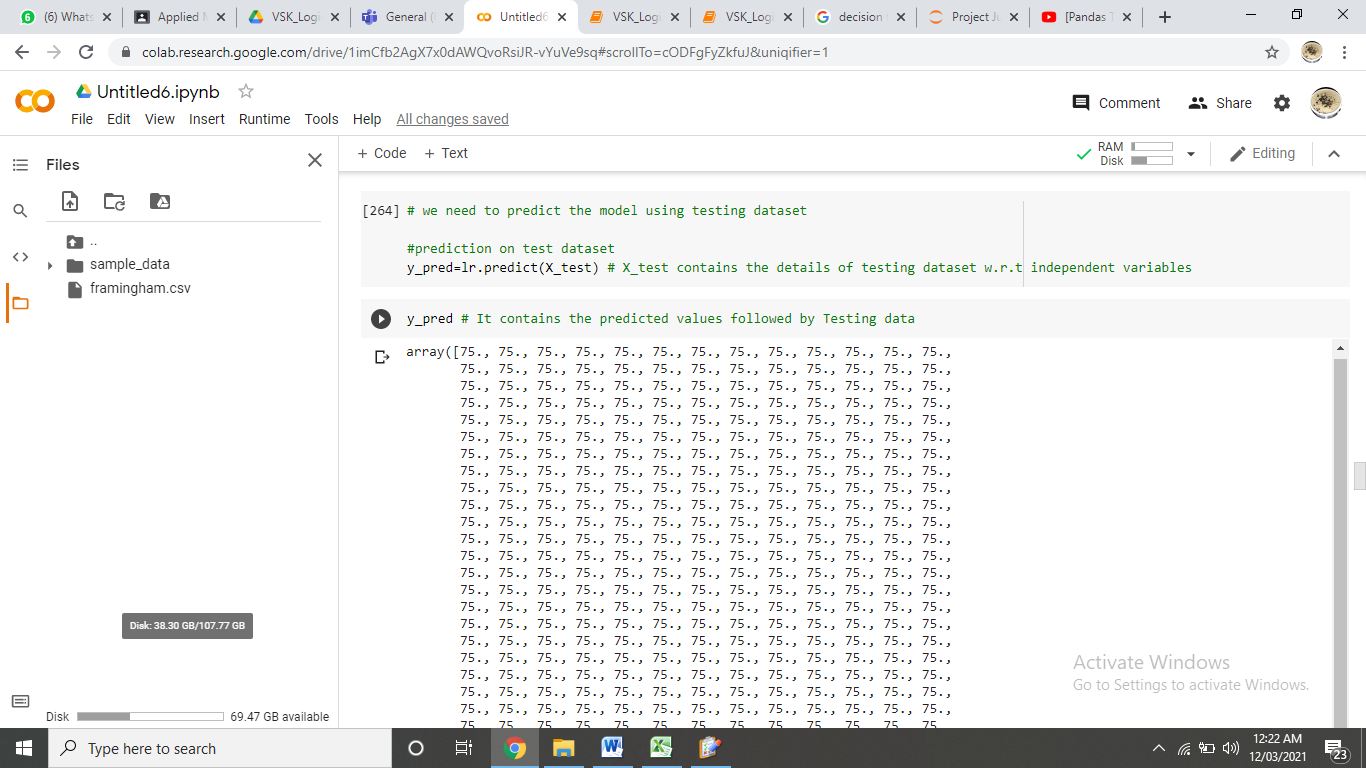


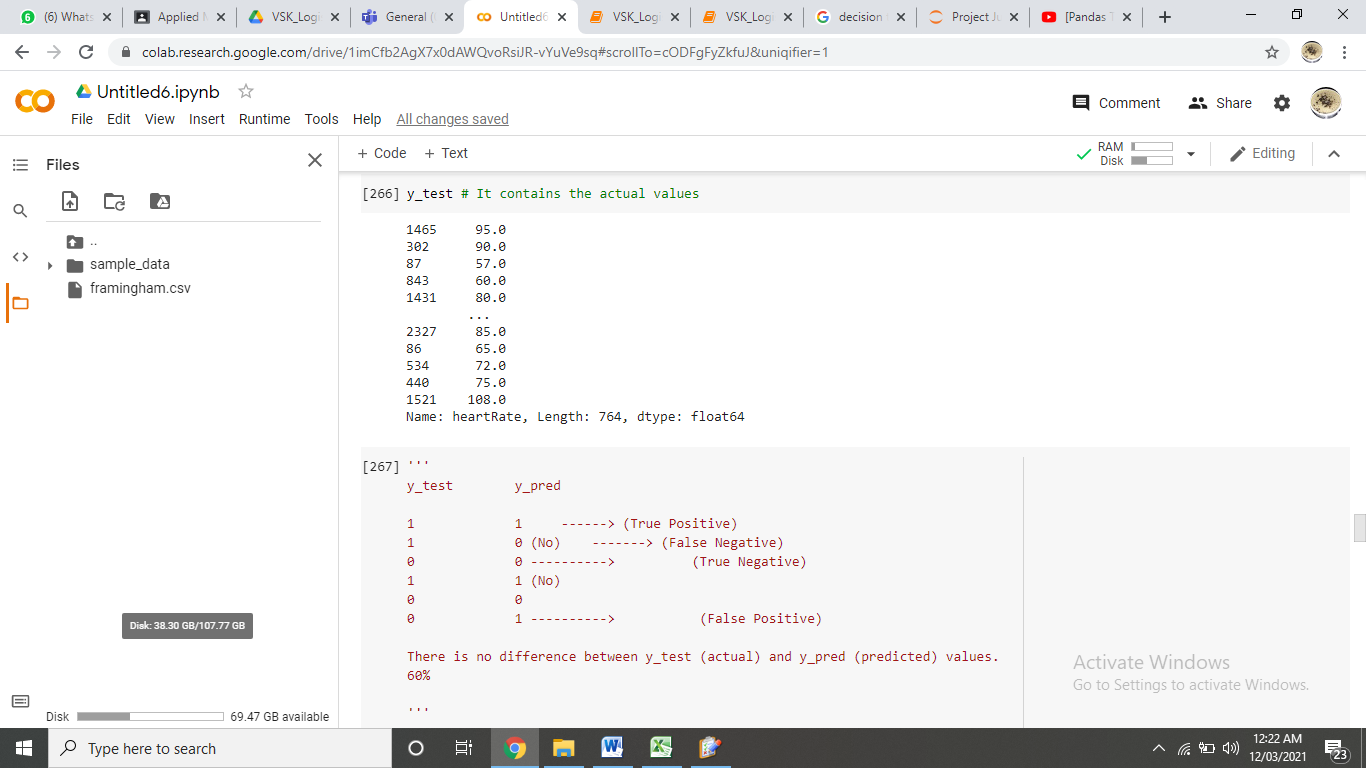




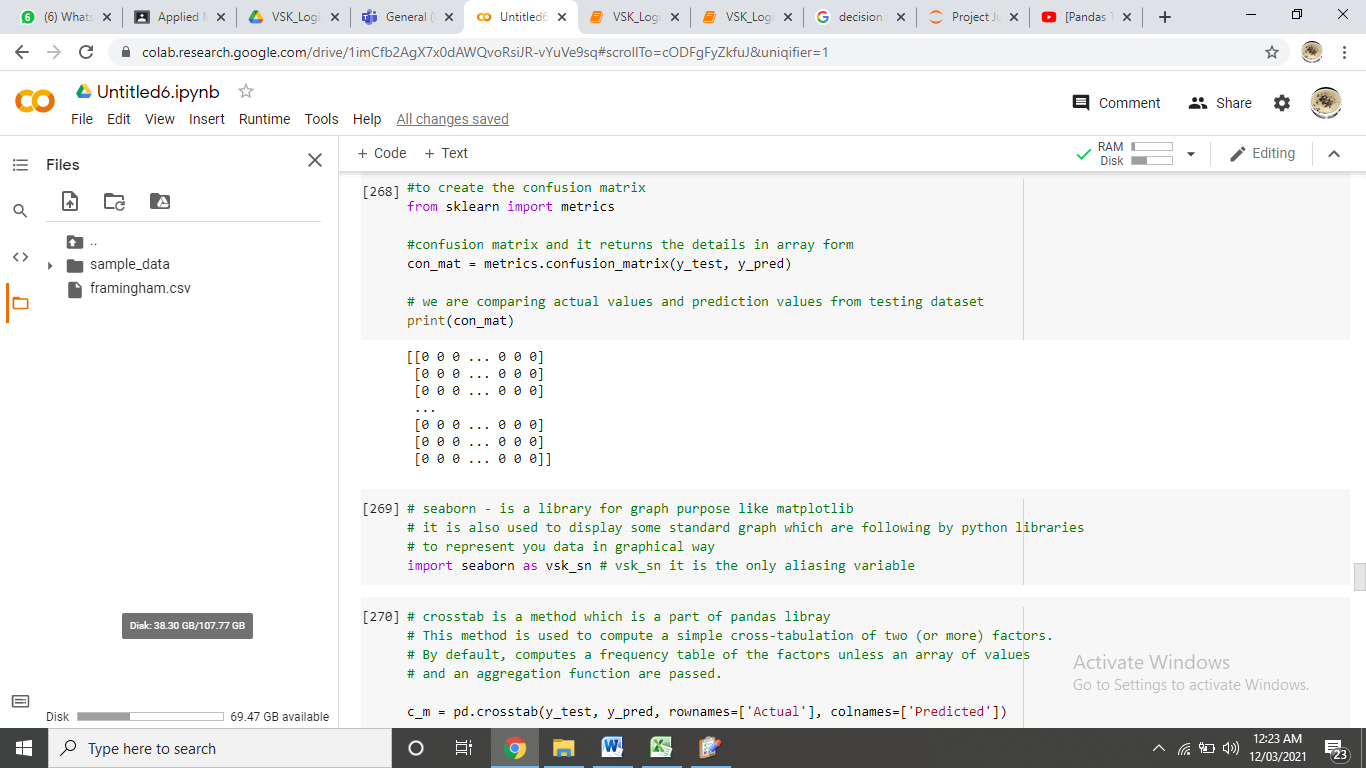


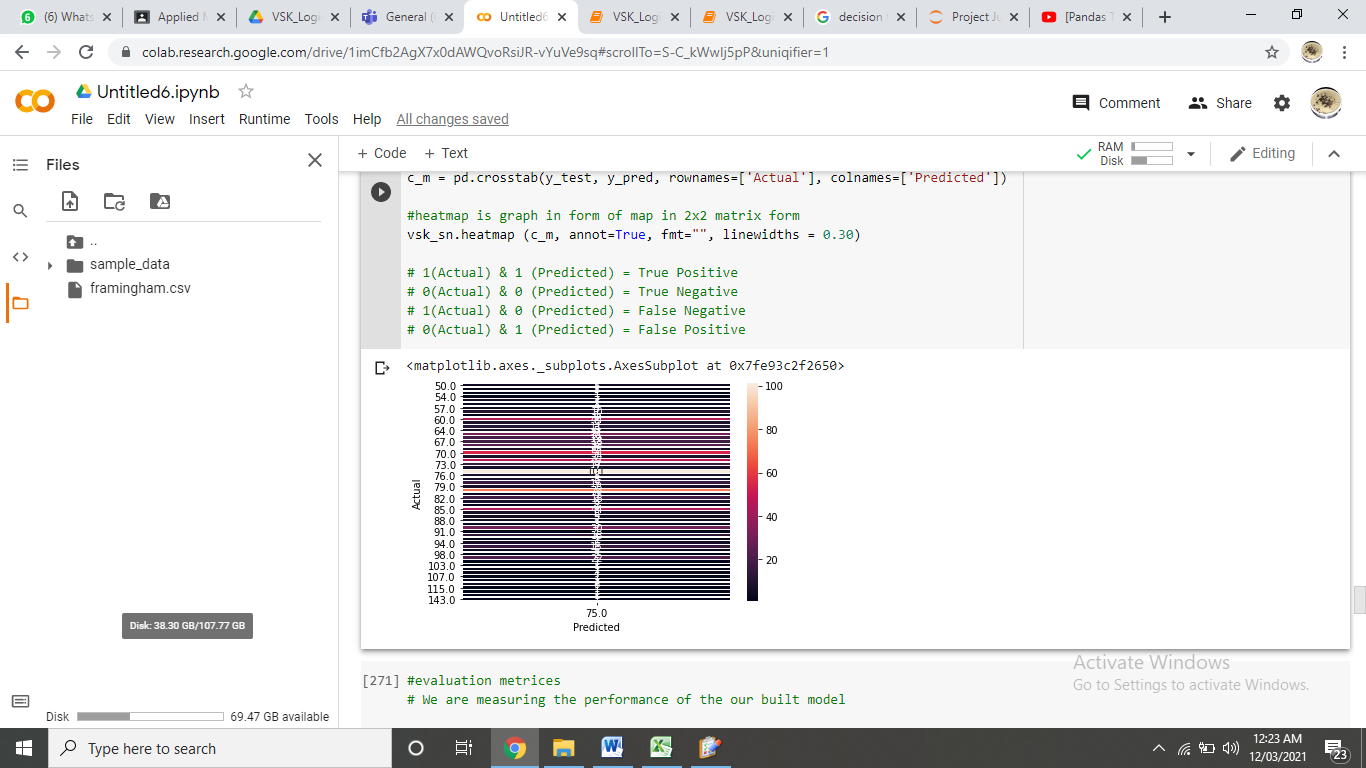


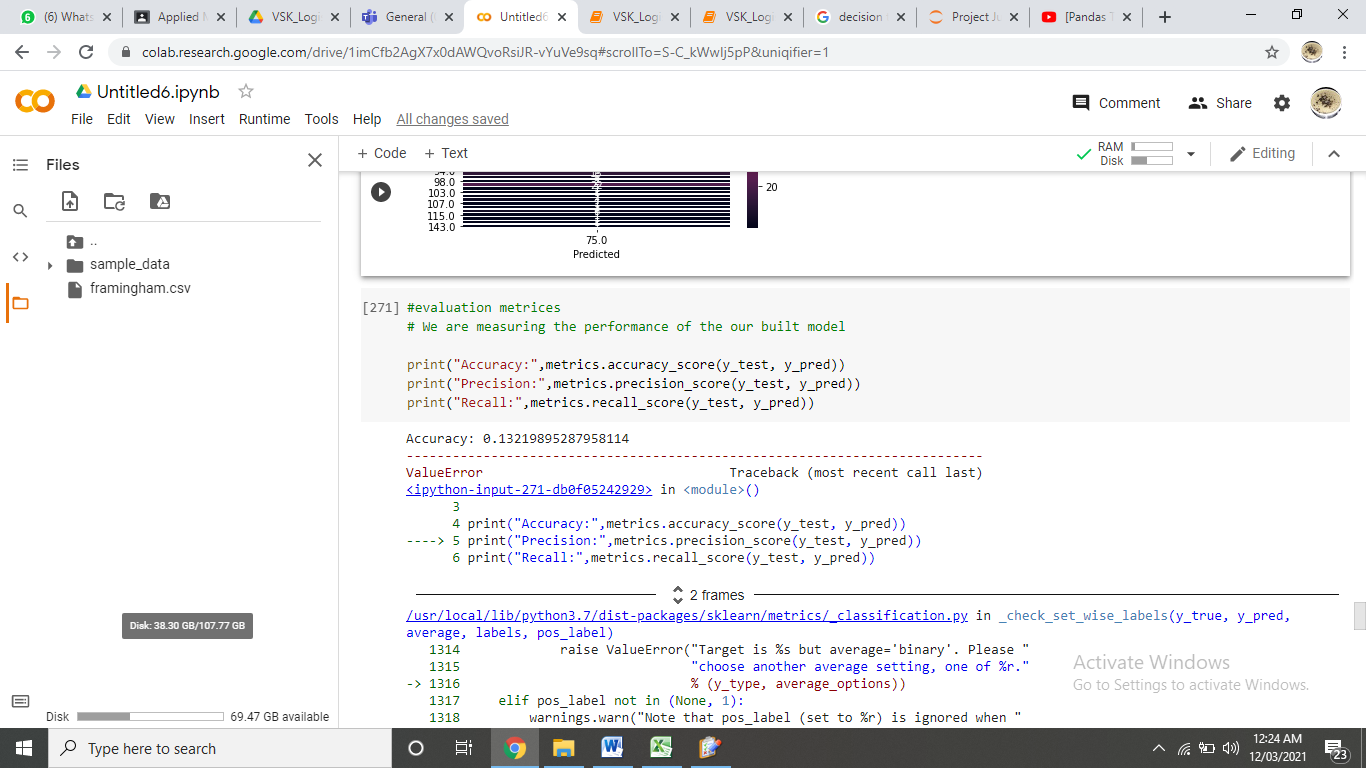


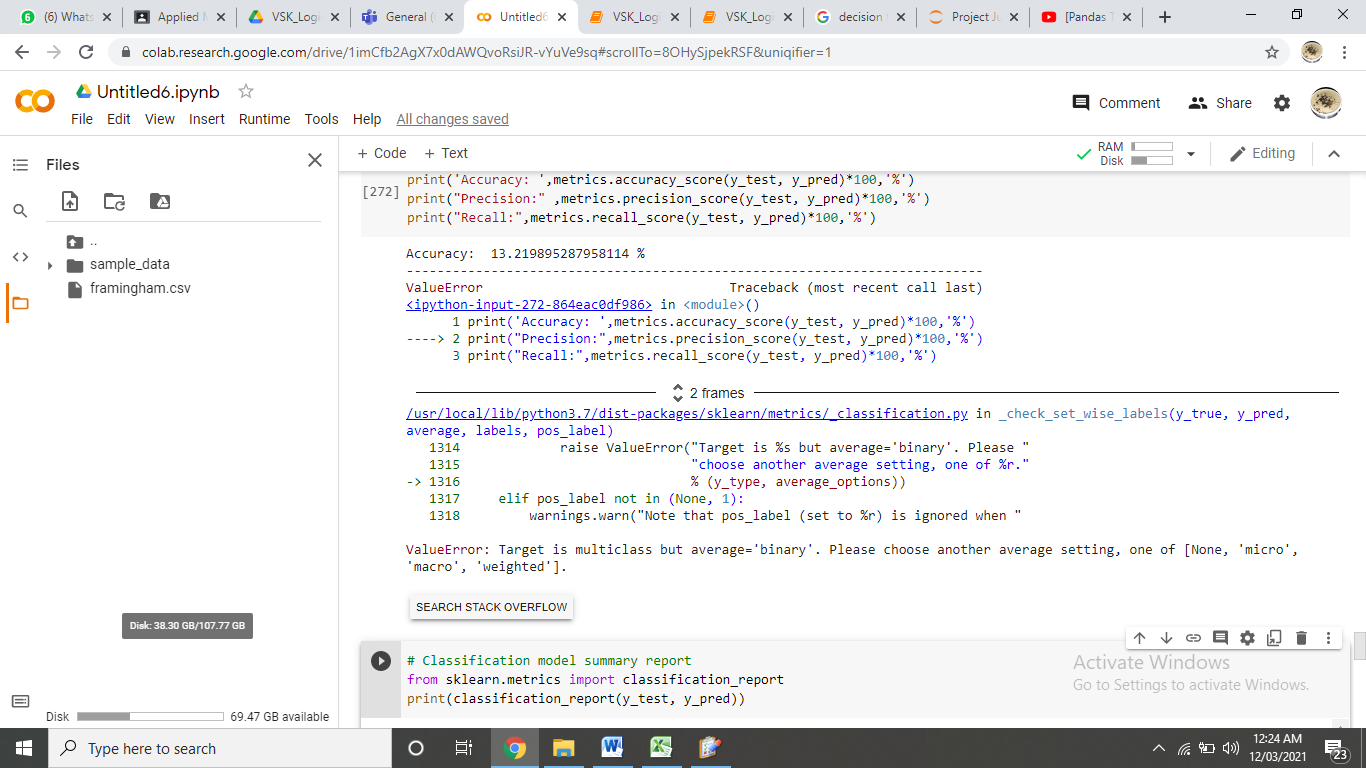
















Programs

Practical 1:

#IMPORTING LIBRARIES

import pandas as pd

import numpy as np

from sklearn.metrics import mean\_squared\_error

#READING xlsx  FILES AS PANDAS DATAFRAME

df=pd.read\_excel('country\_vaccinations.xlsx')

df.head()

#COLLECTING ONLY THE RECORDS OF INDIA

df=df.iloc[1891:1935]

As linear regression model cannot accept dates as a parameter, we take days since January 15th as our independent variable ie 2021-01-15(as our data begins from 15th January). Thus, while predicting we will be start from January 15th. Hence, X will contain values starting from 0, where 0 indicates January 15.

#PREDICTING TOTAL VACCINATIONS USING LINEAR REGRESSION MODEL WITH DATE AS INDEPENDENT VARIABLE

dataset=df

dataset=dataset.iloc[:,2:4]

dataset=dataset.dropna()

dataset.head()

#WE TAKE VACCINATION AS DEPENDENT VARIABLE(Y) AND DATE AS INDEPENDENT VARIABLE(X)

X=pd.DataFrame(np.array(range(0,43))).values

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

Y

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

X\_train,X\_test,Y\_train,Y\_test=train\_test\_split(X,Y,random\_state=1)

from sklearn.linear\_model import LinearRegression

linear=LinearRegression()

linear.fit(X\_train,Y\_train)

import matplotlib.pyplot as plt

plt.scatter(X\_train,Y\_train,color='black')

plt.plot(X\_train,linear.predict(X\_train),color='blue')

plt.show()

import matplotlib.pyplot as plt

plt.scatter(X\_test,Y\_test,color='red')

plt.plot(X\_test,linear.predict(X\_test),color='green')

plt.show()

#PRDEICTION DATA FOR MARCH 15TH

linear.predict([[59]])

print(f'accuraccy is {linear.score(X\_test,Y\_test)\*100}%')

Practical 2:

# y=m1x1+m2x2+c

    # we have two independent variables x1 and x2

    # m1 and m2 are the coef. of x1 and x2 respectively

    # c it is an intercept of the y

     #y = m1x1 + m2x2 + m3x3 + .........+ mnxn + c

# Multiple regression is like linear regression, but with more than one independent value,

# meaning that we try to predict a value based on two or more variables.

#import libraries

#Multiple regression (an extension of simple linear regression) is used to predict the value

# of a dependent variable (also known as an outcome variable) based on the value of two or more

#independent variables (also known as predictor variables).

import pandas as pd # used to represent the data in dataframe

import numpy as np # used for numerical operations

from sklearn import linear\_model #linear

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

#used to split the data into training data and testing data

# Reading the dataset

df = pd.read\_csv("/content/winequality-red.csv") #reading the dataset file

print (df)

X = df[['fixed acidity', 'volatile acidity','citric acid','residual sugar','chlorides', 'free sulfur dioxide','total sulfur dioxide','density','pH' , 'sulphates','alcohol']] #independent variables

y = df['quality'] #dependent variable

print (X)

print (y)

# all column name

df.columns

# split data into train and test

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.3, random\_state = 3)

#0.3 means 30% dataset we are taking and random

#state to choose test dta randomly after 3 runs it choose random set

#random\_state as the name suggests, is used for initializing the internal

#random number generator, which will decide the splitting of data into train and

#test indices in your case. ... If random\_state is None or np. random, then a

#randomly-initialized RandomState object is returned.

print (X\_train)

print (X\_test)

print (y\_train)

print (y\_test)

# Builiding the model

from sklearn.linear\_model import LinearRegression

# instantiate or object

model = LinearRegression() # we are creating object of the linear regression class.

# fit

model.fit(X\_train, y\_train) #Training data set #Fitting the model

# predict

y\_pred = model.predict(X\_test) # Testing data set

#predict the quality of wine  where the weight and the volume are written below:

predictedWineQuality = model.predict([[11.2, 0.1, 0.01, 14, 0.02, 60, 100, 1.003, 3, 0.546, 13]]) #Putting new value to get the prediction of CO2 emission

print(predictedWineQuality)

import sklearn.metrics as sm

print("Mean absolute error =", round(sm.mean\_absolute\_error(y\_test, y\_pred), 2))

print("Mean squared error =", round(sm.mean\_squared\_error(y\_test, y\_pred), 2))

print("R2 score =", round(sm.r2\_score(y\_test, y\_pred), 2))

# R2 score: This is pronounced as R-squared, and this score refers to the coefficient of determination.

# This tells us how well the unknown samples will be predicted by our model.

# The best possible score is 1.0, but the score can be negative as well.

Practical 3:

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.linear\_model import LinearRegression

from sklearn.preprocessing import PolynomialFeatures

from sklearn.preprocessing import LabelEncoder

df = pd.read\_csv('/content/winequality-red.csv')

df.head()

#X = df[['fixed acidity', 'volatile acidity','citric acid','residual sugar','chlorides', 'free sulfur dioxide','total sulfur dioxide','density','pH' , 'sulphates','alcohol']] #independent variables

#y = df['quality'] #dependent variable

#READING MISSING VALUES IN EACH COLUMN

df.isna().sum()

#INDEPENDENT VARIABLE

x = df.iloc[:, :-1].values

#DEPENDENT VARIABLE(QUALITY)

y = df.iloc[:, -1].values

#SPLITTING DATA INTO TRAINING AND TESTING DATASET

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

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size=0.25,random\_state = 19)

#TRAINING THE MODEL

from sklearn.linear\_model import LinearRegression

from sklearn.preprocessing import PolynomialFeatures

poly\_reg = PolynomialFeatures(degree = 2)

x\_poly = poly\_reg.fit\_transform(x\_train)

linear\_reg = LinearRegression()

linear\_reg.fit(x\_poly, y\_train)

#GETTING PREDICTIONS

y\_pred = linear\_reg.predict(poly\_reg.transform(x\_test))

#CALCULATING ACCURACY

from sklearn.metrics import r2\_score, mean\_squared\_error

print("MSE is ",mean\_squared\_error(y\_test, y\_pred))

print("R2 score is ",r2\_score(y\_test, y\_pred))

Comparision of MSE of Multiple Regression vs Polynomial Regression:

From last question, we see that MSE of Multiple regression = 0.4320483258362. While MSE of Polynomial Regression = 0.4090834281195525. Thus,we arrive at a conclusion that Polunomial Regression is better than Multiple regression as it has low MSE.

Practical 4:

import pandas as pd #Used to data in dataframe

import numpy as np

df = pd.read\_excel('/content/melb\_data.xlsx')

df

np.where(dataset.applymap(lambda Suburb: Suburb == ''))

dataset[dataset['Suburb'] == ''].index # checking index of blank dataset

dataset.isna().sum()

dataset.columns

features=['Rooms','Bathroom','Landsize','Lattitude','Longtitude']

X=dataset[features]

X.describe()

y=dataset.Price

y

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

X\_train,X\_test,y\_train,y\_test=train\_test\_split(X,y,test\_size=0.2)

X\_train

X\_test

y\_train

y\_test

from sklearn.tree import DecisionTreeRegressor

regressor = DecisionTreeRegressor()

regressor.fit(X\_train, y\_train)

y\_pred = regressor.predict(X\_test)

print ("Prediction of testing data by decision tree:", y\_pred)

y\_pred= regressor.predict([[2,1.0,202,-37.79960,144.99840]])

print("Predicted price of the house by Decision Tree: % d\n"% y\_pred)

from sklearn.metrics import accuracy\_score

accuracy= regressor.score(X\_test,y\_test)

print("Accuracy of Decision Tree for Regression Model in %:",accuracy\*100)

from sklearn import tree

import matplotlib.pyplot as plt

plt.figure(figsize=(20,20))

tree.plot\_tree(regressor, fontsize=12)

plt.show()

regressor.tree\_.impurity

Practical 5:

#import required libraries

#for data import and basic oprtaion

# The purpose of this library is to create the dataset or dataframe

import pandas as pd

# For numerical operation and other functions

import numpy as np

#for visulization and plotting

import matplotlib.pyplot as plt

import seaborn as sns

#to view the plots in the jupyter notebook inline

%matplotlib inline

#load the dataset or reading the dataset

data = pd.read\_csv(r"/content/framingham.csv") # the dataset will be read out in read only mode.

data # It display the entire dataset details

#check top 10 records

data.head(10) # it will display top 10 records

data['BMI'].fillna(data.groupby('glucose')['BMI'].transform('median'), inplace = True)

data['BPMeds'].fillna(data.groupby('totChol')['BPMeds'].transform('median'), inplace = True)

data['education'].fdillna(data.groupby('cigsPerDay')['education'].transform('median'), inplace = True)

data['glucose'].fillna(data.groupby('totChol')['glucose'].transform('median'), inplace = True)

data

data.isna().sum()

data['heartRate'].fillna(data.groupby('BMI')['heartRate'].transform('median'), inplace = True)

data['BMI'].fillna(data.groupby('cigsPerDay')['BMI'].transform('median'), inplace = True)

data['cigsPerDay'].fillna(data.groupby('totChol')['cigsPerDay'].transform('median'), inplace = True)

data['cigsPerDay'].fillna(data.groupby('education')['cigsPerDay'].transform('median'), inplace = True)

data['glucose'].fillna(data.groupby('diabetes')['glucose'].transform('median'), inplace = True)

data['totChol'].fillna(data.groupby('diaBP')['totChol'].transform('median'), inplace = True)

data['BPMeds'].fillna(data.groupby('age')['BPMeds'].transform('median'), inplace = True)

data['education'].fillna(data.groupby('sysBP')['education'].transform('median'), inplace = True)

data.isna().sum()

# we are Choosing dependent and independent variables

#split dataset in features (Independent) and target (Dependent) variable

ind\_var = ['age', 'BMI']

X = data[ind\_var] # Features or independent variables

y = data.heartRate # Target variable or dependent variable or response variable

X

y

# Next step, we need to classify the data into training and testing dataset

#to split the dataset into train and test

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

#split the dataset in train and test

X\_train,X\_test,y\_train,y\_test=train\_test\_split(X,y,test\_size=0.18,random\_state=0)

X\_train.shape #It will display the size of the training data in terms of X-axis

# total no. of rows, columns

X\_test.shape #It will display the size of the test data in terms of X-axis

# total no. of rows, columns

y\_test.shape

So far we have the followings:

1. Prepare the data

2. Training and Testing Data

X\_train

y\_train

# Model Development

# Our Next step would be to build the model by training data

# We are using logistic regression to build the model

#to apply logistic regresison

from sklearn.linear\_model import LogisticRegression

# instantiate the model using the default parameters

lr = LogisticRegression()

# fit the model with data

lr.fit(X\_train,y\_train) #we are fitting the model as per or towards training dataset

# So far we have the followings:

# 1. Prepare the data

# 2. Training and Testing Data

# 3. Build the model

# we need to predict the model using testing dataset

#prediction on test dataset

y\_pred=lr.predict(X\_test) # X\_test contains the details of testing dataset w.r.t independent variables

y\_pred # It contains the predicted values followed by Testing data

y\_test # It contains the actual values

# So far we have the followings:

# 1. Prepare the data

# 2. Training and Testing Data

# 3. Build the model

# 4. Testing the model

# 5. Next-Model Evaluation or Performance of the Model

# So far we have the followings:

# 1. Prepare the data

# 2. Training and Testing Data

# 3. Build the model

# 4. Testing the model

# 5. Next-Model Evaluation or Performance of the Model

#to create the confusion matrix

from sklearn import metrics

#confusion matrix and it returns the details in array form

con\_mat = metrics.confusion\_matrix(y\_test, y\_pred)

# we are comparing actual values and prediction values from testing dataset

print(con\_mat)

# seaborn - is a library for graph purpose like matplotlib

# it is also used to display some standard graph which are following by python libraries

# to represent you data in graphical way

import seaborn as vsk\_sn # vsk\_sn it is the only aliasing variable

# crosstab is a method which is a part of pandas libray

# This method is used to compute a simple cross-tabulation of two (or more) factors.

# By default, computes a frequency table of the factors unless an array of values

# and an aggregation function are passed.

c\_m = pd.crosstab(y\_test, y\_pred, rownames=['Actual'], colnames=['Predicted'])

#heatmap is graph in form of map in 2x2 matrix form

vsk\_sn.heatmap (c\_m, annot=True, fmt="", linewidths = 0.30)

# 1(Actual) & 1 (Predicted) = True Positive

# 0(Actual) & 0 (Predicted) = True Negative

# 1(Actual) & 0 (Predicted) = False Negative

# 0(Actual) & 1 (Predicted) = False Positive

#evaluation metrices

# We are measuring the performance of the our built model

print("Accuracy:",metrics.accuracy\_score(y\_test, y\_pred))

print("Precision:",metrics.precision\_score(y\_test, y\_pred))

print("Recall:",metrics.recall\_score(y\_test, y\_pred))

print('Accuracy: ',metrics.accuracy\_score(y\_test, y\_pred)\*100,'%')

print("Precision:" ,metrics.precision\_score(y\_test, y\_pred)\*100,'%')

print("Recall:",metrics.recall\_score(y\_test, y\_pred)\*100,'%')

# Classification model summary report

from sklearn.metrics import classification\_report

print(classification\_report(y\_test, y\_pred))

#To predict heartRate

y\_pred\_new=lr.predict([[90,35]])

y\_pred\_new