**Experiment 5**

**Student Name:** Sahil Kaundal **UID:** 21BCS8197

**Branch:** BE CSE (Lateral Entry) **Section/Group:** 616/A

**Semester:** 5th **Date of Performance:** 16/10/2022

**Subject Name:** ML Lab **Subject Code:** 20CSP-317

1. **Aim/Overview of the practical:**

Implement Naive Bayes on any Dataset.

1. **Task To Be Done:**

Implement Naïve Bayes on any data set using sklearn.

**3. Apparatus / Simulator Used:**

1. Windows 7 or above.
2. Google Collab.

**Naive Bayes**

[Bayes’ Theorem](https://machinelearningmastery.com/bayes-theorem-for-machine-learning/) provides a way that we can calculate the probability of a piece of data belonging to a given class, given our prior knowledge. Bayes’ Theorem is stated as:

* P(class|data) = (P(data|class) \* P(class)) / P(data)

Where P(class|data) is the probability of class given the provided data.

For an in-depth introduction to Bayes Theorem, see the tutorial:

* [A Gentle Introduction to Bayes Theorem for Machine Learning](https://machinelearningmastery.com/bayes-theorem-for-machine-learning/)

Naive Bayes is a classification algorithm for binary (two-class) and multiclass classification problems. It is called Naive Bayes or idiot Bayes because the calculations of the probabilities for each class are simplified to make their calculations tractable.

Rather than attempting to calculate the probabilities of each attribute value, they are assumed to be conditionally independent given the class value.

This is a very strong assumption that is most unlikely in real data, i.e. that the attributes do not interact. Nevertheless, the approach performs surprisingly well on data where this assumption does not hold.

**4. Program / Commands:**

#Sahil Kaundal

#21BCS8197

from google.colab import drive

drive.mount('/content/drive')

# importing the libraries

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

import seaborn as sns

# importing the dataset

dataset = pd.read\_csv('/content/drive/MyDrive/Data/NaiveBayes.csv')

# split the data into inputs and outputs

X = dataset.iloc[:, [0,1]].values

y = dataset.iloc[:, 2].values

# training and testing data

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

# assign test data size 25%

X\_train, X\_test, y\_train, y\_test =train\_test\_split(X,y,test\_size= 0.25, random\_state=0)

# importing standard scaler

from sklearn.preprocessing import StandardScaler

# scalling the input data

sc\_X = StandardScaler()

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

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

# importing classifier

from sklearn.naive\_bayes import BernoulliNB

# initializaing the NB

classifer = BernoulliNB()

# training the model

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

# testing the model

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

# importing accuracy score

from sklearn.metrics import accuracy\_score

# printing the accuracy of the model

print(accuracy\_score(y\_pred, y\_test))

# import Gaussian Naive Bayes classifier

from sklearn.naive\_bayes import GaussianNB

# create a Gaussian Classifier

classifer1 = GaussianNB()

# training the model

classifer1.fit(X\_train, y\_train)

# testing the model

y\_pred1 = classifer1.predict(X\_test)

# importing accuracy score

from sklearn.metrics import accuracy\_score

# printing the accuracy of the model

print(accuracy\_score(y\_test,y\_pred1))

# importing the required modules

import seaborn as sns

from sklearn.metrics import confusion\_matrix

# passing actual and predicted values

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

# true write data values in each cell of the matrix

sns.heatmap(cm, annot=True)

plt.savefig('confusion.png')

# importing classification report

from sklearn.metrics import classification\_report

# printing the report

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

# importing the required modules

import seaborn as sns

from sklearn.metrics import confusion\_matrix

# passing actual and predicted values

cm = confusion\_matrix(y\_test, y\_pred1)

# true write data values in each cell of the matrix

sns.heatmap(cm,annot=True)

plt.savefig('confusion.png')

# importing classification report

from sklearn.metrics import classification\_report

# printing the report

print(classification\_report(y\_test, y\_pred1))

# assigning features and label variables

weather = ['Sunny','Sunny','Overcast','Rainy','Rainy','Rainy','Overcast','Sunny','Sunny', 'Rainy','Sunny','Overcast','Overcast','Rainy']

# output class

play = ['No','No','Yes','Yes','Yes','No','Yes','No','Yes','Yes','Yes','Yes','Yes','No']

# Import LabelEncoder

from sklearn import preprocessing

# creating LabelEncoder

labelCode = preprocessing.LabelEncoder()

# Converting string labels into numbers.

wheather\_encoded=labelCode.fit\_transform(weather)

print(wheather\_encoded)

# import LabelEncoder

from sklearn import preprocessing

# creating LabelEncoder

labelCode = preprocessing.LabelEncoder()

# converting string labels into numbers.

label=labelCode.fit\_transform(play)

# import Gaussian Naive Bayes model

from sklearn.naive\_bayes import GaussianNB

# create a Gaussian Classifier

model = GaussianNB()

# train the model using the training sets

model.fit(wheather\_encoded, label)

# importing numpy module

import numpy as np

# converting 1D array to 2D

weather\_2d = np.reshape(wheather\_encoded, (-1, 1))

# import Gaussian Naive Bayes model

from sklearn.naive\_bayes import GaussianNB

# create a Gaussian Classifier

model = GaussianNB()

# train the model using the training sets

model.fit(weather\_2d, label)

# predicting the odel

predicted= model.predict([[0]]) # 0:Overcast

# printing predicted value

print(predicted)

# import scikit-learn dataset library

from sklearn import datasets

# load dataset

dataset = datasets.load\_wine()

# print the names of the 13 features

print ("Inputs: ", dataset.feature\_names)

# print the label type of wine

print ("Outputs: ", dataset.target\_names)

# print the wine data features

print(dataset.data[0:3])

# print the wine labels

print(dataset.target)

# import train\_test\_split function

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

# input and outputs

inputs = dataset.data

outputs = dataset.target

# split dataset into training set and test set

X\_train, X\_test, y\_train, y\_test = train\_test\_split(inputs, outputs, test\_size=0.3, random\_state=1)

# import Gaussian Naive Bayes model

from sklearn.naive\_bayes import GaussianNB

# create a Gaussian Classifier

classifer = GaussianNB()

# train the model using the training sets

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

# predict the response for test dataset

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

# import scikit-learn metrics module for accuracy calculation

from sklearn import metrics

# printing accuracy

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

# importing the required modules

import seaborn as sns

from sklearn.metrics import confusion\_matrix

# passing actual and predicted values

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

# true Write data values in each cell of the matrix

sns.heatmap(cm, annot=True)

plt.savefig('confusion.png')

# Importing classification report

from sklearn.metrics import classification\_report

# printing the report

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

# importring modules

import matplotlib.pyplot as plt

import pandas as pd

# importing the dataset

dataset = pd.read\_csv('/content/drive/MyDrive/Data/NaiveBayes.csv')

# split the data into inputs and outputs

X = dataset.iloc[:, [0,1]].values

y = dataset.iloc[:, 2].values

# training and testing data

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

# assign test data size 25%

X\_train, X\_test, y\_train, y\_test =train\_test\_split(X, y, test\_size=0.25, random\_state=0)

# importing StandardScaler

from sklearn.preprocessing import StandardScaler

# scalling the input data

sc\_X = StandardScaler()

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

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

# importing bernoulli NB

from sklearn.naive\_bayes import BernoulliNB

# initializaing the NB

classifer=BernoulliNB()

# training the model

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

# testing the model

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

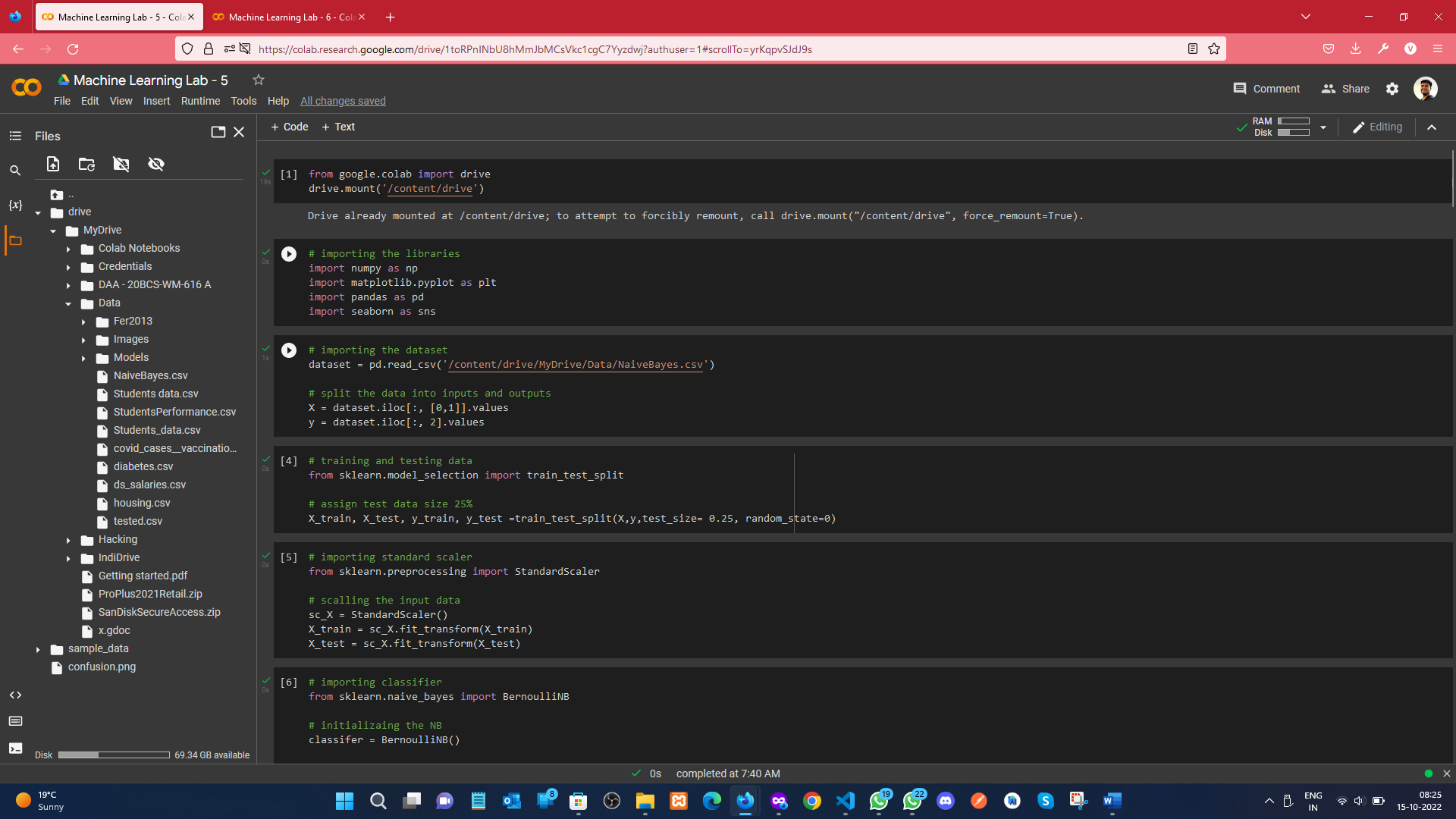
# importing accuracy score

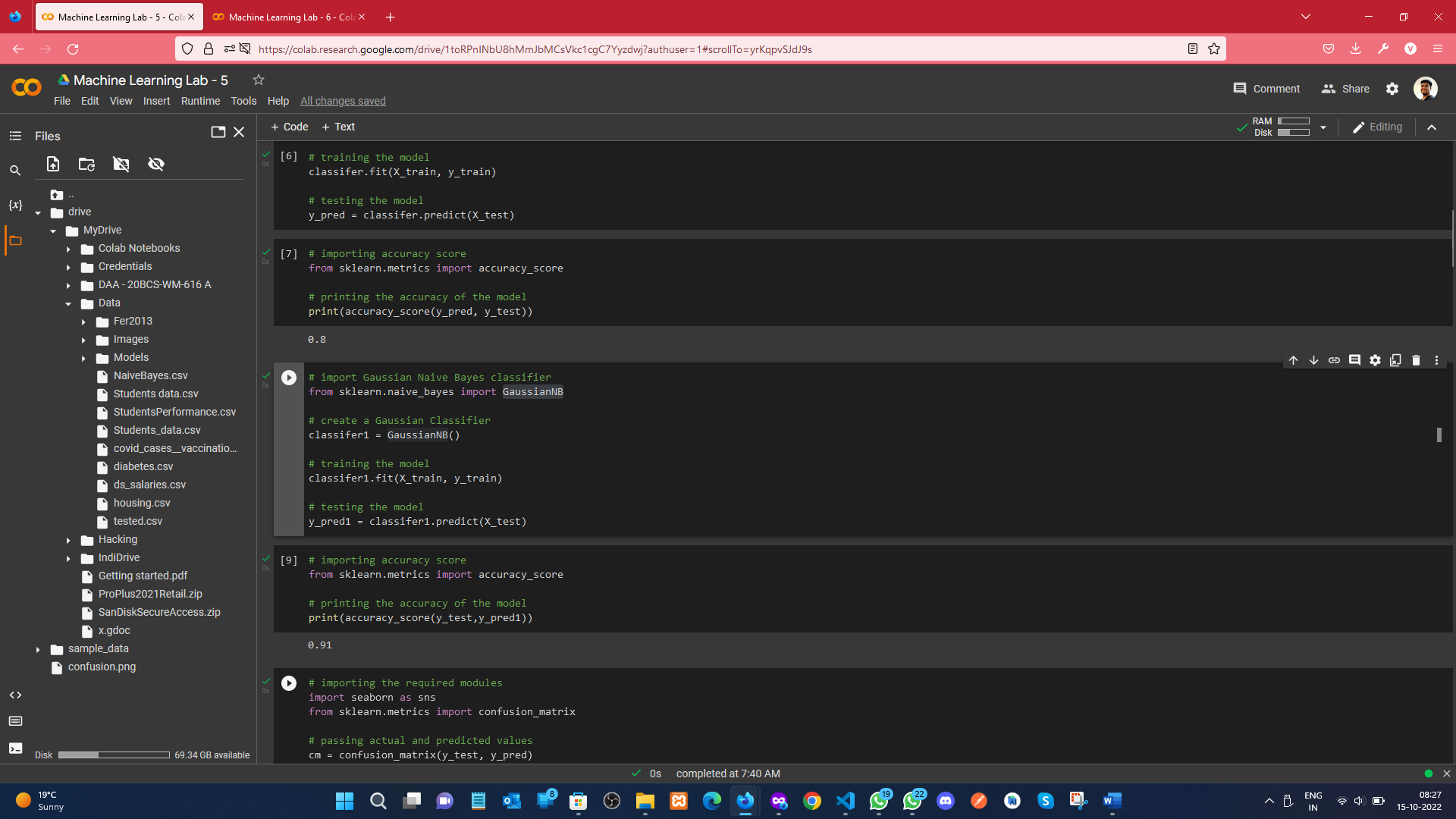
from sklearn.metrics import accuracy\_score

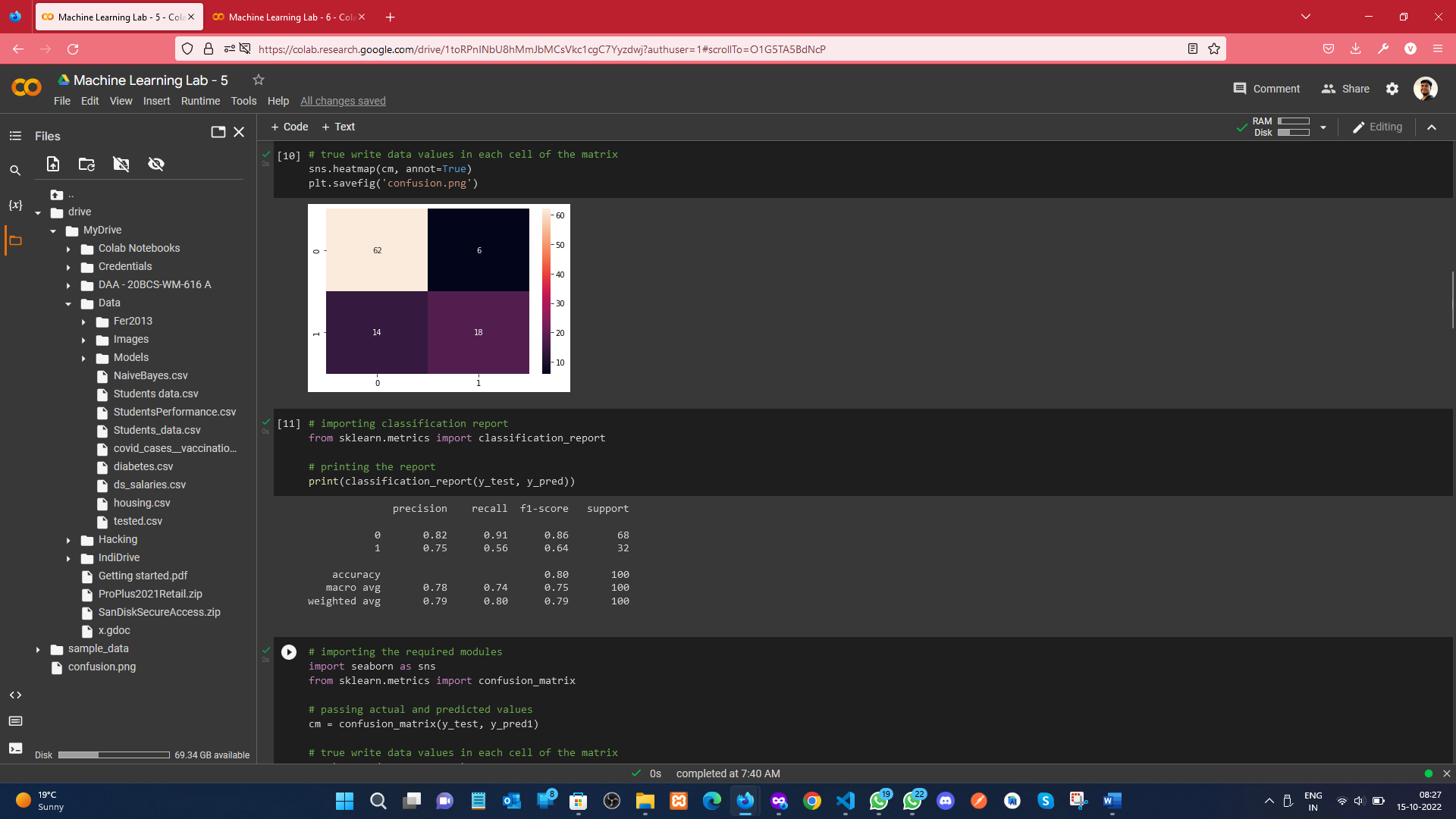
# printing the accuracy of the model

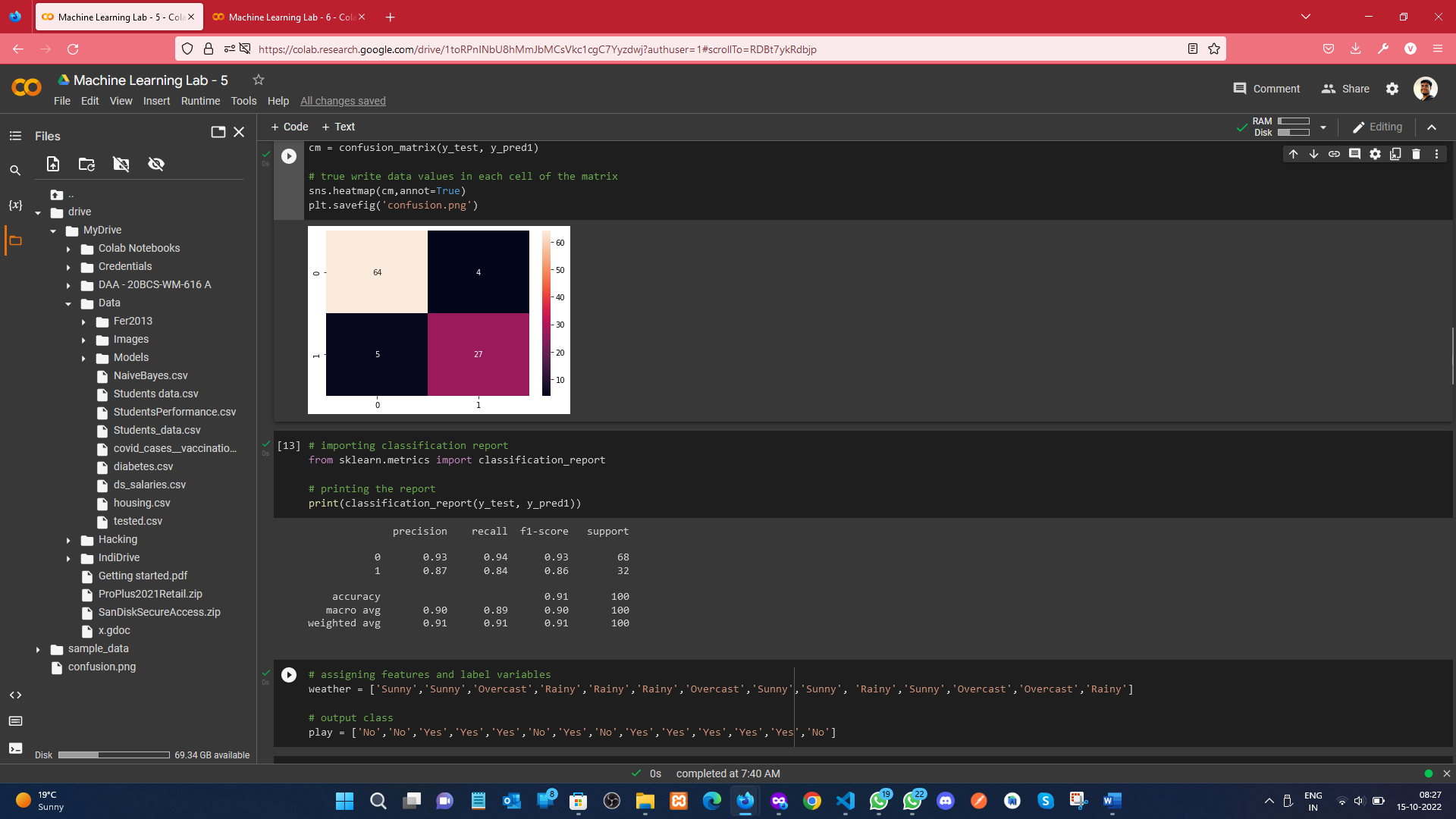
print(accuracy\_score(y\_test, y\_pred))

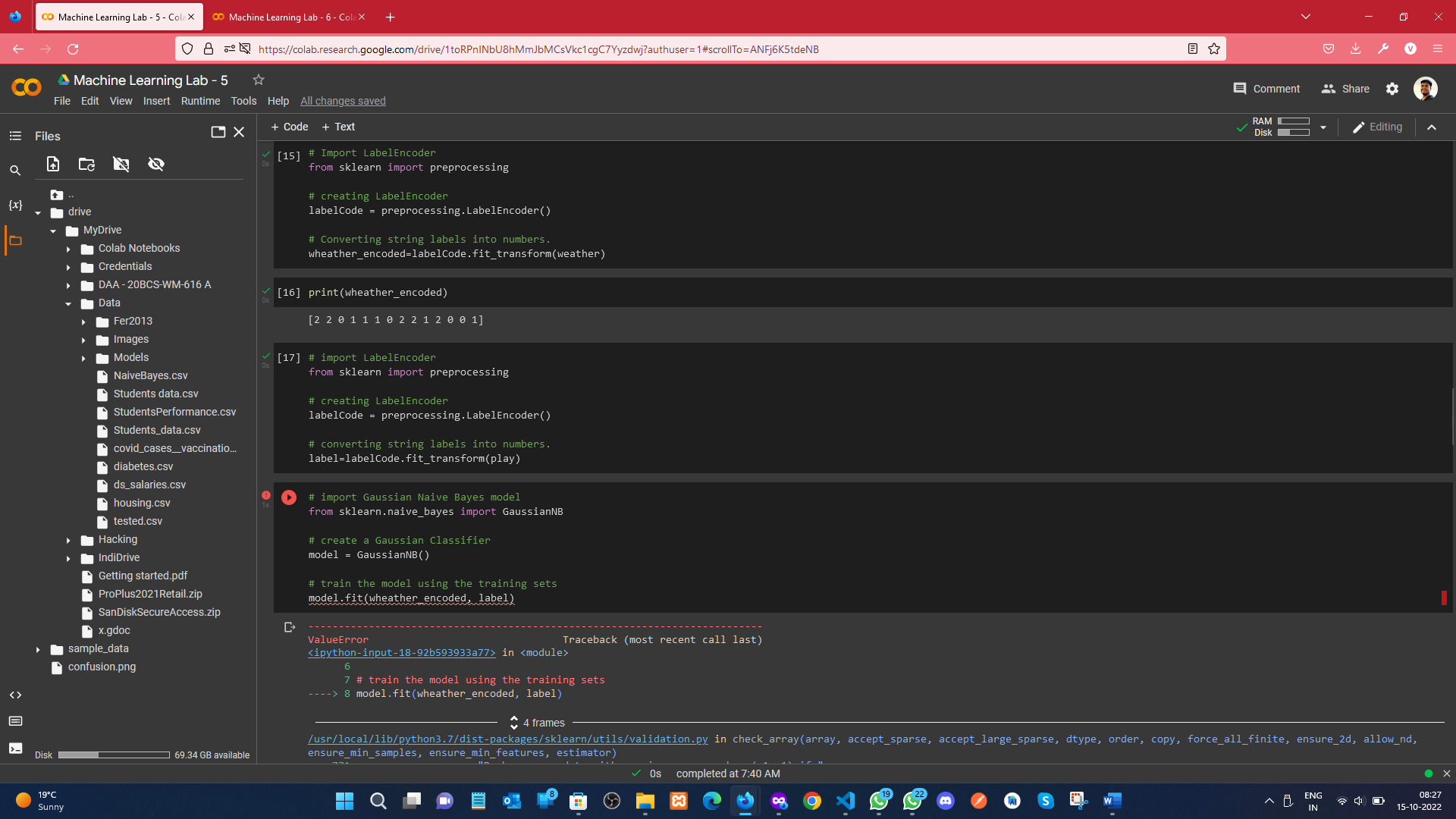
1. **Result/Output/Writing Summary:**

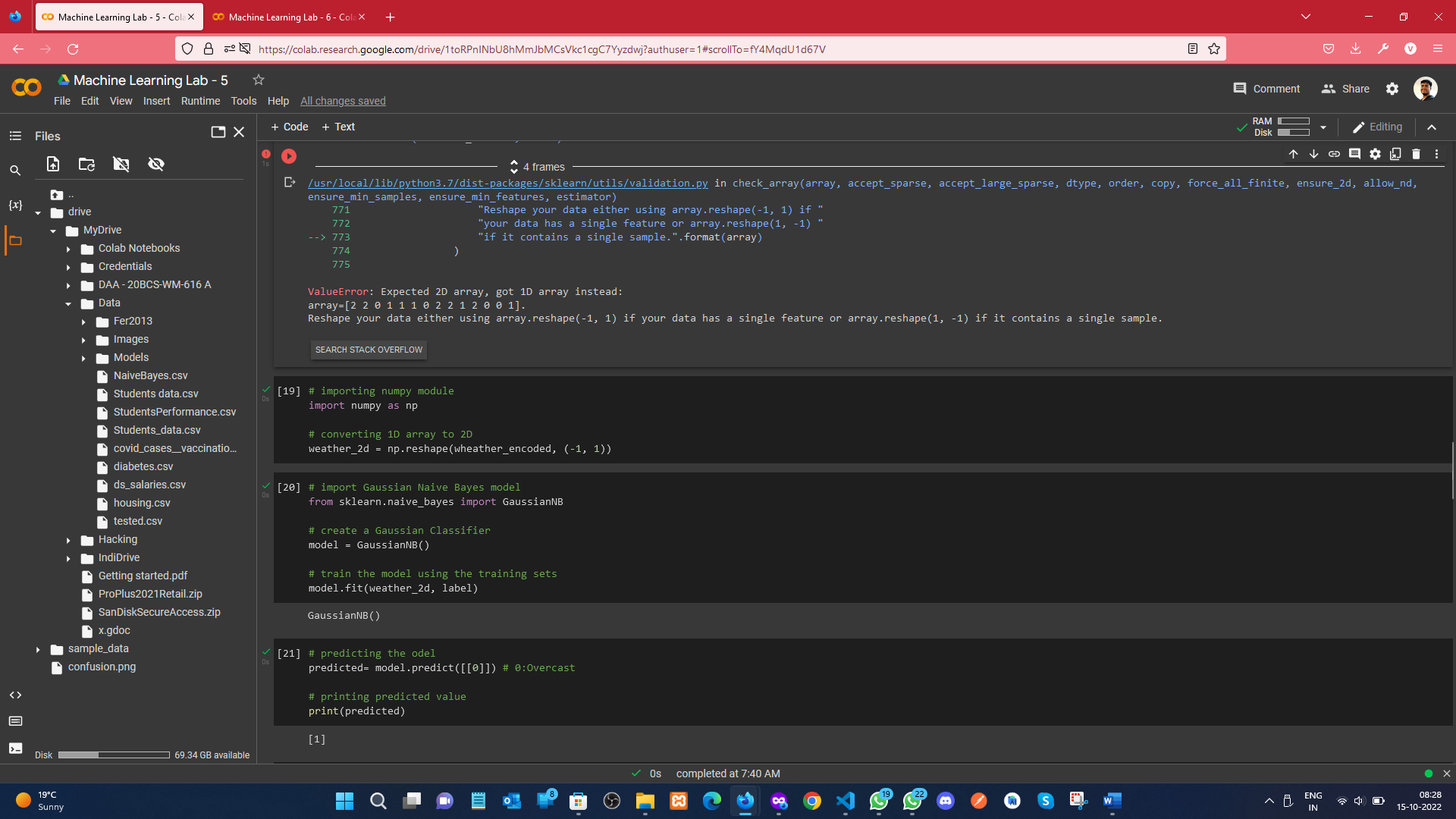


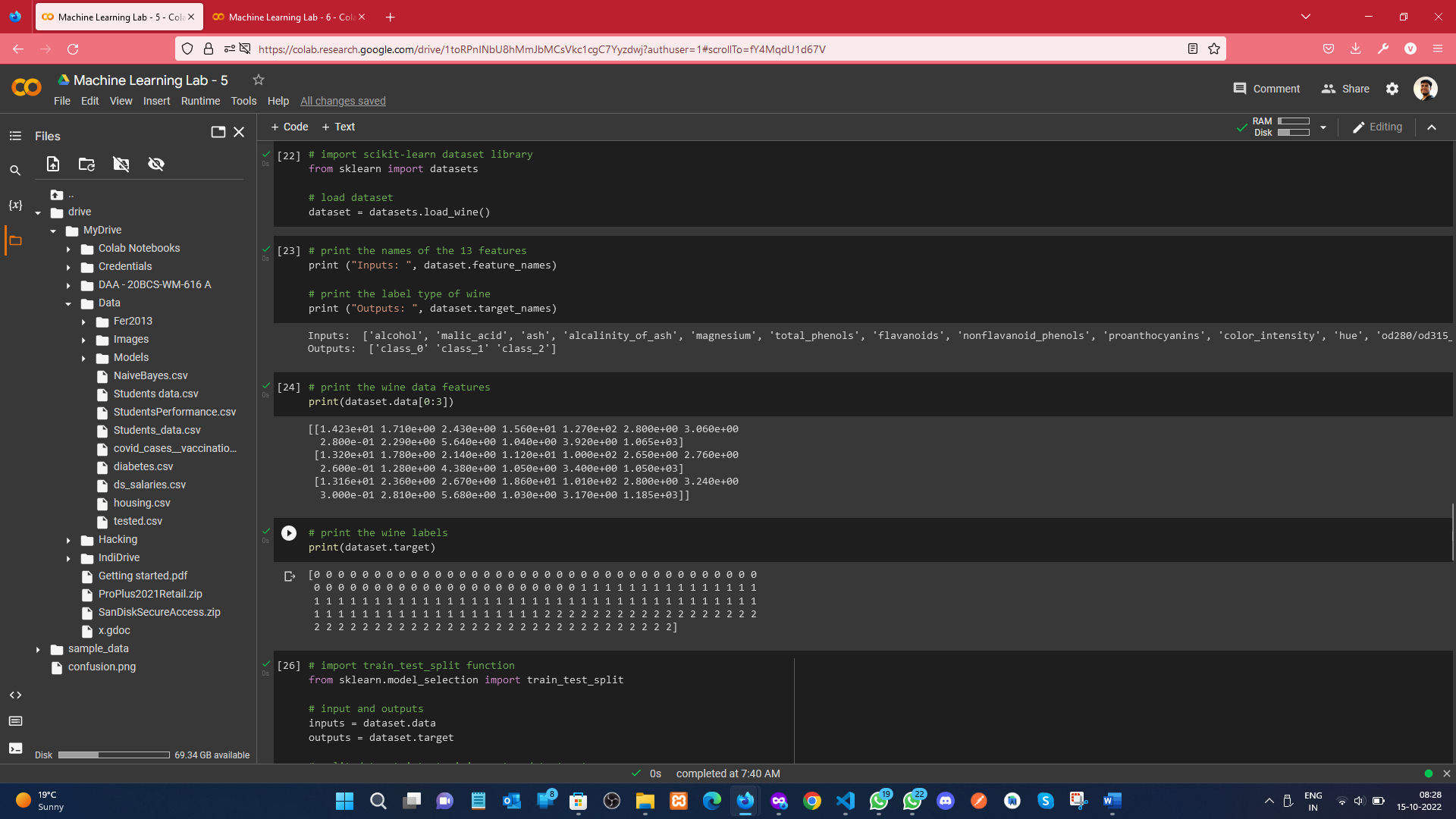


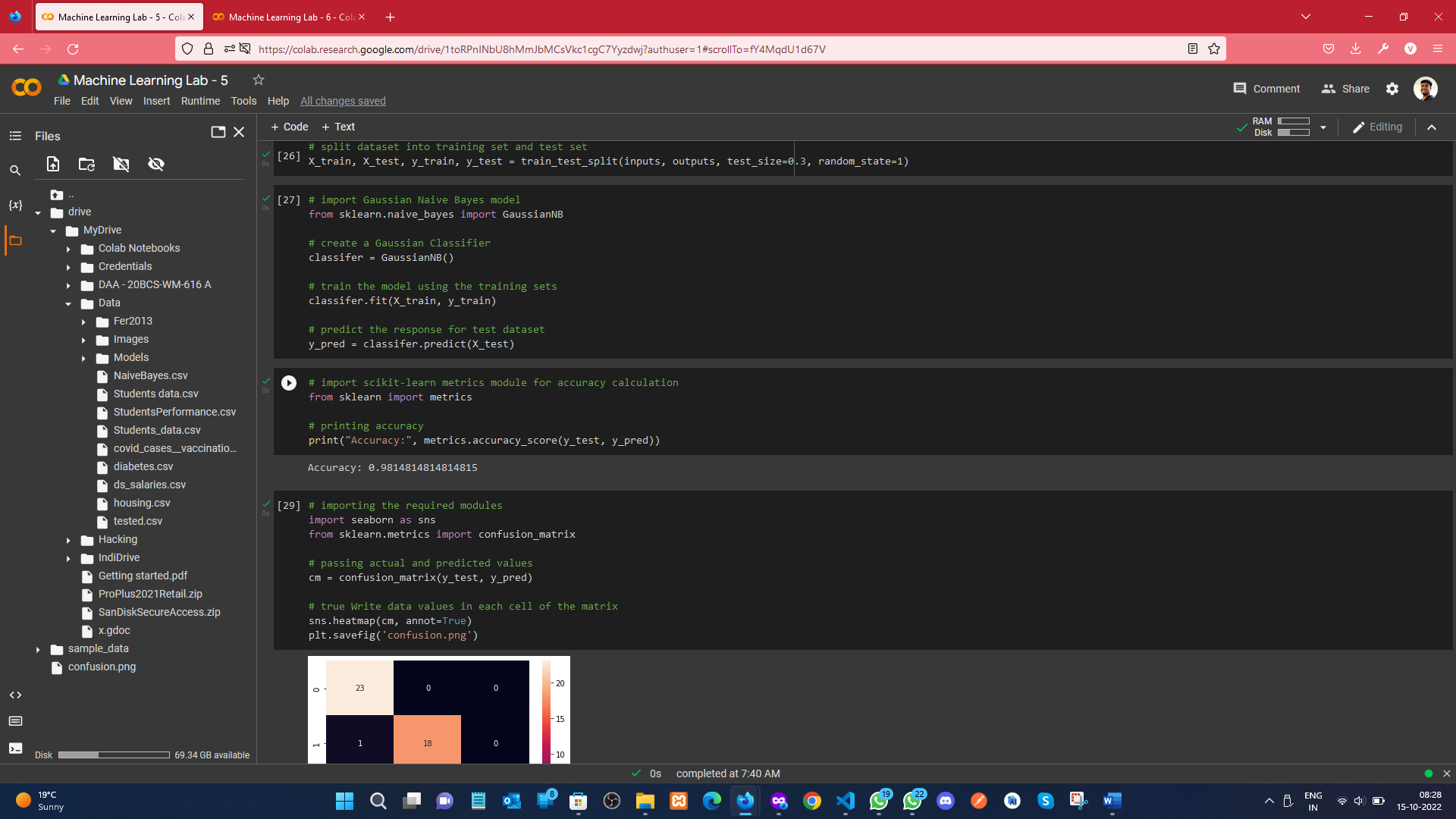


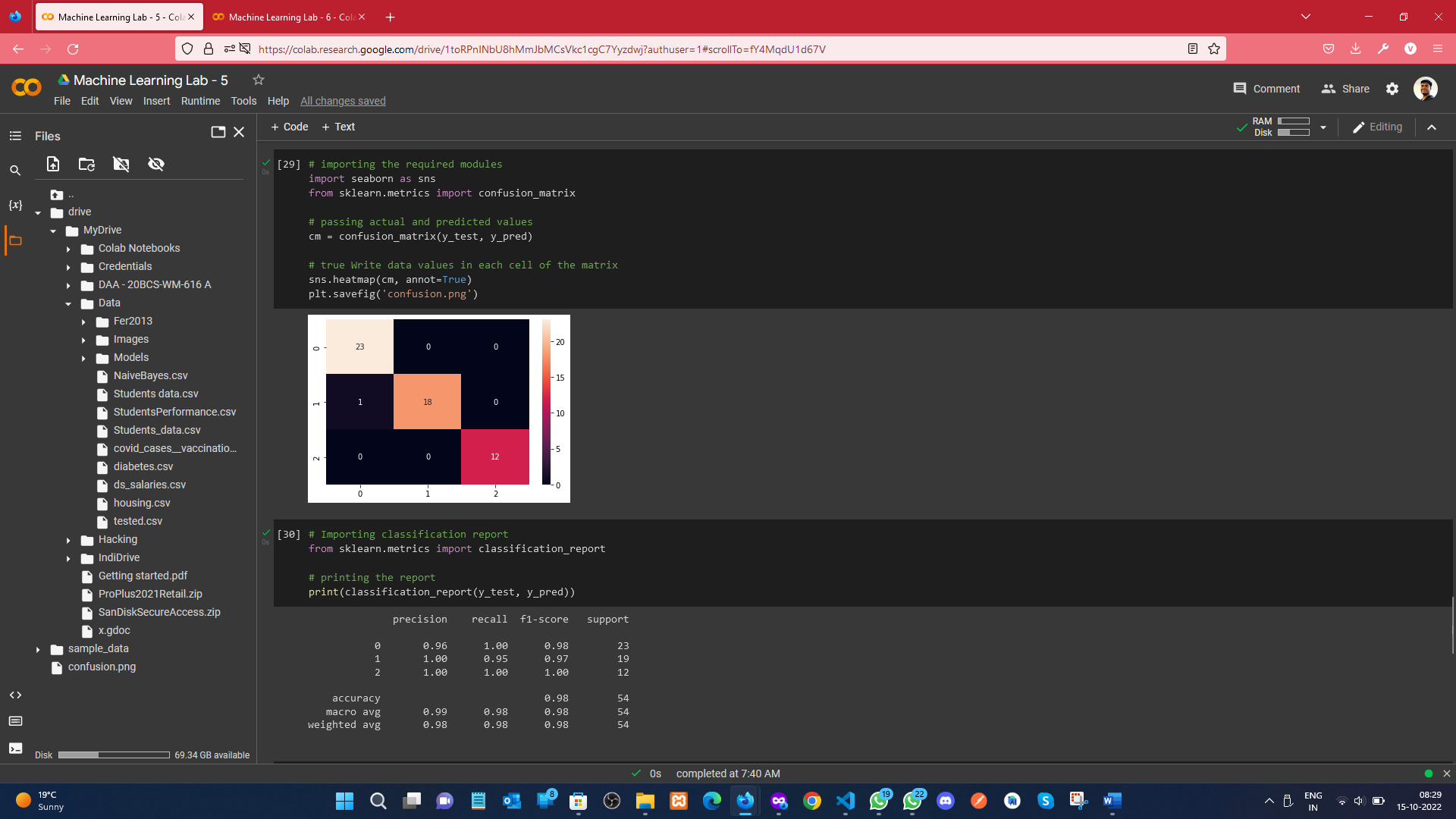












**Learning outcomes (What I have learnt):**

* Learned about how to calculate the probabilities required by the Naive Bayes algorithm.
* Learned about how to implement the Naive Bayes algorithm from scratch.
* Learned about how to apply Naive Bayes to a real-world predictive modeling problem.
* Understood the concept of Naïve Bayes (NB)
* Learnt how to split the data into training and testing parts and perform operation on it.
* Understood the concept of GaussianNB, BernoulliNB, and confusion matrix.
* Finally plotted the classification report.
* **Evaluation Grid (To be created as per the SOP and Assessment guidelines by the faculty):**

|  |  |  |  |
| --- | --- | --- | --- |
| Sr. No. | Parameters | Marks Obtained | Maximum Marks |
| 1. |  |  |  |
| 2. |  |  |  |
| 3. |  |  |  |
|  |  |  |  |