**ML LAB ASSIGNMENT 3**

**Name:** Abhay Vijay Jagtap

**Class:** BE-B

**Problem Statement:**

# Given a bank customer, build a neural network-based classifier that can determine whether they will leave or not in the next 6 months.

Dataset Description: The case study is from an open-source dataset from Kaggle. The dataset contains 10,000 sample points with 14 distinct features such as CustomerId, CreditScore, Geography, Gender, Age, Tenure, Balance, etc.

Link to the Kaggle project: <https://www.kaggle.com/barelydedicated/bank-customer-churn-modeling>

Perform following steps:

1. Read the dataset.
2. Distinguish the feature and target set and divide the data set into training and test sets.
3. Normalize the train and test data.
4. Initialize and build the model. Identify the points of improvement and implement the same.
5. Print the accuracy score and confusion matrix.

#Importing the libraries

import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

df = pd.read\_csv("/content/Churn\_Modelling.csv")

# 

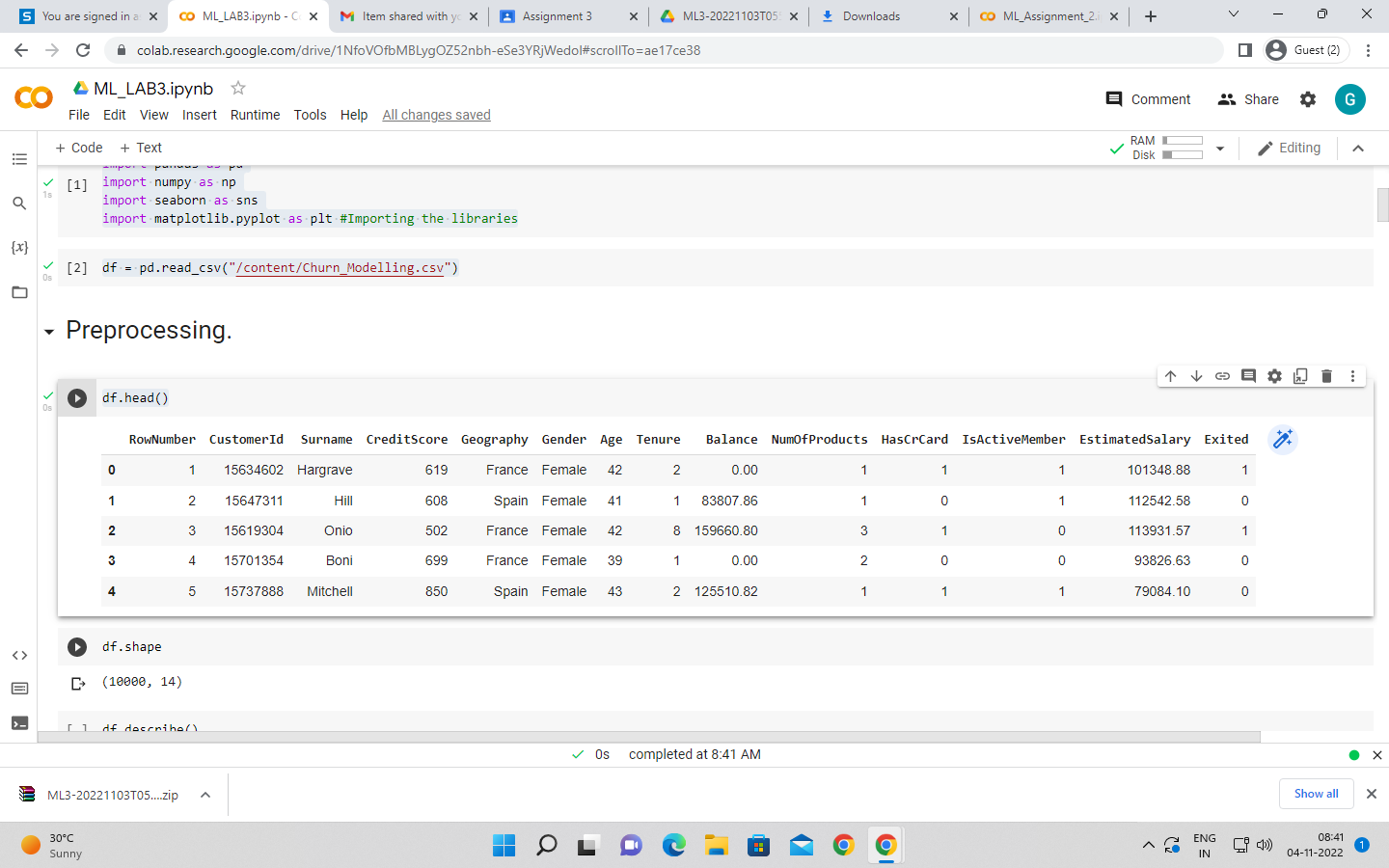
# 

# 

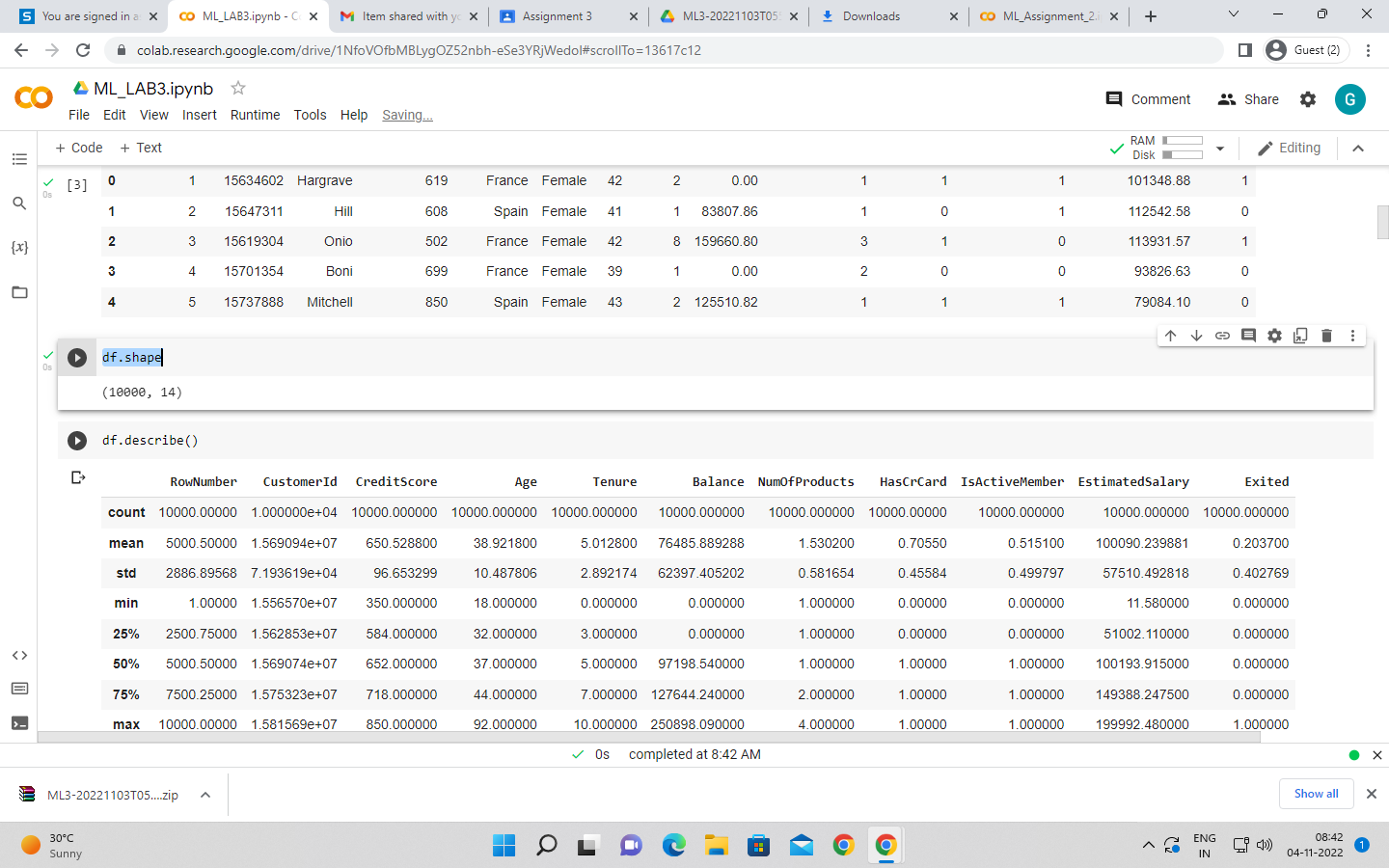
# 

# Preprocessing.

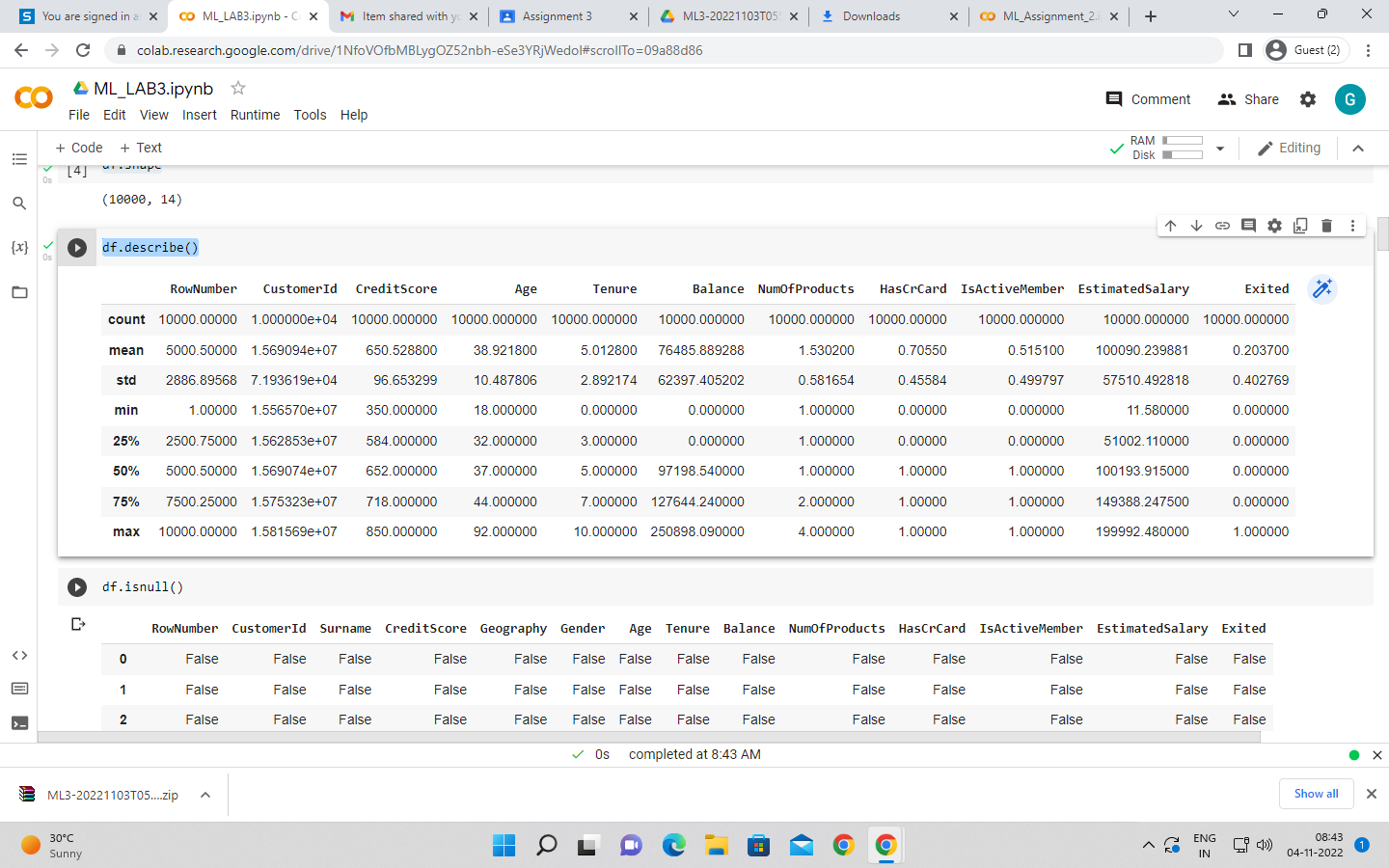
df.head()



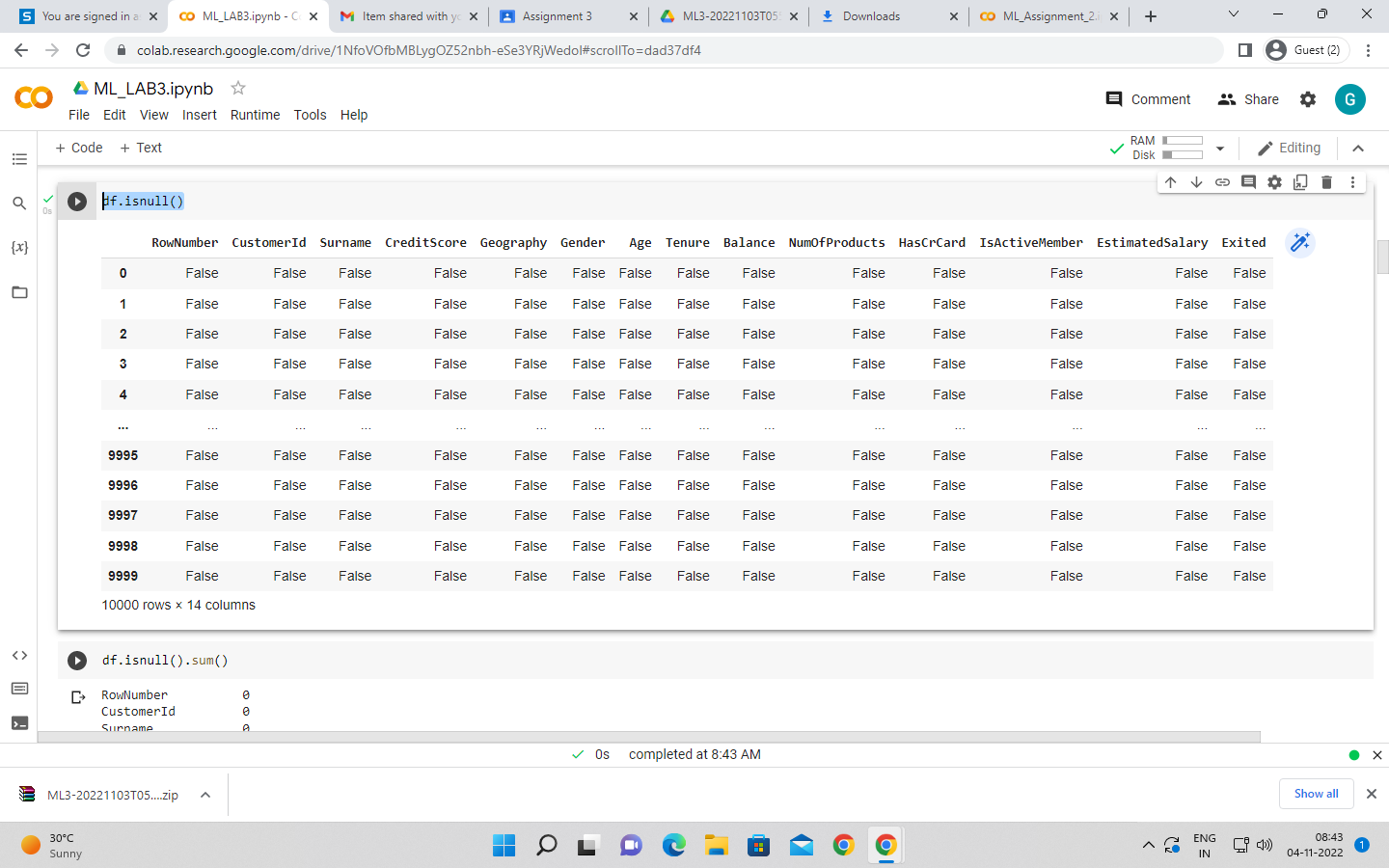
df.shape



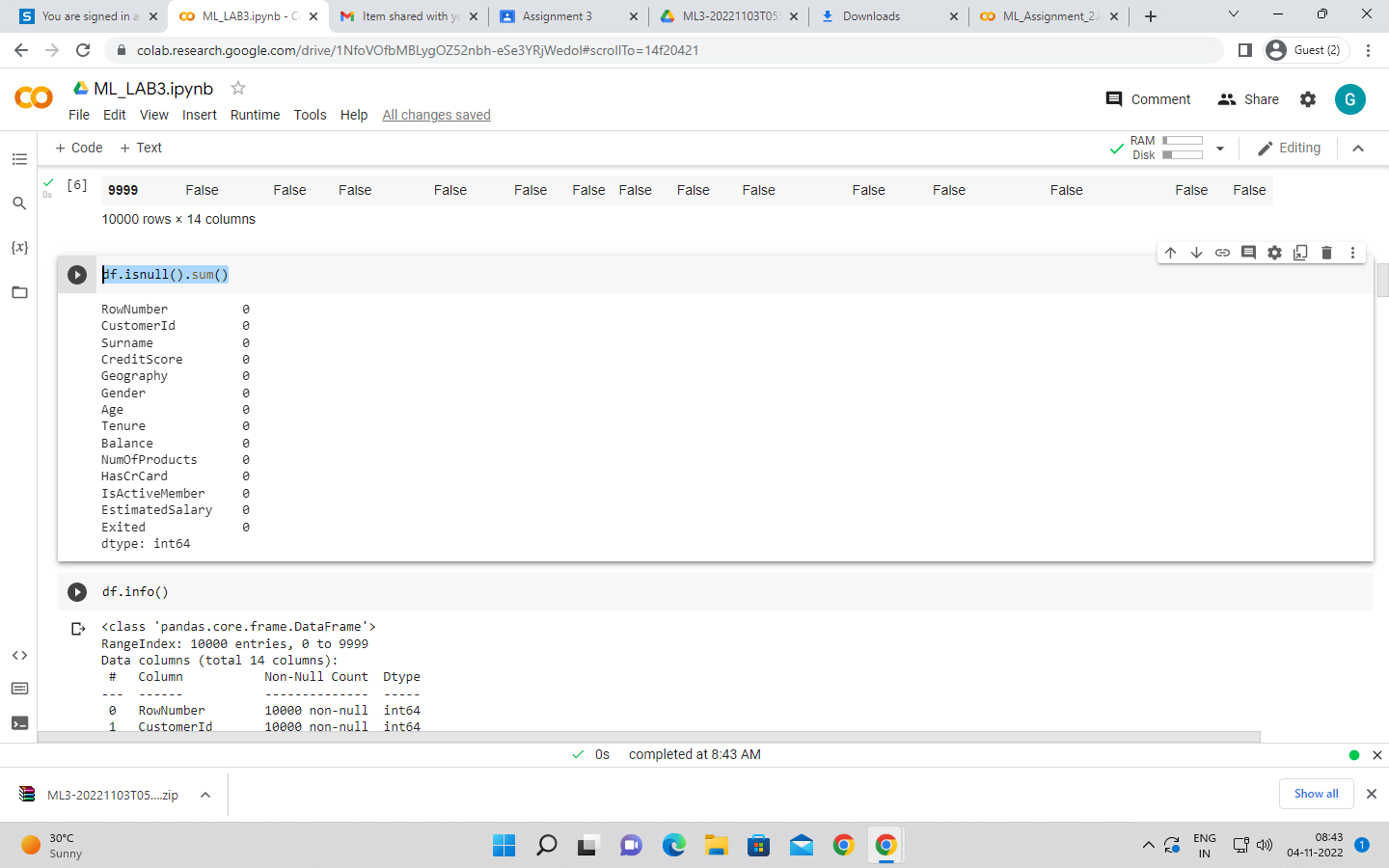
df.describe()



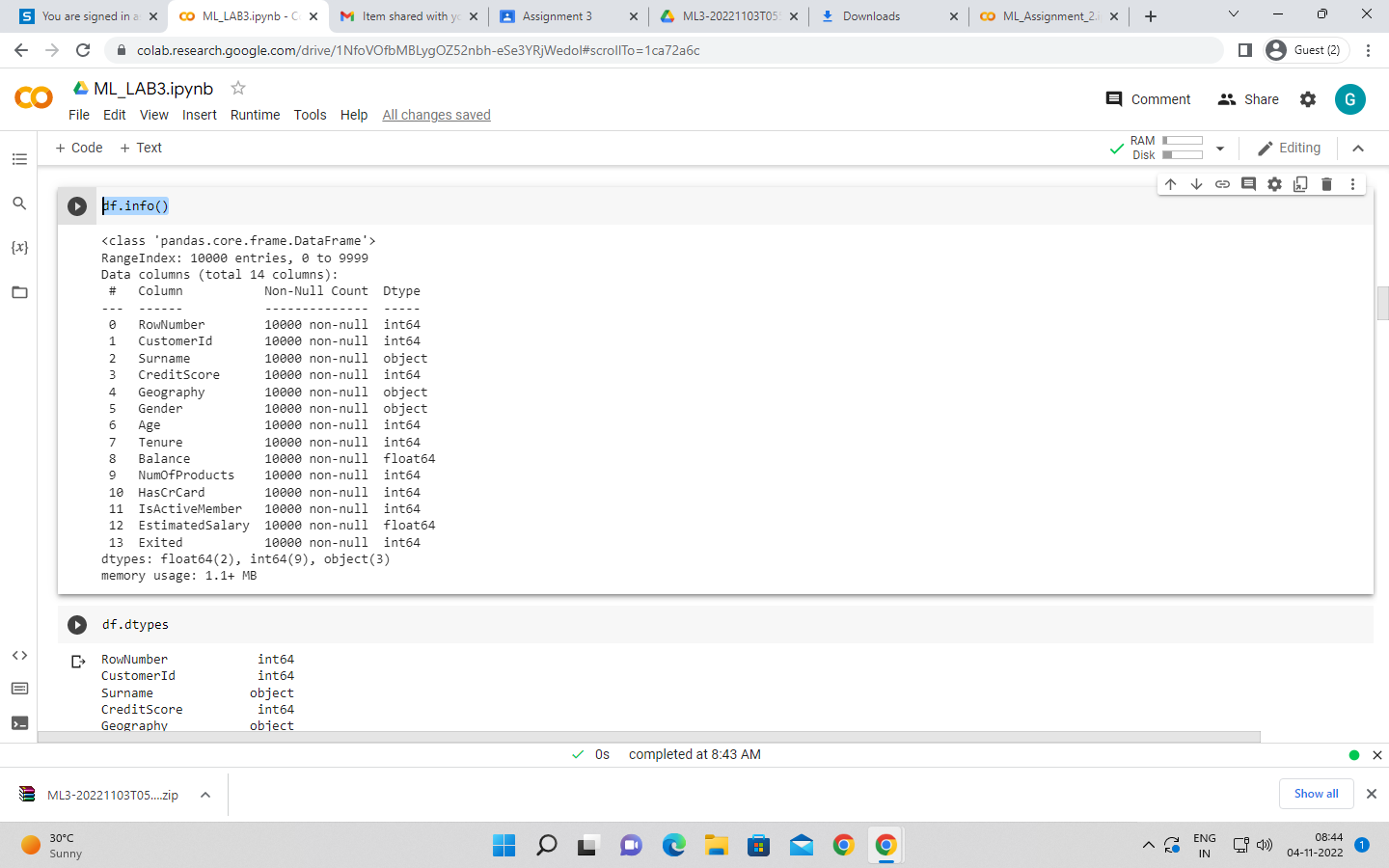
df.isnull()



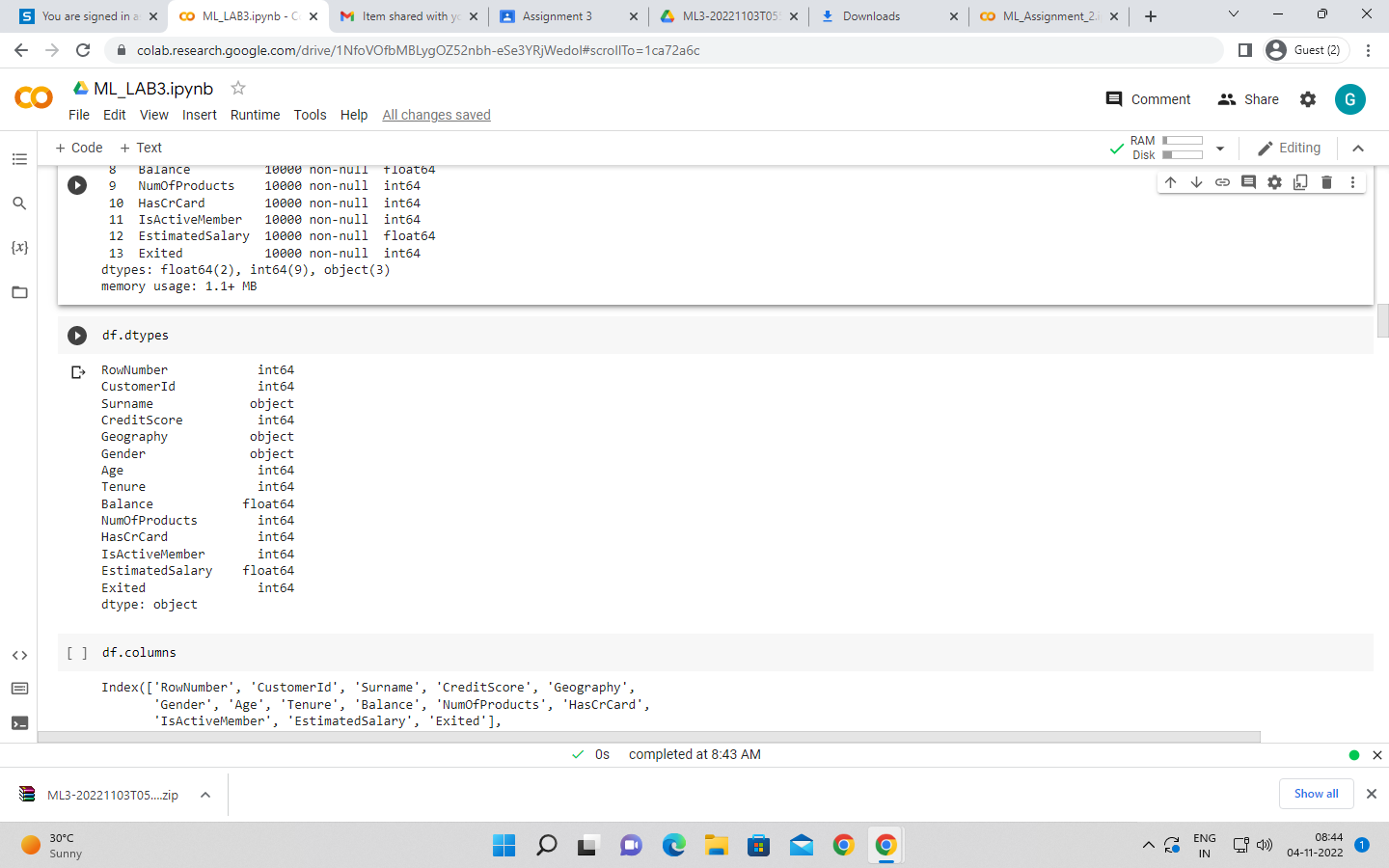
df.isnull().sum()



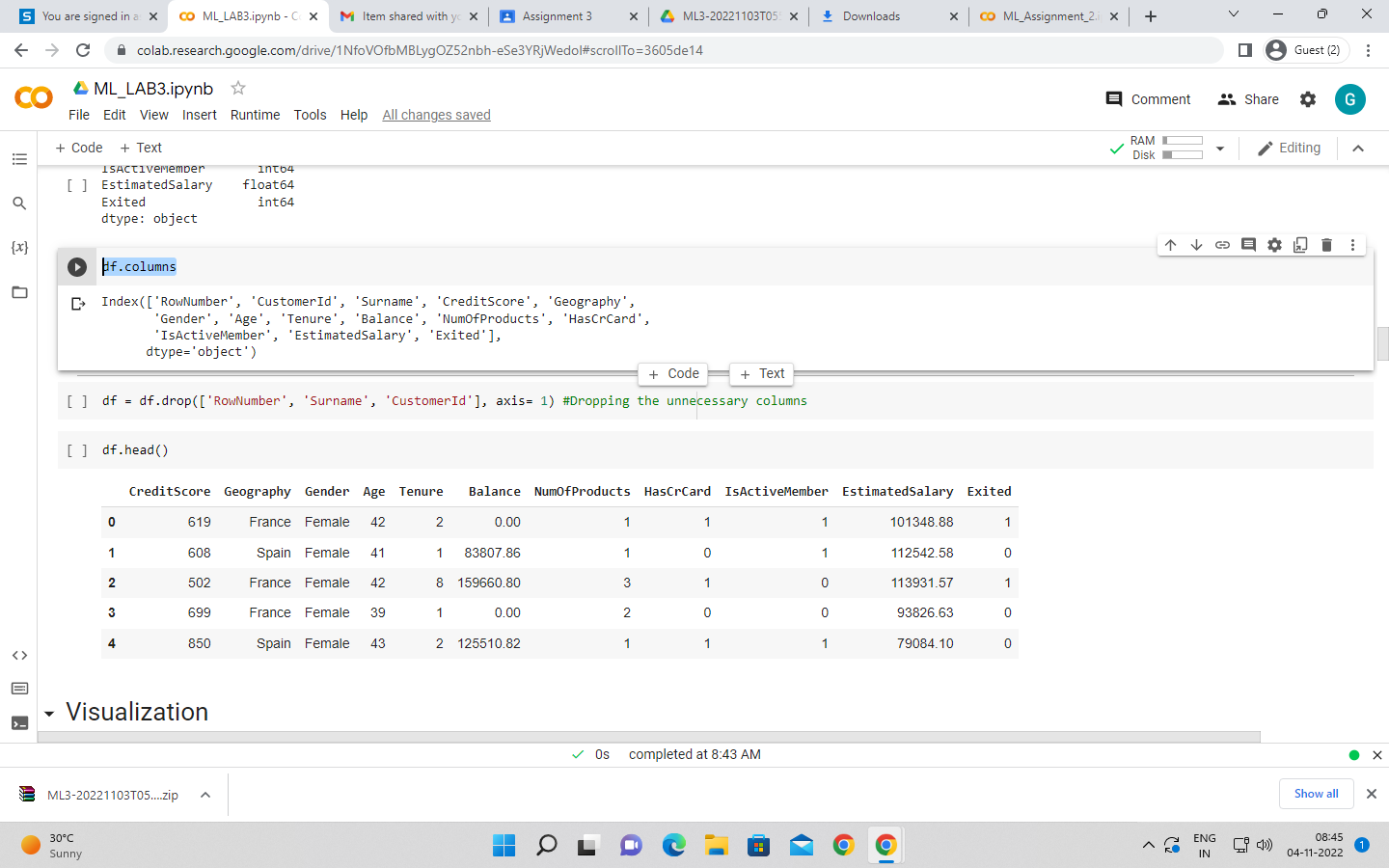
df.info()



df.dtypes



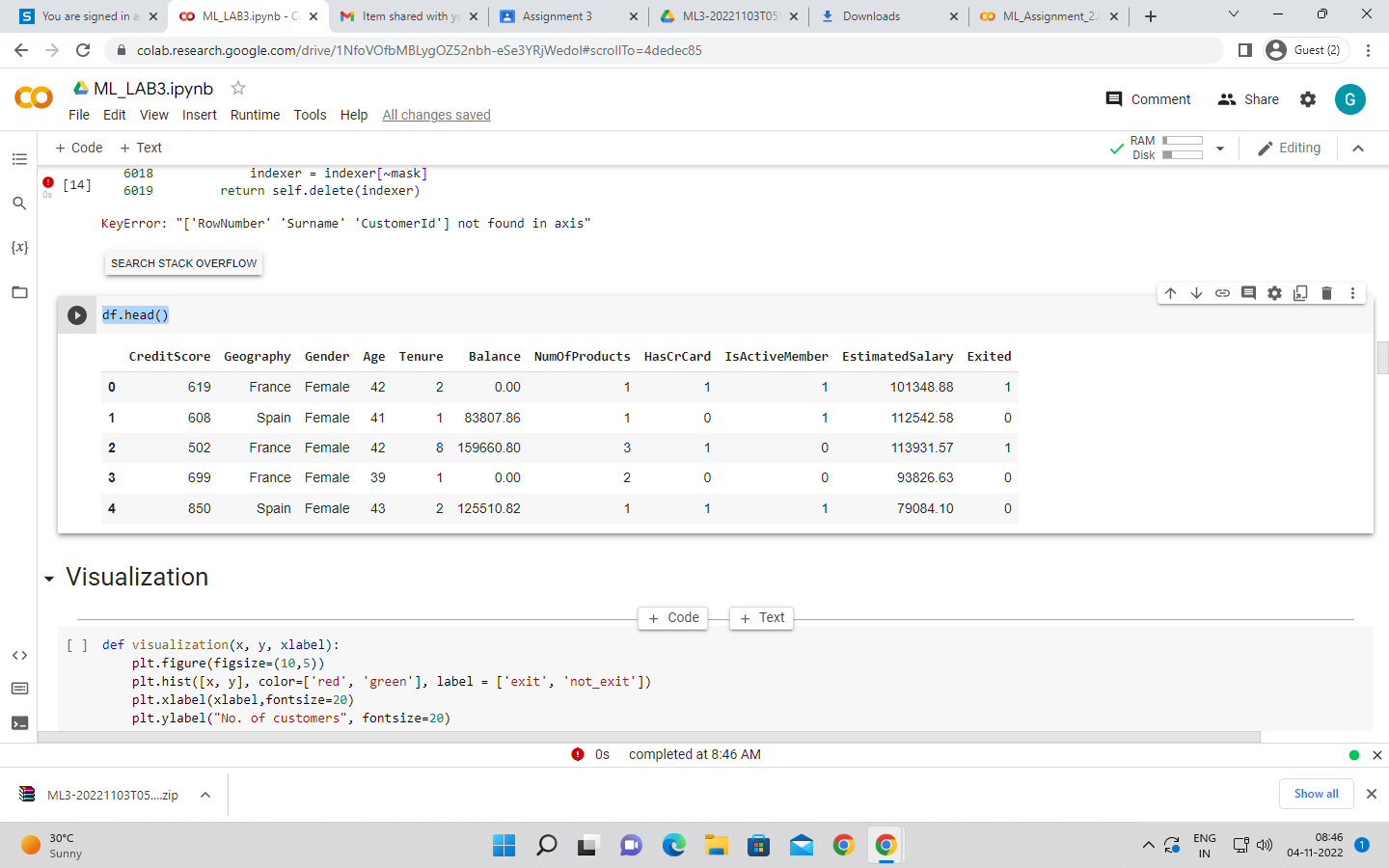
df.columns



#Dropping the unnecessary columns

df = df.drop(['RowNumber', 'Surname', 'CustomerId'], axis= 1)

df.head()



# Visualization

def visualization(x, y, xlabel):

plt.figure(figsize=(10,5))

plt.hist([x, y], color=['red', 'green'], label = ['exit', 'not\_exit'])

plt.xlabel(xlabel,fontsize=20)

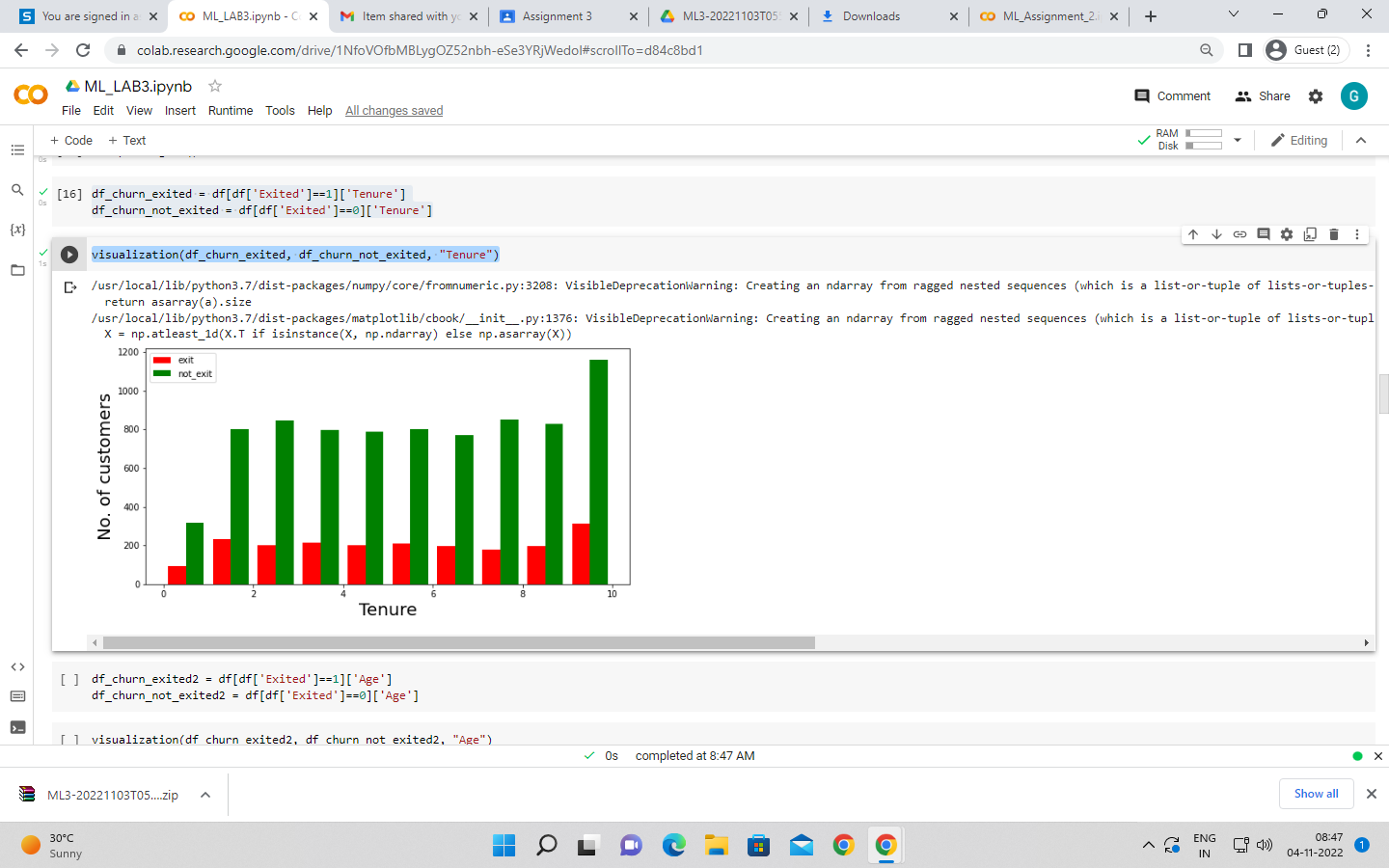
plt.ylabel("No. of customers", fontsize=20)

plt.legend()

df\_churn\_exited = df[df['Exited']==1]['Tenure']

df\_churn\_not\_exited = df[df['Exited']==0]['Tenure']

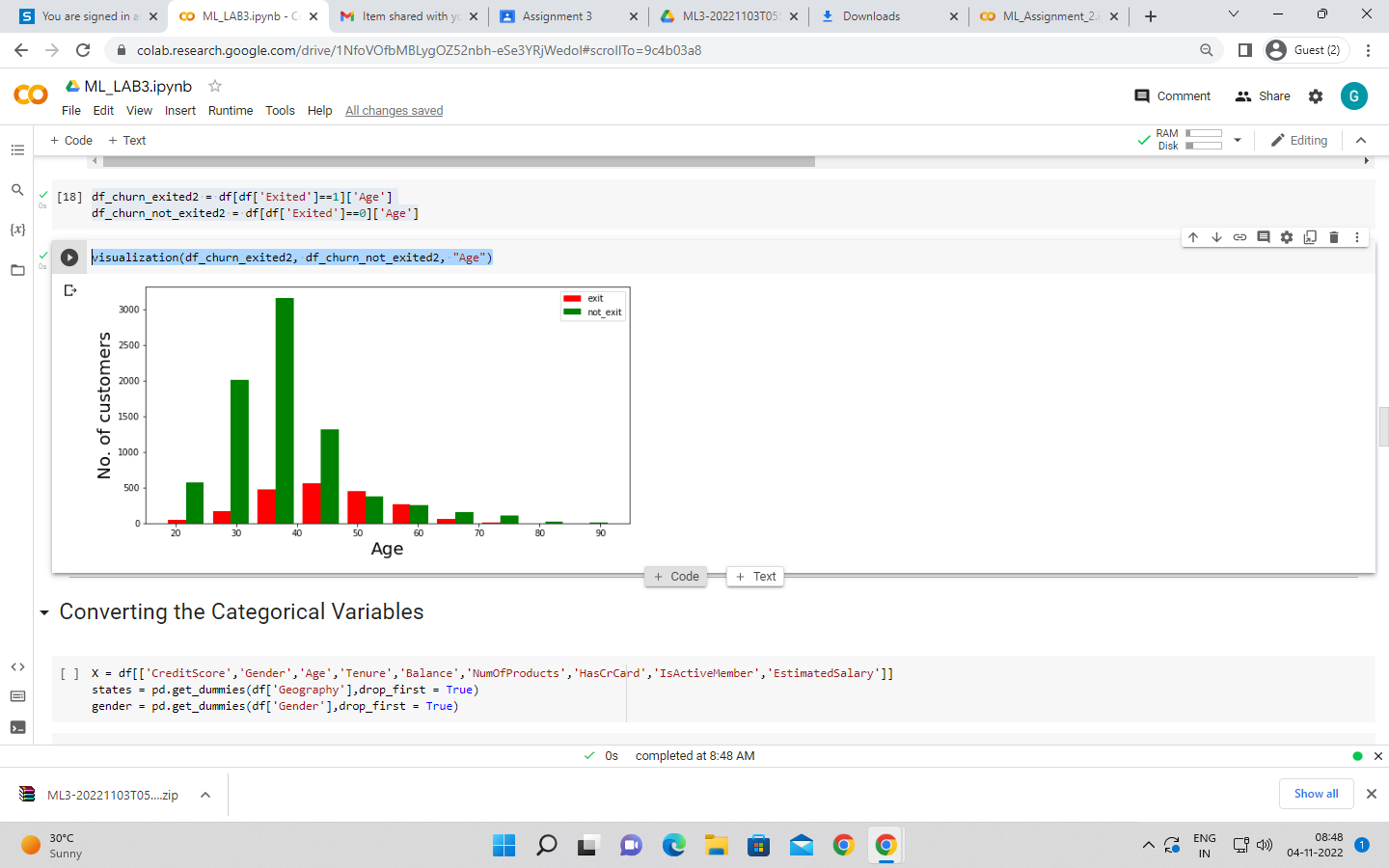
visualization(df\_churn\_exited, df\_churn\_not\_exited, "Tenure")



df\_churn\_exited2 = df[df['Exited']==1]['Age']

df\_churn\_not\_exited2 = df[df['Exited']==0]['Age']

visualization(df\_churn\_exited2, df\_churn\_not\_exited2, "Age")



# Converting the Categorical Variables

X = df[['CreditScore','Gender','Age','Tenure','Balance','NumOfProducts','HasCrCard','IsActiveMember','EstimatedSalary']]

states = pd.get\_dummies(df['Geography'],drop\_first = True)

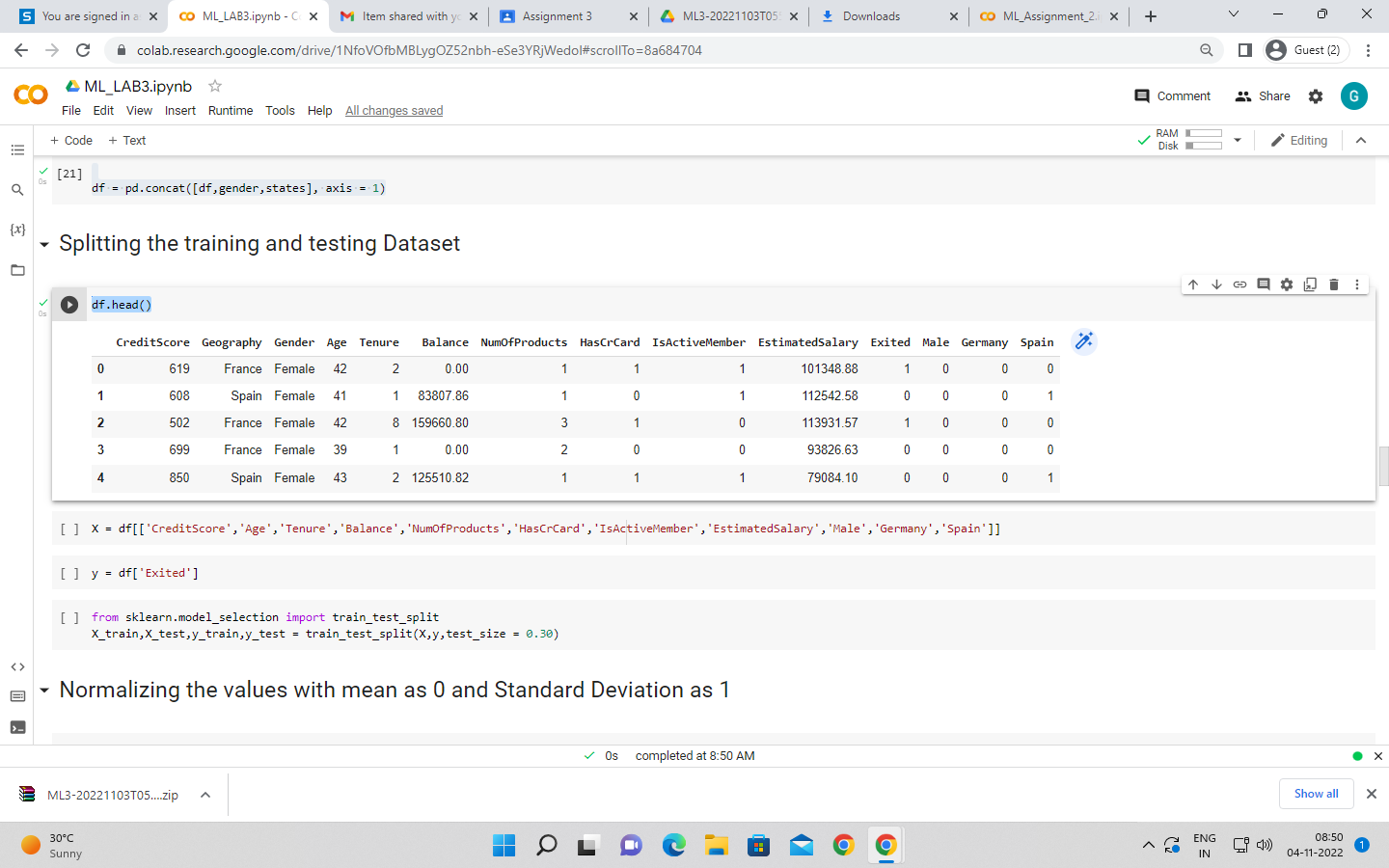
gender = pd.get\_dummies(df['Gender'],drop\_first = True)

df = pd.concat([df,gender,states], axis = 1)

# 

# Splitting the training and testing Dataset

df.head()



X = df[['CreditScore','Age','Tenure','Balance','NumOfProducts','HasCrCard','IsActiveMember','EstimatedSalary','Male','Germany','Spain']]

y = df['Exited']

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

X\_train,X\_test,y\_train,y\_test = train\_test\_split(X,y,test\_size = 0.30)

# Normalizing the values with mean as 0 and Standard Deviation as 1

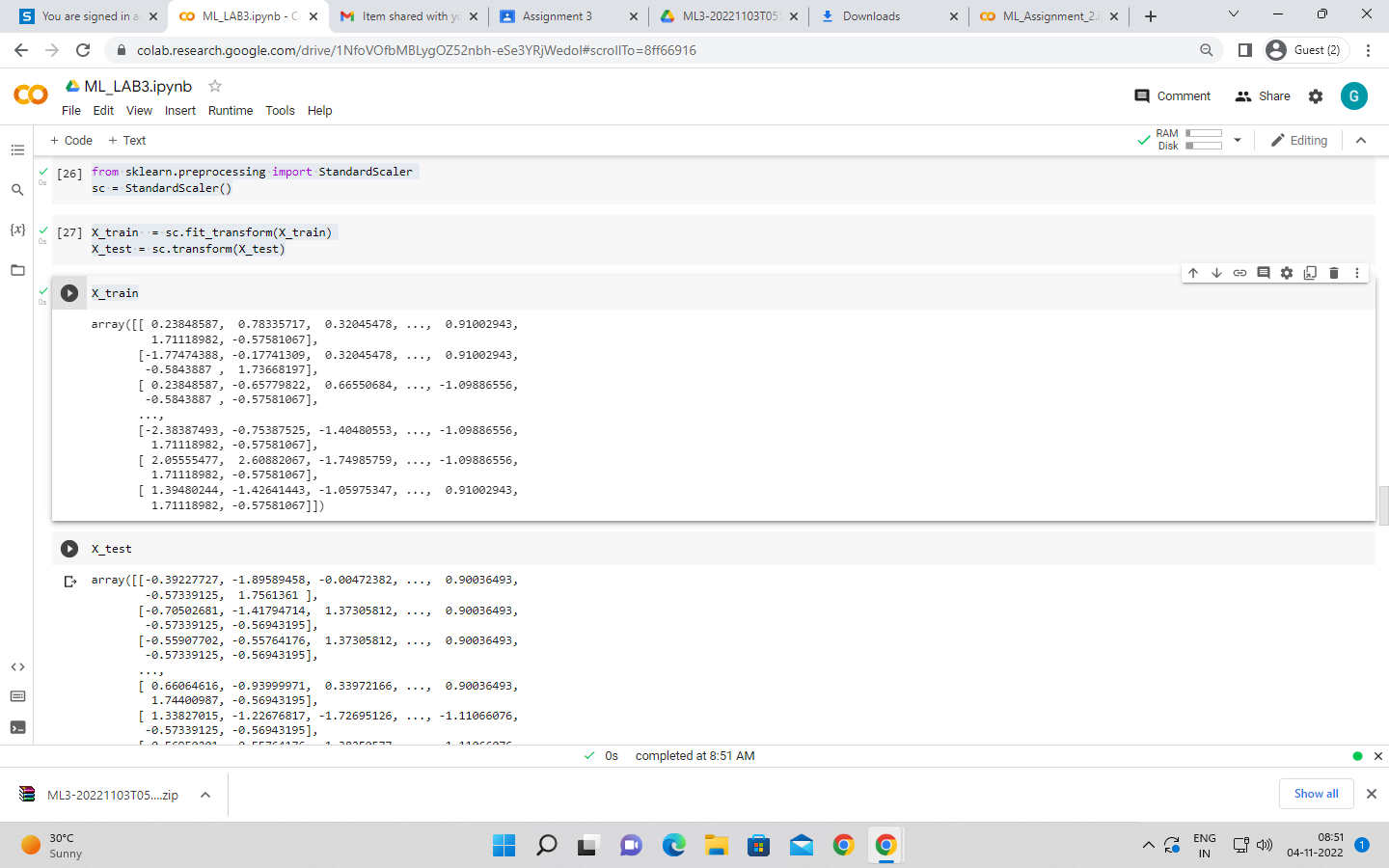
from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

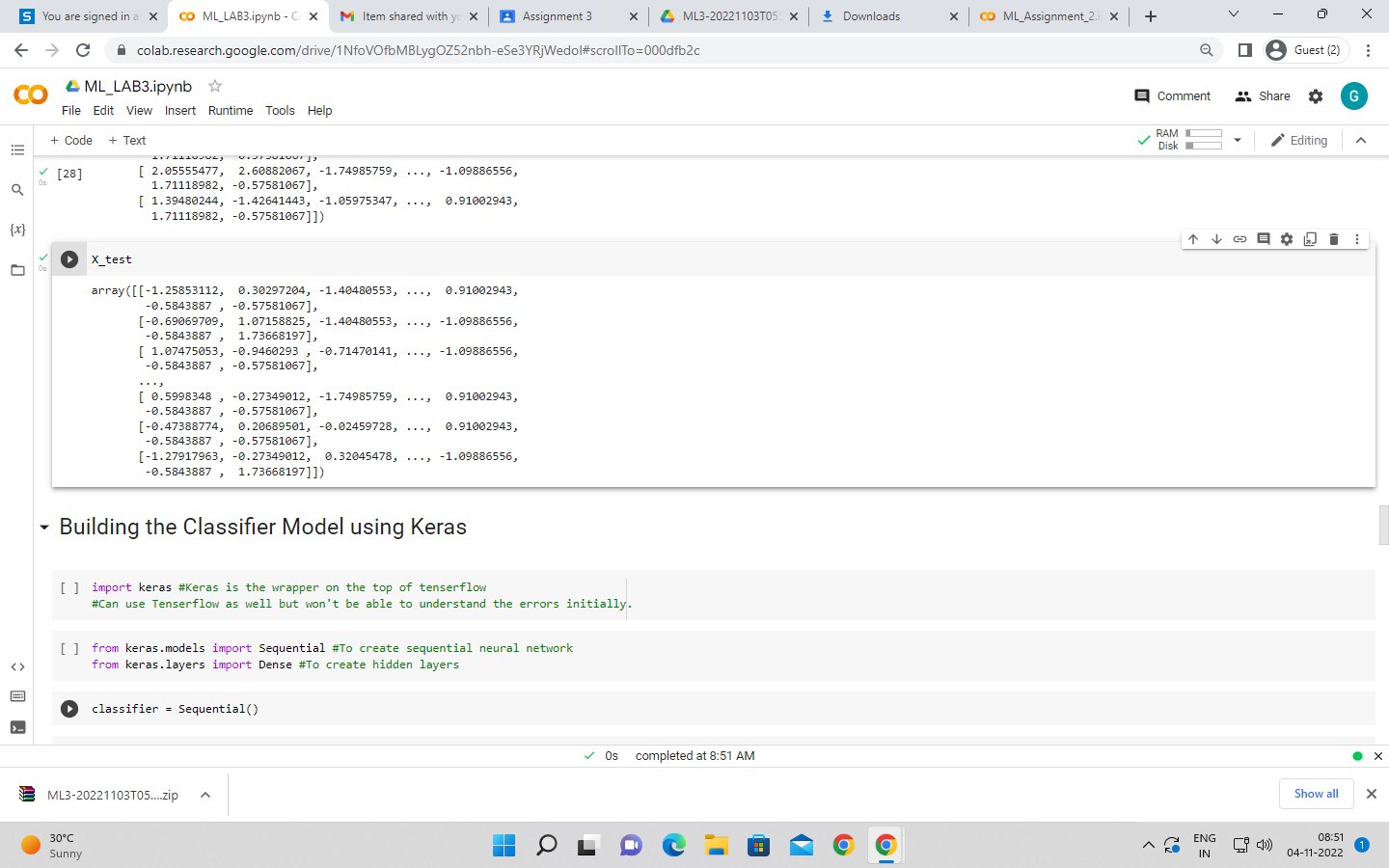
X\_train = sc.fit\_transform(X\_train)

X\_test = sc.transform(X\_test)

X\_train



X\_test



# Building the Classifier Model using Keras

import keras #Keras is the wrapper on the top of tenserflow

#Can use Tenserflow as well but won't be able to understand the errors initially.

from keras.models import Sequential #To create sequential neural network

from keras.layers import Dense #To create hidden layers

classifier = Sequential()

#To add the layers

#Dense helps to contruct the neurons

#Input Dimension means we have 11 features

# Units is to create the hidden layers

#Uniform helps to distribute the weight uniformly

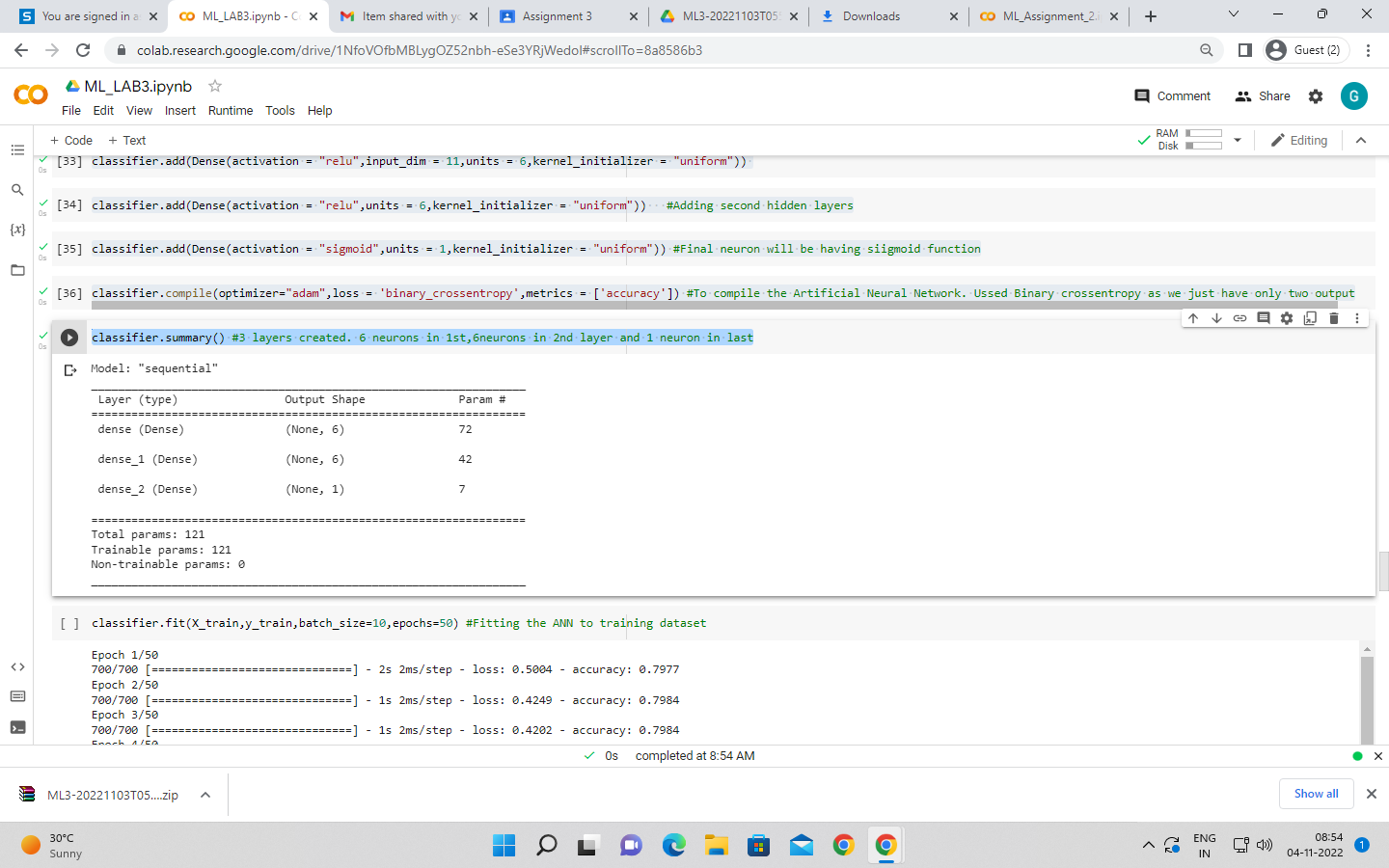
classifier.add(Dense(activation = "relu",input\_dim = 11,units = 6,kernel\_initializer = "uniform"))

classifier.add(Dense(activation = "relu",units = 6,kernel\_initializer = "uniform")) #Adding second hidden layers

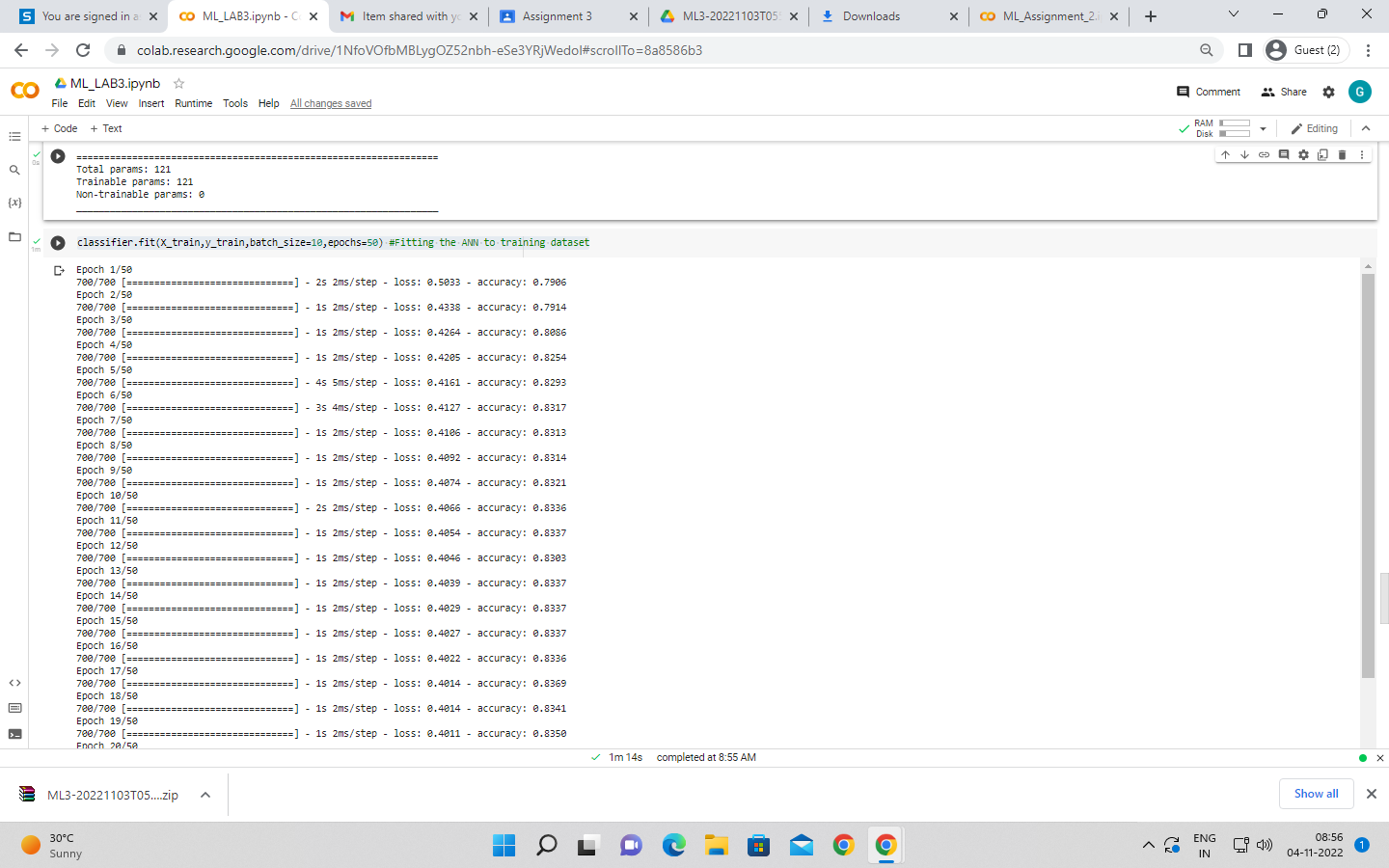
classifier.add(Dense(activation = "sigmoid",units = 1,kernel\_initializer = "uniform")) #Final neuron will be having siigmoid function

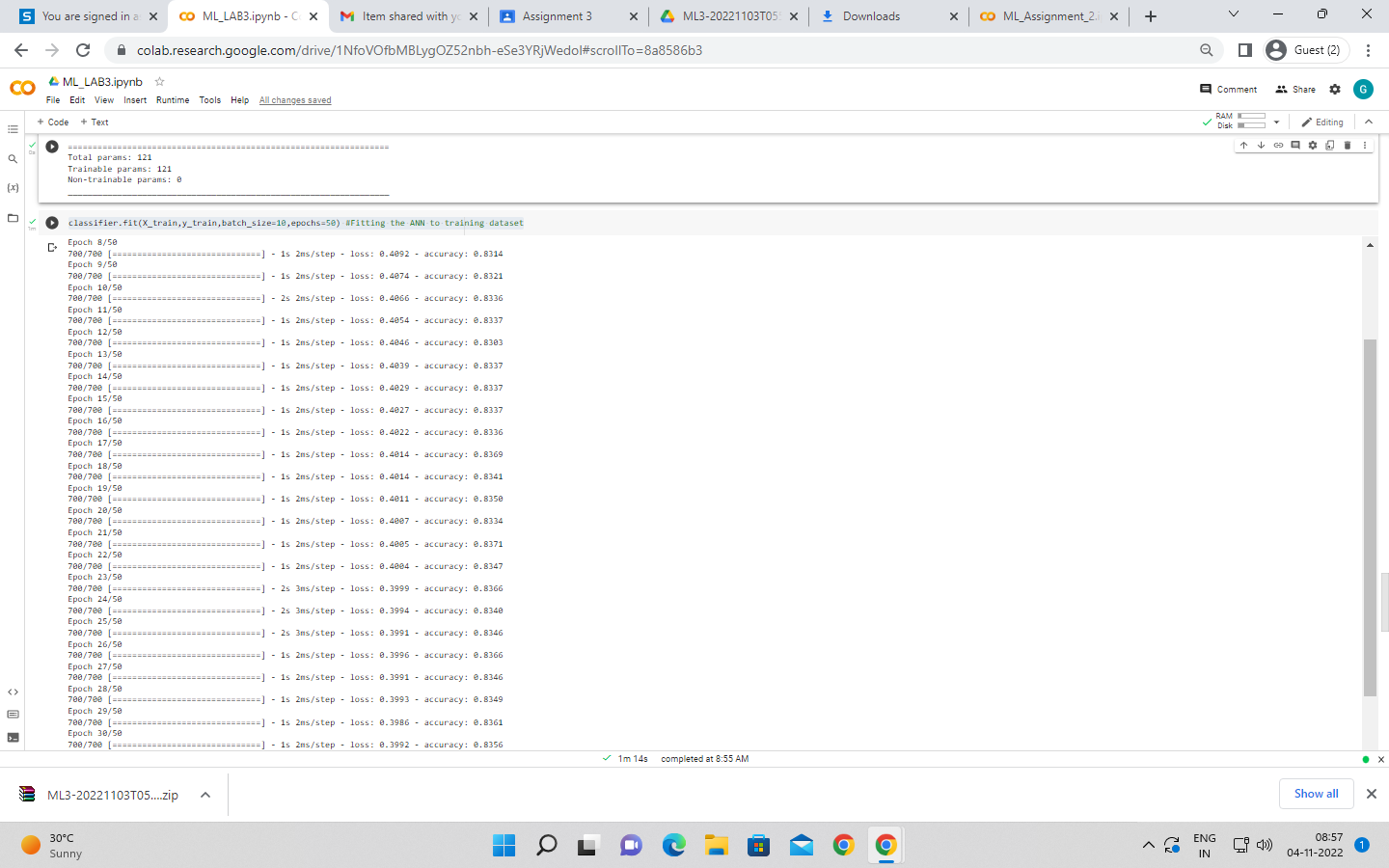
classifier.compile(optimizer="adam",loss = 'binary\_crossentropy',metrics = ['accuracy']) #To compile the Artificial Neural Network. Ussed Binary crossentropy as we just have only two output

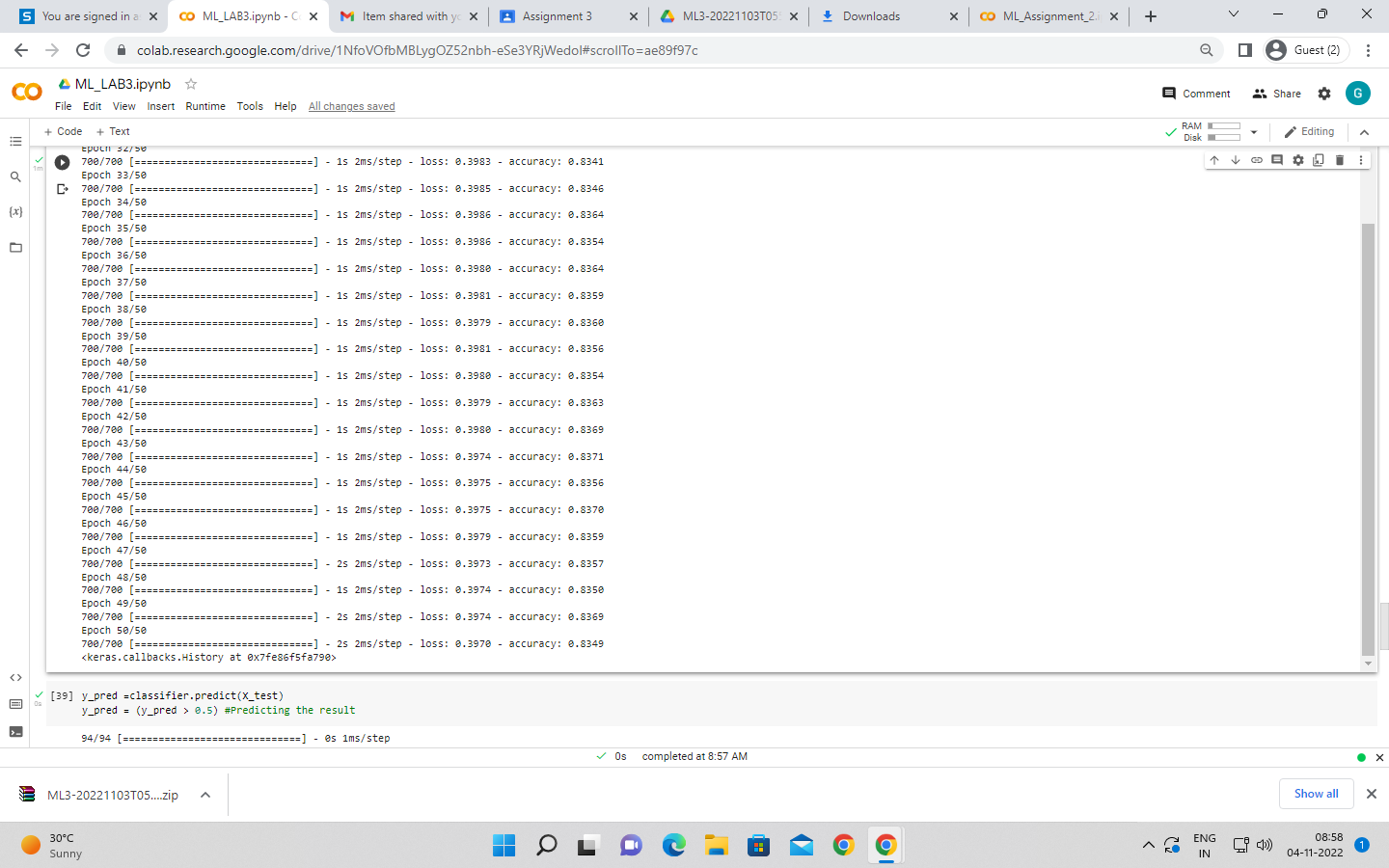
classifier.summary() #3 layers created. 6 neurons in 1st,6neurons in 2nd layer and 1 neuron in last



classifier.fit(X\_train,y\_train,batch\_size=10,epochs=50) #Fitting the ANN to training dataset

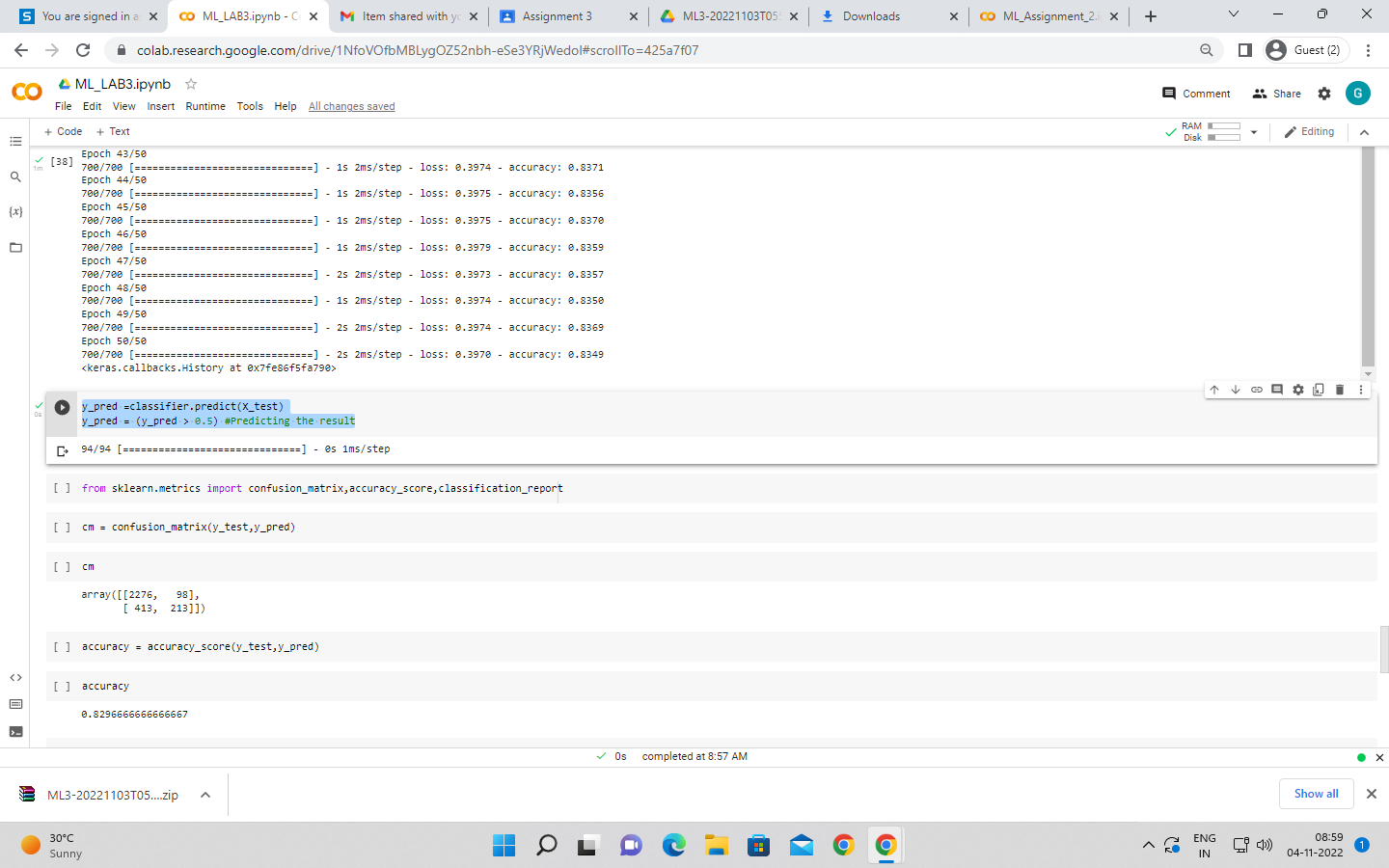






y\_pred =classifier.predict(X\_test)

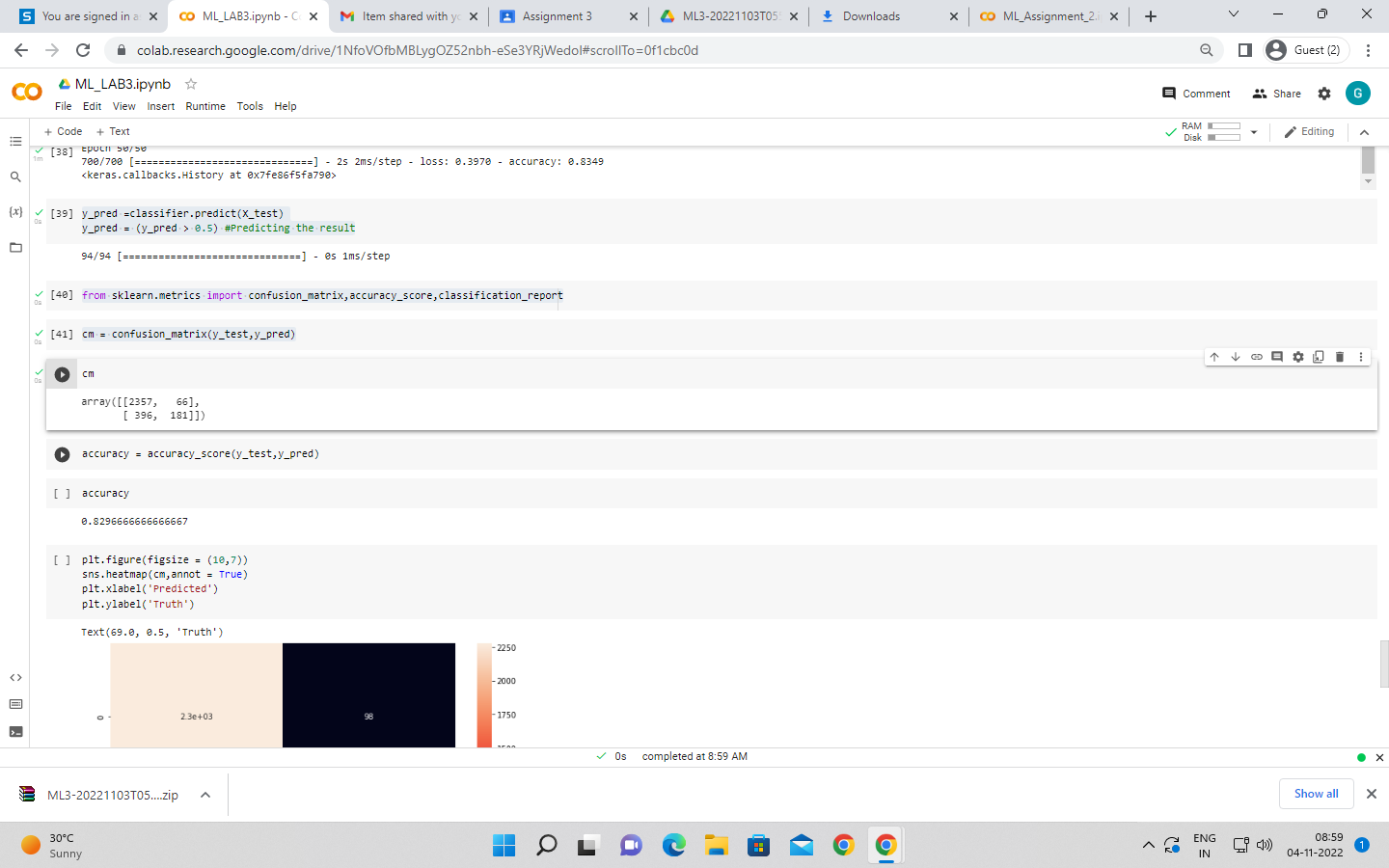
y\_pred = (y\_pred > 0.5) #Predicting the result



from sklearn.metrics import confusion\_matrix,accuracy\_score,classification\_report

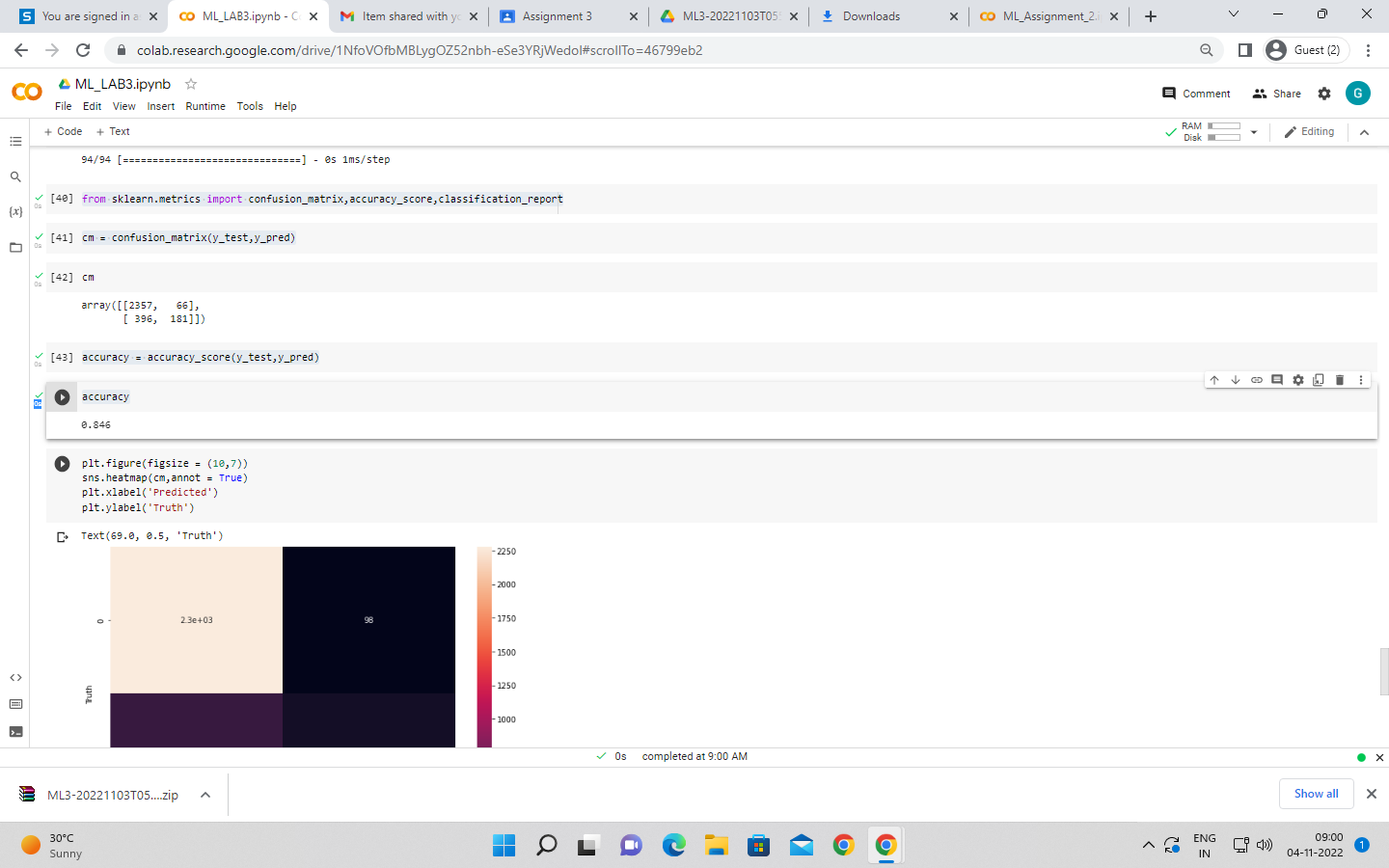
cm = confusion\_matrix(y\_test,y\_pred)

cm



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

accuracy

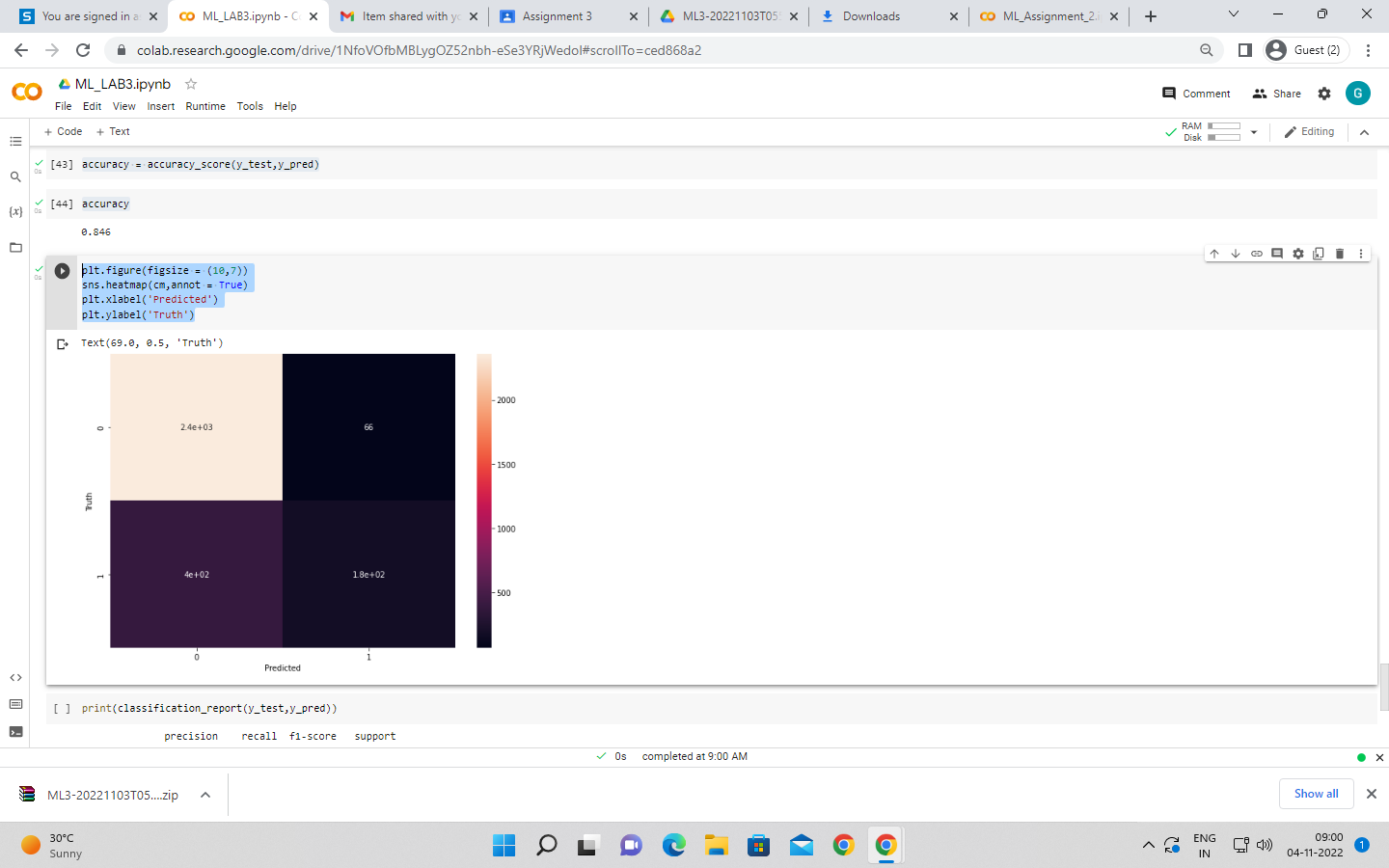


plt.figure(figsize = (10,7))

sns.heatmap(cm,annot = True)

plt.xlabel('Predicted')

plt.ylabel('Truth')



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

