**DL Lab Assignment 04**

Importing libraries

import matplotlib.pyplot as plt

import pandas as pd

import tensorflow as tf

import seaborn as sns

from tensorflow.keras.models import Model

from sklearn.metrics import accuracy\_score, precision\_score, recall\_score

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

from keras import Sequential

sns.set()

import numpy as np

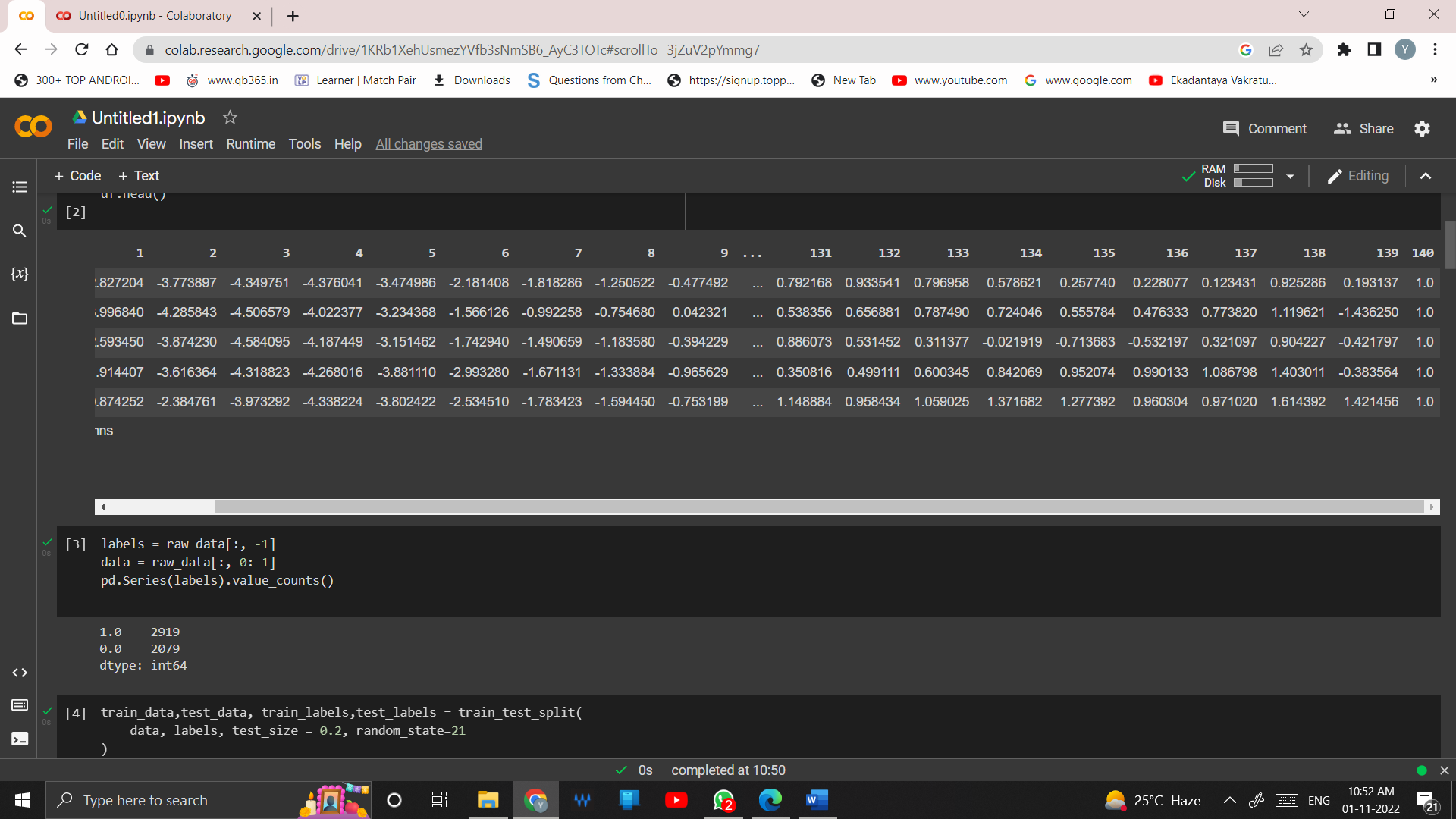
from tensorflow.keras import layers, losses

Importing data and Checking the data

df = pd.read\_csv('http://storage.googleapis.com/download.tensorflow.org/data/ecg.csv', header=None)

raw\_data = df.values

df.head()

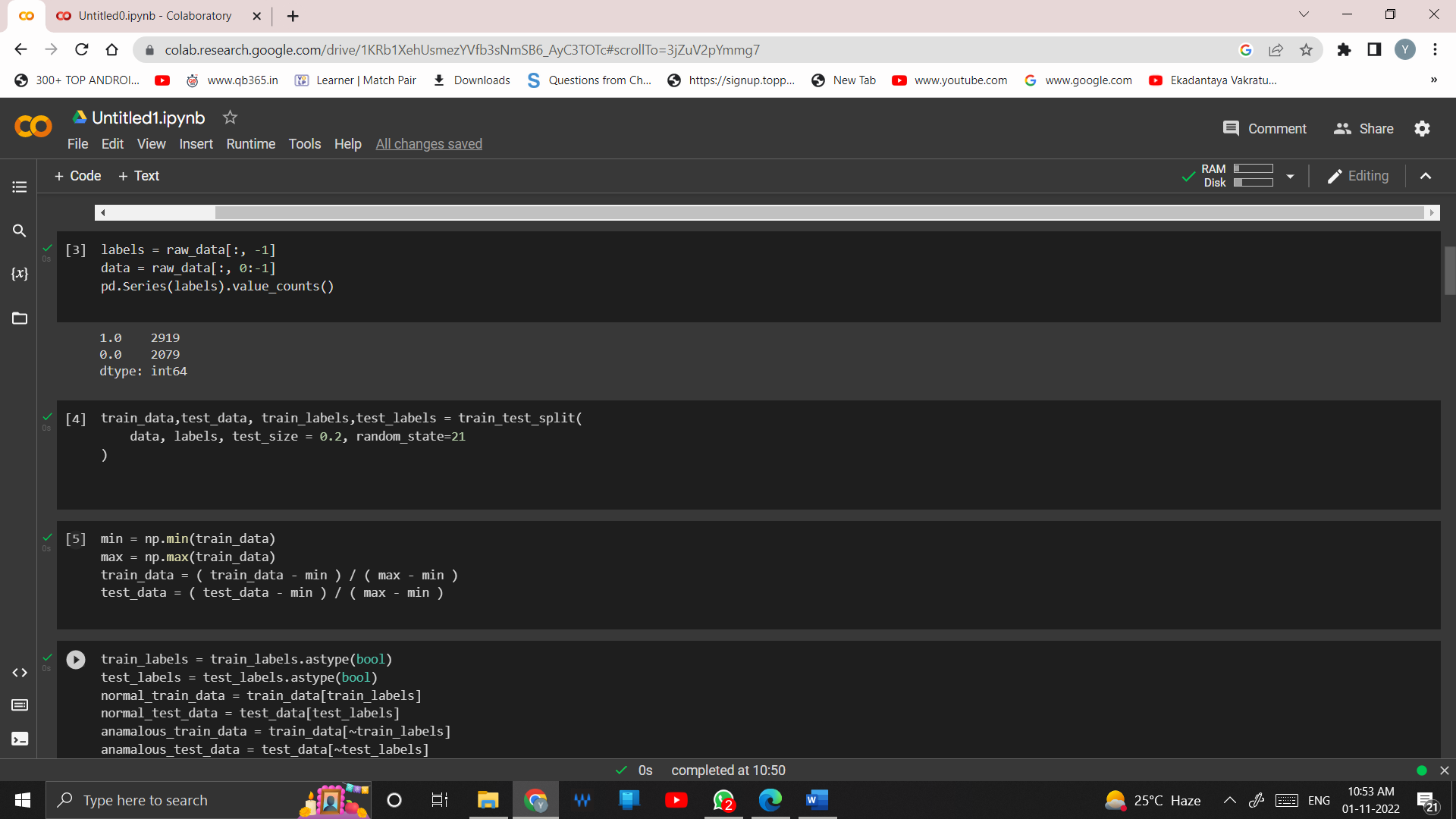


Splitting the data and labels

labels = raw\_data[:, -1]

data = raw\_data[:, 0:-1]

pd.Series(labels).value\_counts()



Splitting the training and testing data

train\_data,test\_data, train\_labels,test\_labels = train\_test\_split(

data, labels, test\_size = 0.2, random\_state=21

)

Structuring the data

min = np.min(train\_data)

max = np.max(train\_data)

train\_data = ( train\_data - min ) / ( max - min )

test\_data = ( test\_data - min ) / ( max - min )

**Dividing the data using true / false value**

train\_labels = train\_labels.astype(bool)

test\_labels = test\_labels.astype(bool)

normal\_train\_data = train\_data[train\_labels]

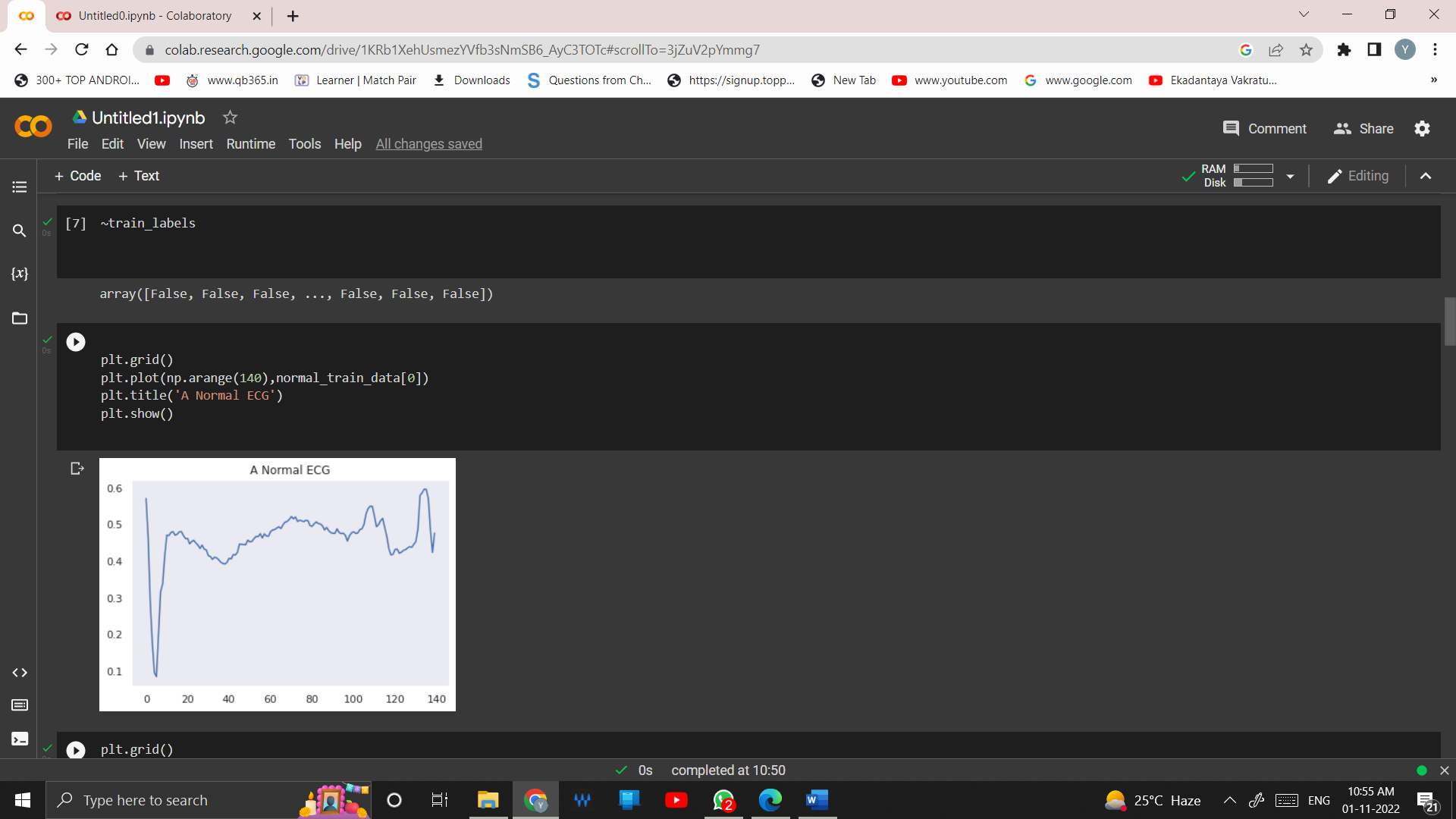
normal\_test\_data = test\_data[test\_labels]

anamalous\_train\_data = train\_data[~train\_labels]

anamalous\_test\_data = test\_data[~test\_labels]

**Printing Values**

~train\_labels



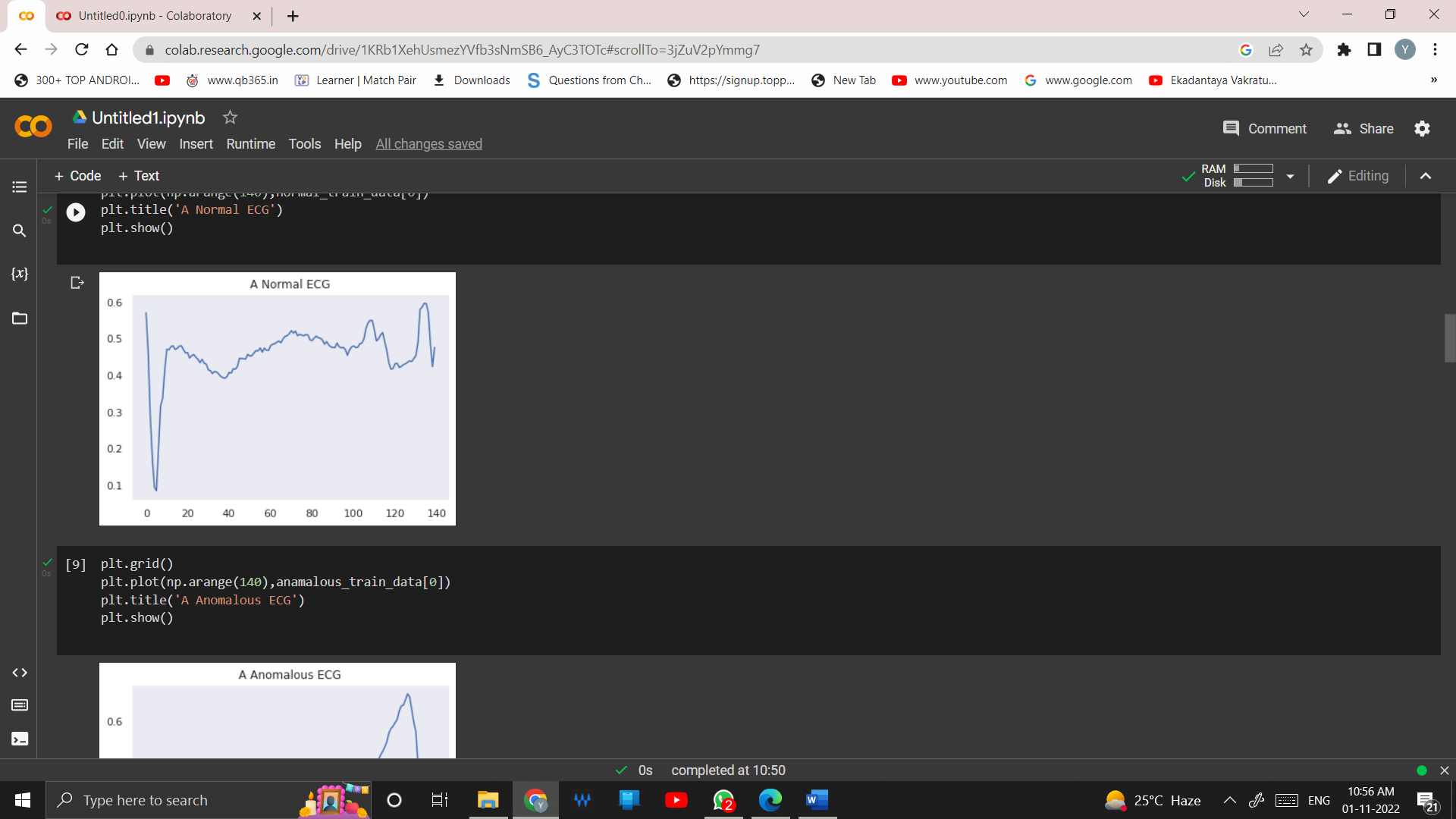
**Plotting normal ECG**

plt.grid()

plt.plot(np.arange(140),normal\_train\_data[0])

plt.title('A Normal ECG')

plt.show()



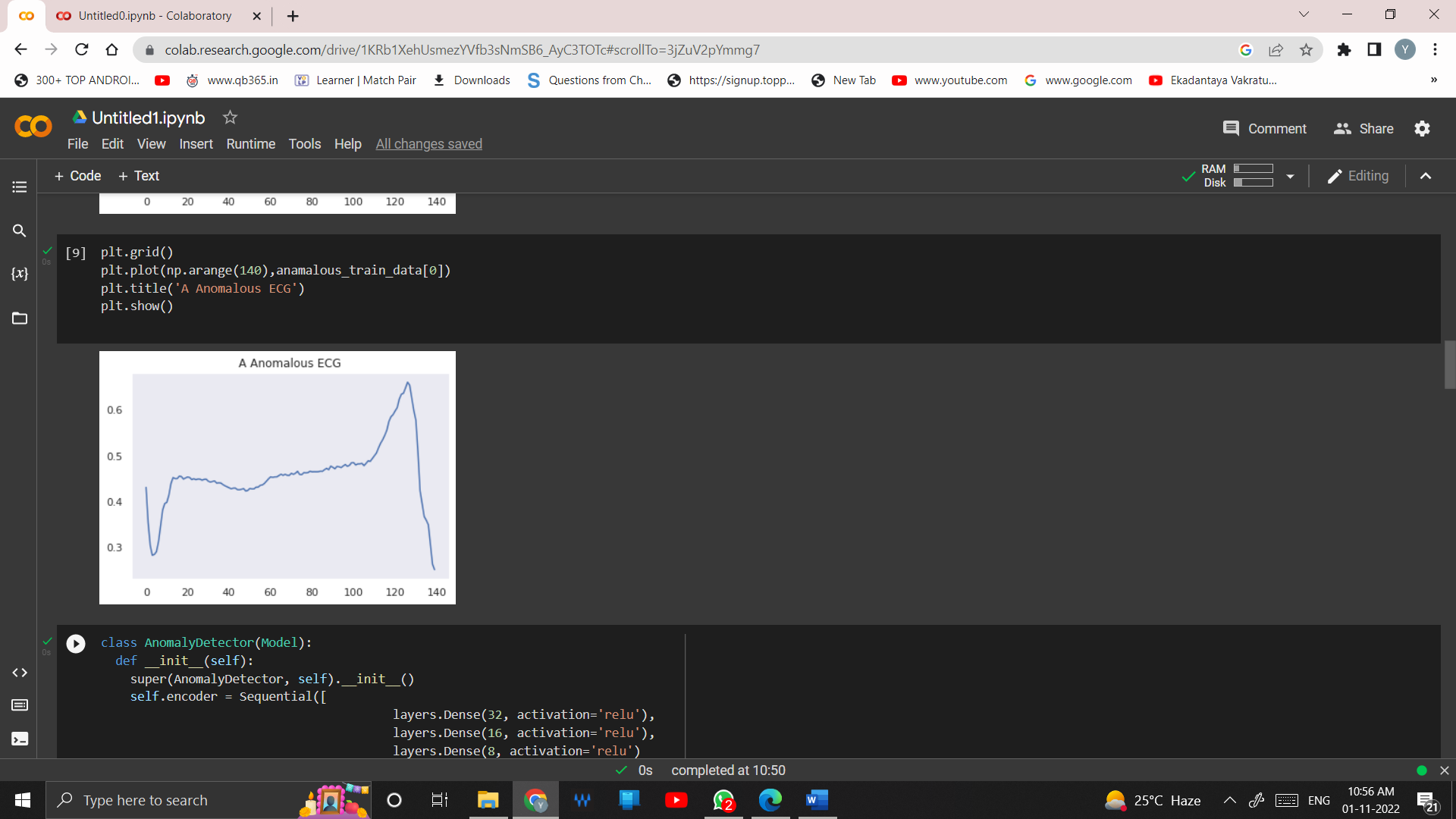
**Plotting a anomalous ECG**

plt.grid()

plt.plot(np.arange(140),anamalous\_train\_data[0])

plt.title('A Anomalous ECG')

plt.show()



**Model Building**

class AnomalyDetector(Model):

def \_\_init\_\_(self):

super(AnomalyDetector, self).\_\_init\_\_()

self.encoder = Sequential([

layers.Dense(32, activation='relu'),

layers.Dense(16, activation='relu'),

layers.Dense(8, activation='relu')

])

self.decoder = tf.keras.Sequential([

layers.Dense(16, activation='relu'),

layers.Dense(32, activation='relu'),

layers.Dense(140, activation='sigmoid')

])

def call(self, x):

encoded = self.encoder(x)

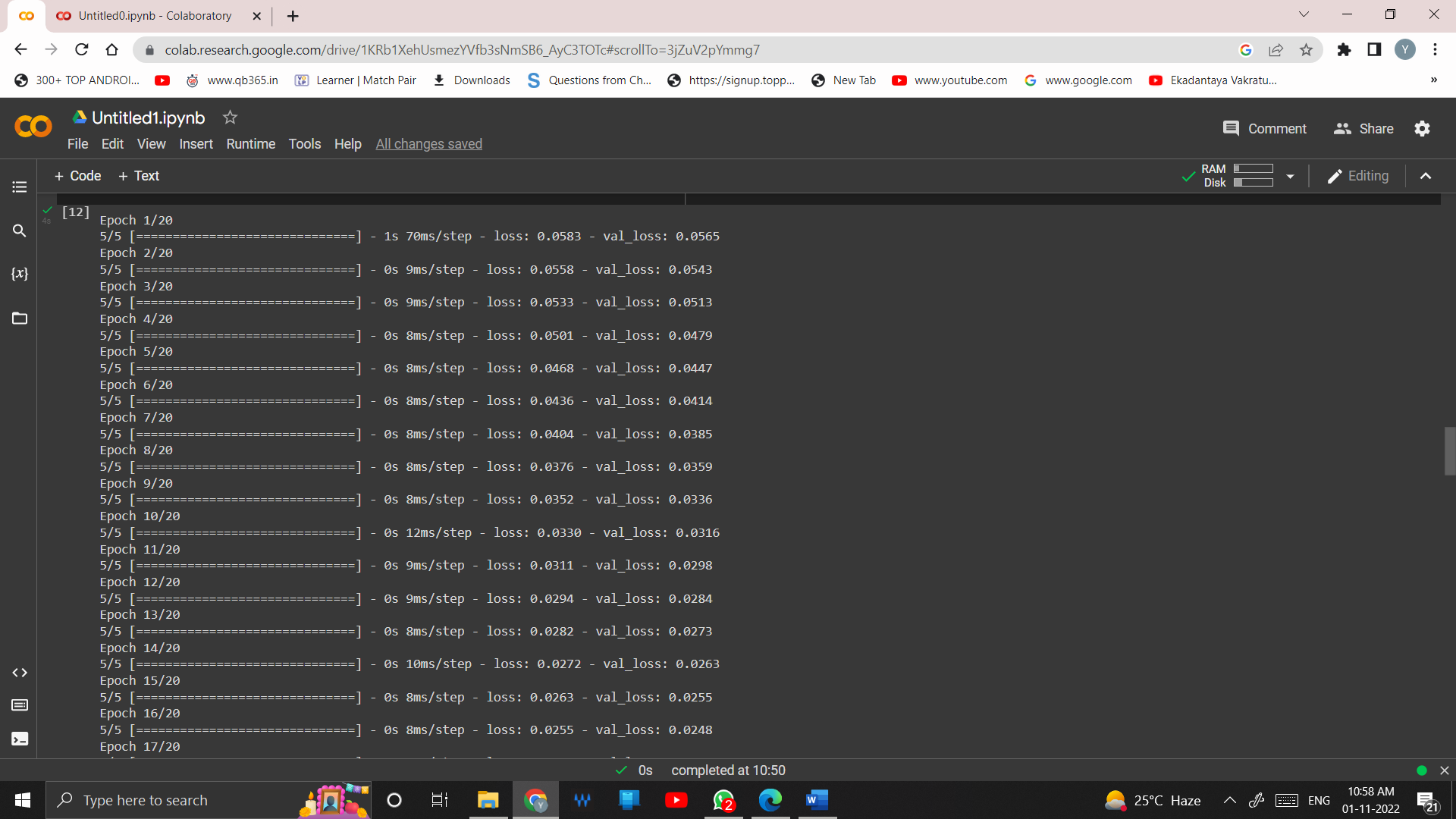
decoded = self.decoder(encoded)

return decoded

autoencoder = AnomalyDetector()

**Compiling model using loss function and optimizer**

autoencoder.compile(optimizer='adam', loss='mae')



**Fitting the data**

history = autoencoder.fit(normal\_train\_data, normal\_train\_data,

epochs = 20,

batch\_size=512,

validation\_data=(normal\_test\_data, normal\_test\_data),

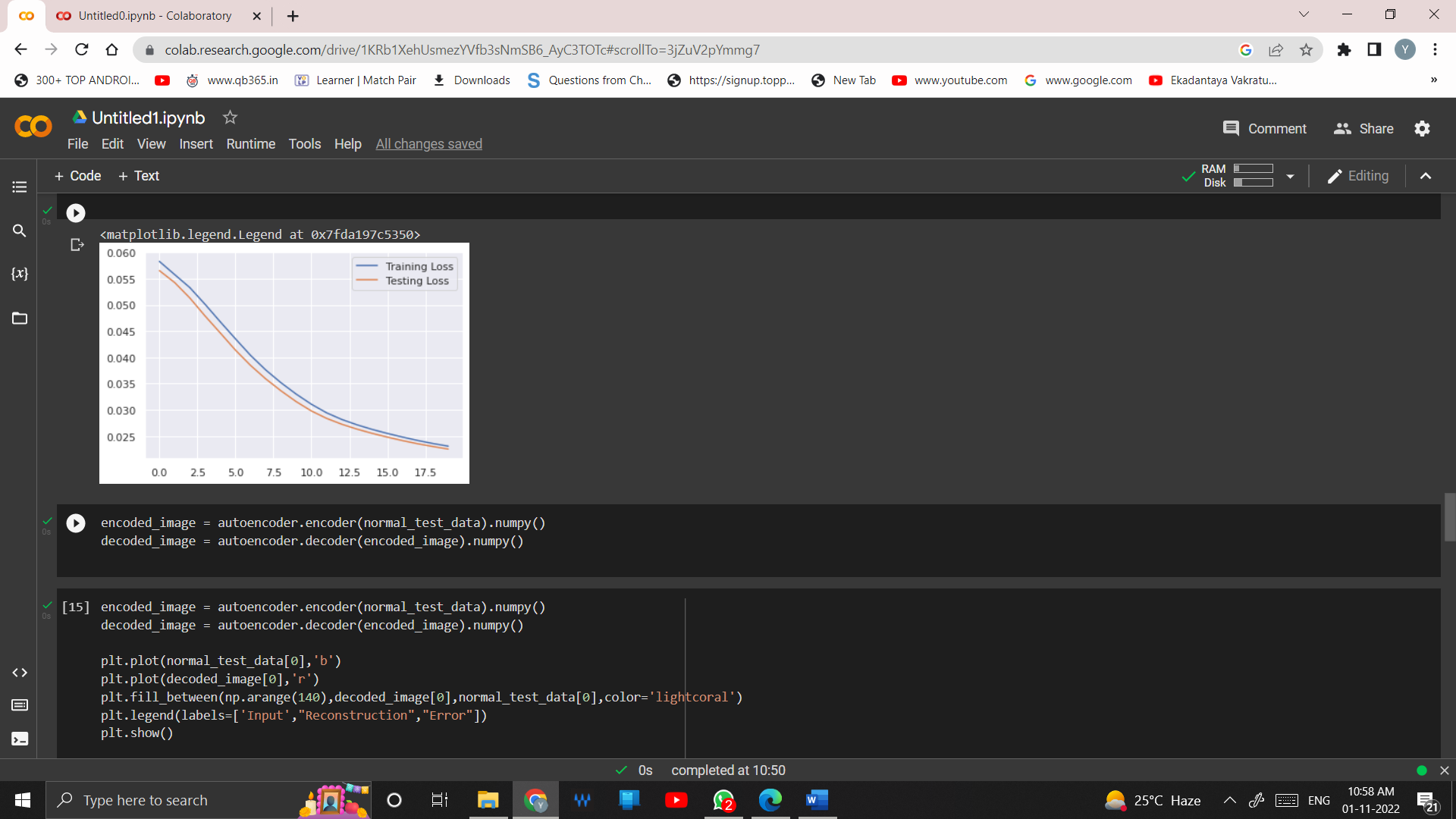
shuffle=True)

Plotting the difference between loss and validation loss of data

plt.plot(history.history['loss'],label='Training Loss')

plt.plot(history.history['val\_loss'],label='Testing Loss')

plt.legend()



**Plotting the difference between input and reconstruction and error or normal test data set**

encoded\_image = autoencoder.encoder(normal\_test\_data).numpy()

decoded\_image = autoencoder.decoder(encoded\_image).numpy()

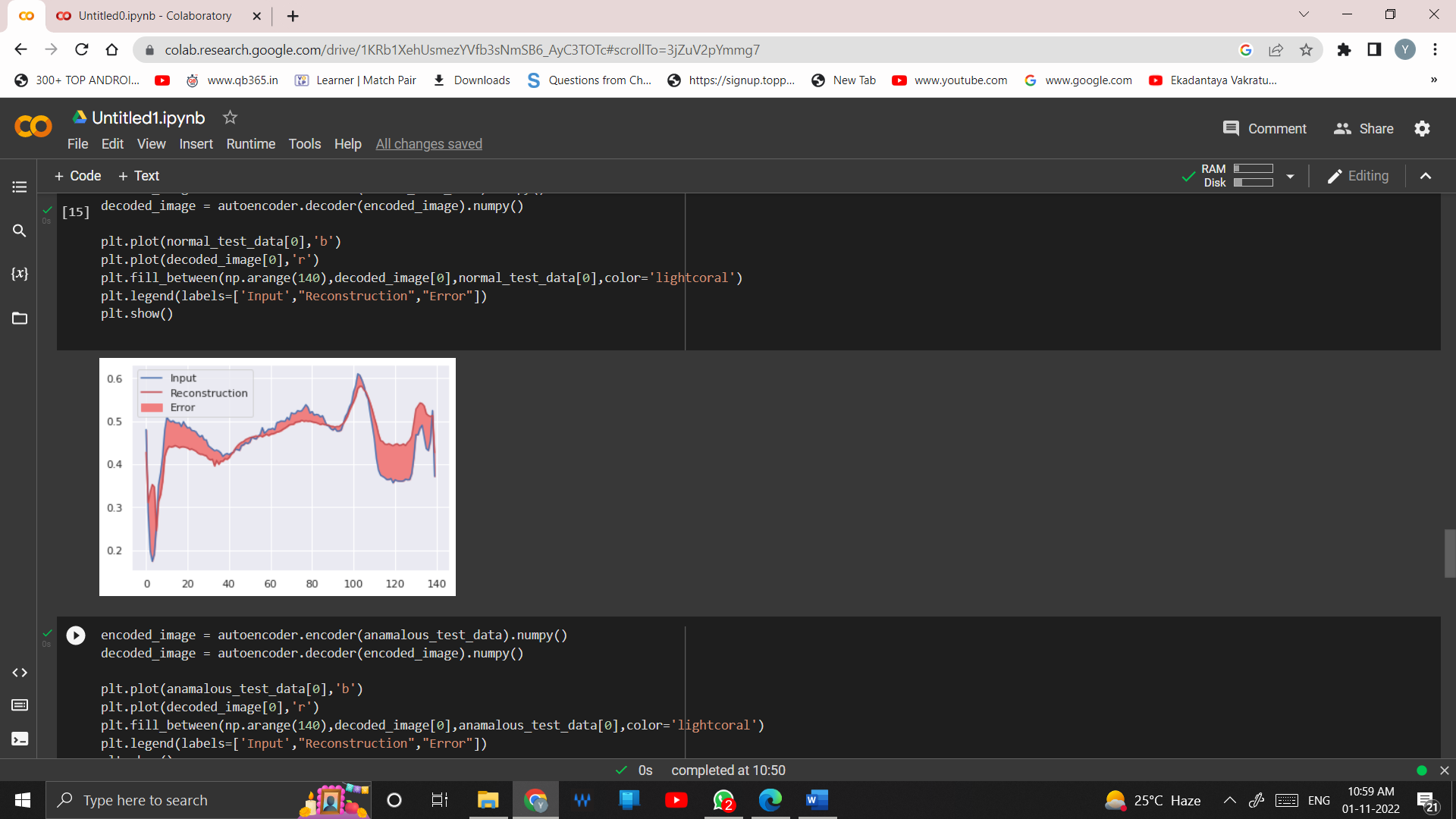
plt.plot(normal\_test\_data[0],'b')

plt.plot(decoded\_image[0],'r')

plt.fill\_between(np.arange(140),decoded\_image[0],normal\_test\_data[0],color='lightcoral')

plt.legend(labels=['Input',"Reconstruction","Error"])

plt.show()



**Plotting the difference between input and reconstruction and error or anomalous test data set**

encoded\_image = autoencoder.encoder(anamalous\_test\_data).numpy()

decoded\_image = autoencoder.decoder(encoded\_image).numpy()

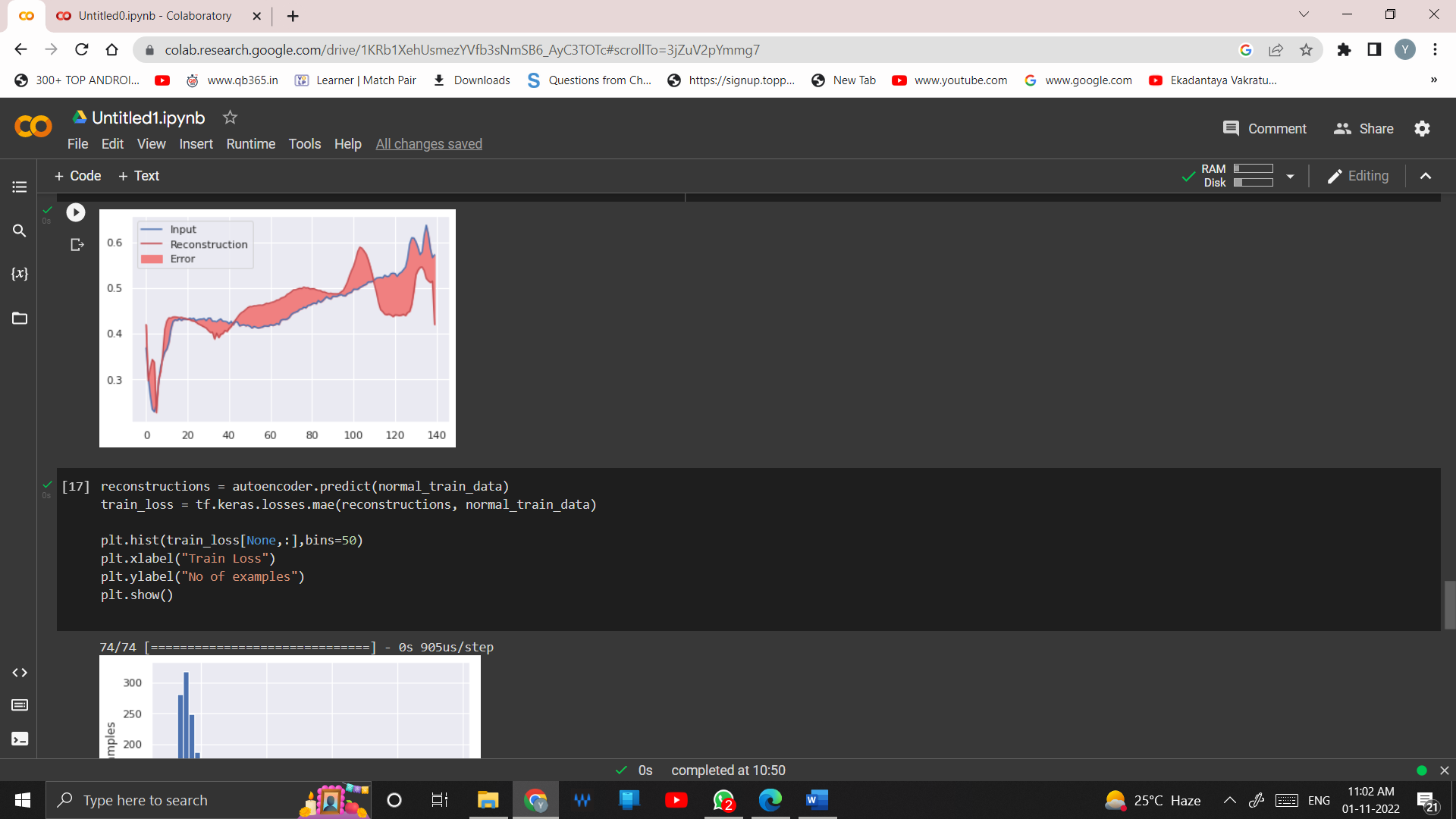
plt.plot(anamalous\_test\_data[0],'b')

plt.plot(decoded\_image[0],'r')

plt.fill\_between(np.arange(140),decoded\_image[0],anamalous\_test\_data[0],color='lightcoral')

plt.legend(labels=['Input',"Reconstruction","Error"])

plt.show()



**Plotting histogram prediction for normal test data**

reconstructions = autoencoder.predict(normal\_train\_data)

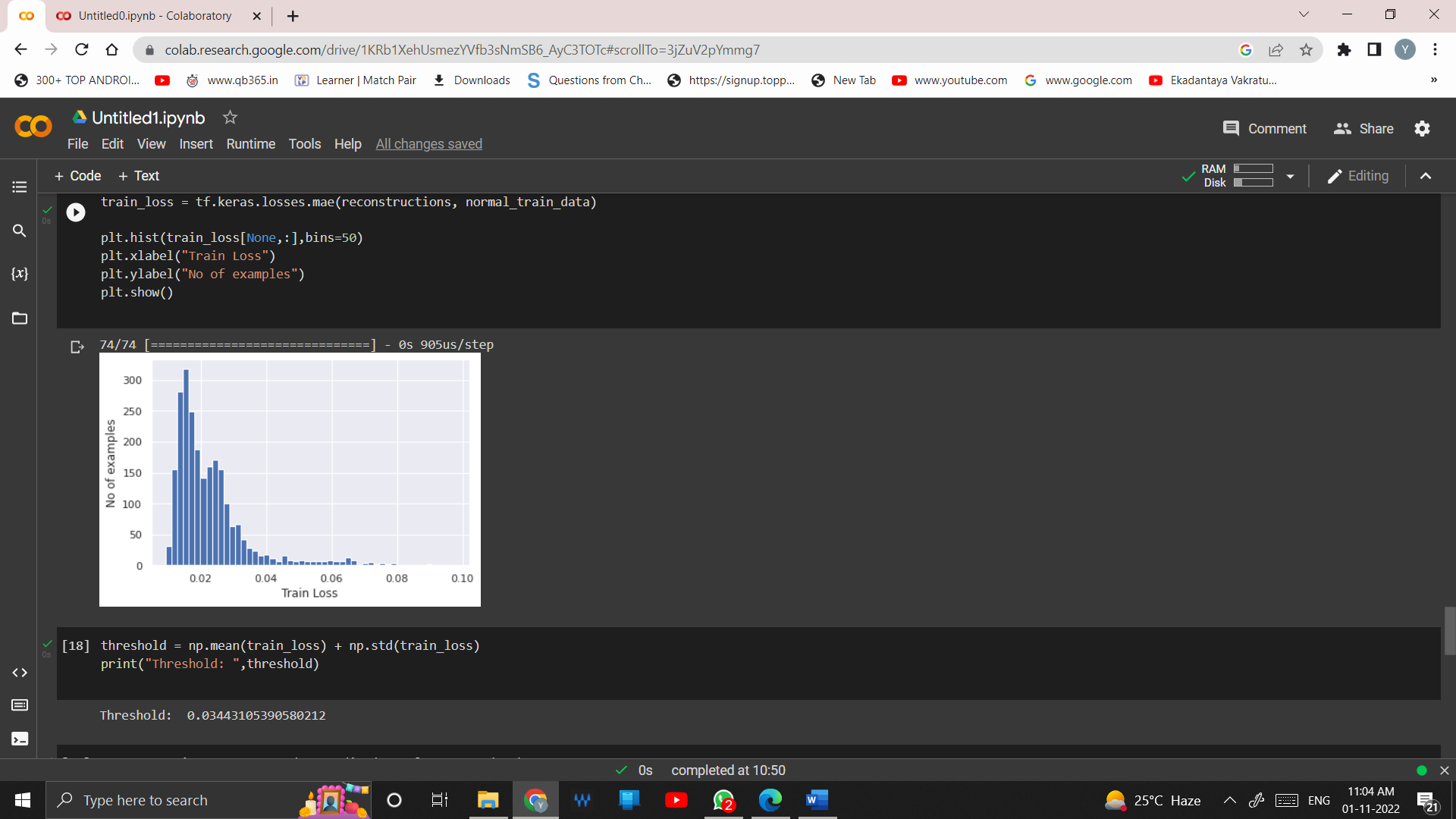
train\_loss = tf.keras.losses.mae(reconstructions, normal\_train\_data)

plt.hist(train\_loss[None,:],bins=50)

plt.xlabel("Train Loss")

plt.ylabel("No of examples")

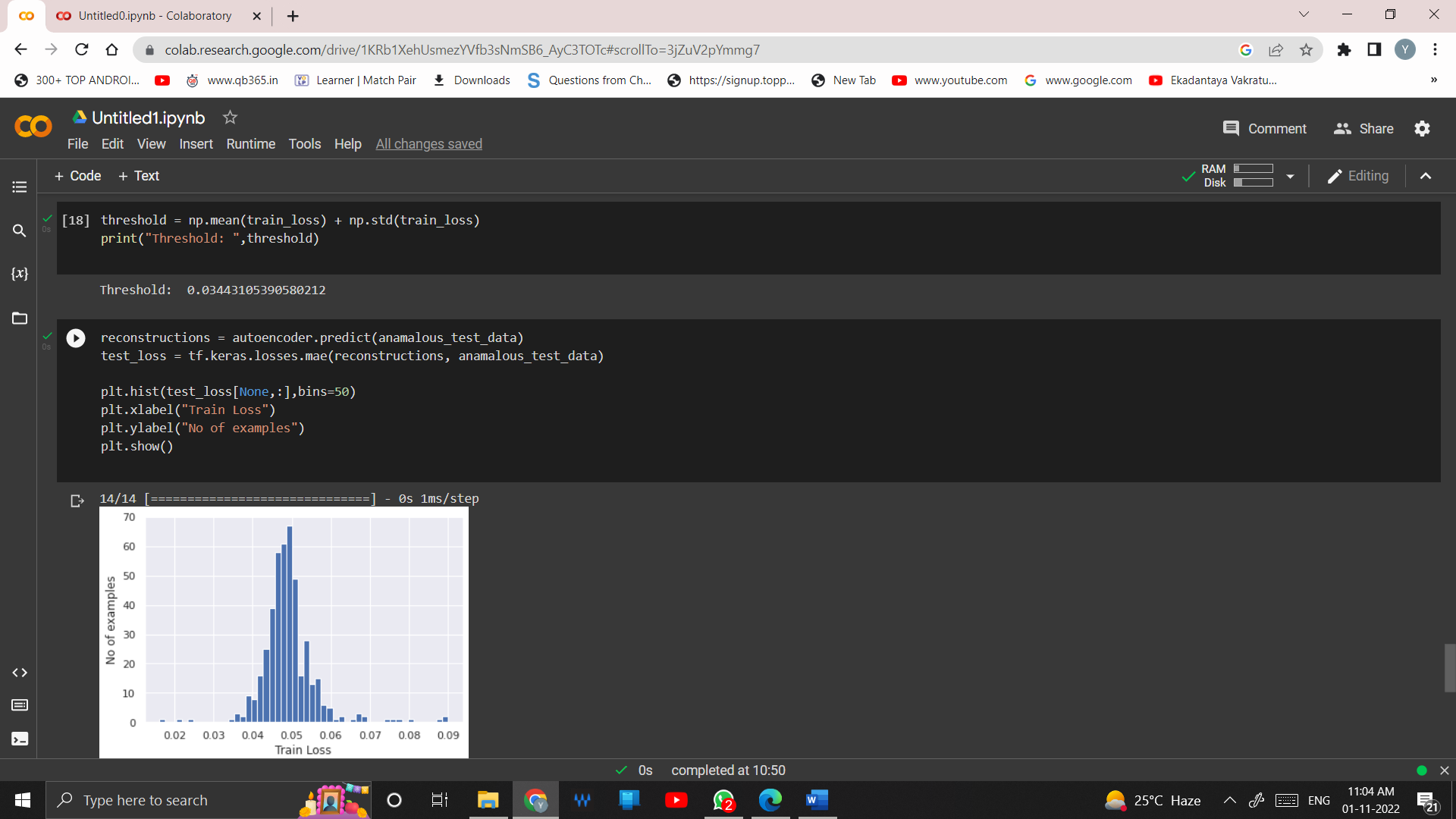
plt.show()



**Threshold value**

threshold = np.mean(train\_loss) + np.std(train\_loss)

print("Threshold: ",threshold)



Plotting histogram for prediction of anomalous test data

reconstructions = autoencoder.predict(anamalous\_test\_data)

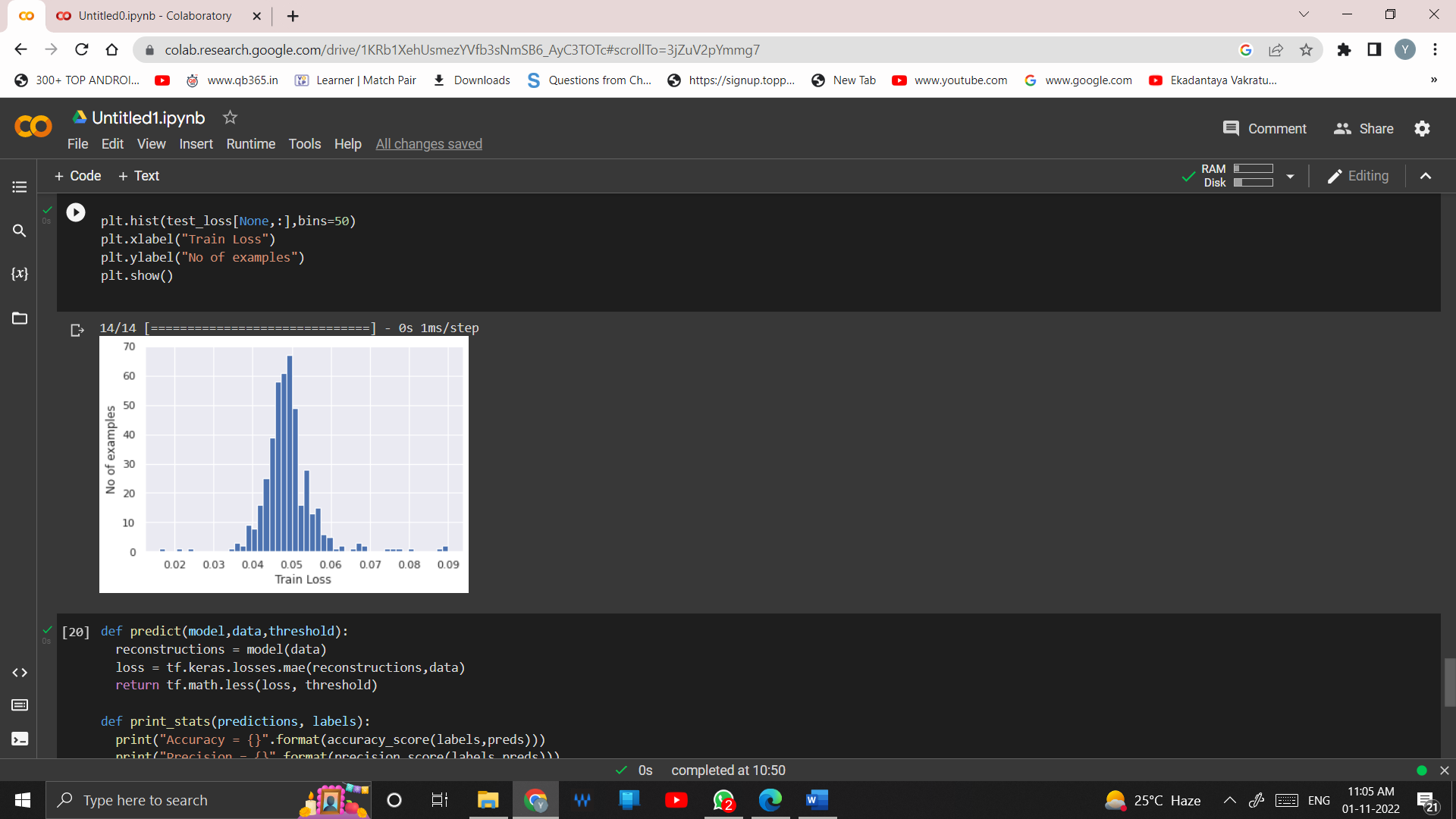
test\_loss = tf.keras.losses.mae(reconstructions, anamalous\_test\_data)

plt.hist(test\_loss[None,:],bins=50)

plt.xlabel("Train Loss")

plt.ylabel("No of examples")

plt.show()



**Predicting the Accuracy**

def predict(model,data,threshold):

reconstructions = model(data)

loss = tf.keras.losses.mae(reconstructions,data)

return tf.math.less(loss, threshold)

def print\_stats(predictions, labels):

print("Accuracy = {}".format(accuracy\_score(labels,preds)))

print("Precision = {}".format(precision\_score(labels,preds)))

print("Recall = {}".format(recall\_score(labels,preds)))

preds = predict(autoencoder, test\_data, threshold)

print\_stats(preds, test\_labels)

