

Handwritten digit recognition in python using scikit-learn and Random Forest Classifier

By

A. Raghul (roll no:39) Reg.no:11805979

K. Saivivek Reddy(roll.no:37) Reg.no:11805765

R. Hrithiksai (roll.no:34) Reg.no:11805273

Github link of the project:

https://github.com/raghuln26/Hand-writtten-digit-recognition-using-random-forest-/blob/main/Untitled4.ipynb

**Introduction:**

Our project is to identify the hand written digits recognition using python in Scikit-learn. In this report we propose a fundamental work to contemplate the Irregular Forest system in a businesslike manner, by taking a specialist perspective. O ur point isn't to look for best inborn execution but instead to investigate the worldwide conduct of this group of techniques concerning their boundary settings. For that reason we have examined one variation of RF called Forest-RI(Random Forest)

**Scikit learn :**

[Scikit-learn](http://scikit-learn.org/stable/) is an open source Python library that implements a range of machine learning, pre-processing, cross-validation and visualization algorithms using a unified interface.

**Important features of scikit-learn:**

* Simple and efficient tools for data mining and data analysis. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means, etc.
* Accessible to everybody and reusable in various contexts.
* Built on the top of NumPy, SciPy, and matplotlib.
* Open source, commercially usable – BSD license.

**What is a classification?**

Classification is defined as the process of predicting class or category from the observed values or given points.

Mathematically classification is the work of approximating a mapping function (f) from the input variable(x) to output variable(Y). it is basically belong to be supervised machine learning in which target are also provided along with the input data set.

**Type of learners in classification:**

1. Lazy learners
2. Eager learners
3. **Lazy learners**:

The lazy learners waits for the testing data to be appeared after storing the training data. Classification is done onlt after getting the testing data. They spend less time on training but more time on predicting.

**Eg**: k-nearest neighbor

1. **Eager learners**

The Eager learner construct ac classification model without waiting for the testing data to be appeared after storing the training data. They spend more time on training but less time on predicting.

**E.g:**  Decision tree, Navie Bayes, ANN

**There are 7 types of Classification Algorithms**

1. Logistic Regression
2. Naïve Bayes
3. Stochastic Gradient Descent
4. K-Nearest Neighbours
5. Decision Tree
6. Random Forest
7. Support Vector Machine

We have used **Random** **Forest classification algorithm** for the classification of **MNIST (“Modified National Institute of Standards and Technology”)** is dataset of computer vision and this dataset of handwritten images used as the basis for benchmarking classification algorithms.

**Random Forest :**

Random forest is a supervised learning algorithm which is used for both classification as well as regression. But however, it is mainly used for classification problems. As we know that a forest is made up of trees and more trees means more robust forest. Similarly, random forest algorithm creates decision trees on data samples and then gets the prediction from each of them and finally selects the best solution by means of voting. It is an ensemble method which is better than a single decision tree because it reduces the over-fitting by averaging the result.

Random forests are an ensemble learning method for classification operate by constructing a multitude of decision trees at training time and outputting the class of the individual trees. Random decision forests correct for decision tree’s habit of overfitting to their training set.

Building a classifier :

1. Importing the necessary packages
2. Importing the dataset
3. Organizing data into training and testing data
4. Model evalution
5. Finding the accuracy

Code walk through and Explanation:

1. Importing the necessary Packages

**import** **numpy** **as** **np**

**import** **pandas** **as** **pd** **#------------------>used to load the dataset**

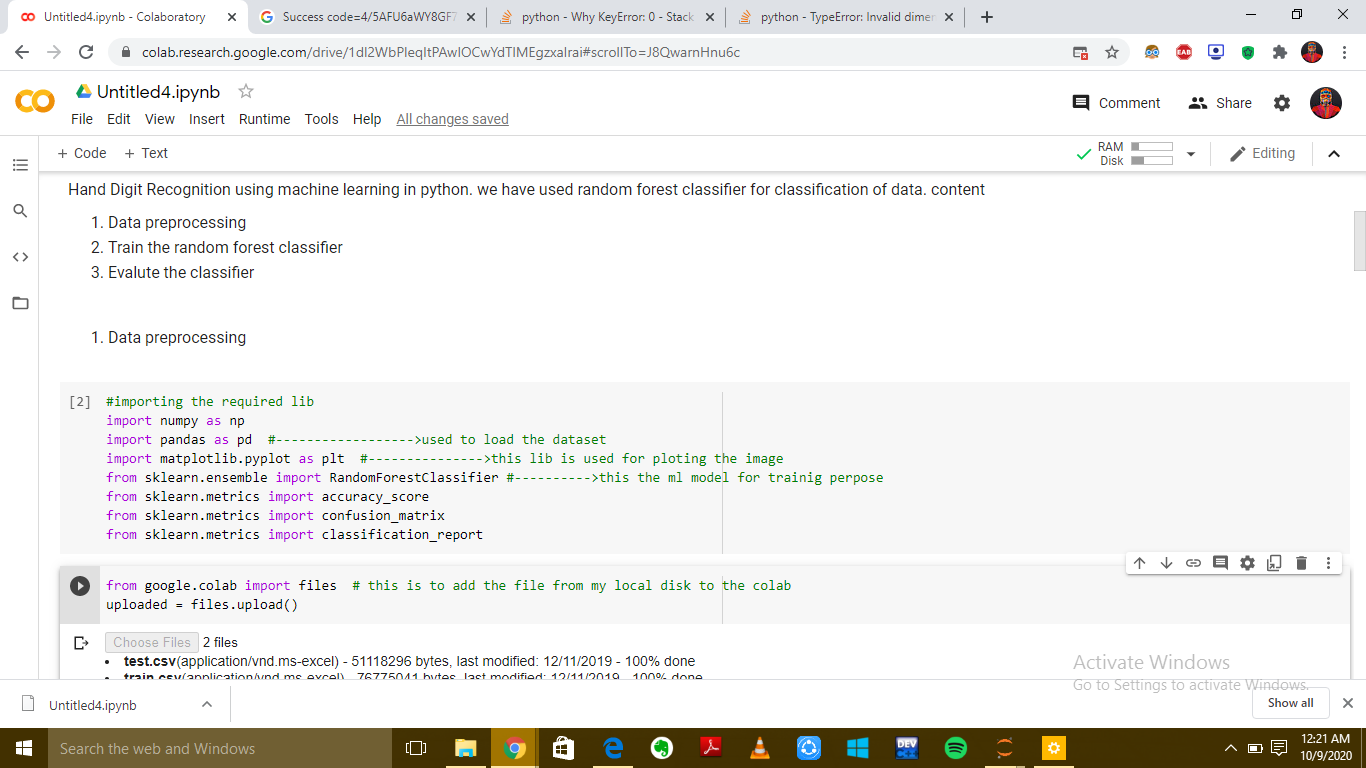
**import** **matplotlib.pyplot** **as** **plt** ***#------------->this lib is used for ploting the image***

**from** **sklearn.ensemble** **import** RandomForestClassifier **#---------->this the ml model for trainig perpose**

**from** **sklearn.metrics** **import** accuracy\_score

**from** **sklearn.metrics** **import** confusion\_matrix

**from** **sklearn.metrics** **import** classification\_report

screenshot of the code

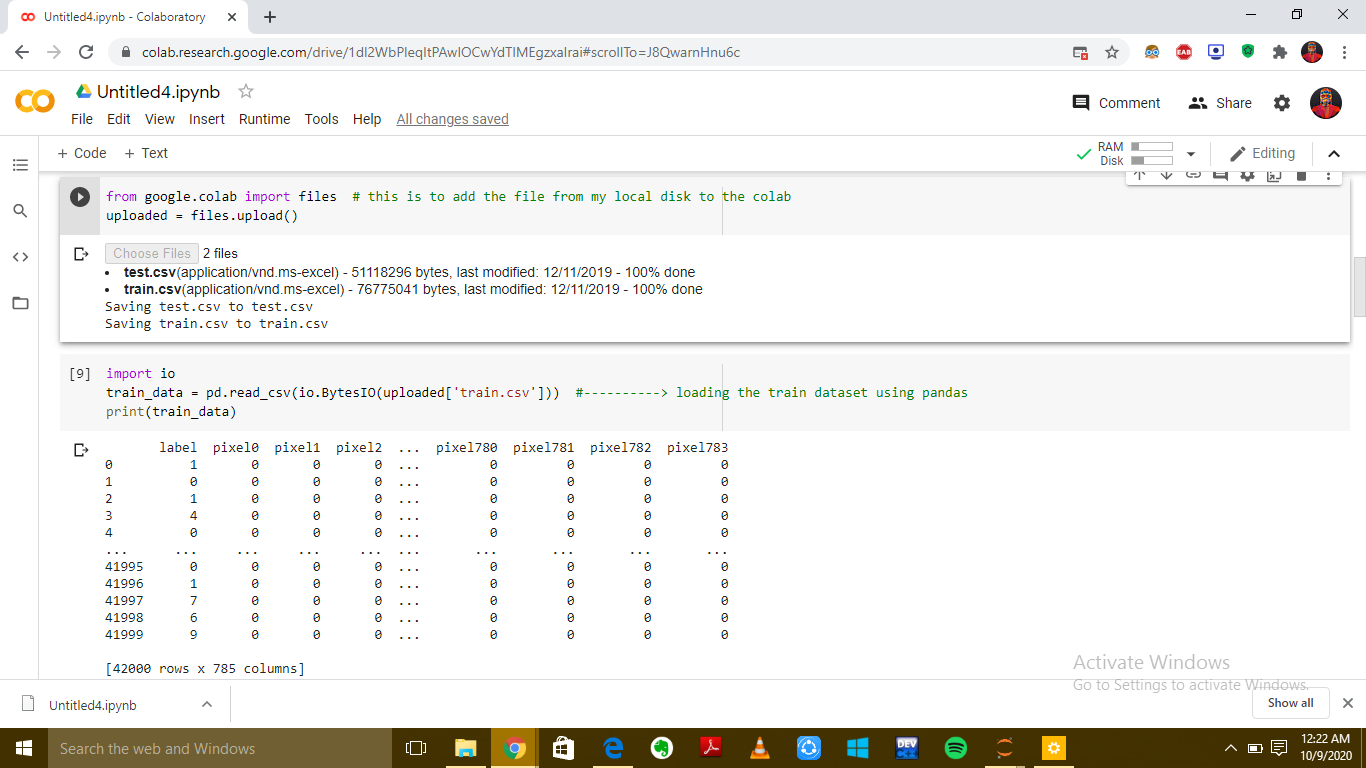
1. Importing the dataset:

* After importing all the necessary package we need a dataset to build classification prediction model. We are using MNIST data set each image is a 28 by 28 pixel square (784 pixel total). Training dataset contains 42000 entries and test dataset contains 28000 entries. Since it is a digit recognition task, it has 10 classes to predict.
* Loading the dataset :
* We using pandas for loading a dataset

**from** **google.colab** **import** files ***# this is to add the file from my local disk to the colab***

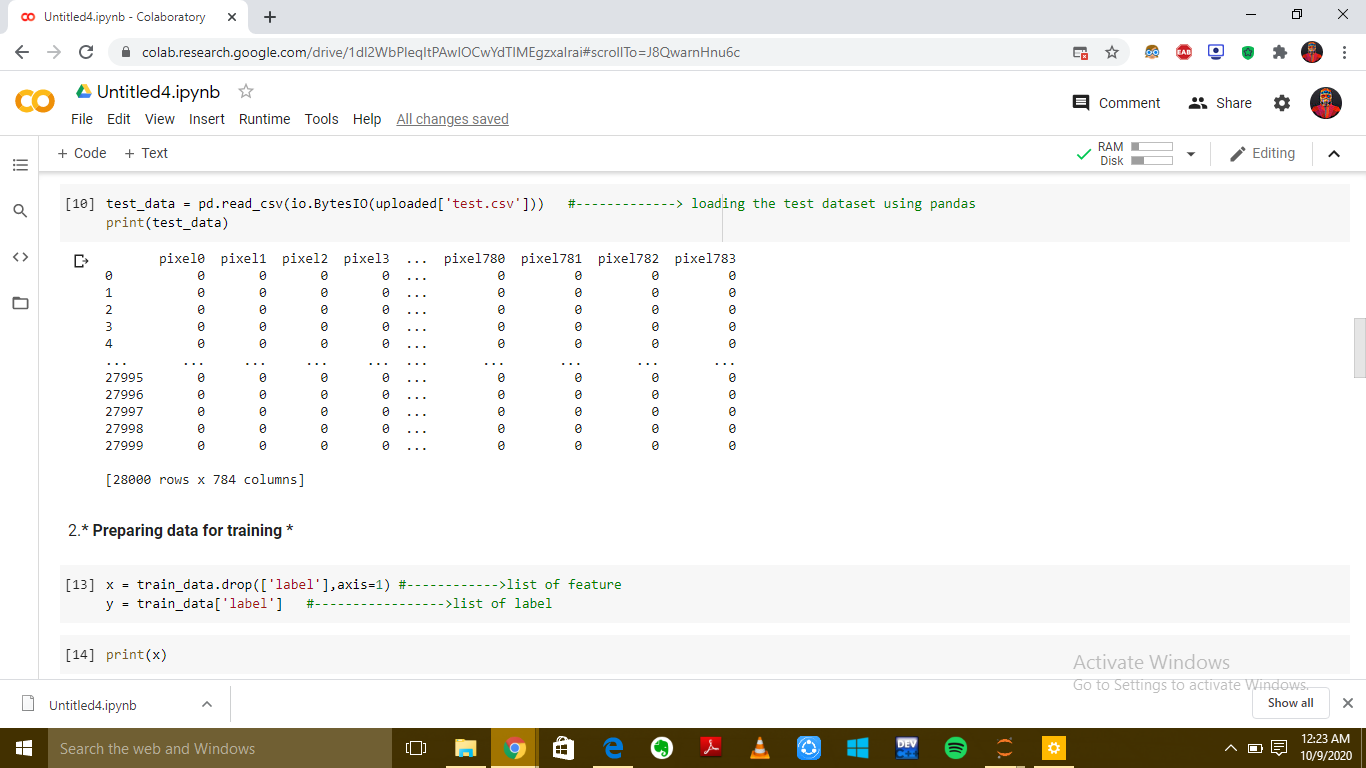
uploaded = files.upload()

**import** **io**

train\_data = pd.read\_csv(io.BytesIO(uploaded['train.csv'**])) *#----------> loading the train dataset using pandas***

print(train\_data)

**and the necessary out is added in the screen shot in this**



**this it the screen shot of the test dataset**

***Preparing data for training***

In [13]:

x = train\_data.drop(['label'],axis=1) *#------------>list of feature*

y = train\_data['label'] *#----------------->list of label*

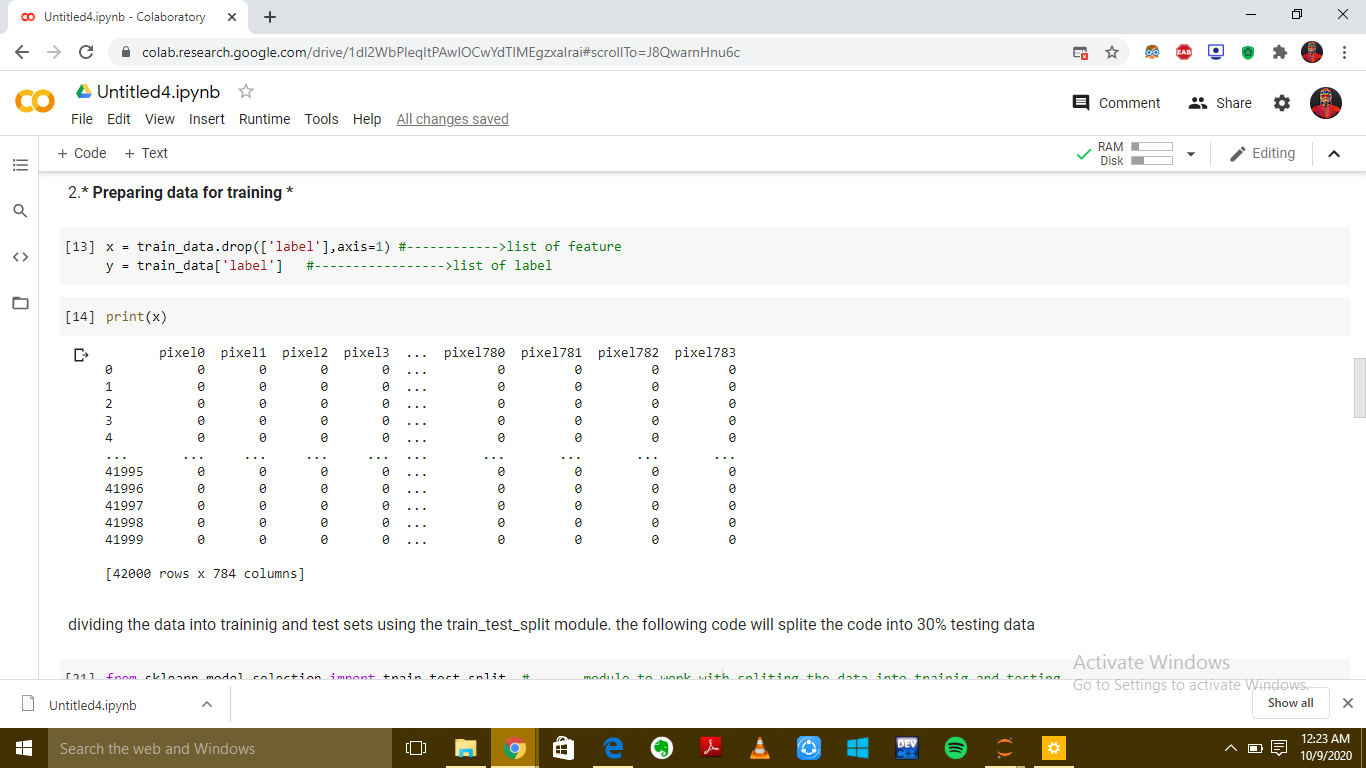
In [14]:

print(x)

A dataset is nothing but a collection of data. A dataset generally has two main components:

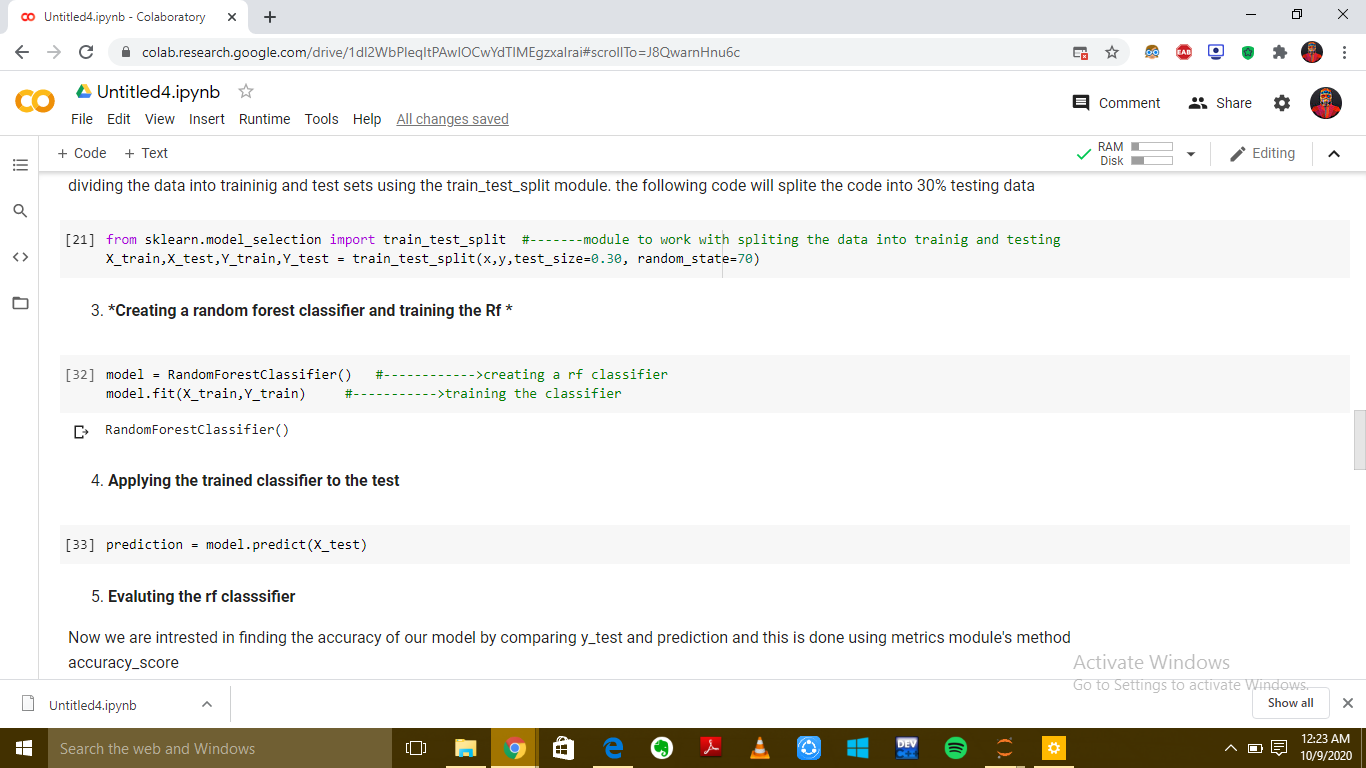
* **Features**: (also known as predictors, inputs, or attributes) they are simply the variables of our data. They can be more than one and hence represented by a **feature matrix** (‘X’ is a common notation to represent feature matrix). A list of all the feature names is termed as **feature names**.
* **Response**: (also known as the target, label, or output) This is the output variable depending on the feature variables. We generally have a single response column and it is represented by a **response vector** (‘y’ is a common notation to represent response vector). All the possible values taken by a response vector is termed as **target names**.

Here we are adding the output of the X after droping the label column



1. **Organizing data into training and testing data**

* Next step we are going to divide the data into train and test split .
* The code with separate the dataset into 30% training set For splitting we have used Train\_test\_split module from scikit-learn
* And syntax
* Sklearn.model\_selection.Train\_test\_split(array,test\_size,random\_state,options)
* Here ,Array- is sequence of input of same length
* Allowed inputs are lists, numpy, array etc,
* Test\_size – it represent the absolute number of test samples
* Test\_size can be float or int default = none if the test\_size is float then it should me between 0.0 to 1.0 it represent the part of the data to included in the test split
* Random\_state will return same result for each execution if the reandom\_state is null then it will returns different result for each execution.



**4.Model evalution**

Now we are goint to train the model with the help of random forest classifier class of sklearn. This two following code are the important lines

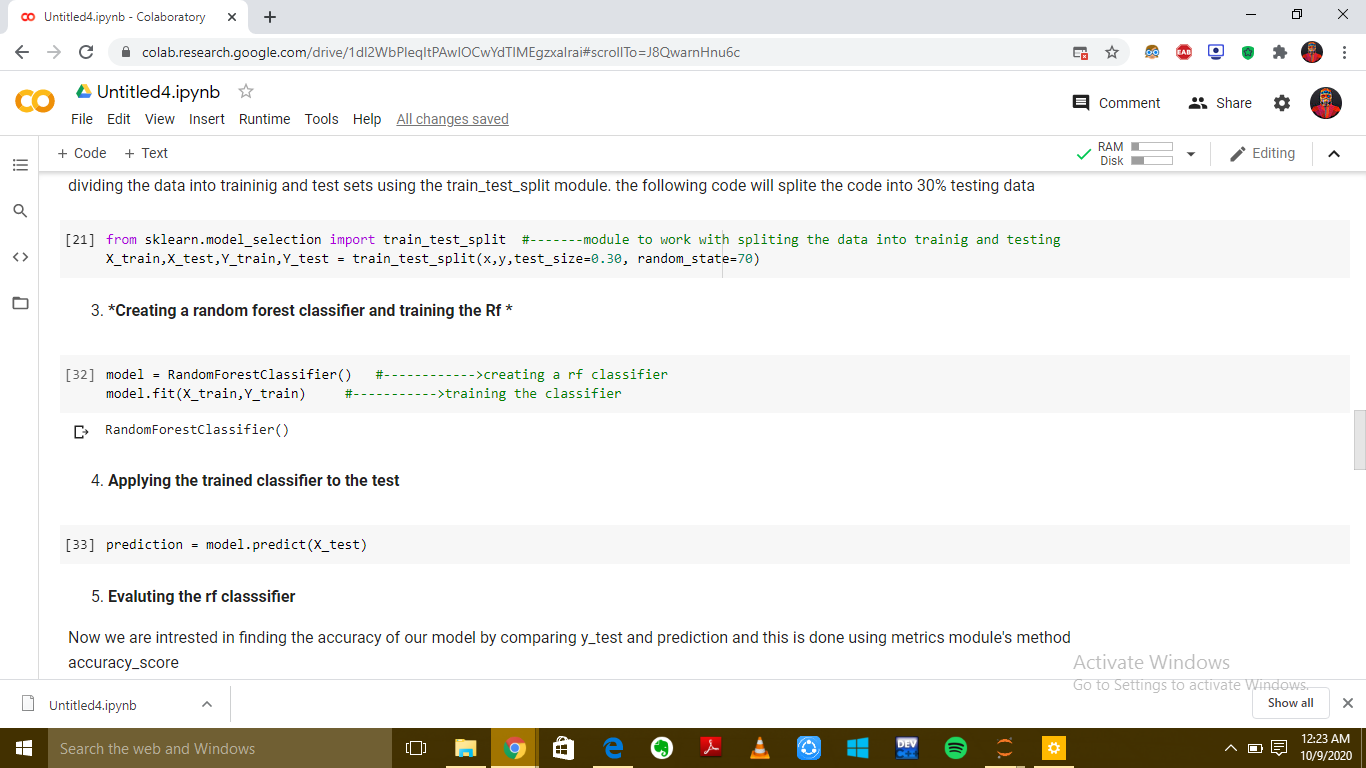
model = RandomForestClassifier() *#------------>creating a rf classifier*

model.fit(X\_train,Y\_train) *#----------->training the classifier*

* + we are creating a variable called model and passing the random classifier model
  + the classifier is trained using X\_train data the process is called fitting. We pass the feature matrix and the corresponding responsr vector.
  + Model.fit(X\_train,Y\_train)
* model.fit ->we are fitting/we are creating a random forest classifier

**Testing the trained classifier:**

* Now we need to test our classifier on the x\_test data. Model.predict method isused for this purpose. It returns the predicted response vector prediction.
* Prediction = Model.predict(x\_test)



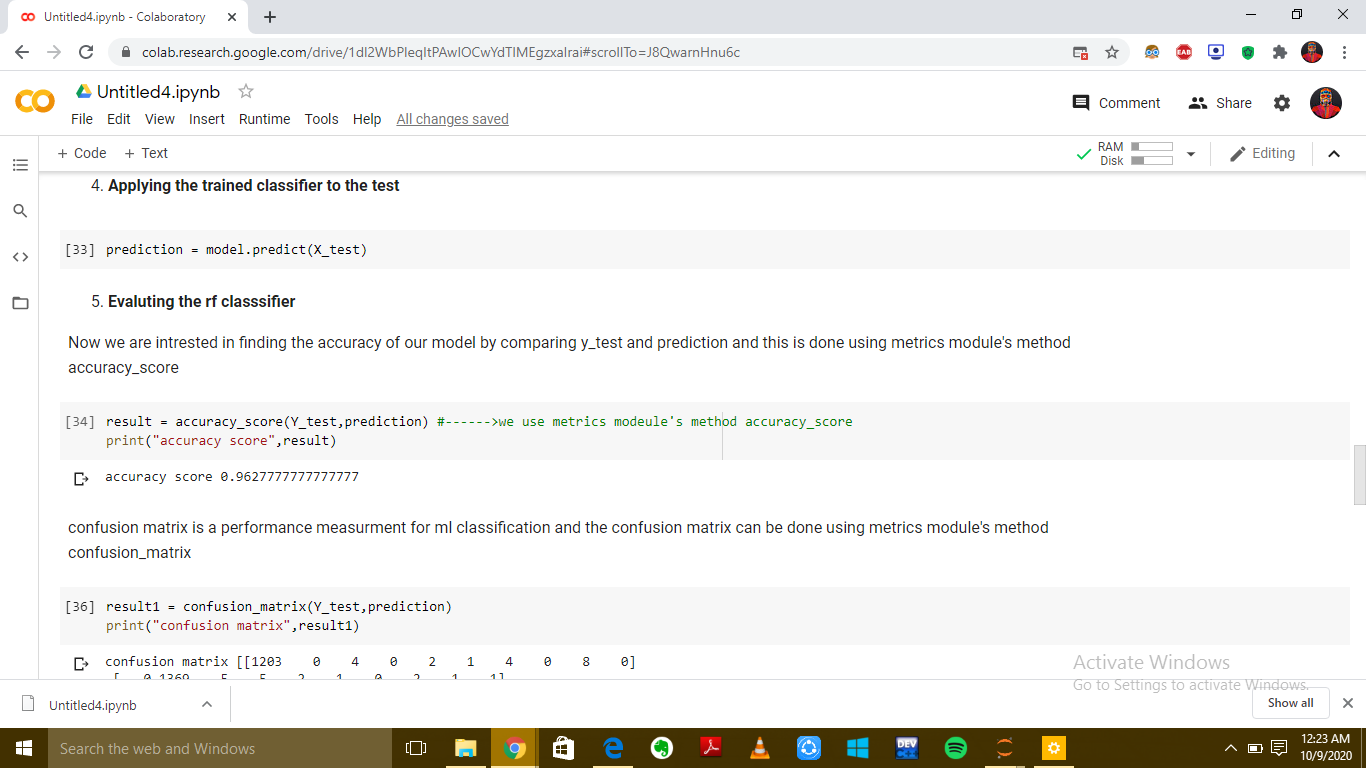
1. **Finding the accuracy**

* we are finding the accuracy of our model by comparing the prediction and Y\_test.
* And we can find the accuracy by using the module’s method accuracy\_score.

result = accuracy\_score(Y\_test,prediction) *#------>we use metrics modeule's method accuracy\_score*

print("accuracy score",result)

* and we **got 96% accuracy from our model.**

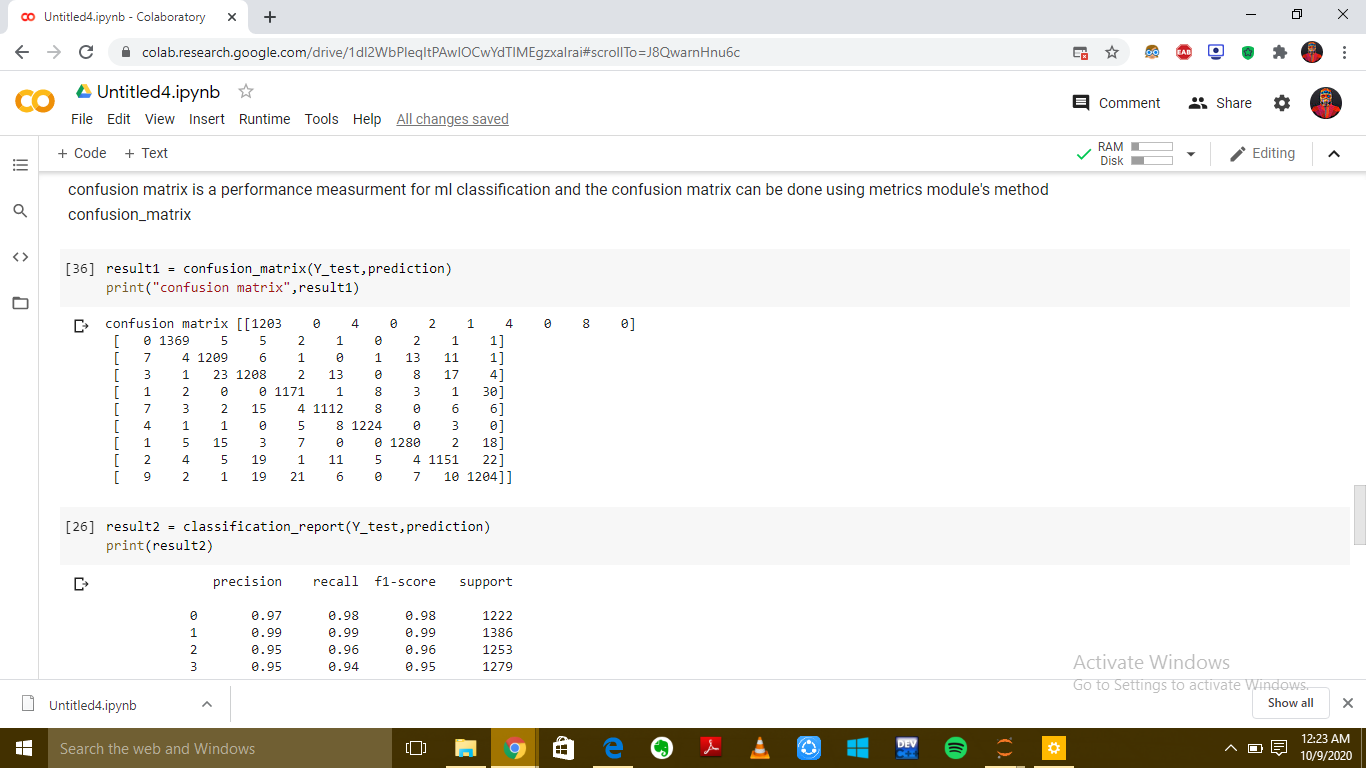
****

**6.a confusion matrix**

* confusion matrix is a performance measurment for ml classification and the confusion matrix can be done using metrics module's method confusion\_matrix

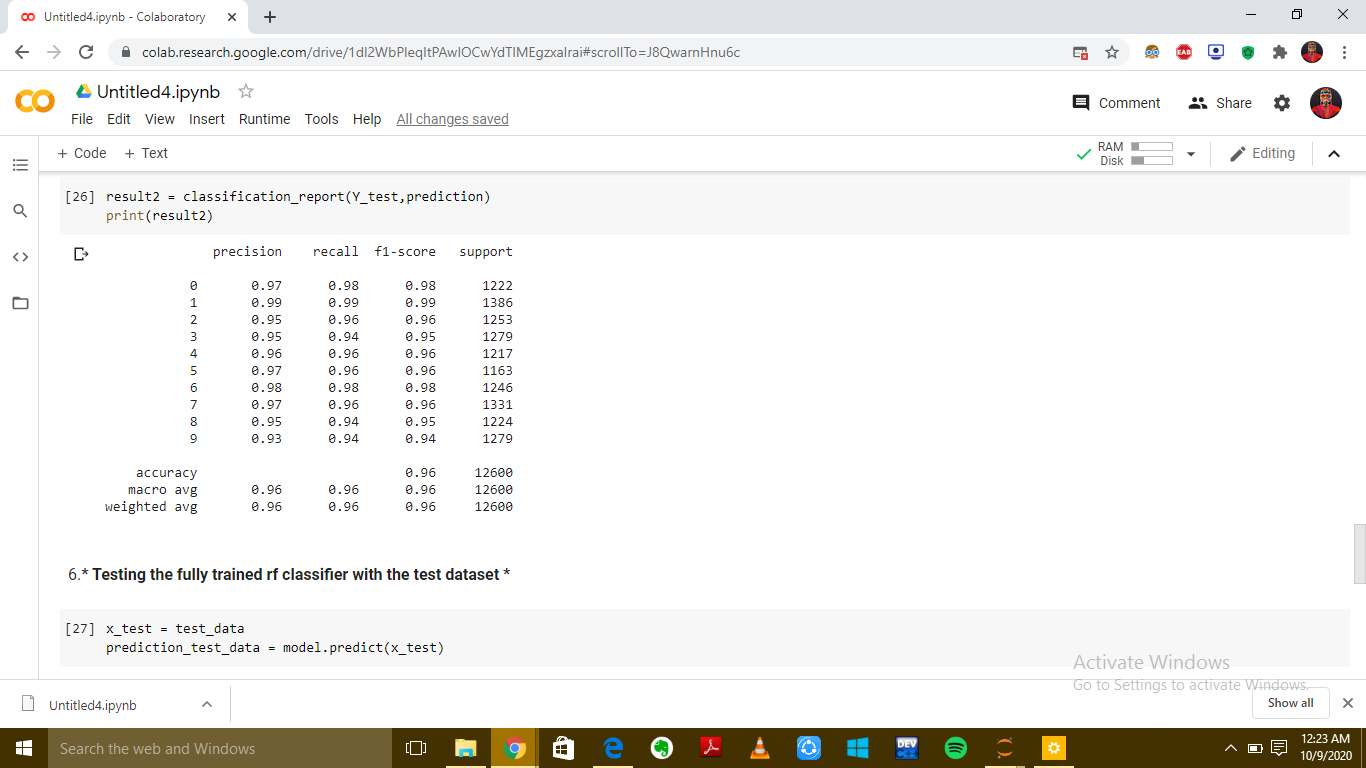
result1 = confusion\_matrix(Y\_test,prediction)

print("confusion matrix",result1)



6.b. **classification report:**

* The classification report visualizer displays the precision, recall, F1, and support scores for the model.
* We can done by using matrix modules classification\_report
* result2 = classification\_report(Y\_test,prediction)
* print(result2)



6**.c.Testing the fully trained rf classifier with the test dataset**

x\_test = test\_data

prediction\_test\_data = model.predict(x\_test)

the following code is used to convert the csv row into image

index\_data = 9

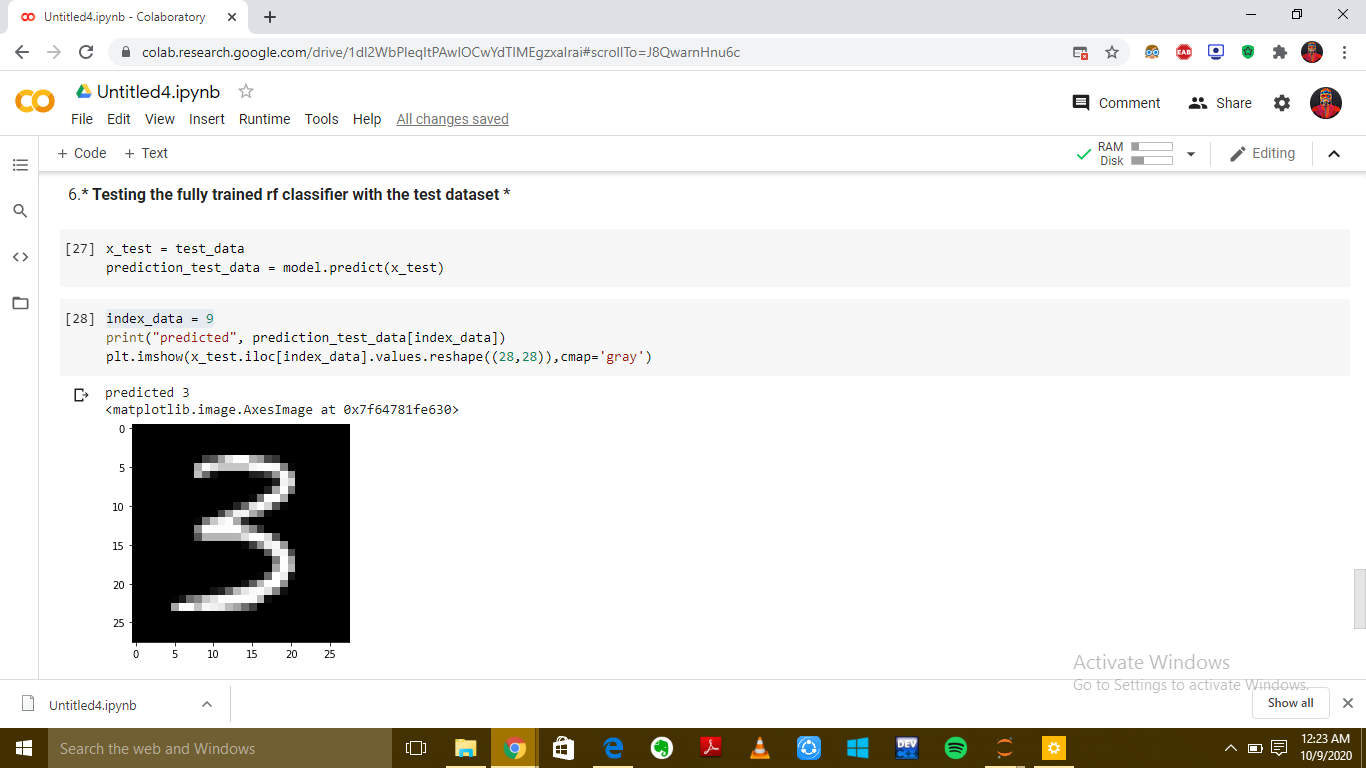
print("predicted", prediction\_test\_data[index\_data])

we have used imshow() from mateplotlib after reshaping the data in 3d with the function values.reshape()

plt.imshow(x\_test.iloc[index\_data].values.reshape((28,28)),cmap='gray')--------🡪

imshow() function is used to plot the numpy as the map , cmap() function is used to colormap for the existing plot object

* Now we need to test our classifier on the x\_test data. Model.predict method isused for this purpose. It returns the predicted response vector prediction.
* Imshow() is a function used to plot the numpy data
* If we have your data in a numpy array either by importing or by generating it.
* Cmap is used to colormap the existing plot object



THANK YOU