

ML-MAJOR-MAY

INTRODUCTION TO MACHINE LEARNING WITH PYTHON

MAJOR PROJECT SUBMISSION

PROJECT TITLE

**HANDWRITTEN DIGIT RECOGNITION USING SVM
ALGORITHM**

BATCH ID: ML05-MB09

Name: Ven Leonin

Email ID: nissyleo.007@gmail.com

Date of submission: 20-JULY-2021

Batch No.: 09

DIGIT CLASSIFICATION

Objective:

To develop a model to recognize the handwritten digits.

Introduction:

The MNIST dataset is the data which is used for training the Machine Learning models based on hand written digit classification problem. In the data set there are various numbers which have different pixels as they are the scanned copies of handwritten digits. There are 10 digits (0-9) which are of different pixels. The complexity in the above data set is that digits are in different styles for different writers so we use build a model to detect the digits. We train our model using Support Vector Machine (SVM) algorithm.

Problem Statement:

To create a model that detects the hand written digits from the MNIST dataset using SVM algorithm.

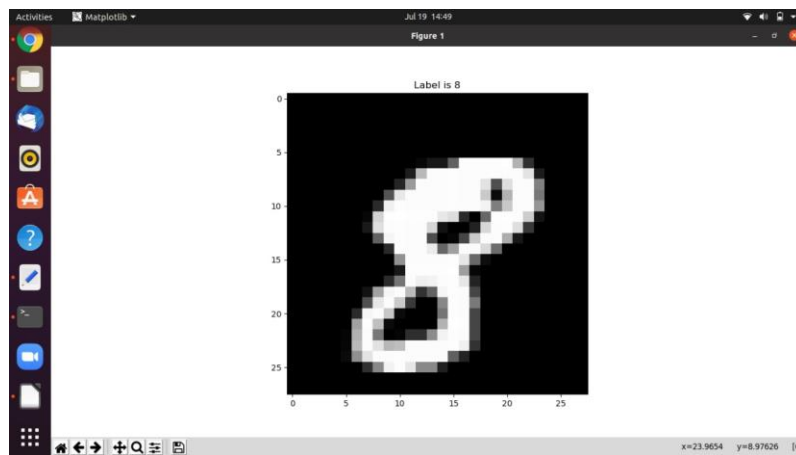
Data set:

The MNIST database is the large database of hand written digits that is commonly used for training various image processing systems and also used for training and testing in the field of machine learning. The MNIST data set stands for Modified National Institute of Standards and Technology dataset.

It is a data set of 60,000 small square 28 X 28 pixels grayscale images of handwritten single digits between 0 – 9.

Ex:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
df = pd.read_csv("/home/rishi/Downloads/digit_svm.csv")
pixel = df.iloc[10,1:].values
label = df.iloc[10,0]
pixel = pixel.reshape((28, 28))
plt.title('Label is {label}'.format(label=label))
plt.imshow(pixel, cmap='gray')
plt.show()
```



Link:

<https://drive.google.com/file/d/1PulBpS2m2X8M1Z9vqwZPCjBuR-ZxQ6j/view?usp=sharing>

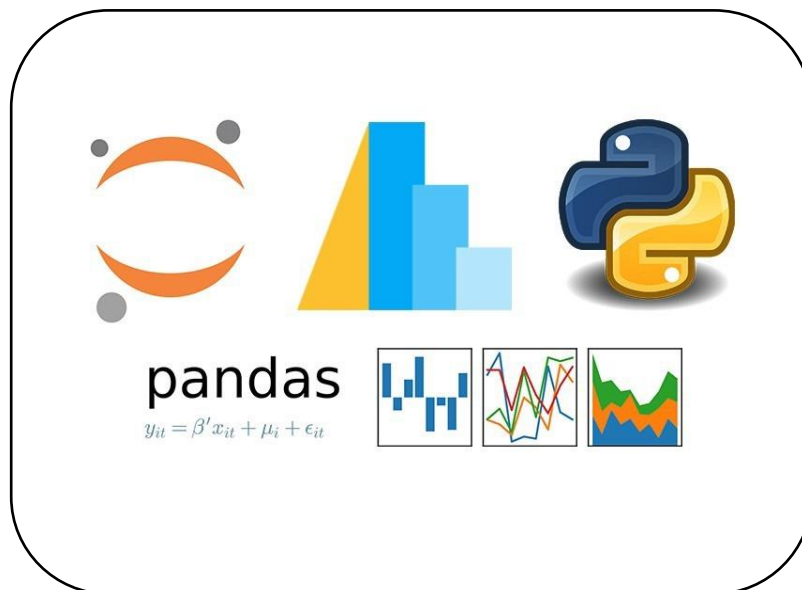
Libraries:

1. Pandas:

- Pandas is an open source Python package that is most widely used for Data Analysis and Machine Learning tasks.
- Pandas is used to read the data from the user and convert the dataset into data frames

Eg: Import pandas as pd:

```
Df=pd.read_csv("File name")
```



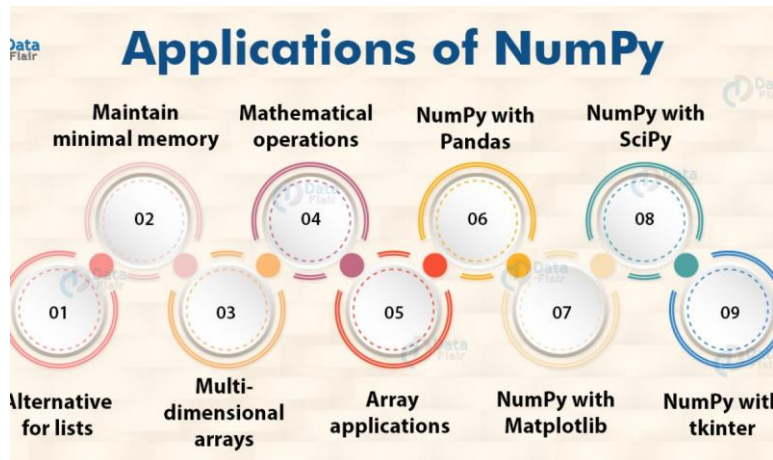
2. Numpy:

- Numpy is the fundamental package for scientific computing in python
- Numpy arrays facilitate advanced mathematical and other type of operations on large numbers of data

Eg: import numpy as np

```
X=df.iloc[:,1:].values
```

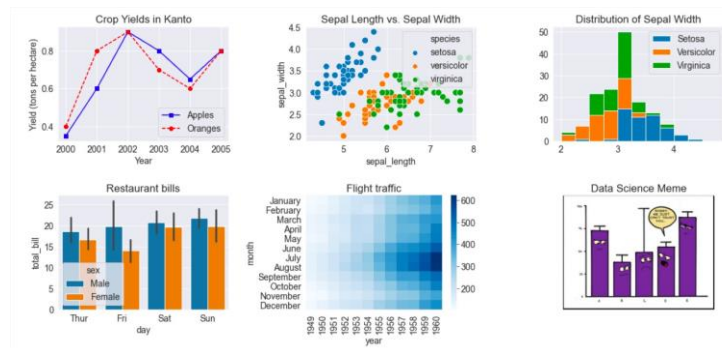
```
Y=df.iloc[:,0].values
```



Matplotlib:

Matplotlib is a collection of functions that make Matplotlib work like MATLAB. Each pyplot function makes some changes to a figure, creates a plotting area in a figure, plots some lines in a plotting area, decorates the plot with labels etc.

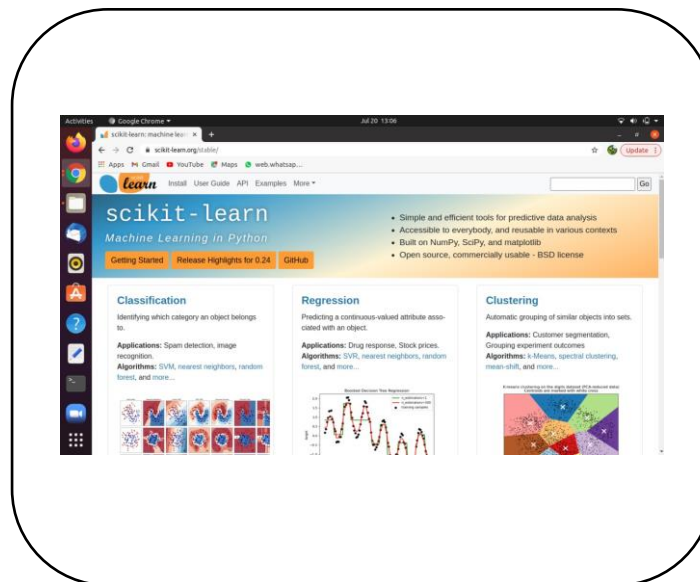
Ex: `import matplotlib.pyplot as plt`



Sklearn:

- Sklearn is probably the most usable library for machine learning in python.
- The Sklearn library contains a lot of efficient tools for machine learning and statistical modelling including classification, regression, clustering and dimensionality reduction.

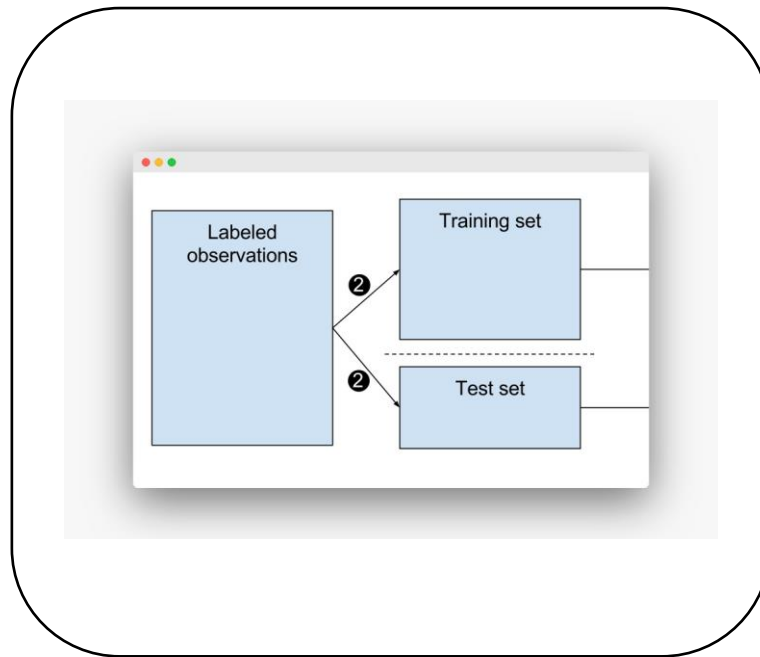
Ex: import Sklearn



Train_Test_Split:

- Train test Split is a function in Sklearn model selection for splitting data arrays into two subsets; for training data and testing data.
- By default Sklearn train test split will make random partitions for the two subsets however you can also specify a random state for the operation.

Ex: from Sklearn.model selection import train_test_split



Standard Scaler:

- Standard scaler follows standard normal distribution (Snd) therefore, it makes mean=0 and scales the data to unit variants.
- This method removes the median and scales the data in range between first quartile and third quartile.
- Standard scaler is the function in Sklearn preprocessing for scaling the input data.

Ex: `from Sklearn.preprocessing import StandardScaler()`

c

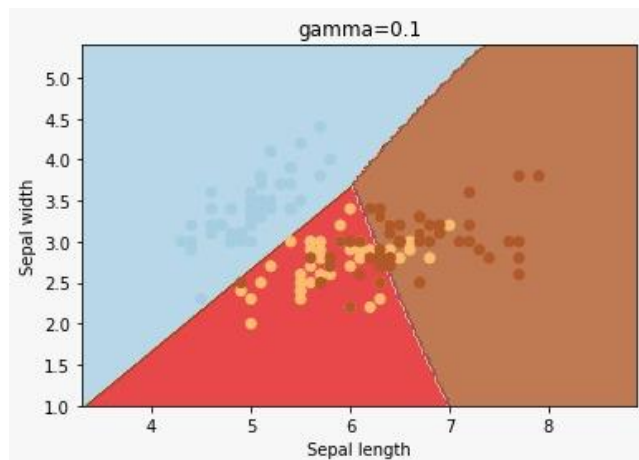
Standard Scaler	$\frac{x_i - \text{mean}(\mathbf{x})}{\text{stdev}(\mathbf{x})}$
MinMax Scaler	$\frac{x_i - \min(\mathbf{x})}{\max(\mathbf{x}) - \min(\mathbf{x})}$
Robust Scaler	$\frac{x_i - Q_1(\mathbf{x})}{Q_3(\mathbf{x}) - Q_1(\mathbf{x})}$

SVC:

- The objective of a linear SVC (Support Vector Classifier) is to fit to the data we provide returning a best fit hyperplane.
- That divides or categorizes your data. SVC is a classifier in the module of SVM (Support Vector Machine) from SKlearn library.

Ex: `from sklearn.svm import SVC`

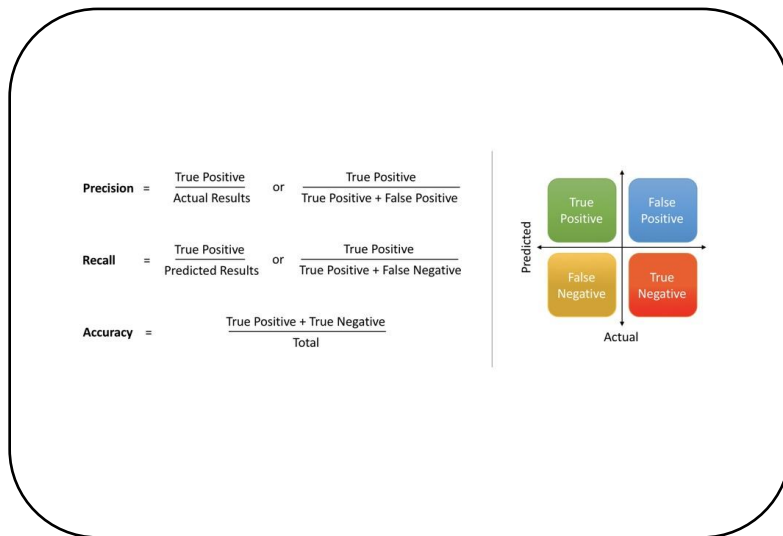
`Clf=SVC ()`



Accuracy Score:

Accuracy score is a function from Sklearn metrics. In multi label classification this function computes subset accuracy: the set of labels predicted (predy) for a sample must exactly match the corresponding set of labels in ytrue.

Ex: `from sklearn.metrics import accuracy score.`



Confusion matrix:

Confusion matrix is a function from sklearn metrics. In conclusion matrix is such that is equal to number of observations known to be in group but predicted to be in group.

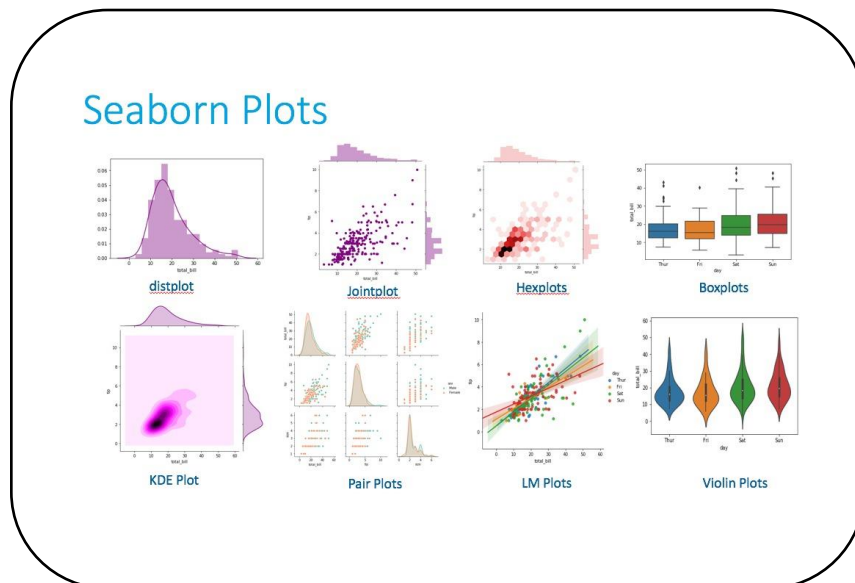
Ex: `from sklearn.metrics import conclusion_matrix`

		Predicted:		
		NO	YES	
Actual:	NO	TN = 50	FP = 10	60
	YES	FN = 5	TP = 100	105
		55	110	

Seaborn:

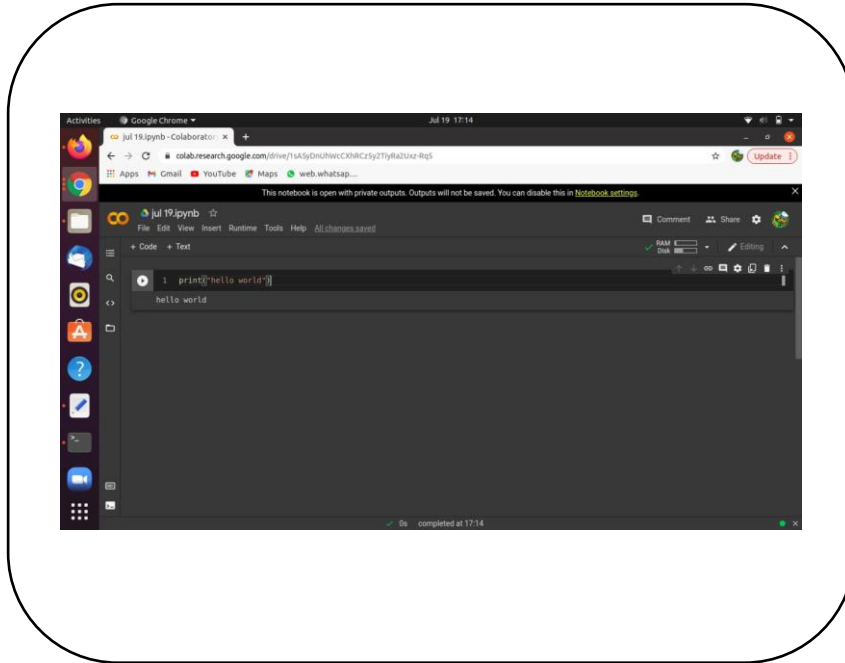
Seaborn is a data visualization library built on top of matplotlib and closely integrated with pandas data structures in python and mainly used for making statistical graphs. Visualization is the central part of seaborn which helps in exploration and understanding of data.

Ex: import seaborn as SNS



Platform:

Collabratory, or collab for short, is a product from google research. Collab allows anybody to write and execute arbitrary python code through the browser and especially well suited to machine learning, data analysis and education.

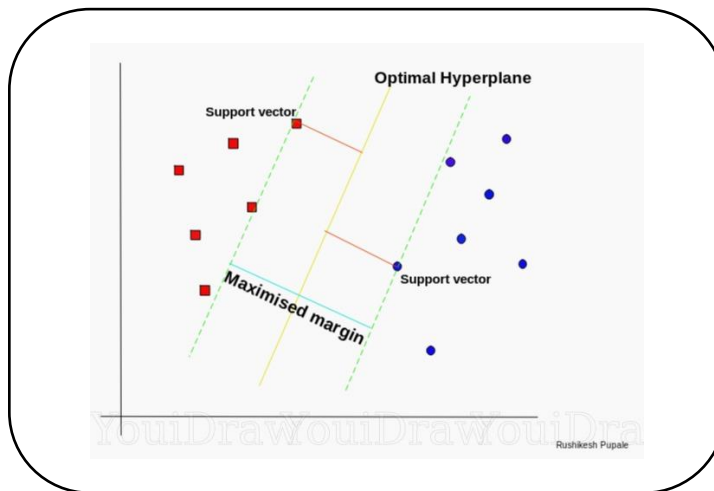


Support Vector Machine: (SVM)

A support vector machine is a supervised machine learning model that uses the classification algorithm for two group classification programs. After giving SVM model sets of labelled training data for each category they are able to categorize new text. Compared to newer algorithms like neural networks they have two main advantages:

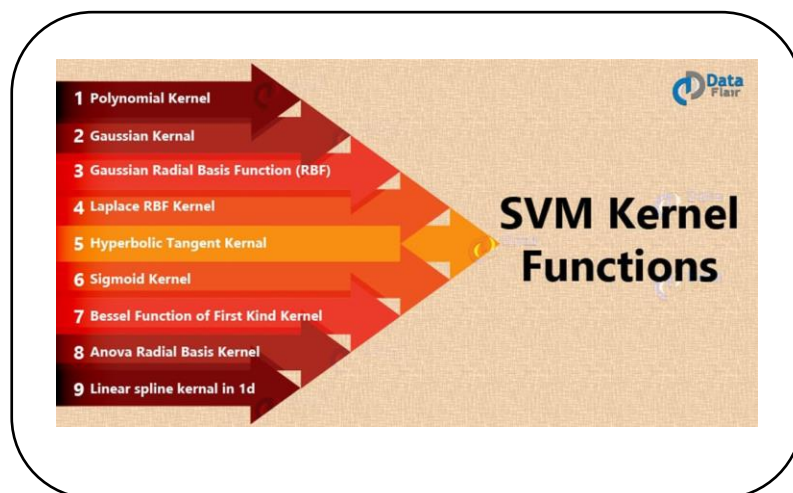
1. Higher Speed
2. Better Performance.

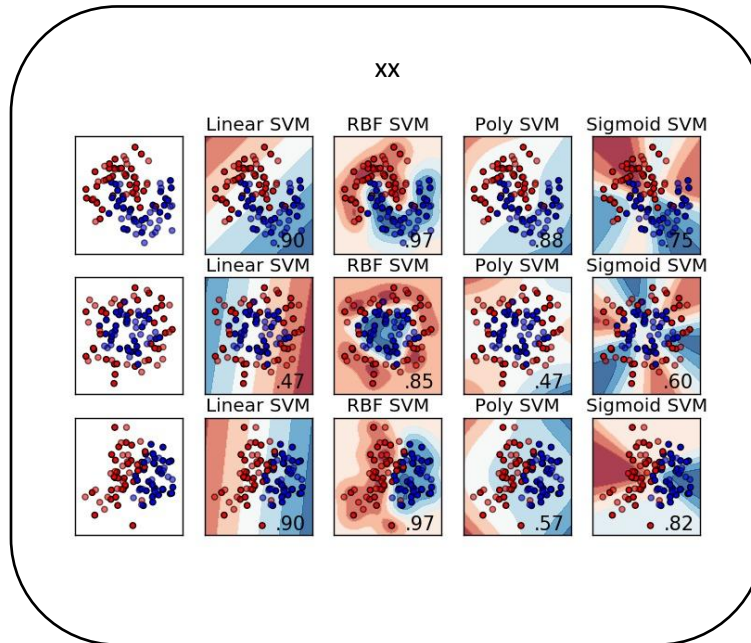
SVM algorithm finds the points closest to the line from both the classes these points are called support vectors. Now, we compute the distance between the line and support vector. This distance is called margin. Our goal is to maximize the margin. The hyper plane for which the margin is maximum is the optimal hyper plane.



Kernels:

Kernel is used due to set of mathematical functions used in support vector machine provides the window to manipulate the data. So, Kernel function generally transforms the training set of data so, that a non-linear decision surface is able to transform to a linear equation in a higher number of dimensional spaces. We have used polynomial kernel to get the maximum accuracy.





Implementation and Analysis:

1. Taking the data:

We haven't taken the data from the file provided. Using the pandas library, we had read the data and converted it into data frames. So, that we can access the values from it.

```

1 import pandas as pd
2 df = pd.read_csv("digit_svm.csv")
3 df

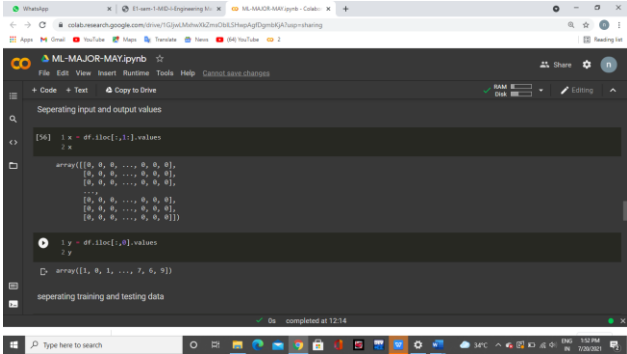
```

label	pixel0	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	pixel9	pixel10	pixel11	pixel12	pixel13	pixel14	pixel15	pixel16	pixel17
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
...
41995	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
41996	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
41997	7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
41998	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Out[]: completed at 12:14

2. Separating the input and output:

Using Numpy library, we can separate the input and output data. The label columns are to be taken as output (y), and the pixel columns as input data (x).



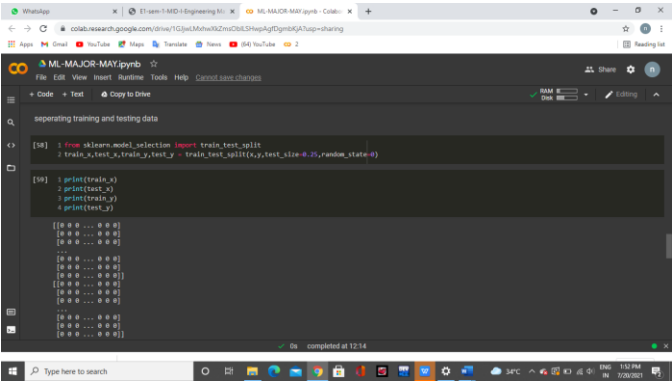
```
ML-MAJOR-MAY.ipynb
File Edit View Insert Runtime Tools Help Cancel save changes
+ Code + Text Copy to Drive
Separating input and output values
[54]: x = df.iloc[:,1:].values
      x
array([[0, 0, ..., 0, 0, 0],
       [0, 0, ..., 0, 0, 0],
       [0, 0, ..., 0, 0, 0],
       ...,
       [0, 0, ..., 0, 0, 0],
       [0, 0, ..., 0, 0, 0],
       [0, 0, ..., 0, 0, 0]])

[55]: y = df.iloc[:,0].values
      y
array([1, 0, 1, ..., 7, 6, 9])

separating training and testing data
completed at 12:14
```

3. Separating training and testing data:

We use Sklearn library, we have to split the data into training and testing data to check the intelligence of the model. We use the function `train_test_split`, to separate the training and testing data instead of separating manually.



```
ML-MAJOR-MAY.ipynb
File Edit View Insert Runtime Tools Help Cancel save changes
+ Code + Text Copy to Drive
separating training and testing data
[56]: from sklearn.model_selection import train_test_split
      train_x, test_x, train_y, test_y = train_test_split(x, y, test_size=0.2, random_state=0)

[57]: print(train_x)
      print(test_x)
      print(train_y)
      print(test_y)

[[0 0 ... 0 0]
 [0 0 ... 0 0]
 [0 0 ... 0 0]
 ...
 [0 0 ... 0 0]
 [0 0 ... 0 0]
 [0 0 ... 0 0]]

[[0 0 ... 0 0]
 [0 0 ... 0 0]
 [0 0 ... 0 0]
 ...
 [0 0 ... 0 0]
 [0 0 ... 0 0]
 [0 0 ... 0 0]]

[[1 0 ... 7 6]
 [0 0 ... 9 0]
 [0 0 ... 0 0]
 ...
 [0 0 ... 0 0]
 [0 0 ... 0 0]
 [0 0 ... 0 0]]

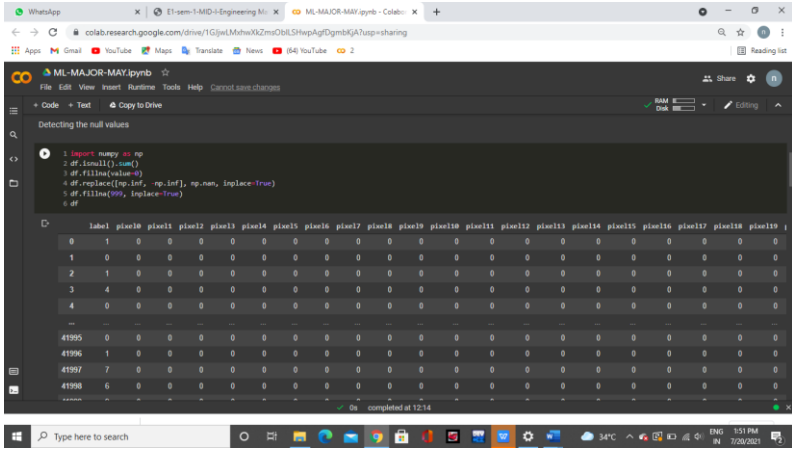
[[0 0 ... 0 0]
 [0 0 ... 0 0]
 [0 0 ... 0 0]
 ...
 [0 0 ... 0 0]
 [0 0 ... 0 0]
 [0 0 ... 0 0]]

completed at 12:14
```

4. Data Preprocessing:

a. Checking for Null values:

There are null values (or NAN values) present in the given data set. By using numpy library, we will find out the null values and replace them with zeroes (0).



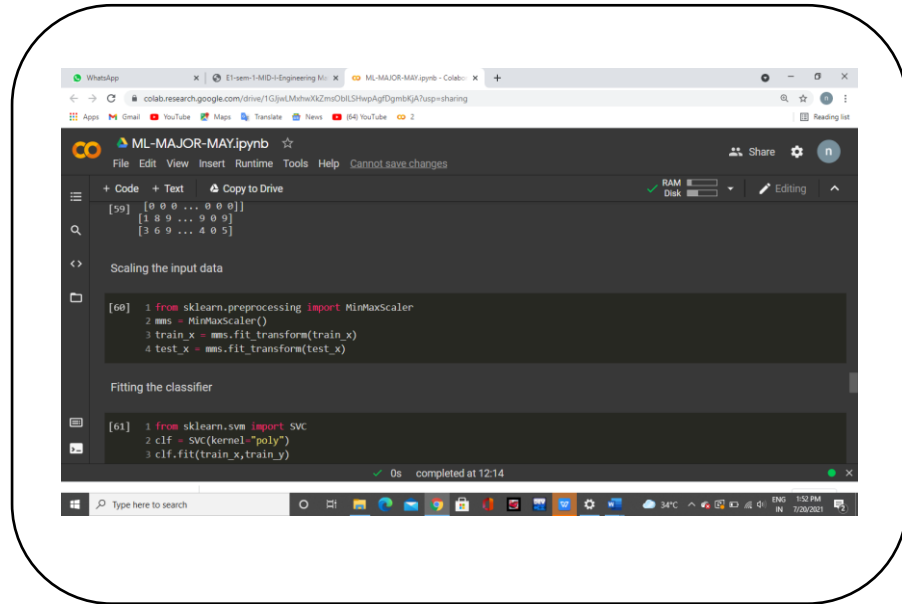
The screenshot shows a Jupyter Notebook interface with the following code and output:

```
1 import numpy as np
2 df.isnull().sum()
3 df.fillna(value=0)
4 df.replace([np.inf, -np.inf], np.nan, inplace=True)
5 df.fillna(999, inplace=True)
6 df
```

The output displays a table with columns labeled 'label' and 'pixel0' through 'pixel19'. The first few rows show data for labels 0, 1, 2, 3, and 4, with most pixel values being 0. The last few rows show data for labels 41995, 41996, 41997, and 41998, with pixel values being 0 or 1. The status bar at the bottom indicates 'completed at 12:14'.

b. Scaling:

We have to scale the input data to the same range. We have to scale data by using standard scaler function from sklearn preprocessing module it scales the data to its unit variants.



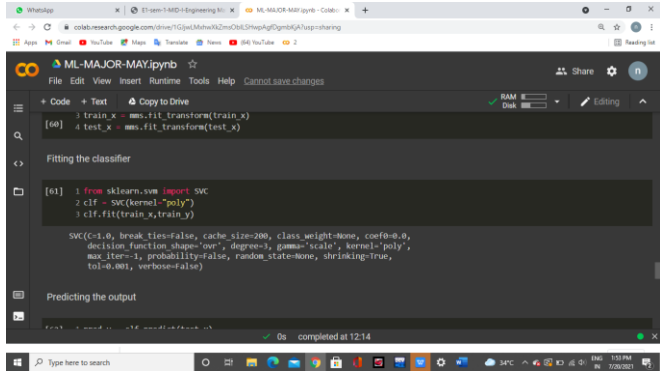
Classifier:

Support Vector Machine (SVM):

It is a supervised learning classification and it uses the classification algorithm which is SVC gets best fit hyperplane by dividing two groups and it will get support vector lines and the length between them is maximum margin. With the help of the support vector lines we get an optimal hyperplane or best fit hyperplane is drawn. From this we can get the data gets categorized or classified.

Fit the data:

Training data is fitted to the model with the help of SVC. So that our data is classified into categories.



```
1 train_x = mms.fit_transform(train_x)
2 test_x = mms.fit_transform(test_x)

Fitting the classifier

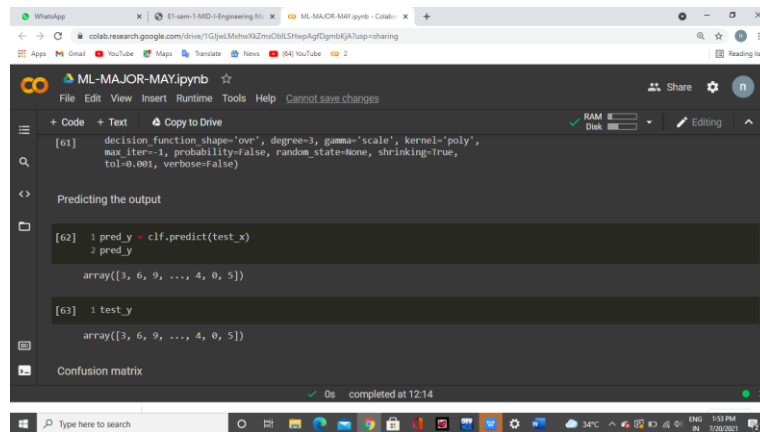
[01] 1 from sklearn.svm import SVC
      2 clf = SVC(kernel='poly')
      3 clf.fit(train_x, train_y)

SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='poly',
    max_iter=1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)

Predicting the output
```

Predicting the Output:

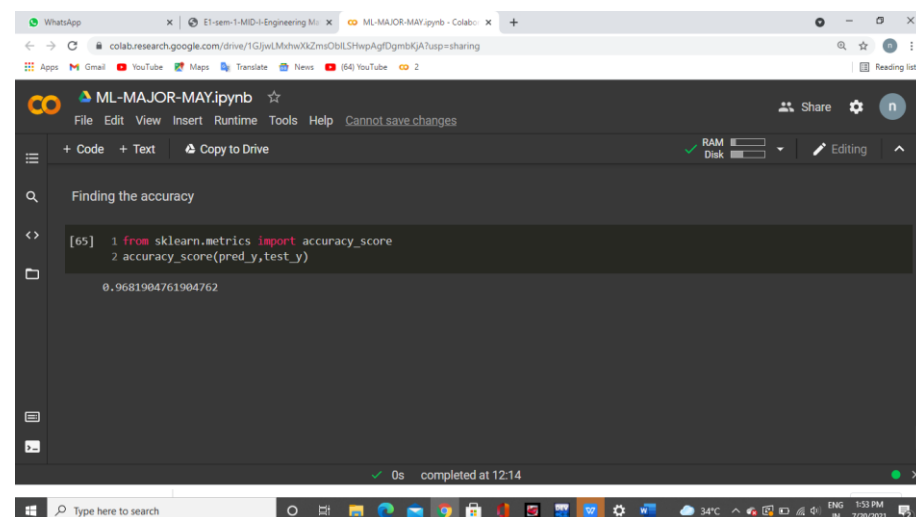
So, with the help of our model we predict the output values for the separated test input values. This predicted are compared with our true testing values.



```
[61] decision_function shap='ovr', degree=3, gamma='scale', kernel='poly',  
max_iter=1, probability=False, random_state=None, shrinking=True,  
tol=0.001, verbose=False)  
  
Predicting the output  
  
[62] 1 pred_y = clf.predict(test_x)  
2 pred_y  
  
array([3, 6, 9, ..., 4, 0, 5])  
  
[63] 1 test_y  
  
array([3, 6, 9, ..., 4, 0, 5])  
  
Confusion matrix
```

Accuracy:

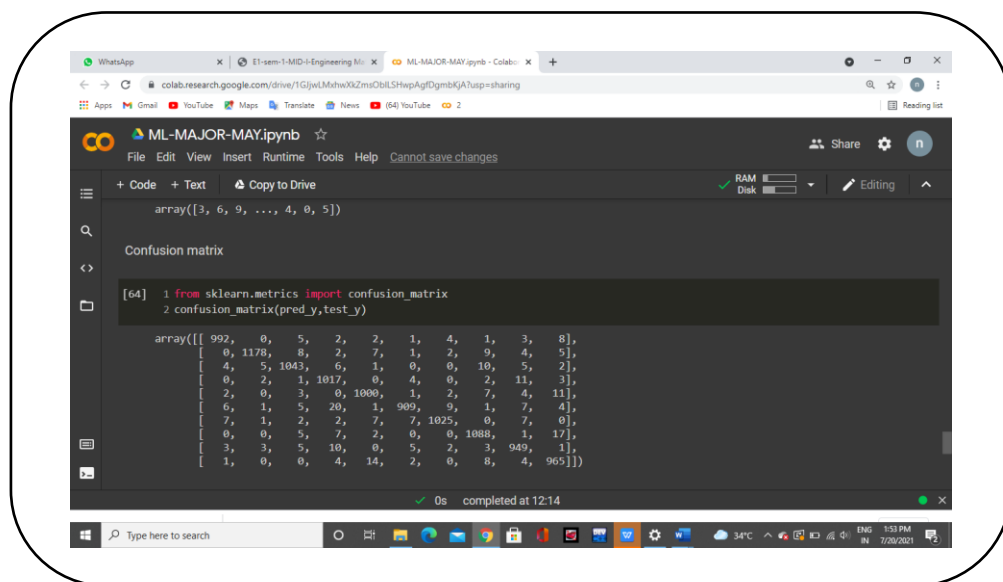
Accuracy score is a function from Sklearn metrics. In multi label classification this function computes subset accuracy: the set of labels predicted (predy) for a sample must exactly match the corresponding set of labels in ytrue. The accuracy score is 94.9%.



```
[65] 1 from sklearn.metrics import accuracy_score  
2 accuracy_score(pred_y, test_y)  
  
0.9681904761904762
```

Confusion matrix:

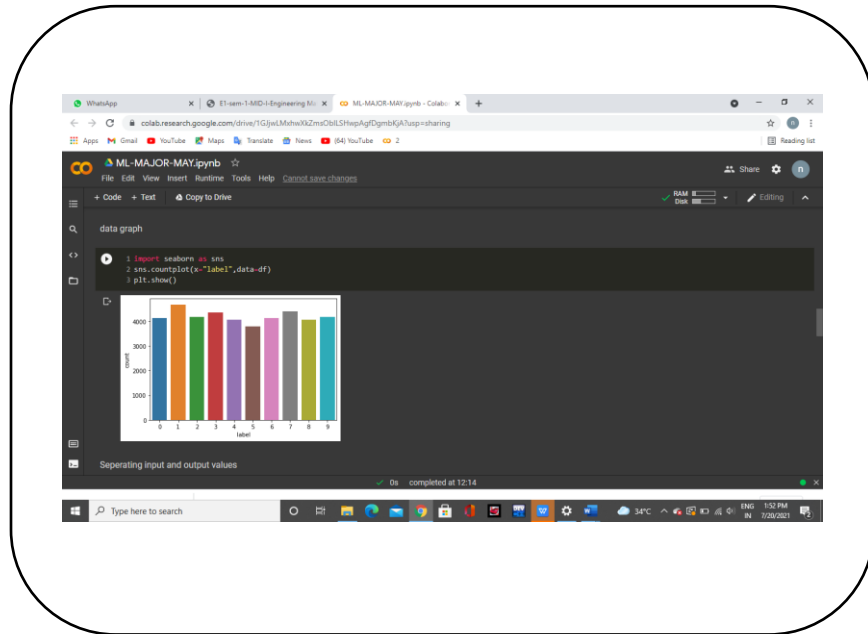
Confusion matrix is a function from sklearn metrics. In confusion matrix is such that it is equal to number of observations known to be in group but predicted to be in group. Thus in binary classification, the count of true negatives is C 0 , 0 , false negatives is C 1 , 0 , true positives is C 1 , 1 and false positives is C 0 , 1.

A screenshot of a Jupyter Notebook interface. The notebook is titled 'ML-MAJOR-MAY.ipynb'. The code cell shows the import of 'confusion_matrix' from 'sklearn.metrics' and its application to 'pred_y' and 'test_y'. The output is a 10x10 confusion matrix array. The array is displayed as follows:

```
array([[ 992,  0,  5,  2,  2,  1,  4,  1,  3,  8],
       [ 0, 1178,  8,  2,  7,  1,  2,  9,  4,  5],
       [ 4,  5, 1043,  6,  1,  0,  0, 10,  5,  2],
       [ 0,  2,  1, 1017,  0,  4,  0,  2, 11,  3],
       [ 2,  0,  3,  0, 1000,  1,  2,  7,  4, 11],
       [ 6,  1,  5, 20,  1, 900,  9,  1,  7,  4],
       [ 7,  1,  2,  2,  7, 7, 1025,  0,  7,  0],
       [ 0,  0,  5,  7,  2,  0,  0, 1088,  1, 17],
       [ 3,  3,  5, 10,  0,  5,  2,  3, 949,  1],
       [ 1,  0,  0,  4, 14,  2,  0,  8,  4, 965]])
```

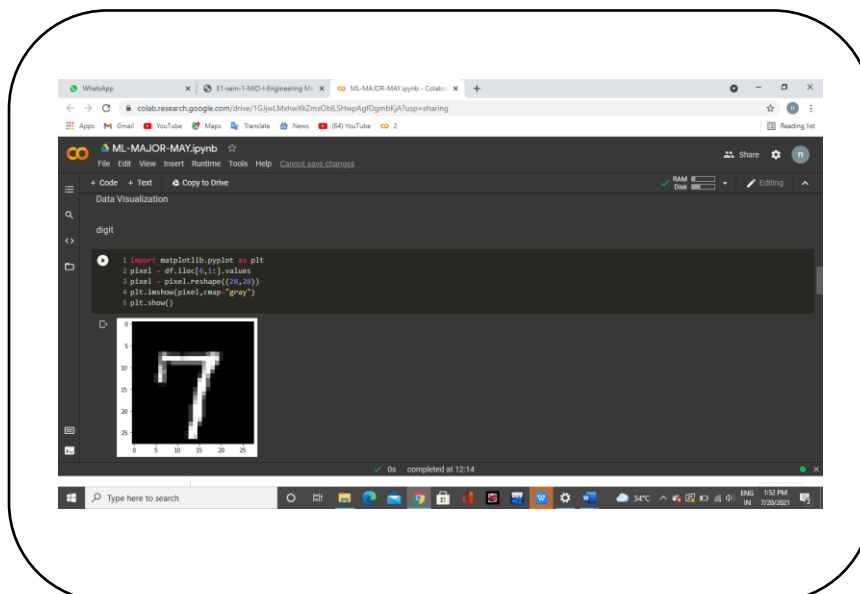
Seaborn:

Seaborn is a data visualization library built on top of matplotlib and closely integrated with pandas data structures in python and mainly used for making statistical graphs. Visualization is the central part of seaborn which helps in exploration and understanding of data.



Matplotlib:

Matplotlib is a collection functions that make Matplotlib work like MATLAB. Each py plot function makes some changes to a figure. Ex: Creates a figure, creates a plotting area in a figure, plot some lines in a plotting area, decorates the plot with labels etc.



IPYNB Link:

<https://colab.research.google.com/drive/1GJjwLMxhwXkZmsOb1LSHwpAgfDgmbKjA?usp=sharing>

APPLICATIONS OF SVM:

- This system is useful in many business purposes including office automation, bank check verification, postal automation, and a large variety of business and data entry applications.
- By also including Alphabets, Special Characters in the training data, this upgraded model can be implemented and used by traffic monitoring system for charging fine and penalties. The retrieved data of vehicles can be sent to RTO for further processing.
- The model can be used to digitize the old paper numerical records whose condition are fragile. This will lead to stable economy in industry.

- The model can be perfectly used in Hospitals and Universities for patients and students respectively who are suffering from trembling disorder (particular hands and fingers). The letters and numbers can be traced on Human interface device like LCDs, which further can be processed SVM algorithm.

Conclusion:

The main objective of this model is to predict the handwritten digits by our Machine Learning Model and it has been achieved successfully. We have got 96.8% accuracy by using SVM algorithm. By this we can predict the hand written digits from MNIST data set.