MNIST dataset to identify digit classification (Machine Learning July Major Project, Ritvik Chawla)

=>THE PROBLEM: The dataset is of 0-9 digits converted from image to input features; every row represents a specific digit image along with labels for every row. You can view the image using Matplotlib if you wish to.

Design a project from the MNIST dataset to identify digit classification using any of the classification algorithms.

=>TOOLS TO BE USED: The tools I along with the rest of the team have used in this projects are:

- Python programming language
- Google Collaboratory
- Pandas Software Library
- Numpy Package
- Matplotlib plotting library
- Sklearn Library
- Excel sheet data in CSV(Comma Separated Values) format

=>WHY SVM AND LOGISTIC REGRESSION ARE USED?:

- SVMs are supervised learning methods and can be used for both classification and regression problems. SVMs are well known for their effectiveness in high dimensional spaces, where the number of features is greater than the number of observations.

- Logistic regression is a supervised learning algorithm which is mostly used to solve binary "**classification"** tasks although it

contains the word “**regression**” . “Regression” contradicts with “classification” but the focus of logistic regression is on the word “logistic” referring to **logistic function** which actually does the classification task in the algorithm. Logistic regression is a simple yet very effective classification algorithm, so it is commonly used for many binary classification tasks.

=>**EXPLAINING THE ALGORITHM:** To do the project, an algorithm was used which has been explained in the following few lines-->

1)First, NumPy was imported for linear algebra and then pandas library was imported for data processing of files (CSV in this case) along with Matplotlib library for future representation of data in form of graphs, plots, etc.

```
[1]  import numpy as np # linear algebra
     import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
     import matplotlib.pyplot as plt #for plotting
```

2)A data frame in pandas called df was created to read the excel sheet data in CSV format and df.dropna(inplace=True) was used to remove all null values in the data frame.

```
#creating a dataframe in pandas
df=pd.read_csv('/content/digit_svm.csv')
```

```
[4]  #to drop null values
     df.dropna(inplace=True)
```

3)Now, to visualize the digits, a Y-label and X-label have been made using iloc function which helps us take certain range of data and locate it. These labels will be used to plot.

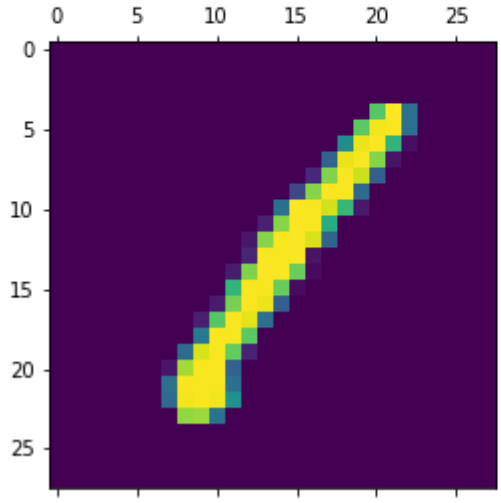
Visualizing the digits

```
[6]  y_label=df.iloc[:,0].values
     y_label
     x_label=df.iloc[:,1:785].values
```

4)Since, the dataset is of 0-9 digits converted from image to input features, every row represents a specific digit image along with labels for every row. The image can be viewed using Matplotlib using X-label and by reshaping the quantities on X and Y axis.

```
[7]  plt.matshow(x_label[0].reshape(28,28))
```

```
<matplotlib.image.AxesImage at 0x7f47a4ee9ad0>
```



5)Now, we must fit and evaluate the model which is to be done by importing libraries like sklearn. Sklearn library is used to import functions related to regression like LinearRegression,LogisticRegression, train_test_split,etc. Sklearn's model selection method which helps in setting a blueprint to analyze data and then using it to measure new data has a function train_test_split for splitting data arrays into two subsets: for training data and for testing data. With this function, you don't need to

divide the dataset manually. By default, Sklearn train_test_split will make random partitions for the two subsets.

```
[8]  #setting a blueprint to analyze data and then using it to measure new data
     from sklearn.model_selection import train_test_split
```

6) Both, x_label and y_label are divided into testing and training set by assigning them variables as x_train,x_test,y_train and y_test. Then a logistic regression model is created to fit the data in it using the variable name model,as a function of LogisticRegression.

```
+ Code  + Text                                                    RAM Disk   Editing

[10] x_train,x_test,y_train,y_test=train_test_split(x_label,y_label,test_size=0.2)

[11] len(x_test)

     802

[12] from sklearn.linear_model import LogisticRegression
     model=LogisticRegression()

[13] #fitting data
     model.fit(x_train,y_train)

     /usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_logistic.py:940: ConvergenceWarning: lbfgs failed to conve
     STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

     Increase the number of iterations (max_iter) or scale the data as shown in:
         https://scikit-learn.org/stable/modules/preprocessing.html
     Please also refer to the documentation for alternative solver options:
         https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
       extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
     LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                        intercept_scaling=1, l1_ratio=None, max_iter=100,
                        multi_class='auto', n_jobs=None, penalty='l2',
                        random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
```

7)An accuracy score of the rough model is checked using model.score with the x_test and y_test values.

```
[13] LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                        intercept_scaling=1, l1_ratio=None, max_iter=100,
                        multi_class='auto', n_jobs=None, penalty='l2',
                        random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                        warm_start=False)

[14] model.score(x_test,y_test)

     0.8690773067331671
```

8) A prediction variable called y_pred is created using the test cases of x_test.This y_pred and y_test variables are printed which show the digits of data in the form of an array.Both of them will be compared to check the accuracy of our prediction model.

+ Code + Text

RAM
Disk

```
[15] y_pred=model.predict(x_test)
     y_pred

     array([1, 4, 3, 7, 4, 5, 3, 5, 5, 0, 8, 0, 5, 3, 9, 9, 9, 5, 7, 3, 2, 0,
            5, 2, 5, 5, 5, 3, 2, 9, 0, 6, 4, 3, 0, 9, 6, 2, 3, 8, 6, 4, 5, 3,
            2, 2, 4, 2, 8, 7, 8, 4, 8, 3, 7, 1, 5, 6, 7, 9, 2, 9, 3, 1, 4, 9,
            7, 6, 4, 5, 7, 4, 7, 0, 1, 9, 5, 6, 2, 4, 5, 3, 0, 6, 7, 6, 0, 0,
            7, 5, 2, 9, 4, 2, 0, 2, 1, 6, 6, 8, 5, 6, 5, 5, 7, 5, 3, 7, 8, 2,
            9, 3, 1, 9, 1, 6, 4, 3, 2, 1, 4, 2, 1, 6, 8, 1, 2, 4, 3, 8, 0, 8,
            9, 5, 8, 0, 3, 1, 3, 1, 0, 4, 1, 5, 7, 1, 6, 6, 3, 2, 1, 0, 7, 6,
            2, 3, 1, 5, 7, 8, 9, 6, 8, 8, 1, 1, 0, 9, 5, 2, 4, 5, 5, 7, 4, 5,
            2, 3, 4, 7, 7, 5, 2, 0, 4, 5, 7, 9, 0, 9, 3, 7, 1, 2, 7, 3, 6, 5,
            4, 4, 4, 0, 9, 1, 6, 5, 8, 1, 0, 7, 3, 4, 8, 3, 0, 9, 6, 8, 7, 5,
            1, 3, 2, 7, 2, 1, 0, 6, 9, 1, 9, 3, 6, 7, 1, 1, 1, 1, 1, 0, 4, 5,
            4, 6, 5, 1, 3, 3, 5, 7, 1, 7, 7, 7, 4, 9, 5, 9, 8, 9, 6, 4, 4, 4,
            1, 7, 2, 4, 3, 1, 3, 3, 9, 2, 5, 3, 9, 0, 5, 7, 2, 6, 6, 0, 3, 5,
            3, 9, 2, 5, 1, 4, 2, 8, 8, 8, 1, 2, 8, 0, 8, 2, 4, 6, 7, 9, 7, 2,
            4, 9, 7, 3, 6, 9, 2, 5, 5, 5, 1, 5, 1, 6, 1, 7, 1, 9, 3, 1, 9, 0,
            6, 3, 7, 0, 7, 6, 5, 7, 3, 5, 3, 1, 6, 7, 8, 3, 8, 6, 0, 1, 8, 2,
            7, 2, 3, 7, 2, 4, 6, 9, 1, 6, 0, 5, 6, 7, 0, 6, 4, 9, 6, 5, 3, 9,
            9, 0, 7, 1, 9, 0, 5, 4, 1, 5, 1, 9, 1, 2, 5, 9, 0, 8, 2, 7, 6, 6,
            5, 9, 0, 1, 8, 3, 8, 5, 2, 2, 8, 2, 3, 6, 1, 1, 3, 6, 7, 7, 3, 2,
            8, 6, 9, 7, 1, 2, 3, 7, 7, 7, 0, 7, 0, 2, 4, 3, 9, 1, 0, 8, 3, 6,
            2, 3, 2, 8, 8, 0, 0, 5, 0, 4, 9, 7, 4, 8, 6, 2, 4, 9, 3, 3, 1, 5,
            7, 4, 3, 4, 5, 4, 9, 8, 9, 8, 4, 9, 5, 7, 0, 9, 9, 6, 1, 0, 0, 9,
            7, 9, 9, 7, 3, 2, 0, 4, 8, 8, 9, 2, 7, 5, 9, 0, 8, 9, 5, 8, 0, 2,
            2, 9, 7, 8, 0, 0, 3, 8, 2, 3, 9, 6, 2, 0, 2, 4, 1, 1, 3, 3, 4, 9,
```

0s completed at 11:52 AM

```
[15]
       0, 5, 2, 4, 2, 8, 4, 6, 7, 2, 9, 7, 4, 8, 5, 6, 2, 0, 5, 7, 2, 7,
       4, 5, 4, 9, 9, 9, 1, 2, 2, 4, 0, 6, 5, 4, 2, 0, 7, 5, 8, 1, 3, 4,
       0, 1, 7, 5, 5, 2, 9, 1, 2, 0, 5, 2, 4, 7, 5, 0, 5, 0, 2, 1, 8, 1,
       0, 8, 3, 2, 3, 6, 1, 8, 1, 6, 3, 1, 6, 2, 4, 5, 7, 7, 5, 5, 0, 6,
       4, 3, 6, 5, 1, 6, 4, 4, 8, 9, 7, 6, 9, 6, 1, 2, 3, 6, 7, 9, 6, 3,
       0, 9, 9, 3, 9, 1, 2, 9, 6, 9, 6, 5, 1, 8, 9, 1, 0, 2, 7, 8, 2, 1,
       2, 2, 1, 1, 9, 2, 1, 6, 0, 5, 7, 1, 6, 7, 3, 1, 6, 6, 9, 8, 8, 7,
       9, 3, 3, 8, 5, 6, 6, 2, 1, 6, 5, 2, 7, 7, 3, 5, 0, 4, 7, 4, 4, 8,
       2, 0, 0, 4, 7, 6, 3, 8, 8, 4, 2, 9, 2, 8, 7, 9, 1, 8, 0, 7, 8, 9,
       5, 8, 7, 4, 0, 7, 2, 2, 1, 6, 2, 3, 5, 8, 2, 6, 8, 2, 1, 0, 0, 1,
       8, 3, 5, 6, 7, 8, 1, 6, 4, 9])
```

```
[16] y_test
```

```
array([1, 4, 3, 7, 4, 5, 3, 8, 5, 0, 2, 0, 8, 3, 9, 9, 8, 9, 7, 3, 2, 0,
       5, 2, 5, 8, 5, 3, 2, 9, 0, 6, 4, 5, 0, 9, 6, 2, 3, 8, 6, 4, 5, 3,
       2, 2, 4, 2, 8, 7, 8, 4, 3, 3, 7, 1, 5, 6, 7, 9, 2, 9, 3, 1, 4, 9,
       7, 6, 0, 5, 7, 4, 7, 0, 1, 9, 0, 6, 2, 7, 8, 3, 0, 6, 7, 6, 0, 0,
       7, 5, 2, 3, 4, 2, 0, 2, 1, 1, 6, 3, 1, 6, 6, 5, 7, 5, 8, 7, 8, 2,
       9, 3, 1, 9, 1, 6, 4, 3, 2, 1, 4, 2, 1, 6, 8, 1, 2, 4, 3, 2, 0, 8,
       9, 8, 8, 0, 3, 1, 3, 1, 0, 4, 1, 8, 7, 1, 6, 6, 3, 2, 1, 0, 7, 6,
       2, 3, 1, 5, 7, 8, 7, 6, 8, 8, 1, 1, 0, 9, 5, 2, 8, 8, 5, 7, 4, 5,
       2, 3, 4, 7, 7, 8, 2, 0, 4, 5, 7, 9, 0, 9, 3, 7, 1, 8, 7, 3, 6, 5,
       4, 5, 4, 0, 9, 1, 6, 5, 8, 1, 0, 7, 3, 4, 8, 3, 0, 9, 6, 8, 7, 5,
       1, 3, 2, 7, 2, 1, 0, 6, 9, 1, 9, 3, 6, 7, 1, 1, 1, 1, 1, 0, 4, 5,
       4, 6, 8, 1, 3, 3, 3, 7, 1, 7, 7, 7, 4, 9, 5, 9, 8, 4, 6, 4, 4, 4,
       1, 7, 3, 4, 3, 1, 3, 3, 9, 2, 5, 3, 9, 0, 5, 7, 2, 6, 4, 0, 3, 6,
```

0s completed at 11:52 AM

```
[16]
       3, 9, 2, 5, 1, 4, 2, 8, 4, 8, 1, 2, 8, 0, 8, 2, 4, 6, 7, 9, 7, 2,
       4, 9, 7, 3, 6, 9, 2, 5, 1, 0, 1, 5, 8, 6, 1, 9, 3, 9, 2, 1, 9, 0,
       6, 3, 7, 0, 7, 6, 5, 7, 3, 5, 3, 1, 5, 7, 8, 8, 8, 6, 0, 1, 8, 3,
       7, 2, 3, 7, 2, 4, 6, 9, 1, 6, 0, 5, 6, 9, 0, 6, 4, 9, 6, 5, 3, 9,
       9, 0, 9, 1, 9, 0, 1, 4, 1, 5, 1, 5, 1, 2, 8, 9, 0, 8, 2, 7, 6, 6,
       5, 9, 0, 1, 8, 3, 8, 5, 6, 2, 8, 2, 3, 6, 1, 2, 7, 6, 7, 7, 3, 6,
       8, 6, 3, 2, 1, 2, 3, 7, 7, 9, 0, 7, 0, 2, 4, 3, 9, 3, 0, 8, 3, 6,
       2, 5, 5, 8, 8, 0, 0, 5, 0, 4, 9, 7, 4, 8, 6, 2, 4, 9, 3, 3, 3, 1,
       7, 4, 3, 4, 5, 4, 9, 8, 9, 8, 4, 9, 5, 7, 0, 9, 0, 6, 1, 0, 0, 9,
       7, 9, 9, 7, 3, 2, 0, 4, 8, 8, 9, 2, 7, 8, 9, 0, 8, 9, 5, 5, 0, 2,
       7, 9, 7, 8, 0, 0, 3, 8, 2, 3, 9, 0, 2, 0, 2, 4, 8, 8, 8, 5, 4, 8,
       4, 1, 8, 7, 0, 2, 1, 7, 3, 9, 5, 2, 1, 2, 7, 7, 7, 7, 1, 4, 6, 8,
       2, 5, 3, 2, 6, 6, 7, 6, 4, 7, 8, 8, 4, 5, 0, 7, 3, 6, 8, 3, 8, 4,
       0, 5, 2, 4, 2, 8, 4, 6, 7, 2, 3, 7, 4, 8, 5, 6, 2, 0, 5, 7, 2, 7,
       4, 5, 4, 9, 5, 9, 1, 2, 5, 4, 0, 6, 5, 4, 2, 0, 7, 5, 0, 1, 3, 2,
       0, 1, 7, 5, 5, 2, 9, 1, 2, 0, 5, 3, 4, 7, 5, 0, 5, 0, 5, 1, 9, 1,
       0, 8, 3, 6, 3, 6, 1, 8, 8, 6, 3, 1, 6, 2, 4, 5, 7, 7, 5, 5, 0, 6,
       2, 3, 6, 3, 1, 6, 4, 4, 8, 1, 7, 6, 9, 6, 1, 2, 8, 6, 6, 9, 0, 3,
       0, 9, 9, 3, 4, 1, 2, 2, 6, 9, 6, 5, 1, 8, 7, 1, 0, 6, 7, 8, 2, 1,
       8, 2, 1, 1, 9, 2, 1, 6, 0, 0, 9, 1, 6, 7, 3, 1, 6, 6, 9, 8, 8, 6,
       9, 2, 3, 8, 5, 6, 6, 2, 1, 6, 5, 2, 7, 7, 3, 8, 0, 4, 7, 6, 4, 8,
       2, 0, 0, 8, 7, 6, 3, 8, 8, 4, 2, 9, 2, 8, 7, 9, 1, 8, 0, 7, 8, 9,
       5, 8, 7, 4, 0, 7, 2, 2, 8, 6, 2, 3, 5, 8, 6, 6, 0, 2, 1, 0, 0, 1,
       8, 3, 2, 6, 7, 8, 1, 6, 4, 9])
```

9) Accuracy_score and confusion_matrix are imported from the metrics method of sklearn library. The y_pred and y_test values are then taken into each function in order to check accuracy of predicted model from accuracy_score and performance of the model using confusion_matrix.

```
[17] from sklearn.metrics import accuracy_score,confusion_matrix

[18] accuracy_score(y_test,y_pred)

     0.8690773067331671

[19] confusion_matrix(y_test,y_pred)

     array([[71,  0,  0,  0,  1,  3,  2,  0,  2,  1],
            [ 0, 77,  0,  0,  0,  4,  1,  0,  0,  1],
            [ 0,  1, 76,  2,  2,  1,  0,  1,  2,  1],
            [ 0,  3,  3, 67,  0,  2,  0,  0,  2,  3],
            [ 0,  0,  0,  0, 64,  0,  1,  0,  1,  2],
            [ 0,  0,  3,  4,  1, 58,  1,  0,  1,  2],
            [ 0,  0,  6,  0,  1,  2, 73,  2,  0,  0],
            [ 0,  0,  1,  1,  1,  0,  0, 83,  0,  2],
            [ 0,  5,  2,  4,  2, 13,  0,  0, 62,  2],
            [ 0,  0,  0,  0,  0,  1,  0,  5,  1, 66]])

[20] model.predict(x_label[0:5])

     array([1, 0, 1, 4, 0])
```

10) The accuracy of the model came out to be 0.8690773067331671 Which is fairly good but we tried SVM method also to check accuracy.

11) From the sklearn library we use svm(Support Vector Machines) method to import SVC(Support Vector Classifier) function and make an object called newModel which is equal to SVC().

SVM Algorithm //Works the best

```
[22] from sklearn.svm import SVC

[23] newModel=SVC()
```

12) The x_train and y_train values of the dataframe are fir into the newmodel object and a score is taken out which will be checked later through the test and prediction values.

```
[24] newModel.fit(x_train,y_train)

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='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

```
[25] newModel.score(x_test,y_test)

0.9364089775561097
```

13) Again a y_pred variable is created for the newModel using x_test values and y_pred and y_test is printed in form of arrays.

```
[26] y_pred=newModel.predict(x_test)
     y_pred

array([1, 4, 3, 7, 4, 5, 5, 8, 5, 0, 8, 0, 8, 3, 9, 9, 8, 9, 7, 2, 2, 0,
       5, 2, 5, 8, 5, 3, 2, 9, 0, 6, 4, 5, 0, 9, 6, 2, 3, 8, 6, 4, 5, 3,
       2, 2, 4, 2, 8, 7, 8, 4, 3, 3, 9, 1, 5, 6, 7, 9, 2, 9, 3, 1, 4, 9,
       7, 6, 6, 5, 7, 4, 7, 0, 1, 9, 0, 6, 2, 4, 8, 1, 0, 6, 7, 6, 0, 0,
       7, 5, 2, 9, 4, 2, 0, 2, 1, 6, 6, 2, 5, 6, 6, 5, 7, 5, 8, 7, 8, 2,
       9, 3, 1, 9, 1, 6, 4, 3, 2, 1, 4, 2, 1, 6, 8, 1, 2, 4, 3, 3, 0, 8,
       9, 8, 8, 0, 3, 1, 3, 1, 0, 4, 1, 8, 7, 1, 6, 6, 3, 2, 1, 0, 7, 6,
       2, 3, 1, 5, 7, 8, 7, 6, 8, 8, 1, 1, 0, 9, 5, 2, 4, 8, 5, 7, 4, 5,
       2, 3, 4, 7, 7, 5, 2, 0, 4, 5, 7, 9, 0, 9, 3, 7, 1, 2, 7, 3, 6, 5,
       4, 9, 4, 0, 9, 1, 6, 5, 8, 1, 0, 7, 3, 4, 8, 3, 0, 9, 6, 8, 7, 5,
       1, 3, 2, 7, 2, 1, 0, 6, 9, 1, 9, 3, 6, 7, 1, 1, 1, 1, 1, 0, 4, 5,
       4, 4, 8, 1, 3, 3, 3, 7, 1, 7, 7, 7, 4, 9, 5, 9, 8, 4, 6, 4, 4, 4,
       1, 7, 3, 4, 3, 1, 3, 3, 9, 2, 5, 3, 9, 0, 5, 7, 2, 6, 6, 0, 3, 6,
       3, 9, 2, 5, 1, 4, 2, 8, 4, 8, 1, 2, 8, 0, 8, 2, 4, 6, 7, 9, 7, 2,
       4, 9, 2, 3, 6, 9, 2, 5, 1, 0, 1, 5, 8, 6, 1, 4, 1, 9, 2, 1, 9, 0,
       6, 3, 7, 0, 7, 6, 5, 7, 3, 5, 3, 1, 5, 7, 8, 8, 8, 6, 0, 1, 8, 3,
       7, 2, 3, 7, 2, 4, 6, 9, 1, 6, 0, 8, 6, 7, 0, 6, 4, 9, 6, 5, 3, 9,
       9, 0, 9, 1, 9, 0, 1, 4, 1, 5, 1, 5, 1, 2, 8, 9, 0, 8, 2, 7, 6, 6,
       5, 9, 0, 1, 8, 3, 8, 5, 6, 2, 8, 2, 3, 6, 1, 1, 9, 6, 7, 7, 3, 6,
       8, 6, 9, 2, 1, 2, 3, 7, 7, 9, 0, 7, 0, 2, 4, 3, 9, 3, 0, 8, 3, 6,
       2, 5, 5, 8, 8, 0, 0, 5, 0, 4, 9, 7, 4, 8, 6, 2, 4, 9, 3, 3, 3, 1,
       7, 4, 3, 4, 5, 4, 9, 8, 9, 8, 4, 9, 5, 7, 0, 9, 0, 6, 1, 0, 0, 9,
       7, 9, 9, 7, 3, 2, 0, 4, 8, 8, 9, 2, 7, 5, 9, 0, 8, 9, 5, 5, 0, 2,
       7, 9, 7, 8, 0, 0, 3, 8, 2, 3, 9, 6, 2, 0, 2, 4, 8, 8, 8, 5, 4, 8,

 0s  completed at 11:52 AM
```

14) The accuracy_score and confusion_matrix are printed and the accuracy score comes out to be 0.9212827988338192 which is way better than the accuracy of the model using Logistic Regression classifier.

```
[27] accuracy_score(y_test,y_pred)

     0.9364089775561097

[28] confusion_matrix(y_test,y_pred)

     array([[77,  0,  0,  0,  0,  0,  3,  0,  0,  0],
            [ 0, 80,  0,  0,  0,  1,  1,  1,  0,  0],
            [ 0,  1, 78,  2,  2,  0,  1,  0,  2,  0],
            [ 0,  2,  2, 71,  0,  1,  0,  0,  2,  2],
            [ 0,  0,  0,  0, 66,  0,  1,  0,  0,  1],
            [ 1,  0,  1,  0,  0, 66,  0,  0,  1,  1],
            [ 0,  0,  1,  0,  1,  0, 82,  0,  0,  0],
            [ 0,  0,  1,  0,  1,  0,  0, 84,  0,  2],
            [ 0,  2,  2,  0,  1,  2,  1,  0, 82,  0],
            [ 0,  0,  0,  1,  2,  0,  0,  4,  1, 65]])
```

=><u>CONCLUSION:</u> From the above lines, we can conclude that while analysing the data frame the final accuracy on test data is approx. 93% and SVM algorithm works the best (comparing SVM and Logistic Regression Algorithm).