DIGIT CLASSIFICATION USING SVM ALGORITHM
(ML MAJOR PROJECT)

Problem Statement :

Design a project from the MNIST dataset to identify digit classification using the SVM algorithm.

Packages, Libraries, Modules used :

- pandas
- matplotlib.pyplot
- numpy
- sklern - package
    1. sklearn.model_selection
    2. sklearn.preprocessing
    3. sklearn.svm
    4. Sklearn.metrics

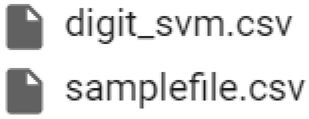
How I solved it :

- First I imported the necessary packages and libraries.

- Then I read the csv file provided using the read_csv() function.
- Then I have divided the input and output into 2 arrays x and y respectively.
- x and y are expressed as 2D arrays. This is because the fit function only accepts 2D input and if we pass 1D input we will get an error.
- Now we have to split the data set into 2 sets. One for test and other for train.
- For this, I have imported test_train_split from sklearn.model_selection
- We should pass x and y to test_train_split function. We will also mention the test_size. This test size determines how many fractions of the data must be given for testing and the remaining portion of the data will be given to training the model.
- This function returns 4 values(training input, testing input, training output, testing output) which are stored in 4 variables(x_train, x_test, y_train, y_test).
- Now we have normalise the data within a particular range. So now we will be using Feature scaling. For this, import StandardScalar class from sklearn.preprocessing. Now create an instance for the class (named as sc in the code). Use fit_transform to scale the data present in x_train and x_test.
- Now use the SVM algorithm. Assign kernel = "linear". Train the model with x_train and y_train.

- Create a predicted value array (named as pred_y) which is the final predicted values by the model.
- Now calculate the accuracy of the model.
- To know about the true negatives, false negatives, true positives, false positives use the confusion matrix.

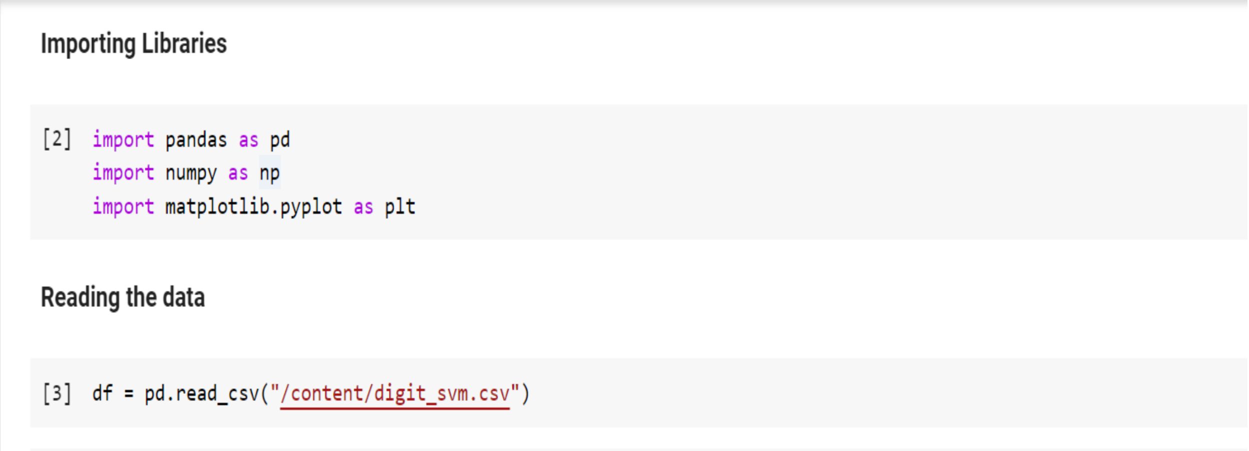
Files used:



digit_svm.csv

samplefile.csv

Code screenshot

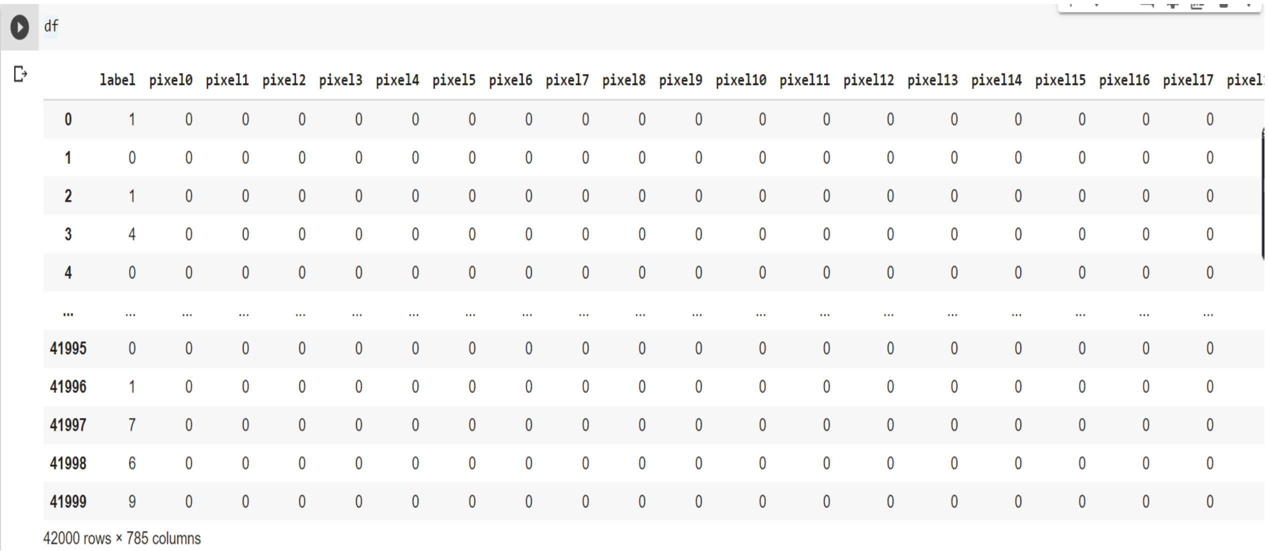


Importing Libraries

```
[2] import pandas as pd
    import numpy as np
    import matplotlib.pyplot as plt
```

Reading the data

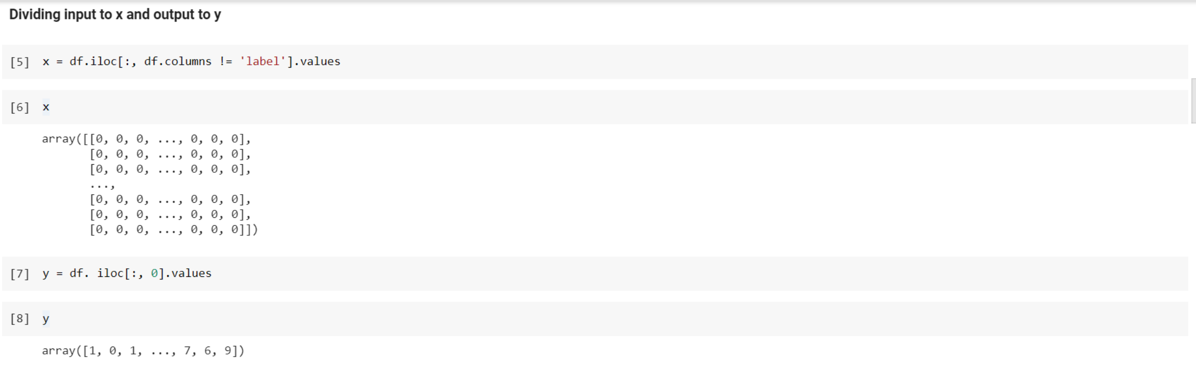
```
[3] df = pd.read_csv("/content/digit_svm.csv")
```



```
df
```

	label	pixel0	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	pixel9	pixel10	pixel11	pixel12	pixel13	pixel14	pixel15	pixel16	pixel17	pixel
0	1	0	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	0	
2	1	0	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	0	
4	0	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	0	
41996	1	0	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	0	
41998	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
41999	9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

42000 rows × 785 columns



Dividing input to x and output to y

```
[5]  x = df.iloc[:, df.columns != 'label'].values
```

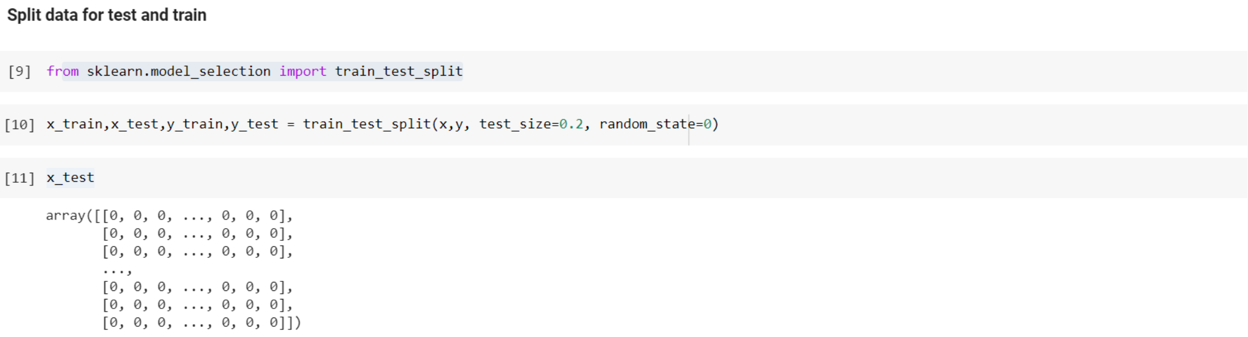
```
[6]  x
```

```
     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],
            [0, 0, 0, ..., 0, 0, 0]])
```

```
[7]  y = df. iloc[:, 0].values
```

```
[8]  y
```

```
     array([1, 0, 1, ..., 7, 6, 9])
```



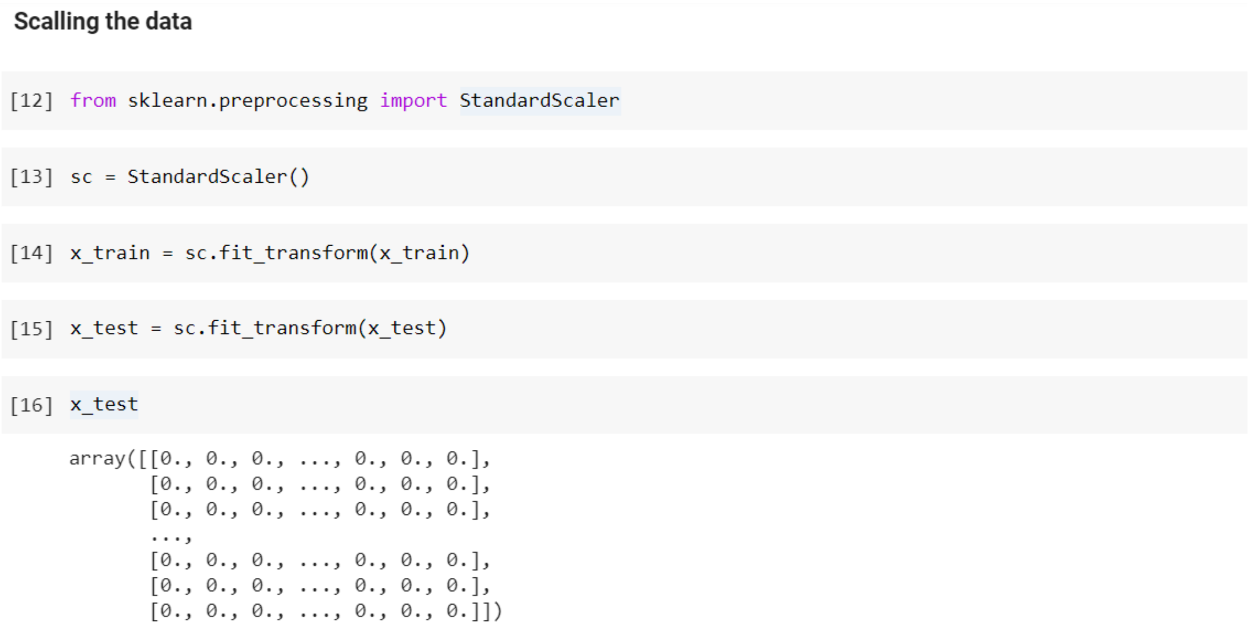
Split data for test and train

```
[9]  from sklearn.model_selection import train_test_split
```

```
[10]  x_train,x_test,y_train,y_test = train_test_split(x,y, test_size=0.2, random_state=0)
```

```
[11]  x_test
```

```
      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],
             [0, 0, 0, ..., 0, 0, 0]])
```



Scalling the data

```
[12]  from sklearn.preprocessing import StandardScaler
```

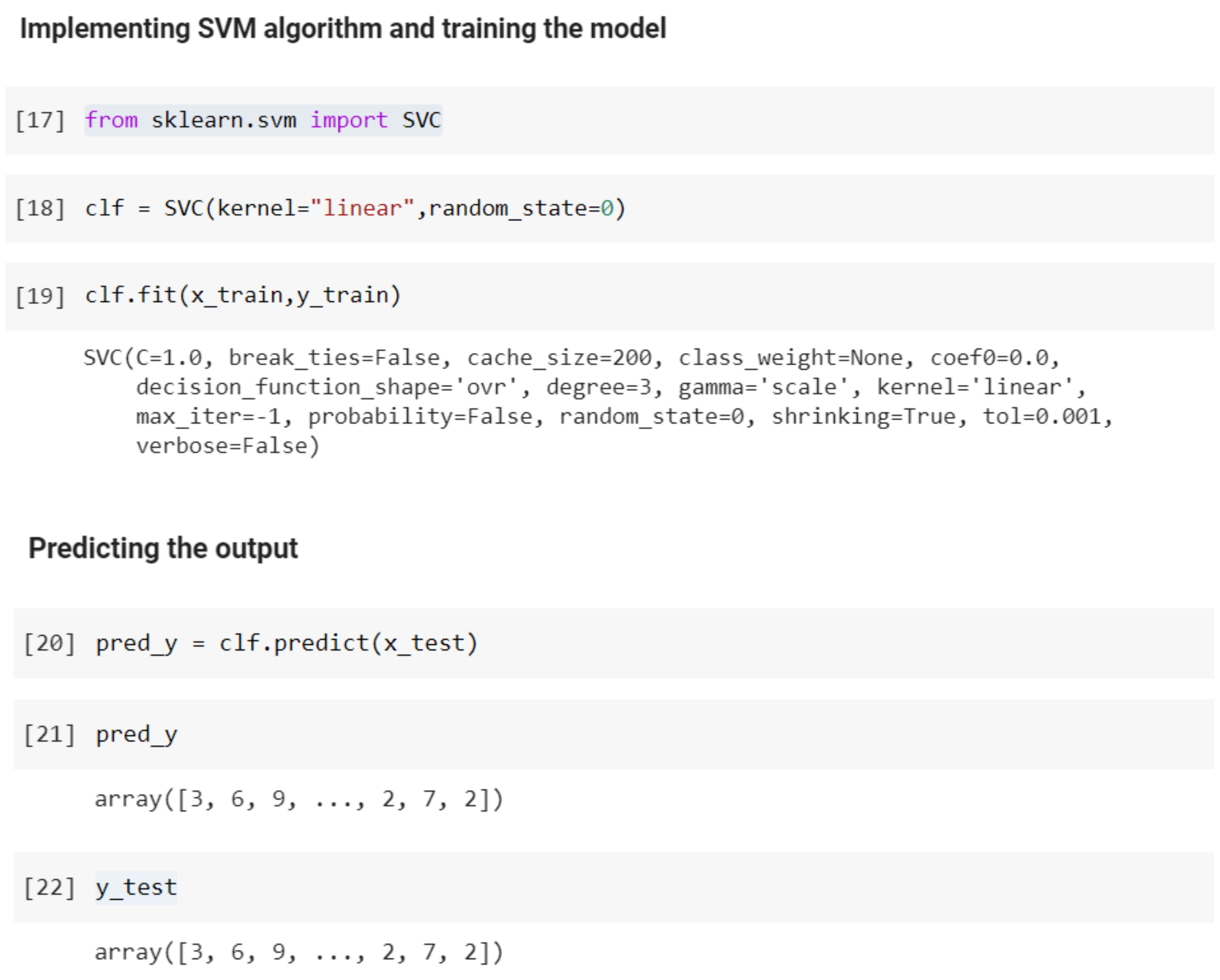
```
[13]  sc = StandardScaler()
```

```
[14]  x_train = sc.fit_transform(x_train)
```

```
[15]  x_test = sc.fit_transform(x_test)
```

```
[16]  x_test
```

```
      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.],
             [0., 0., 0., ..., 0., 0., 0.]])
```



Implementing SVM algorithm and training the model

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

```
[18] clf = SVC(kernel="linear",random_state=0)
```

```
[19] clf.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='linear',
         max_iter=-1, probability=False, random_state=0, shrinking=True, tol=0.001,
         verbose=False)
```

Predicting the output

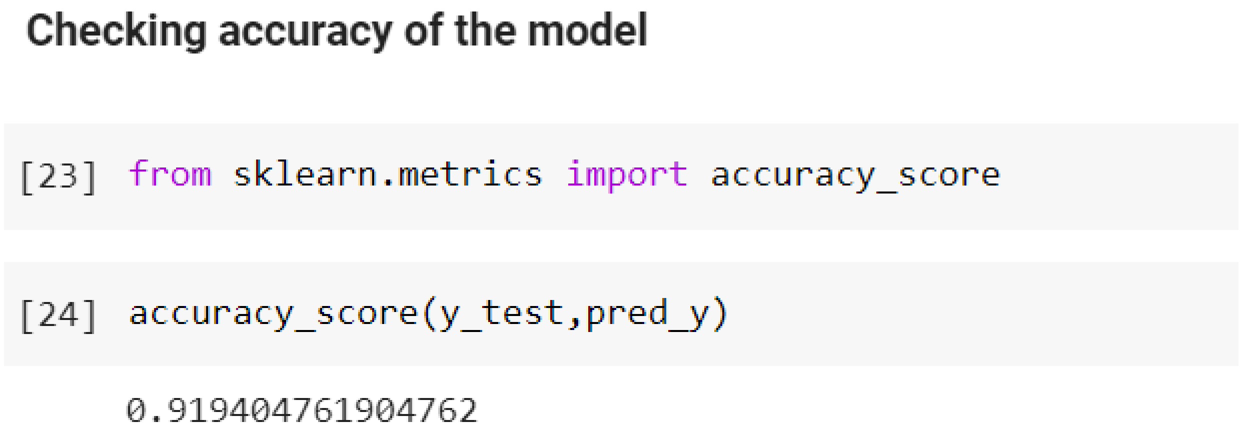
```
[20] pred_y = clf.predict(x_test)
```

```
[21] pred_y

     array([3, 6, 9, ..., 2, 7, 2])
```

```
[22] y_test

     array([3, 6, 9, ..., 2, 7, 2])
```

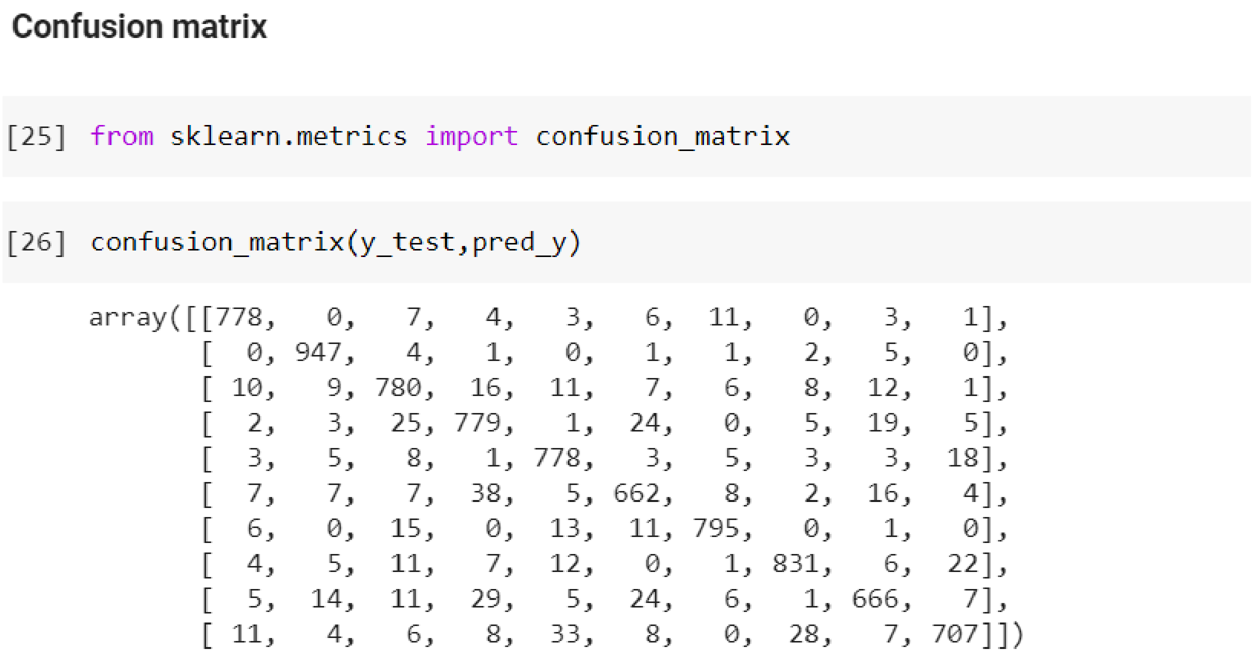


Checking accuracy of the model

```
[23] from sklearn.metrics import accuracy_score
```

```
[24] accuracy_score(y_test,pred_y)

     0.919404761904762
```

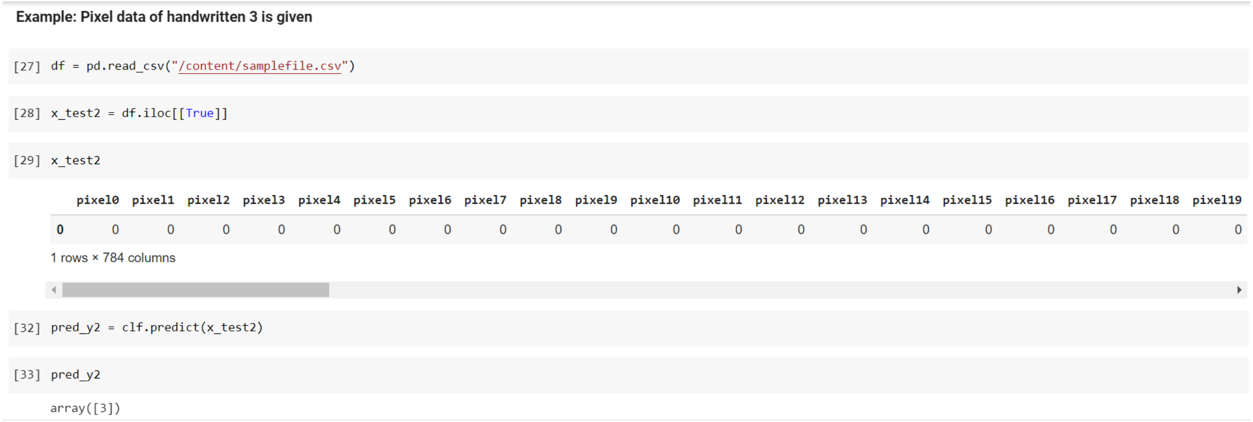


Confusion matrix

```
[25] from sklearn.metrics import confusion_matrix
```

```
[26] confusion_matrix(y_test,pred_y)
```

```
     array([[778,   0,   7,   4,   3,   6,  11,   0,   3,   1],
            [  0, 947,   4,   1,   0,   1,   1,   2,   5,   0],
            [ 10,   9, 780,  16,  11,   7,   6,   8,  12,   1],
            [  2,   3,  25, 779,   1,  24,   0,   5,  19,   5],
            [  3,   5,   8,   1, 778,   3,   5,   3,   3,  18],
            [  7,   7,   7,  38,   5, 662,   8,   2,  16,   4],
            [  6,   0,  15,   0,  13,  11, 795,   0,   1,   0],
            [  4,   5,  11,   7,  12,   0,   1, 831,   6,  22],
            [  5,  14,  11,  29,   5,  24,   6,   1, 666,   7],
            [ 11,   4,   6,   8,  33,   8,   0,  28,   7, 707]])
```



Example: Pixel data of handwritten 3 is given

```
[27] df = pd.read_csv("/content/samplefile.csv")
```

```
[28] x_test2 = df.iloc[[True]]
```

```
[29] x_test2
```

	pixel0	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	pixel9	pixel10	pixel11	pixel12	pixel13	pixel14	pixel15	pixel16	pixel17	pixel18	pixel19
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1 rows × 784 columns

```
[32] pred_y2 = clf.predict(x_test2)
```

```
[33] pred_y2

     array([3])
```

Conclusion

The model has been trained successfully using the SVM **algorithm** with an **accuracy of 91.9%** and performance is calculated using confusion matrix.