1-Logistic Regression Model

a.*General Information on dataset*

Name of dataset:

https://www.kaggle.com/code/schmoyote/breast-cancer-classification-beginner-friendly/data

Number of classes: **2 classes**

Total number of samples : **569 samples**

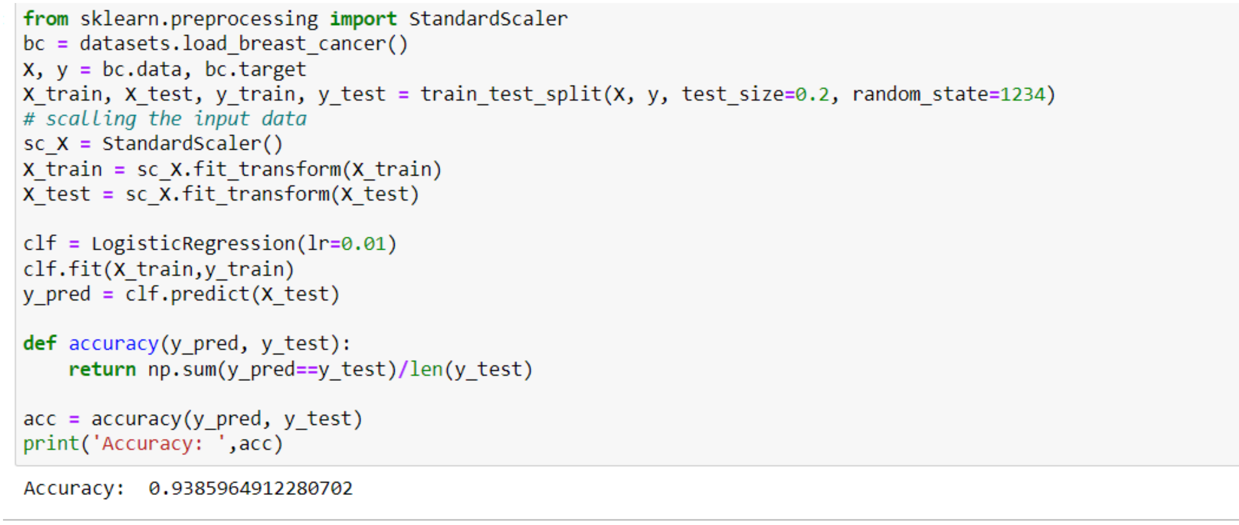
Number of training samples: **398 samples**
Number of test samples: **171 samples**

b.*Implementation details:*

Number features : **2 features.**
Name of features : **Real, Positive.**
Dimension of resulted features : **30**

c. *Results details:*

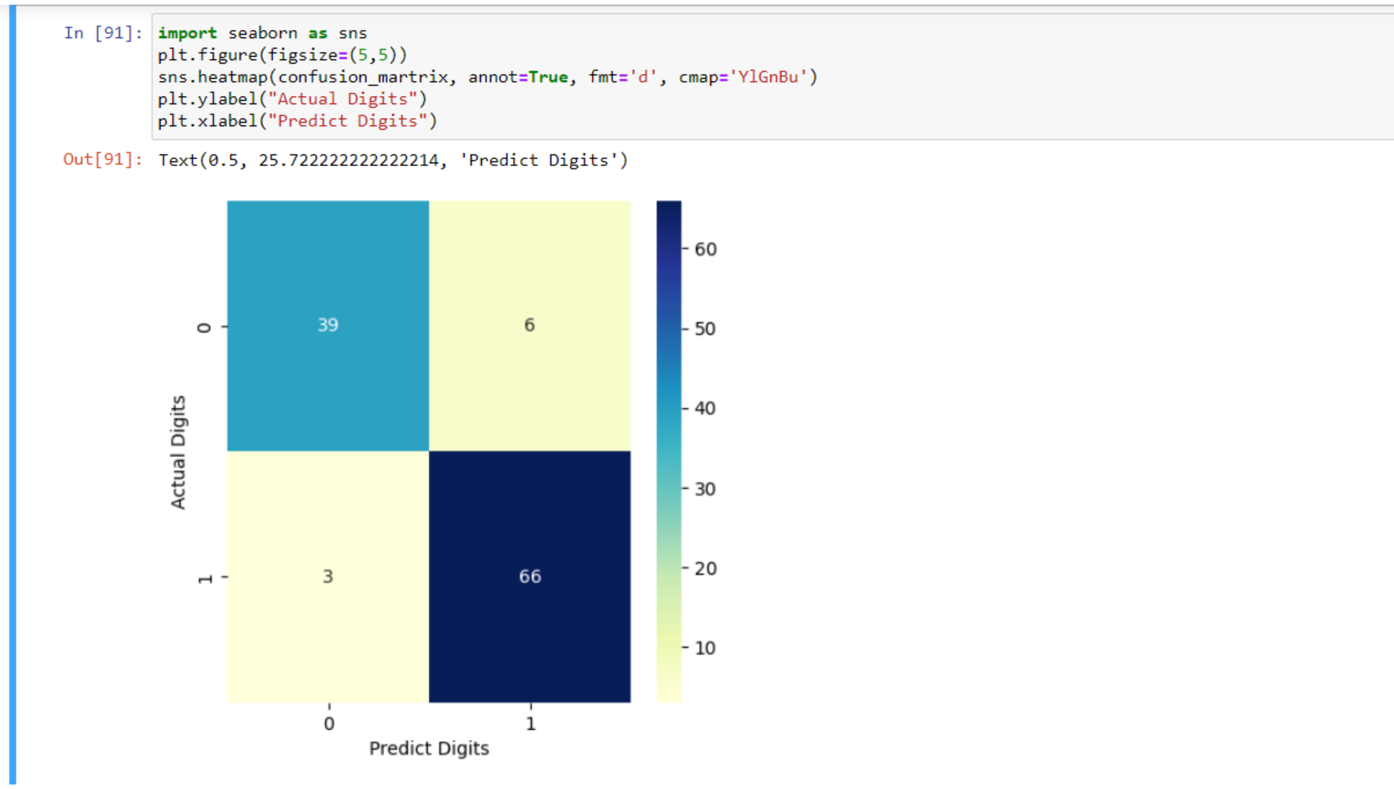
accuracy



```
from sklearn.preprocessing import StandardScaler
bc = datasets.load_breast_cancer()
X, y = bc.data, bc.target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=1234)
# scalling the input data
sc_X = StandardScaler()
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.fit_transform(X_test)

clf = LogisticRegression(lr=0.01)
clf.fit(X_train,y_train)
y_pred = clf.predict(X_test)

def accuracy(y_pred, y_test):
    return np.sum(y_pred==y_test)/len(y_test)

acc = accuracy(y_pred, y_test)
print('Accuracy: ',acc)
```

```
Accuracy:  0.9385964912280702
```

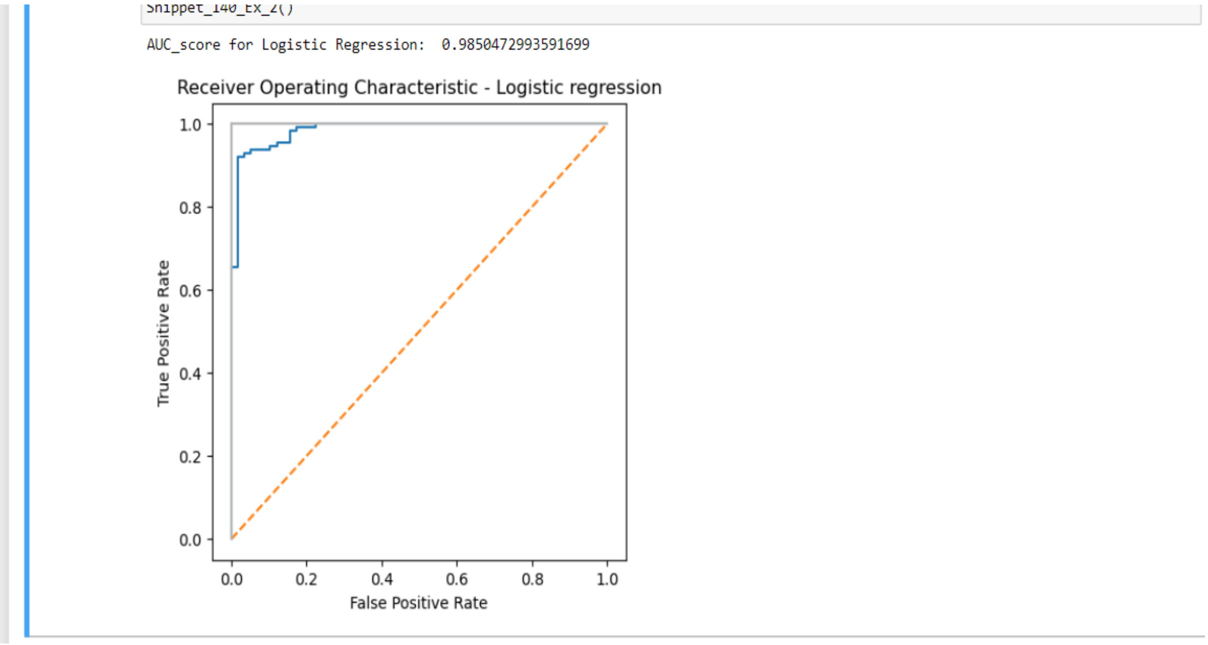
confusion matrix



```
In [91]: import seaborn as sns
         plt.figure(figsize=(5,5))
         sns.heatmap(confusion_martrix, annot=True, fmt='d', cmap='YlGnBu')
         plt.ylabel("Actual Digits")
         plt.xlabel("Predict Digits")
```

```
Out[91]: Text(0.5, 25.722222222222214, 'Predict Digits')
```

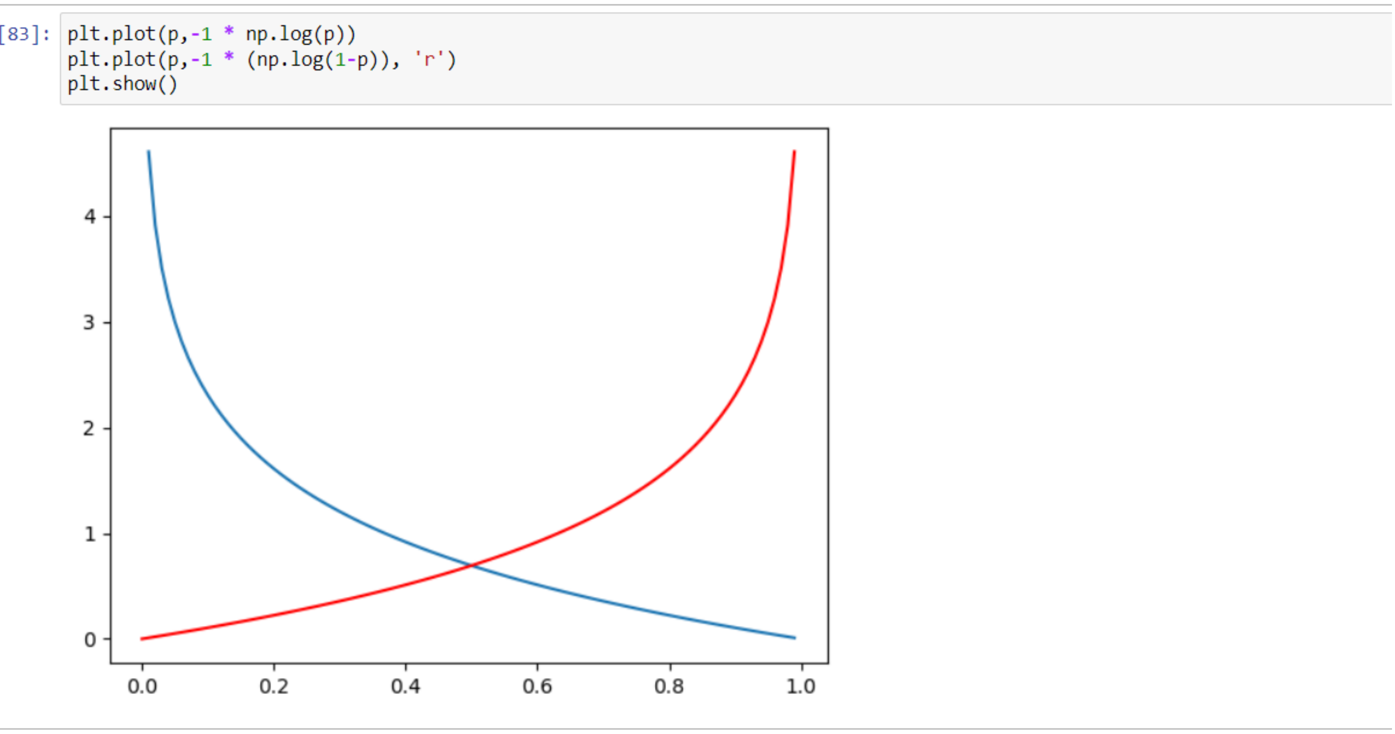
ROC curve



```
Snippet_140_Ex_2()
```

```
AUC_score for Logistic Regression:  0.9850472993591699
```

loss curve



```
[83]: plt.plot(p,-1 * np.log(p))
      plt.plot(p,-1 * (np.log(1-p)), 'r')
      plt.show()
```

2-ANN Model

a.*General Information on dataset*

Name of dataset:

The MNIST database of handwritten digits.
https://www.tensorflow.org/datasets/catalog/mnist

Number of classes:

10 classes

ClassLabel(shape=( ), dtype=int64, num_classes=10)

Total number of samples :

70,000 samples , size of each image 28x28

Image(shape=(28, 28, 1), dtype=uint8)

Number of training samples: 60,000 samples
Number of test samples: 10,000 samples

b. *Implementation details:*

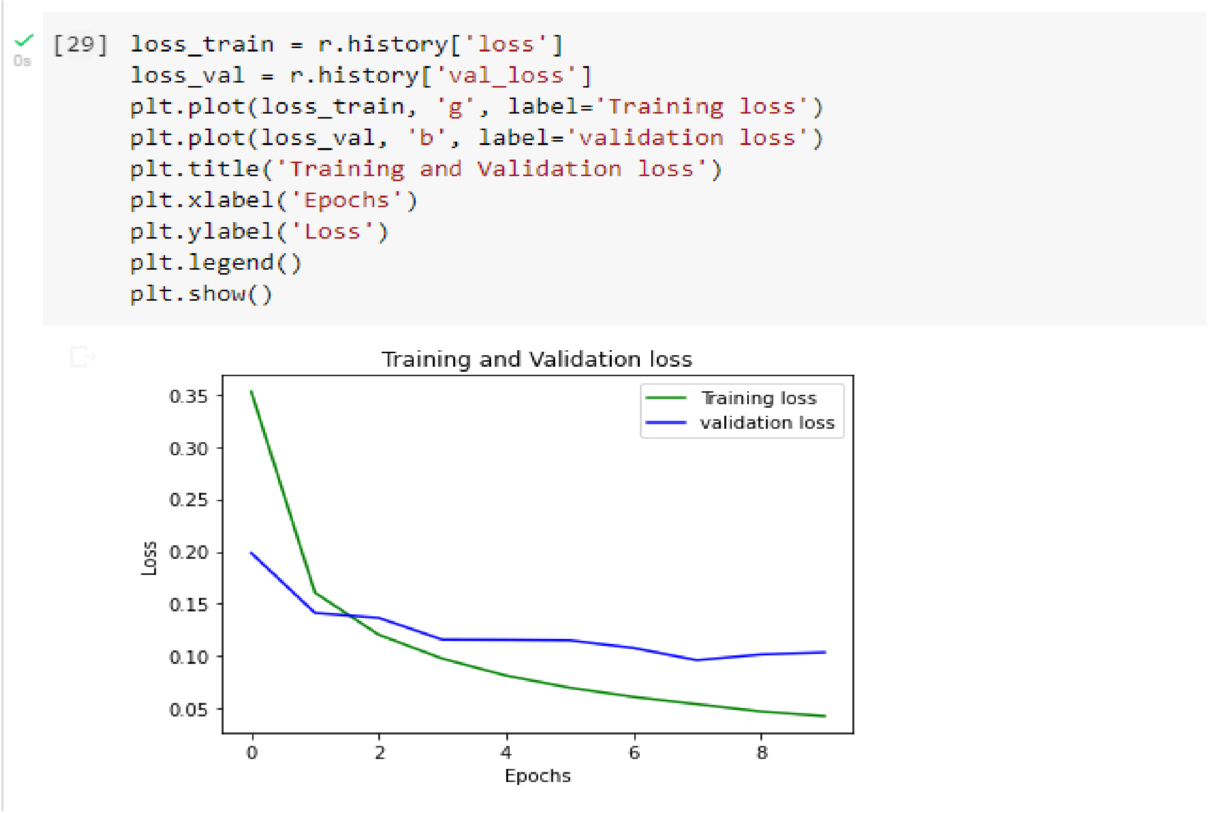
Number features : 10 features.
Name of features : `['0', '1', '2', '3', '4','5', '6', '7', '8', '9']`
Number of samples: 10000

Hyperparameters used in your model

batch size:64
no.epochs=10
`optimizer=adam`

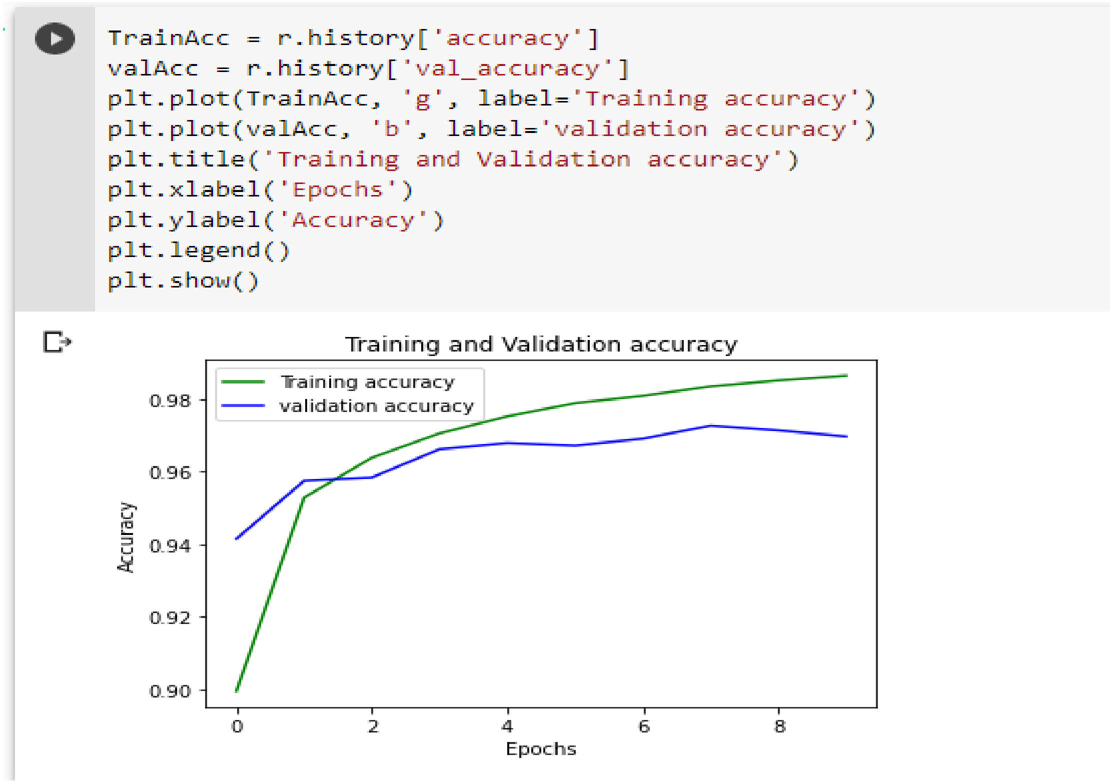
c. Results details:

loss curve



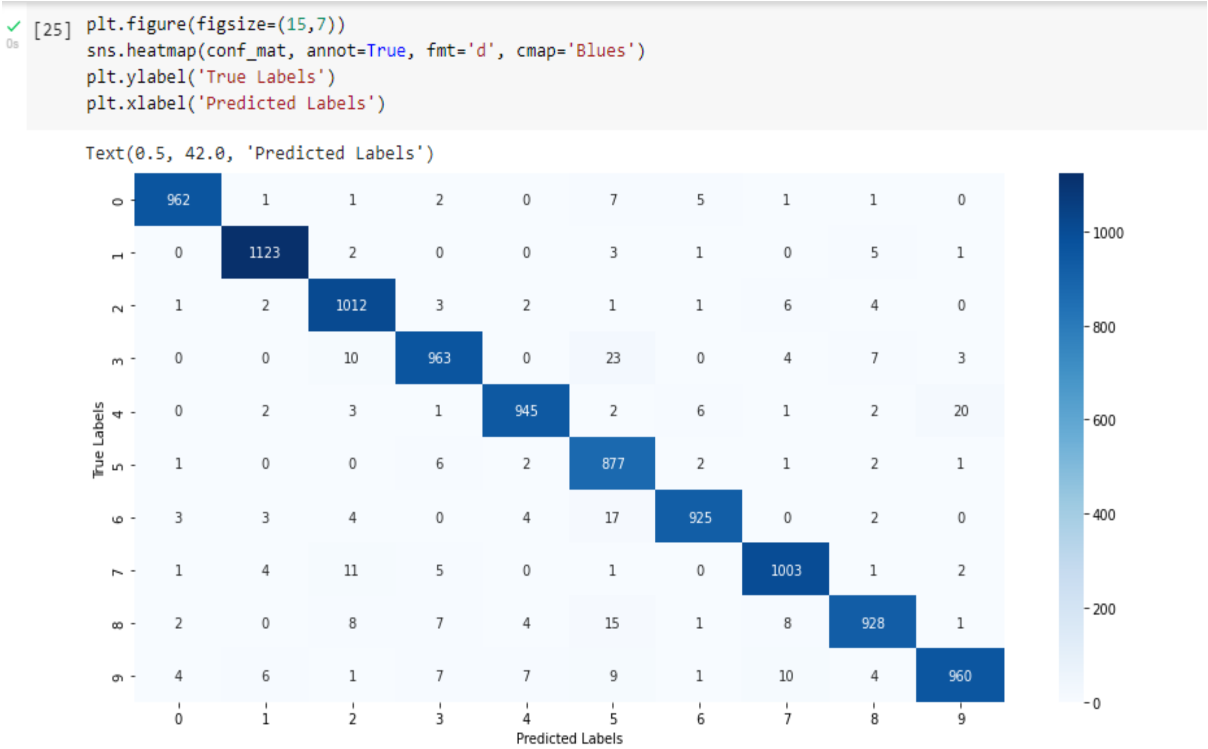
```
loss_train = r.history['loss']
loss_val = r.history['val_loss']
plt.plot(loss_train, 'g', label='Training loss')
plt.plot(loss_val, 'b', label='validation loss')
plt.title('Training and Validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```

accuracy



```
TrainAcc = r.history['accuracy']
valAcc = r.history['val_accuracy']
plt.plot(TrainAcc, 'g', label='Training accuracy')
plt.plot(valAcc, 'b', label='validation accuracy')
plt.title('Training and Validation accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```

confusion matrix

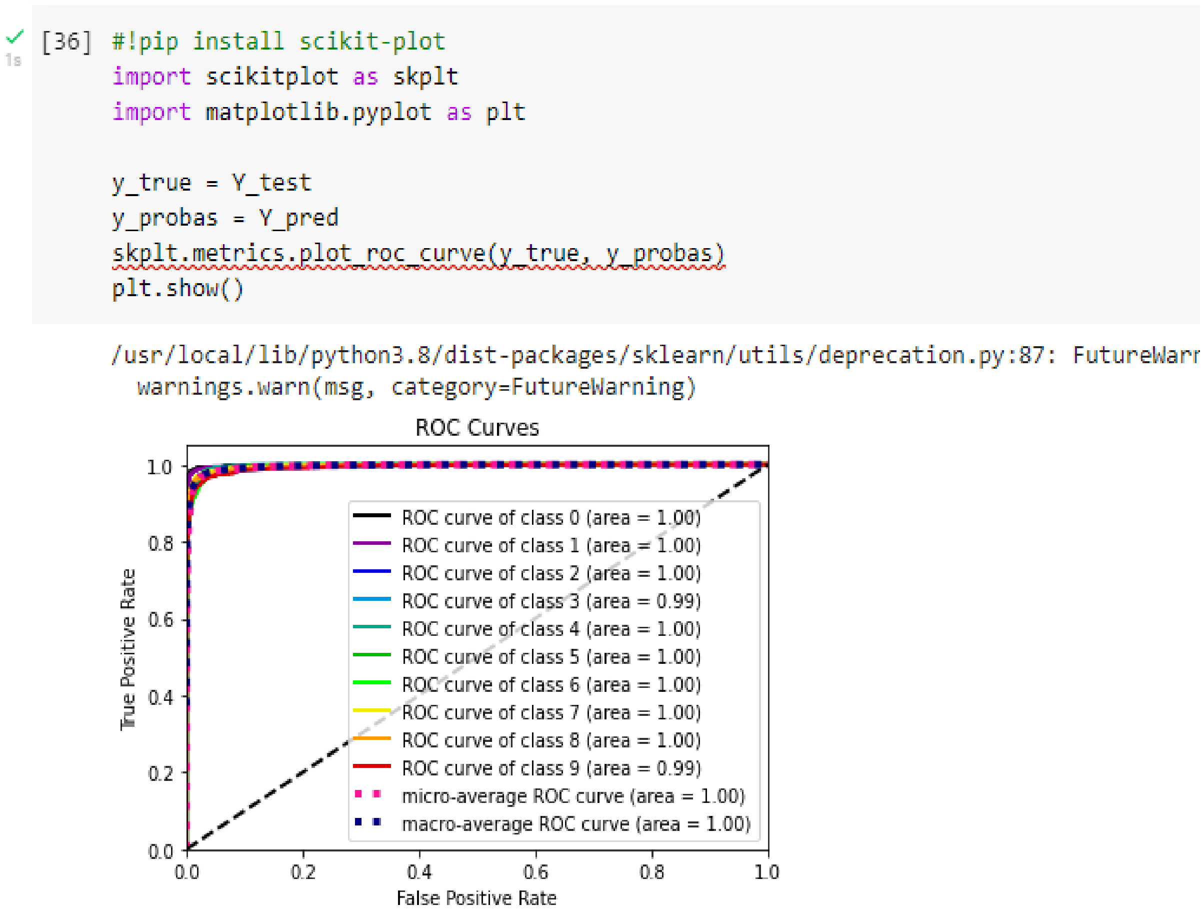


```
[25] plt.figure(figsize=(15,7))
     sns.heatmap(conf_mat, annot=True, fmt='d', cmap='Blues')
     plt.ylabel('True Labels')
     plt.xlabel('Predicted Labels')
```

```
Text(0.5, 42.0, 'Predicted Labels')
```

	0	1	2	3	4	5	6	7	8	9
0	962	1	1	2	0	7	5	1	1	0
1	0	1123	2	0	0	3	1	0	5	1
2	1	2	1012	3	2	1	1	6	4	0
3	0	0	10	963	0	23	0	4	7	3
4	0	2	3	1	945	2	6	1	2	20
5	1	0	0	6	2	877	2	1	2	1
6	3	3	4	0	4	17	925	0	2	0
7	1	4	11	5	0	1	0	1003	1	2
8	2	0	8	7	4	15	1	8	928	1
9	4	6	1	7	7	9	1	10	4	960

ROC curve



```
[36] #!pip install scikit-plot
     import scikitplot as skplt
     import matplotlib.pyplot as plt

     y_true = Y_test
     y_probas = Y_pred
     skplt.metrics.plot_roc_curve(y_true, y_probas)
     plt.show()
```

```
/usr/local/lib/python3.8/dist-packages/sklearn/utils/deprecation.py:87: FutureWarr
  warnings.warn(msg, category=FutureWarning)
```

1s
ROC Curves
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
ROC curve of class 0 (area = 1.00)
ROC curve of class 1 (area = 1.00)
ROC curve of class 2 (area = 1.00)
ROC curve of class 3 (area = 0.99)
ROC curve of class 4 (area = 1.00)
ROC curve of class 5 (area = 1.00)
ROC curve of class 6 (area = 1.00)
ROC curve of class 7 (area = 1.00)
ROC curve of class 8 (area = 1.00)
ROC curve of class 9 (area = 0.99)
micro-average ROC curve (area = 1.00)
macro-average ROC curve (area = 1.00)

SUPPORT VECTOR MACHINE

1- On Numerical dataset

a.*General Information on dataset*

Name of dataset:

https://www.kaggle.com/code/schmoyote/breast-cancer-classification-beginner-friendly/data

Number of classes:

2classes

Total number of samples : 569 samples

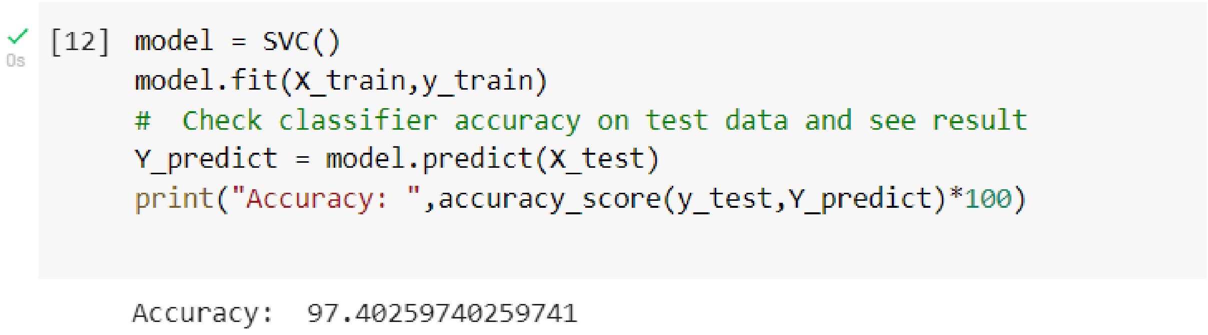
Number of training samples: 398 samples
Number of test samples: 171 samples

b.*Implementation details:*

Number features : 2 features.
Name of features : Real, Positive.
Dimension of resulted features : 30

c. *Results details:*

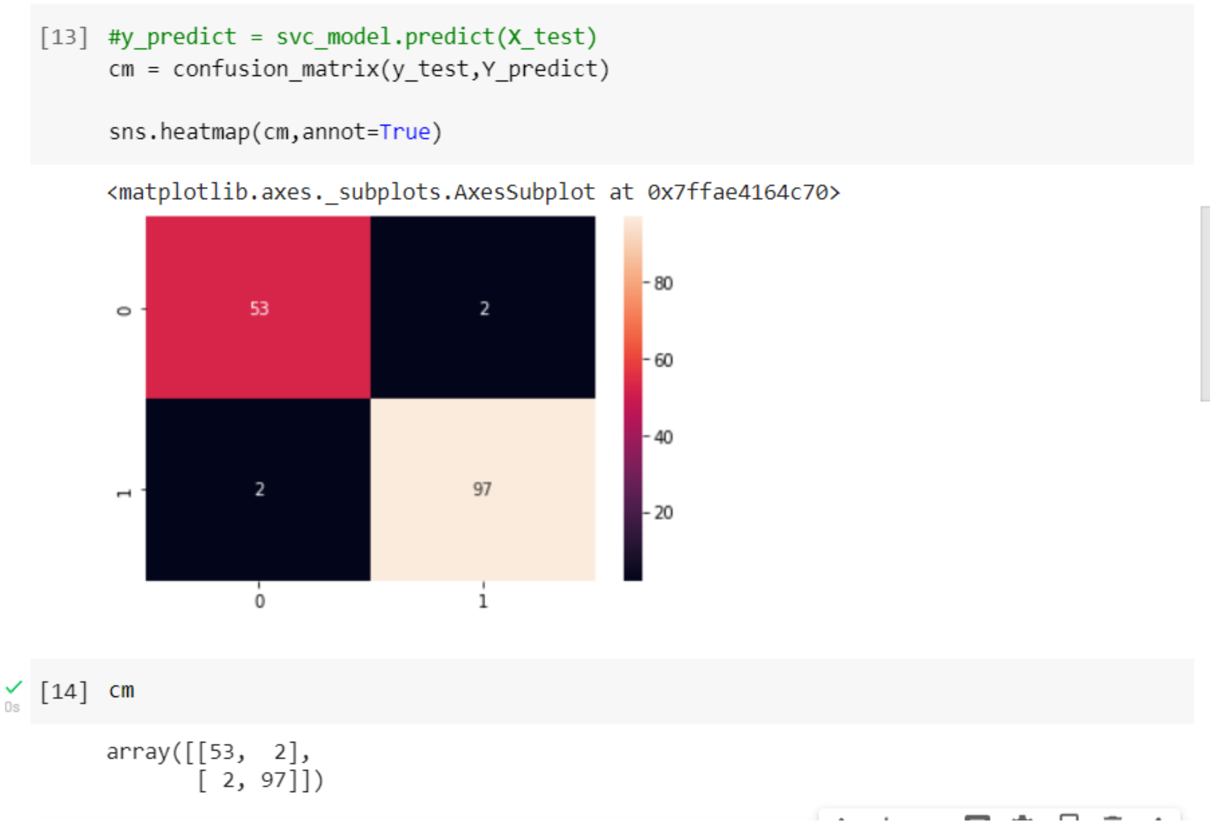
1- accuracy



```
[12] model = SVC()
     model.fit(X_train,y_train)
     #  Check classifier accuracy on test data and see result
     Y_predict = model.predict(X_test)
     print("Accuracy: ",accuracy_score(y_test,Y_predict)*100)
```

```
Accuracy:  97.40259740259741
```

2- confusion matrix



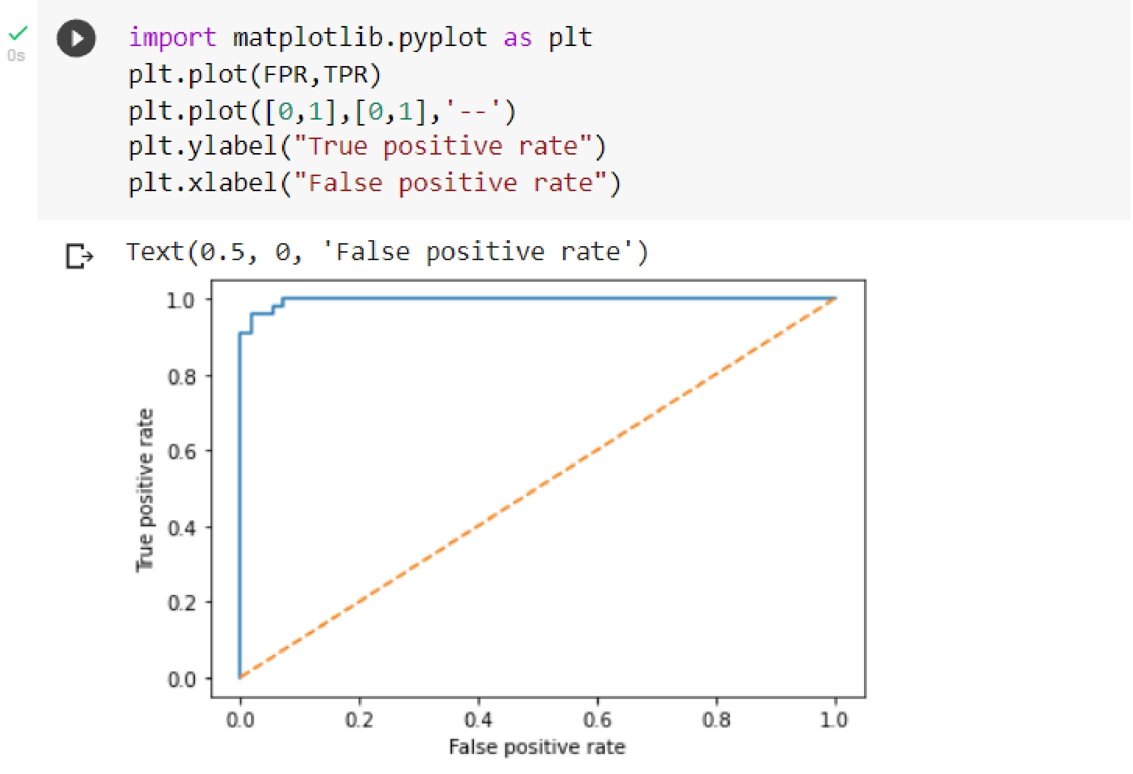
```
[13] #y_predict = svc_model.predict(X_test)
     cm = confusion_matrix(y_test,Y_predict)

     sns.heatmap(cm,annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7ffae4164c70>
```

```
[14] cm
```

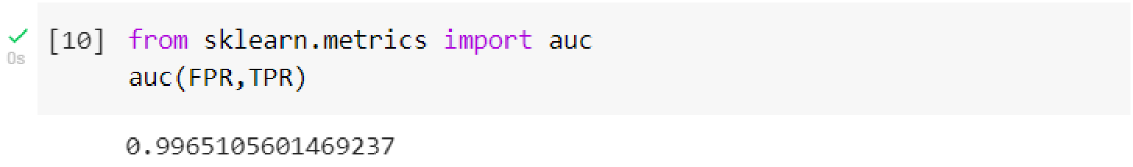
```
array([[53,  2],
       [ 2, 97]])
```

3-ROC Curve



```
import matplotlib.pyplot as plt
plt.plot(FPR,TPR)
plt.plot([0,1],[0,1],'--')
plt.ylabel("True positive rate")
plt.xlabel("False positive rate")
```

```
Text(0.5, 0, 'False positive rate')
```



```
[10] from sklearn.metrics import auc
     auc(FPR,TPR)
```

```
0.9965105601469237
```

2- On image dataset

a. *General Information on dataset*

Name of dataset:

The MNIST database of handwritten digits.
https://www.tensorflow.org/datasets/catalog/mnist

Number of classes:

10 classes

ClassLabel(shape=( ), dtype=int64, num_classes=10)

Total number of samples :

70,000 samples , size of each image 28x28

Image(shape=(28, 28, 1), dtype=uint8)

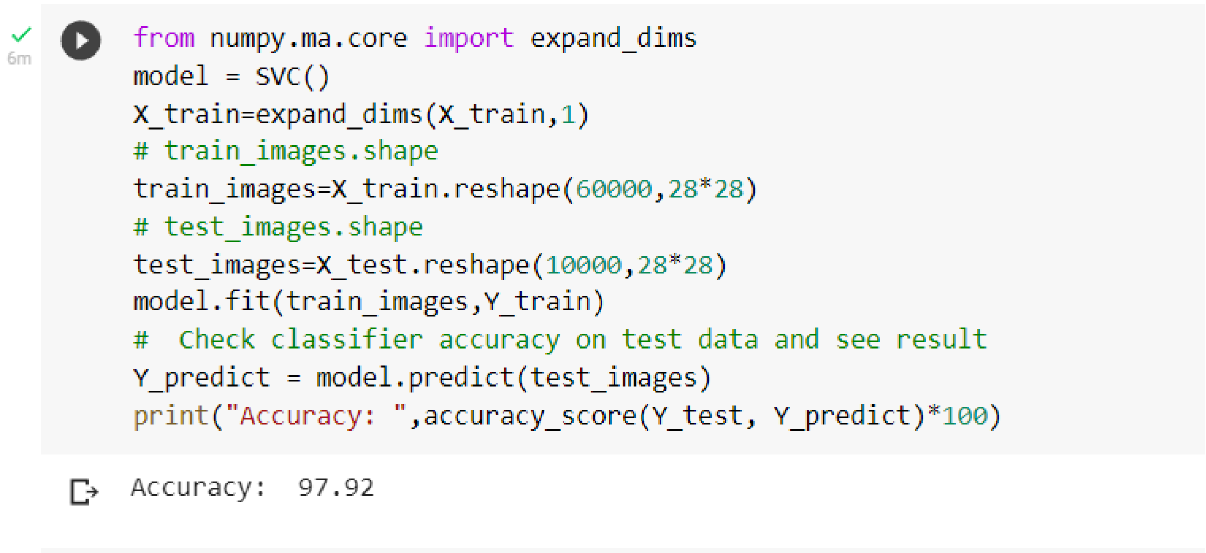
Number of training samples: 60,000 samples
Number of test samples: 10,000 samples

b.*Implementation details:*

Number features : 10 features.
Name of features : `['0', '1', '2', '3', '4','5', '6', '7', '8', '9']`
Number of samples: 10000

c. *Results details:*

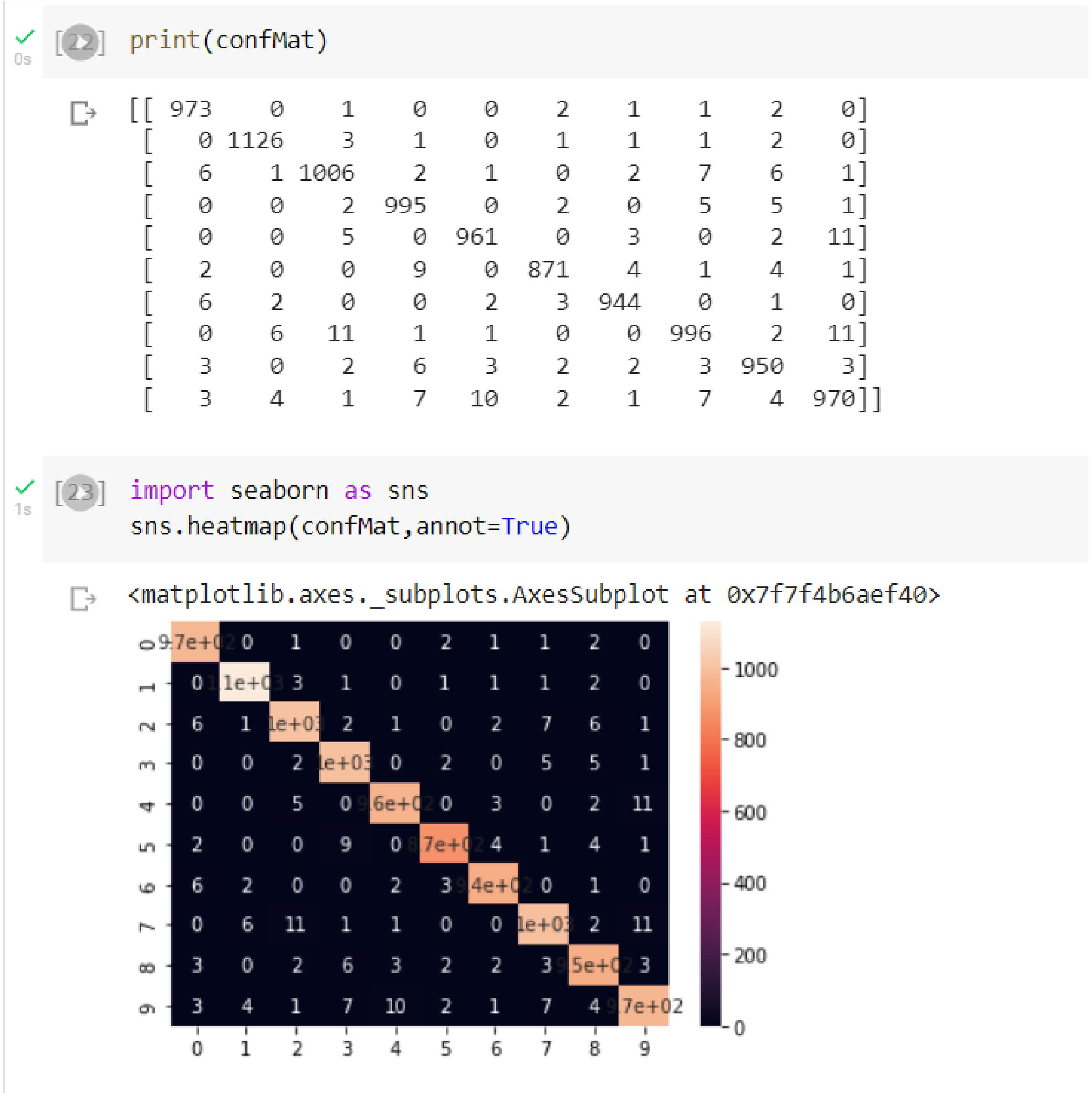
1- accuracy



```
from numpy.ma.core import expand_dims
model = SVC()
X_train=expand_dims(X_train,1)
# train_images.shape
train_images=X_train.reshape(60000,28*28)
# test_images.shape
test_images=X_test.reshape(10000,28*28)
model.fit(train_images,Y_train)
#  Check classifier accuracy on test data and see result
Y_predict = model.predict(test_images)
print("Accuracy: ",accuracy_score(Y_test, Y_predict)*100)
```

```
Accuracy:  97.92
```

2- confusion matrix



```
[22] print(confMat)
```

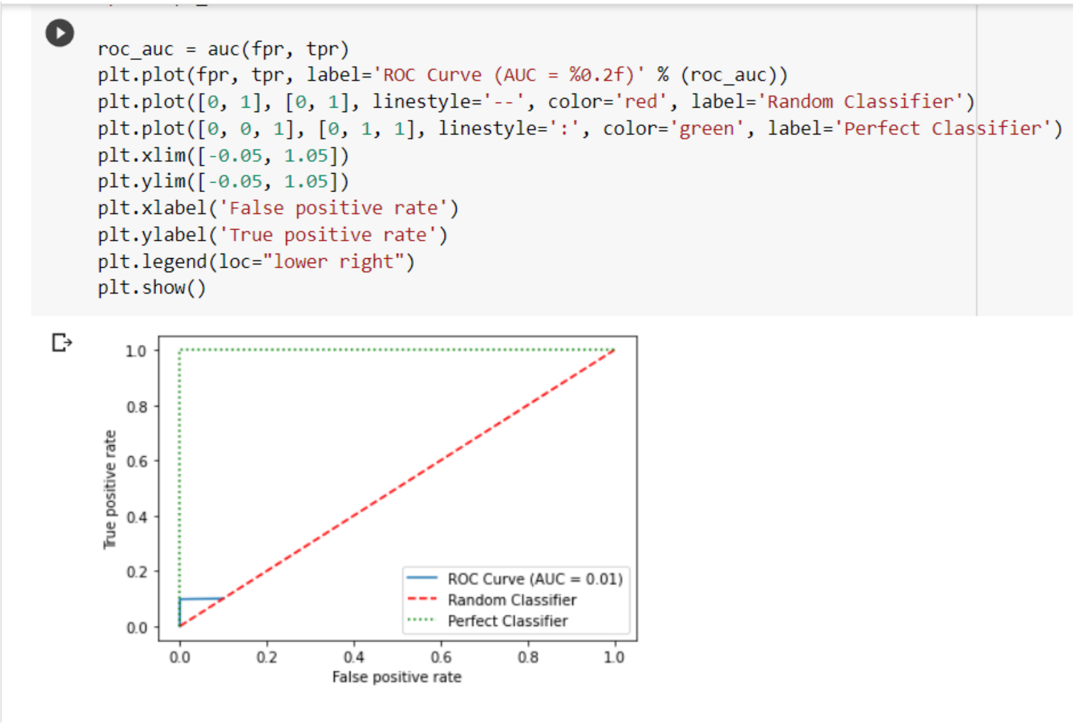
```
[[ 973    0    1    0    0    2    1    1    2    0]
 [    0 1126    3    1    0    1    1    1    2    0]
 [    6    1 1006    2    1    0    2    7    6    1]
 [    0    0    2  995    0    2    0    5    5    1]
 [    0    0    5    0  961    0    3    0    2   11]
 [    2    0    0    9    0  871    4    1    4    1]
 [    6    2    0    0    2    3  944    0    1    0]
 [    0    6   11    1    1    0    0  996    2   11]
 [    3    0    2    6    3    2    2    3  950    3]
 [    3    4    1    7   10    2    1    7    4  970]]
```

```
[23] import seaborn as sns
     sns.heatmap(confMat,annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7f4b6aef40>
```

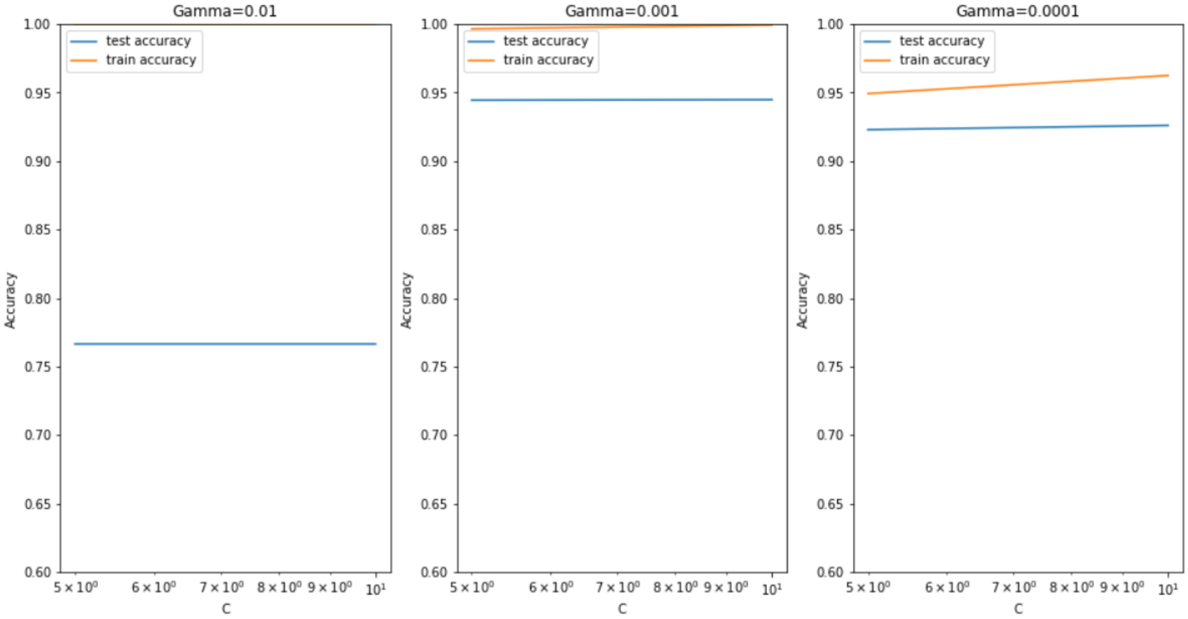
9.7e+02 0 1 0 0 2 1 1 2 0
0 1e+03 3 1 0 1 1 1 2 0
6 1 1e+03 2 1 0 2 7 6 1
0 0 2 1e+03 0 2 0 5 5 1
0 0 5 0 9.6e+02 0 3 0 2 11
2 0 0 9 0 8.7e+02 4 1 4 1
6 2 0 0 2 3 9.4e+02 0 1 0
0 6 11 1 1 0 0 1e+03 2 11
3 0 2 6 3 2 2 3 9.5e+02 3
3 4 1 7 10 2 1 7 4 9.7e+02
0 1 2 3 4 5 6 7 8 9
1000
800
600
400
200
0

3-ROC Curve



```
roc_auc = auc(fpr, tpr)
plt.plot(fpr, tpr, label='ROC Curve (AUC = %0.2f)' % (roc_auc))
plt.plot([0, 1], [0, 1], linestyle='--', color='red', label='Random Classifier')
plt.plot([0, 0, 1], [0, 1, 1], linestyle=':', color='green', label='Perfect Classifier')
plt.xlim([-0.05, 1.05])
plt.ylim([-0.05, 1.05])
plt.xlabel('False positive rate')
plt.ylabel('True positive rate')
plt.legend(loc="lower right")
plt.show()
```

4-accuracy curve



Gamma=0.01
Gamma=0.001
Gamma=0.0001
test accuracy
train accuracy
Accuracy
5×10⁰
6×10⁰
7×10⁰
8×10⁰
9×10⁰
10¹
C