INTRODUCTION TO MACHINE LEARNING

LAB ASSIGNMENT – 5

NAME : PRATHAPANI SATWIKA

REG.NO. :20BCD7160

LOGISTIC REGRESSION

BINARY CLASSIFICATION

```
[1]  import matplotlib.pyplot as plt
     import pandas as pd
     import numpy as np
     from sklearn.linear_model import LogisticRegression
     from sklearn.model_selection import train_test_split
     from sklearn.datasets import load_breast_cancer
     from sklearn.metrics import classification_report, confusion_matrix
```

```
cancer = load_breast_cancer()
print(cancer)
```

```
{'data': array([[1.799e+01, 1.038e+01, 1.228e+02, ..., 2.654e-01, 4.601e-01,
        1.189e-01],
       [2.057e+01, 1.777e+01, 1.329e+02, ..., 1.860e-01, 2.750e-01,
        8.902e-02],
       [1.969e+01, 2.125e+01, 1.300e+02, ..., 2.430e-01, 3.613e-01,
        8.758e-02],
       ...,
       [1.660e+01, 2.808e+01, 1.083e+02, ..., 1.418e-01, 2.218e-01,
        7.820e-02],
       [2.060e+01, 2.933e+01, 1.401e+02, ..., 2.650e-01, 4.087e-01,
        1.240e-01],
       [7.760e+00, 2.454e+01, 4.792e+01, ..., 0.000e+00, 2.871e-01,
        7.039e-02]]), 'target': array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0,
       1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0,
       1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1,
       1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0,
       0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1,
       1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0,
       0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0,
       1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1,
       1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0,
       0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0,
```

```
0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0,
1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1,
1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1,
1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0,
1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1,
1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1,
1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1]), 'frame': None, 'target_names': array(['malignant', 'benign'], dtype='<U9'), 'DESCR':
'mean smoothness', 'mean compactness', 'mean concavity',
'mean concave points', 'mean symmetry', 'mean fractal dimension',
'radius error', 'texture error', 'perimeter error', 'area error',
'smoothness error', 'compactness error', 'concavity error',
'concave points error', 'symmetry error',
'fractal dimension error', 'worst radius', 'worst texture',
'worst perimeter', 'worst area', 'worst smoothness',
'worst compactness', 'worst concavity', 'worst concave points',
'worst symmetry', 'worst fractal dimension'], dtype='<U23'), 'filename': 'breast_cancer.csv', 'data_module': 'sklearn.datasets.data'}
```

```
[14] x_data = cancer.data[:,:1] # mean radius
     labels = cancer.target
```

```
[15] X_train, X_test, y_train, y_test = train_test_split(x_data, labels, test_size=0.2, random_state=0)
```

```
[16] model = LogisticRegression(solver='liblinear', random_state=0)
     H = model.fit(X_train, y_train)
```

```
print('Logistic Regression Model Coeff (m) =' , model.coef_)
print('Logistic Regression Model Coeff (b) =' , model.intercept_)
```

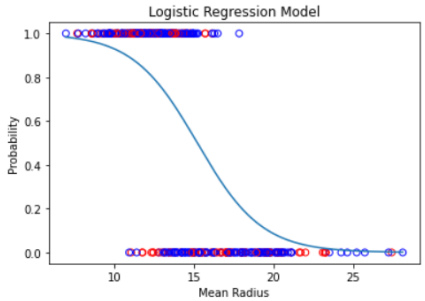
```
Logistic Regression Model Coeff (m) = [[-0.49524206]]
Logistic Regression Model Coeff (b) = [7.54061572]
```

```
y_predict=model.predict(X_test)
print(y_predict)
```

```
[1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 1 0 0 0 1 0 1 1 0 1 1 1 1 0 1 0 1 0 1 1 1
 0 1 1 1 1 1 0 1 1 0 1 1 1 1 1 0 1 0 1 1 1 1 1 1 0 1 0 1 1 0 1 0 0 0 1 1 0 1 1
 0 1 1 1 1 1 1 0 0 0 1 0 1 1 1 0 0 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 0 1 0 1 0 0 1
 0 0 1]
```

```
colors = {0:'red', 1:'blue'}
def sigmoid(x):
    return (1 / (1 +np.exp(-(model.intercept_[0] +(model.coef_[0][0] * x)))))
x1 = np.arange(np.min(X_train), np.max(X_train), 0.01)
y1 = [sigmoid(n) for n in x1]
plt.scatter(X_train,y_train,facecolors='none',edgecolors=pd.DataFrame(cancer.target)[0].apply(lambda X_train:colors[X_train]),cmap=colors)
plt.plot(x1,y1)
plt.xlabel("Mean Radius")
plt.ylabel("Probability")
plt.title('Logistic Regression Model')
```

```
Text(0.5, 1.0, 'Logistic Regression Model')
```



```
print("\nPrediction Probability : \n",model.predict_proba(X_test))

print("\nPrediction : ",model.predict(X_test))

print("\nScore : ",model.score(X_test, y_test))
```

```
Prediction Probability :
 [[0.28815287 0.71184713]
 [0.26924273 0.73075727]
 [0.35495662 0.64504338]
 [0.38261492 0.61738508]
 [0.25206829 0.74793171]
 [0.12735274 0.87264726]
 [0.17179642 0.82820358]
 [0.14851656 0.85148344]
 [0.02382539 0.97617461]
 [0.07875153 0.92124847]
 [0.43400977 0.56599023]
 [0.42551428 0.57448572]
 [0.05280865 0.94719135]
 [0.6898163  0.3101837 ]
 [0.4279373  0.5720627 ]
 [0.52892432 0.47107568]
 [0.04467502 0.95532498]
 [0.93193494 0.06806506]
 [0.89485926 0.10514074]
 [0.94426261 0.05573739]
 [0.1574998  0.8425002 ]
 [0.62295633 0.37704367]
 [0.29531567 0.70468433]
 [0.1574998  0.8425002 ]
```

```
[0.62295633 0.37704367]
[0.29531567 0.70468433]
[0.1574998  0.8425002 ]
[0.95446663 0.04553337]
[0.16350487 0.83649513]
[0.10503629 0.89496371]
[0.32279815 0.67720185]
[0.28815287 0.71184713]
[0.85259212 0.14740788]
[0.23208687 0.76791313]
[0.93379578 0.06620422]
[0.26247677 0.73752323]
[0.61360614 0.38639386]
[0.06192253 0.93807747]
[0.22081103 0.77918897]
[0.27512902 0.72487098]
[0.86634062 0.13365938]
[0.45479412 0.54520588]
[0.48191654 0.51808346]
[0.48810143 0.51189857]
[0.04133403 0.95866597]
[0.59704369 0.40295631]
[0.08093447 0.91906553]
[0.27217591 0.72782409]
```

```
[0.95838557 0.04161443]
[0.03745748 0.96254252]
[0.35609135 0.64390865]
[0.15553836 0.84446164]
[0.41585901 0.58414099]
[0.87579239 0.12420761]
[0.45479412 0.54520588]
[0.8315427  0.1684573 ]
[0.14236215 0.85763785]
[0.20263607 0.79736393]
[0.134681   0.865319  ]
[0.12625603 0.87374397]
[0.18785697 0.81214303]
[0.18115121 0.81884879]
[0.96198065 0.03801935]
[0.32714365 0.67285635]
[0.60298644 0.39701356]
[0.11674528 0.88325472]
[0.23742477 0.76257523]
[0.76416379 0.23583621]
[0.17109292 0.82890708]
[0.97035663 0.02964337]
[0.75144068 0.24855932]
[0.93410129 0.06589871]
```

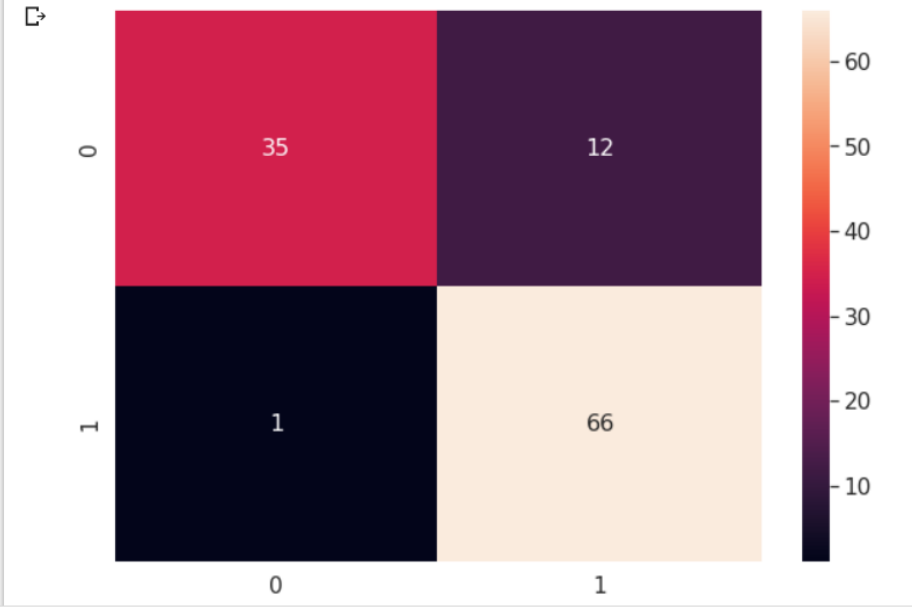
```
[0.05451874 0.94548126]
[0.40149852 0.59850148]
[0.61007769 0.38992231]
[0.13759302 0.86240698]
[0.42309483 0.57690517]
[0.84496559 0.15503441]
[0.24927759 0.75072241]
[0.03571225 0.96428775]
[0.25394023 0.74605977]
[0.14602885 0.85397115]
[0.03859877 0.96140123]
[0.66382854 0.33617146]
[0.92080748 0.07919252]
[0.56814924 0.43185076]
[0.20024611 0.79975389]
[0.59823458 0.40176542]
[0.19166408 0.80833592]
[0.2984169  0.7015831 ]
[0.02983998 0.97016002]
[0.72563651 0.27436349]
[0.93470838 0.06529162]
[0.05552294 0.94447706]
[0.4868641  0.5131359 ]
[0.47079898 0.52920102]
[0.49924213 0.50075787]
[0.05300718 0.94699282]
[0.06601335 0.93398665]
[0.53385718 0.46614282]
[0.23922275 0.76077725]
[0.14055777 0.85944223]
[0.132389   0.867611  ]
[0.04153071 0.95846929]
[0.19475193 0.80524807]
[0.46833194 0.53166806]
```

```
[0.35836566 0.64163434]
[0.8322353  0.1677647 ]
[0.04970286 0.95029714]
[0.79070325 0.20929675]
[0.23832259 0.76167741]
[0.54738559 0.45261441]
[0.74772227 0.25227773]
[0.22683328 0.77316672]
[0.92080748 0.07919252]
[0.82160602 0.17839398]
[0.47697253 0.52302747]]

Prediction :  [1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 1 0 0 0 1 0 1 1 0 1 1 1 1 0 1 0 1 0 1 1 1
 0 1 1 1 1 0 1 1 0 1 1 1 1 0 1 0 1 1 1 1 1 1 0 1 0 1 1 0 1 0 0 0 1 1 0 1 1
 0 1 1 1 1 1 0 0 0 1 0 1 1 1 0 0 1 1 1 1 1 1 0 1 1 1 1 1 1 1 0 1 0 1 0 0 1
 0 0 1]

Score :  0.8859649122807017
```

```
print("\nConfusion Matrix : \n",confusion_matrix(y_test, model.predict(X_test)))
```

```
Confusion Matrix :
 [[35 12]
 [ 1 66]]
```

```
cm = confusion_matrix(y_test, model.predict(X_test))
import seaborn as sn
import pandas as pd
df_cm = pd.DataFrame(cm, range(2), range(2))
plt.figure(figsize=(10,7))
sn.set(font_scale=1.4)
sn.heatmap(df_cm, annot=True, annot_kws={"size": 16})
plt.show()
```



```
print(classification_report(y_test, model.predict(X_test)))
```

```
              precision    recall  f1-score   support

           0       0.97      0.74      0.84        47
           1       0.85      0.99      0.91        67

    accuracy                           0.89       114
   macro avg       0.91      0.86      0.88       114
weighted avg       0.90      0.89      0.88       114
```

MULTI-CLASS CLASSIFICATION

```
[22] import matplotlib.pyplot as plt
     import pandas as pd
     import numpy as np
     from sklearn.linear_model import LogisticRegression
     from sklearn.model_selection import train_test_split
     from sklearn.datasets import load_iris
     from sklearn.metrics import classification_report, confusion_matrix
```

```
iris = load_iris()
print(iris)
```

```
       [5.7, 2.8, 4.1, 1.3],
       [6.3, 3.3, 6. , 2.5],
       [5.8, 2.7, 5.1, 1.9],
       [7.1, 3. , 5.9, 2.1],
       [6.3, 2.9, 5.6, 1.8],
       [6.5, 3. , 5.8, 2.2],
       [7.6, 3. , 6.6, 2.1],
       [4.9, 2.5, 4.5, 1.7],
       [7.3, 2.9, 6.3, 1.8],
       [6.7, 2.5, 5.8, 1.8],
       [7.2, 3.6, 6.1, 2.5],
       [6.5, 3.2, 5.1, 2. ],
       [6.4, 2.7, 5.3, 1.9],
       [6.8, 3. , 5.5, 2.1],
       [5.7, 2.5, 5. , 2. ],
       [5.8, 2.8, 5.1, 2.4],
       [6.4, 3.2, 5.3, 2.3],
       [6.5, 3. , 5.5, 1.8],
       [7.7, 3.8, 6.7, 2.2],
       [7.7, 2.6, 6.9, 2.3],
       [6. , 2.2, 5. , 1.5],
```

```
[6.9, 3.2, 5.7, 2.3],
[5.6, 2.8, 4.9, 2. ],
[7.7, 2.8, 6.7, 2. ],
[6.3, 2.7, 4.9, 1.8],
[6.7, 3.3, 5.7, 2.1],
[7.2, 3.2, 6. , 1.8],
[6.2, 2.8, 4.8, 1.8],
[6.1, 3. , 4.9, 1.8],
[6.4, 2.8, 5.6, 2.1],
[7.2, 3. , 5.8, 1.6],
[7.4, 2.8, 6.1, 1.9],
[7.9, 3.8, 6.4, 2. ],
[6.4, 2.8, 5.6, 2.2],
[6.3, 2.8, 5.1, 1.5],
[6.1, 2.6, 5.6, 1.4],
[7.7, 3. , 6.1, 2.3],
[6.3, 3.4, 5.6, 2.4],
[6.4, 3.1, 5.5, 1.8],
[6. , 3. , 4.8, 1.8],
[6.9, 3.1, 5.4, 2.1],
[6.7, 3.1, 5.6, 2.4],
[6.9, 3.1, 5.1, 2.3],
[5.8, 2.7, 5.1, 1.9],
[6.8, 3.2, 5.9, 2.3],
[6.7, 3.3, 5.7, 2.5],
[6.7, 3. , 5.2, 2.3],
[6.3, 2.5, 5. , 1.9],
[6.5, 3. , 5.2, 2. ],
[6.2, 3.4, 5.4, 2.3],
[5.9, 3. , 5.1, 1.8]]), 'target': 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, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2]), 'frame': None, 'target_names': array(['setosa', 'versicolor', 'virginica'], dtype='<U10'),
```

```
x_data = iris.data
labels = iris.target
```

```
[25] X_train, X_test, y_train, y_test = train_test_split(x_data, labels, test_size=0.2, random_state=0)
```

```
[26] model = LogisticRegression(solver='newton-cg', multi_class='multinomial', random_state=0)
     H = model.fit(X_train, y_train)
```

```
[29] print('Logistic Regression Model Coeff (m) =' , model.coef_)
     print('Logistic Regression Model Coeff (b) =' , model.intercept_)
```

```
Logistic Regression Model Coeff (m) = [[-0.39407173  0.84882842 -2.36093351 -1.01257159]
 [ 0.41036428 -0.3175274  -0.14620809 -0.78740226]
 [-0.01629256 -0.53130101  2.50714161  1.79997385]]
Logistic Regression Model Coeff (b) = [  9.2828424    2.30752786 -11.59037026]
```

```
y_predict=model.predict(X_test)
print(y_predict)
```

```
[2 1 0 2 0 2 0 1 1 1 2 1 1 1 1 0 1 1 0 0 2 1 0 0 2 0 0 1 1 0]
```

```
print("\nPrediction Probability : \n",model.predict_proba(X_test))

print("\nPrediction : ",model.predict(X_test))

print("\nScore : ",model.score(X_test, y_test))
```

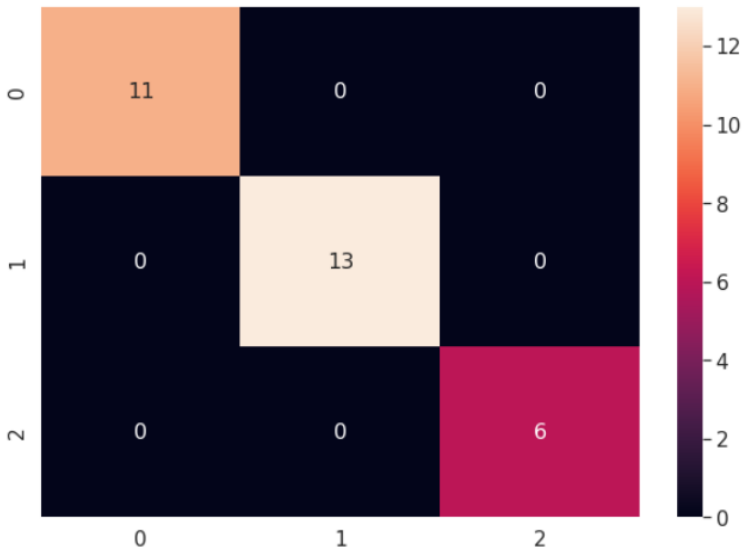
```
Prediction Probability :
 [[1.12598294e-04 5.89208697e-02 9.40966532e-01]
 [1.22106488e-02 9.64585183e-01 2.32041686e-02]
 [9.86665271e-01 1.33346960e-02 3.29728602e-08]
 [1.19530325e-06 2.31815344e-02 9.76817270e-01]
 [9.72121298e-01 2.78785702e-02 1.32007142e-07]
 [1.85378681e-06 6.07638023e-03 9.93921766e-01]
 [9.83515008e-01 1.64849341e-02 5.82145260e-08]
 [2.89680225e-03 7.41944624e-01 2.55158574e-01]
 [1.53481473e-03 7.37586069e-01 2.60879116e-01]
 [2.06193052e-02 9.38045463e-01 4.13352313e-02]
 [9.54078540e-05 1.93896600e-01 8.06007992e-01]
 [7.04941640e-03 8.10539859e-01 1.82410724e-01]
 [3.90131353e-03 8.17846603e-01 1.78252084e-01]
 [3.07798832e-03 7.63177315e-01 2.33744697e-01]
 [3.75939142e-03 7.33638843e-01 2.62601766e-01]
 [9.83846350e-01 1.61536044e-02 4.52851568e-08]
 [6.53945108e-03 7.73357370e-01 2.20103179e-01]
 [1.03944692e-02 8.73778825e-01 1.15826706e-01]
 [9.69079198e-01 3.09206291e-02 1.72546152e-07]
 [9.85384272e-01 1.46156764e-02 5.17460521e-08]
 [8.24659942e-04 2.15603241e-01 7.83572099e-01]
 [9.75685415e-03 7.44302896e-01 2.45940250e-01]
 [9.44096659e-01 5.59025073e-02 8.33203747e-07]
 [9.75218721e-01 2.47811492e-02 1.29363584e-07]
 [1.34605652e-03 4.36836272e-01 5.61817672e-01]
```

```
 [9.94589718e-01 5.41027402e-03 7.72553157e-09]
 [9.52879342e-01 4.71197063e-02 9.51775326e-07]
 [1.06292679e-02 9.06333736e-01 8.30369960e-02]
 [1.29698661e-01 8.65427935e-01 4.87340365e-03]
 [9.63914293e-01 3.60853333e-02 3.73790104e-07]]

Prediction :  [2 1 0 2 0 2 0 1 1 1 2 1 1 1 1 0 1 1 0 0 2 1 0 0 2 0 0 1 1 0]

Score :  1.0
```

```
print("\nConfusion Matrix : \n",confusion_matrix(y_test, model.predict(X_test)))
```

```
Confusion Matrix :
 [[11  0  0]
 [ 0 13  0]
 [ 0  0  6]]
```

```
cm = confusion_matrix(y_test, model.predict(X_test))

import seaborn as sn
import pandas as pd

df_cm = pd.DataFrame(cm, range(3), range(3))
plt.figure(figsize=(10,7))
sn.set(font_scale=1.4) # for label size
sn.heatmap(df_cm, annot=True, annot_kws={"size": 16}) # font size

plt.show()
```



```
[34] print(classification_report(y_test, model.predict(X_test)))
```

```
                  precision    recall  f1-score   support

               0       1.00      1.00      1.00        11
               1       1.00      1.00      1.00        13
               2       1.00      1.00      1.00         6

        accuracy                           1.00        30
       macro avg       1.00      1.00      1.00        30
    weighted avg       1.00      1.00      1.00        30
```