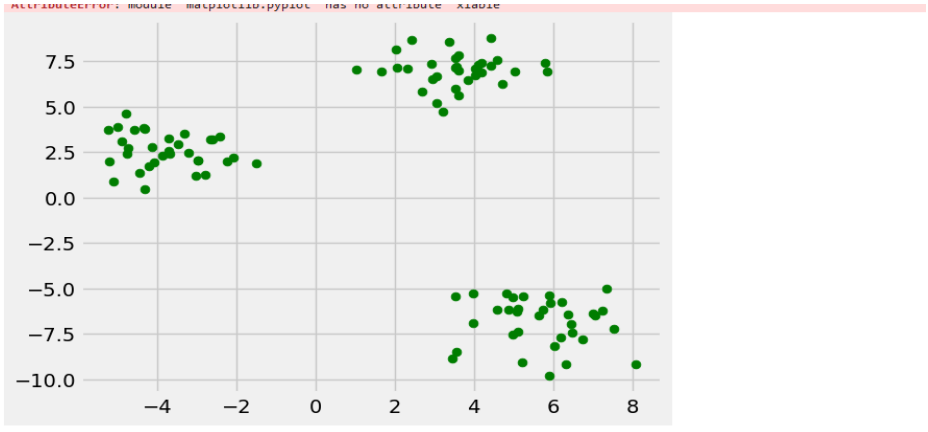
MACHINE LEARNING PROGREMS OUTPUT VERIFIED

PROGREAM: 01

```
from sklearn.datasets import make_blobs

from matplotlib import pyplot as plt

from matplotlib import style


from sklearn.datasets import make_blobs

from matplotlib import pyplot as plt

from matplotlib import style

 style.use("fivethirtyeight")

 X, y = make_blobs(n_samples = 100, centers = 3,

             cluster_std = 1, n_features = 2)

 plt.scatter(X[:, 0], X[:, 1], s = 40, color = 'g')

plt.xlabel("X")

plt.ylabel("Y")

 plt.show()

plt.clf()
```



PROGREAM: 02

```
from IPython.display import HTML

import pandas as pd

import numpy as np

import base64


def create_download_link(df, title = "Download CSV file", filename = "data.csv"):

    csv = df.to_csv()

    b64 = base64.b64encode(csv.encode())

    payload = b64.decode()

    html = '<a download="{filename}" href="data:text/csv;base64,{payload}" target="_blank">{title}</a>'

    html = html.format(payload=payload,title=title,filename=filename)

    return HTML(html)

df = pd.DataFrame(np.random.randn(50, 4), columns=list('ABCD'))

create_download_link(df)
```

PROGREAM: 03

```
from pandas import read_csv

from numpy import set_printoptions

from sklearn.feature_selection import SelectKBest

from sklearn.feature_selection import chi2


path = r'D:\squad\book.csv'


names = ['preg', 'plas', 'hello']

dataframe = read_csv(path, names=names)

array = dataframe.values

print(dataframe)
```

```
        preg     plas     pres     skin   test  mass  pedi  age  class
NaN     preg     plas     pres     skin   test  mass  pedi  age  class
0.0  variance  skewness  Curtosis  Entropy  class   NaN   NaN  NaN    NaN
1.0    3.6216     8.661   -2.8073  -0.4469      0   NaN   NaN  NaN    NaN
2.0    4.5459    8.1674   -2.4586   -1.462      0   NaN   NaN  NaN    NaN
3.0     3.866   -2.6383    1.9242  0.10645      0   NaN   NaN  NaN    NaN
4.0    3.4566    9.5228   -4.0112   -3.594      0   NaN   NaN  NaN    NaN
```

PROGREAM: 04

```
from sklearn.metrics import confusion_matrix

from sklearn.metrics import accuracy_score

from sklearn.metrics import classification_report

from sklearn.metrics import roc_auc_score

from sklearn.metrics import log_loss

X_actual = [1, 1, 0, 1, 0, 0, 1, 0, 0, 0]

Y_predic = [1, 0, 1, 1, 1, 0, 1, 1, 0, 0]

results = confusion_matrix(X_actual, Y_predic)

print ('Confusion Matrix :')

print(results)

print ('Accuracy Score is',accuracy_score(X_actual, Y_predic))

print ('Classification Report : ')

print (classification_report(X_actual, Y_predic))

print('AUC-ROC:',roc_auc_score(X_actual, Y_predic))

print('LOGLOSS Value is',log_loss(X_actual, Y_predic))
```

```
Confusion Matrix :
[[3 3]
 [1 3]]
Accuracy Score is 0.6
Classification Report :
              precision    recall  f1-score   support

           0       0.75      0.50      0.60         6
           1       0.50      0.75      0.60         4

    accuracy                           0.60        10
   macro avg       0.62      0.62      0.60        10
weighted avg       0.65      0.60      0.60        10

AUC-ROC: 0.625
LOGLOSS Value is 14.41746135564686
```

PROGREAM: 05

```
from sklearn.datasets import load_iris

iris = load_iris()

X = iris.data

y=iris.target

 from sklearn.model_selection import train_test_split

X_train,X_test,y_train, y_test=train_test_split(X,y,test_size=0.4,random_state=1)

 from sklearn.naive_bayes import GaussianNB

gnb=GaussianNB()

gnb.fit(X_train, y_train)

 y_pred=gnb.predict(X_test)

from sklearn import metrics

print("GaussianNaiveBayes model accuracy(in %):", metrics.accuracy_score(y_test,y_pred)*100)
```

```
GaussianNaiveBayes model accuracy(in %): 95.0
```

```
[ ]:
```

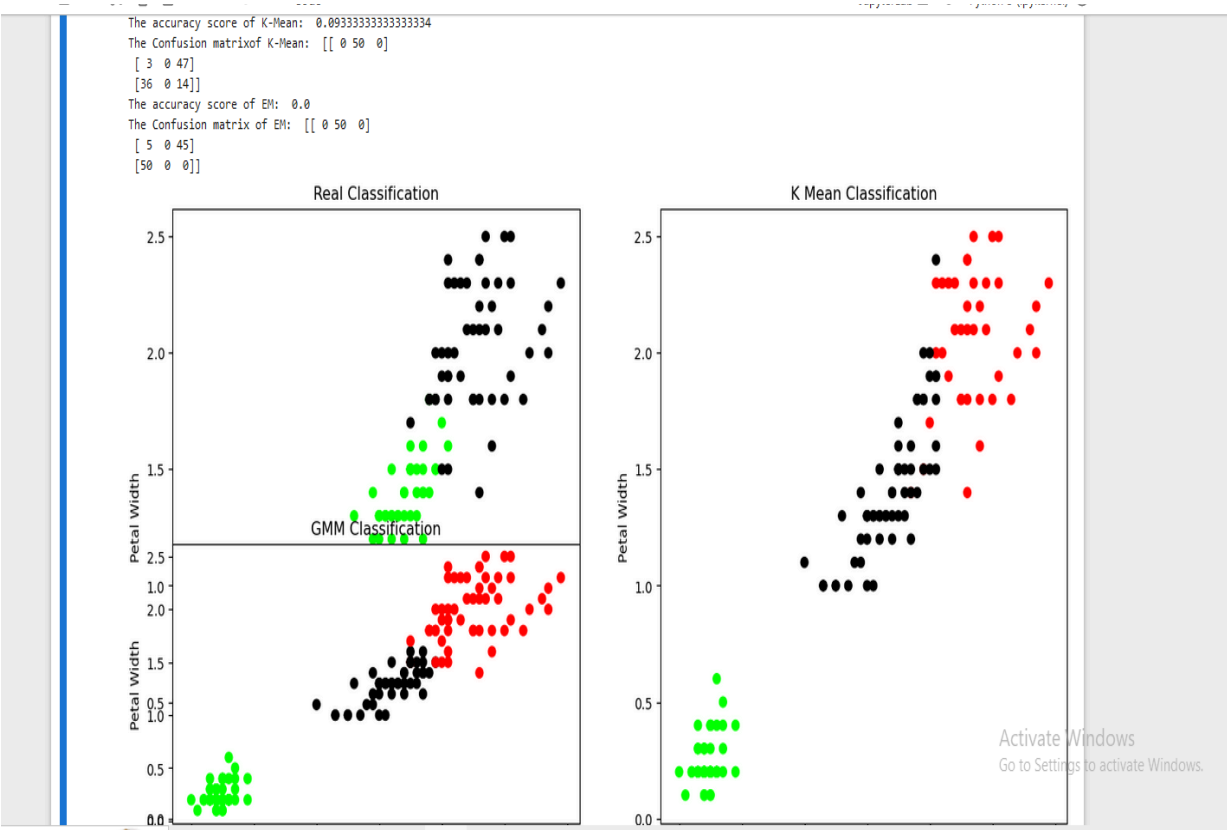
PROGREAM: 06

PROGREAM: 07

```
import matplotlib.pyplot as plt

from sklearn import datasets
```

```
from sklearn.cluster import KMeans
import sklearn.metrics as sm
import pandas as pd
import numpy as np
iris = datasets.load_iris()
X = pd.DataFrame(iris.data)
X.columns = ['Sepal_Length','Sepal_Width','Petal_Length','Petal_Width']
y = pd.DataFrame(iris.target)
y.columns = ['Targets']
model = KMeans(n_clusters=3)
model.fit(X)
plt.figure(figsize=(14,7))
colormap = np.array(['red', 'lime', 'black'])
plt.subplot(1, 2, 1)
plt.scatter(X.Petal_Length, X.Petal_Width, c=colormap[y.Targets], s=40)
plt.title('Real Classification')
plt.xlabel('Petal Length')
plt.ylabel('Petal Width')
plt.subplot(1, 2, 2)
plt.scatter(X.Petal_Length, X.Petal_Width, c=colormap[model.labels_], s=40)
plt.title('K Mean Classification')
plt.xlabel('Petal Length')
plt.ylabel('Petal Width')
print('The accuracy score of K-Mean: ',sm.accuracy_score(y, model.labels_))
```

```
print('The Confusion matrixof K-Mean: ',sm.confusion_matrix(y, model.labels_))

from sklearn import preprocessing

scaler = preprocessing.StandardScaler()

scaler.fit(X)

xsa = scaler.transform(X)

xs = pd.DataFrame(xsa, columns = X.columns)

from sklearn.mixture import GaussianMixture

gmm = GaussianMixture(n_components=3)

gmm.fit(xs)

y_gmm = gmm.predict(xs)

plt.subplot(2, 2, 3)

plt.scatter(X.Petal_Length, X.Petal_Width, c=colormap[y_gmm], s=40)

plt.title('GMM Classification')

plt.xlabel('Petal Length')

plt.ylabel('Petal Width')

print('The accuracy score of EM: ',sm.accuracy_score(y, y_gmm))

print('The Confusion matrix of EM: ',sm.confusion_matrix(y, y_gmm))
```



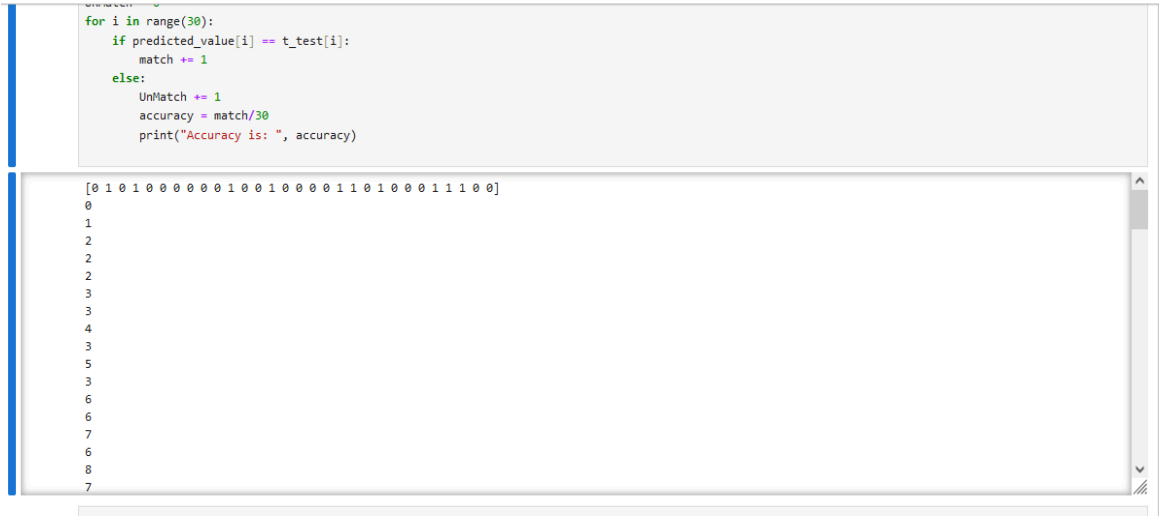
PROGREAM: 08

PROGREAM: 09

```
from sklearn.datasets import make_classification

from sklearn import tree

from sklearn.model_selection import train_test_split

X, t = make_classification(100, 5, n_classes=2, shuffle=True, random_state=10)
```

```
X_train, X_test, t_train, t_test = train_test_split(
X, t, test_size=0.3, shuffle=True, random_state=1)
model = tree.DecisionTreeClassifier()
model = model.fit(X_train, t_train)
predicted_value = model.predict(X_test)
print(predicted_value)
tree.plot_tree(model)
zeroes = 0
ones = 0
for i in range(0, len(t_train)):
    if t_train[i] == 0:
        zeroes += 1
    else:
        ones += 1
        print(zeroes)
        print(ones)
val = 1 - ((zeroes/70)*(zeroes/70) + (ones/70)*(ones/70))
print("Gini :", val)
match = 0
UnMatch = 0
for i in range(30):
    if predicted_value[i] == t_test[i]:
        match += 1
```

```
    else:
        UnMatch += 1
        accuracy = match/30
        print("Accuracy is: ", accuracy)
```

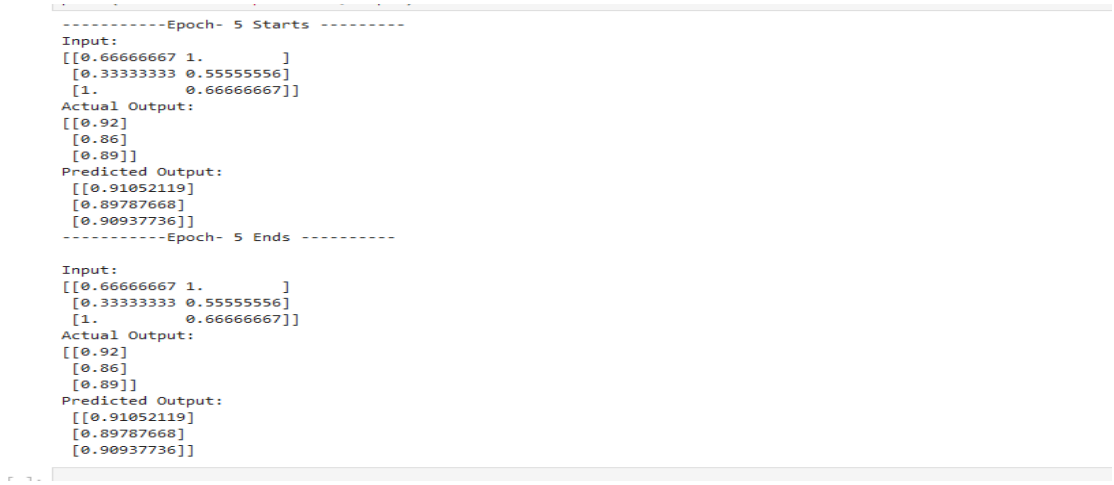


PROGREAM:10

```
import numpy as np
X = np.array(([2, 9], [1, 5], [3, 6]), dtype=float)
y = np.array(([92], [86], [89]), dtype=float)
X = X/np.amax(X,axis=0)
y = y/100
def sigmoid (x):
    return 1/(1 + np.exp(-x))
```

```
def derivatives_sigmoid(x):
    return x * (1 - x)
epoch=5
lr=0.1
inputlayer_neurons = 2
hiddenlayer_neurons = 3
output_neurons = 1
wh=np.random.uniform(size=(inputlayer_neurons,hiddenlayer_neurons))
bh=np.random.uniform(size=(1,hiddenlayer_neurons))
wout=np.random.uniform(size=(hiddenlayer_neurons,output_neurons))
bout=np.random.uniform(size=(1,output_neurons))
for i in range(epoch):
    hinp1=np.dot(X,wh)
    hinp=hinp1 + bh
hlayer_act = sigmoid(hinp)
outinp1=np.dot(hlayer_act,wout)
outinp= outinp1+bout
output = sigmoid(outinp)
EO = y-output
outgrad = derivatives_sigmoid(output)
d_output = EO * outgrad
EH = d_output.dot(wout.T)
hiddengrad = derivatives_sigmoid(hlayer_act)
```

```
d_hiddenlayer = EH * hiddengrad

wout += hlayer_act.T.dot(d_output) *lr

wh += X.T.dot(d_hiddenlayer) *lr

print ("----------Epoch-", i+1, "Starts --------- ")

print("Input: \n" + str(X))

print("Actual Output: \n" + str(y))

print("Predicted Output: \n" ,output)

print ("----------Epoch-", i+1, "Ends ---------- \n")

print("Input: \n" + str(X))

print("Actual Output: \n" + str(y))

print("Predicted Output: \n" ,output)
```

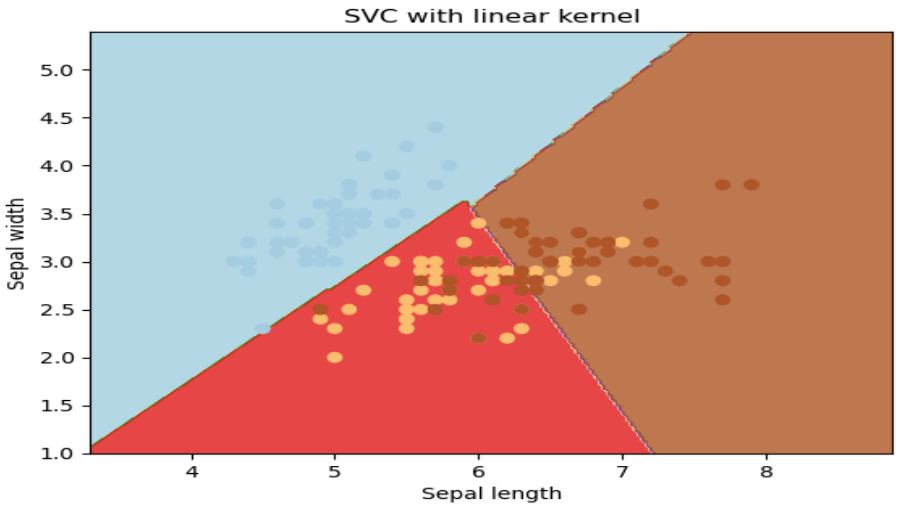


```
----------Epoch- 5 Starts ----------
Input:
[[0.66666667 1.        ]
 [0.33333333 0.55555556]
 [1.         0.66666667]]
Actual Output:
[[0.92]
 [0.86]
 [0.89]]
Predicted Output:
 [[0.91052119]
 [0.89787668]
 [0.90937736]]
----------Epoch- 5 Ends ----------

Input:
[[0.66666667 1.        ]
 [0.33333333 0.55555556]
 [1.         0.66666667]]
Actual Output:
[[0.92]
 [0.86]
 [0.89]]
Predicted Output:
 [[0.91052119]
 [0.89787668]
 [0.90937736]]
```

PROGREAM:11

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn import svm, datasets
iris = datasets.load_iris()
X = iris.data[:, :2]
y = iris.target
C = 1.0
svc = svm.SVC(kernel ='linear', C = 1).fit(X, y)
x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
h = (x_max / x_min)/100
xx, yy = np.meshgrid(np.arange(x_min, x_max, h),
np.arange(y_min, y_max, h))
plt.subplot(1, 1, 1)
Z = svc.predict(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)
plt.contourf(xx, yy, Z, cmap = plt.cm.Paired, alpha = 0.8)
plt.scatter(X[:, 0], X[:, 1], c = y, cmap = plt.cm.Paired)
plt.xlabel('Sepal length')
plt.ylabel('Sepal width')
plt.xlim(xx.min(), xx.max())
plt.title('SVC with linear kernel')
```

plt.show()


SVC with linear kernel
Sepal width
Sepal length
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
4
5
6
7
8

[ ]:

PROGREAM:12

```
import pandas as pd

import numpy as np

from sklearn.linear_model import LogisticRegression

from sklearn.model_selection import train_test_split

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn import metrics

Data= pd.read_csv(r"C:\divya ml\diabetes.csv")

Data.head().transpose()

Data.describe ()
```

[5]:

	0.38076	0.05068	0.061696	0.021872	0.021872.1	-1.044223
count	1.0	1.0	1.0	1.0	1.0	1.0
mean	441.0	441.0	441.0	441.0	441.0	441.0
std	NaN	NaN	NaN	NaN	NaN	NaN
min	441.0	441.0	441.0	441.0	441.0	441.0
25%	441.0	441.0	441.0	441.0	441.0	441.0
50%	441.0	441.0	441.0	441.0	441.0	441.0
75%	441.0	441.0	441.0	441.0	441.0	441.0
max	441.0	441.0	441.0	441.0	441.0	441.0