ANNA UNIVERSITY, CHENNAI

KARPAGA VINAYAGA COLLEGE OF ENGINEERING AND TECHNOLOGY

(Approved by AICTE, Affiliated to Anna University)
GST Road, Karpaga Vinayaga Nagar,
Palayanoor Post, Madhuranthagam Taluk,
Chengalpattu Dist-603308.

LABORATORY RECORD

FACULTY OF INFORMATION AND COMMUNICATION ENGINEERING

(MASTER OF COMPUTER APPLICATIONS)
M.C.A

MC4311 – MACHINE LEARNING LABORATORY

REG.NO :

NAME :

YEAR : MCA II

SEMESTER : III

Karpaga Vinayaga
College of Engineering and Technology

DEPARTMENT OF COMPUTER APPLICATIONS

BONAFIDE CERTIFICATE

Name : ....

Reg. No :

Subject Code : **MC4311**

Subject : **MACHINE LEARNING LABORATORY**

Course : **MASTER OF COMPUTER APPLICATIONS**

Certified that this is the bonafide record of practicals done as a part of 3rd semester during the academic year 2023- 2024.

Staff In-charge **Head of the Department**

Submitted for University Practical Examination held on ______________________

Internal Examiner **External Examiner**

INDEX

Ex: 1 **Demonstrate How Do You Structure Data in Machine Learning**

Date: 27/09/2023

Aim:

To write a program to demonstrate the structure data in Machine Learning.

Algorithm:

Step 1: Start the program.

Step 2: Preparing the Data: After you have your data, you have to prepare it.

Step 3: Importing libraries.

Step 4: Importing Matplotlib for ploting.

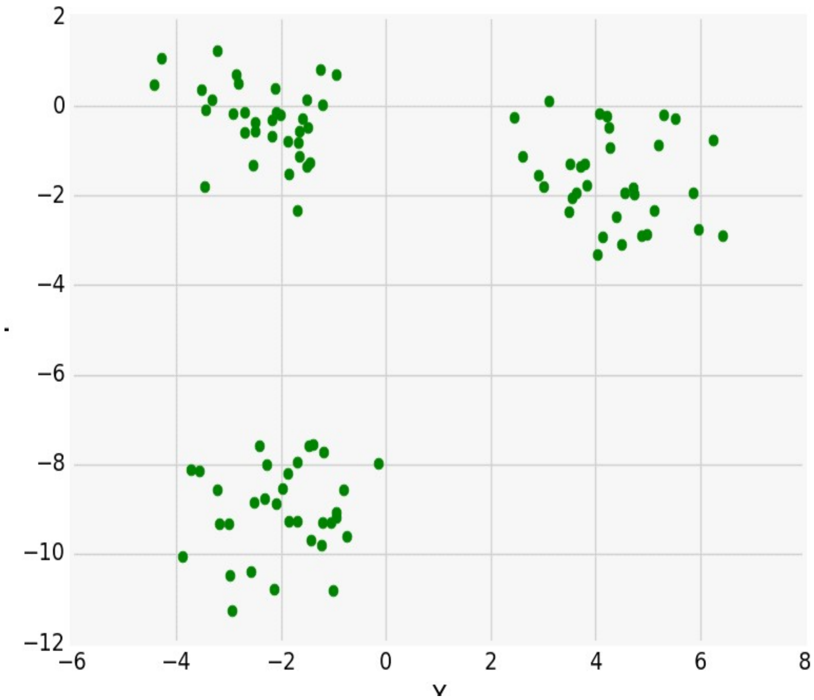
Step 5: Creating Test Data Sets using sklearn.datasets.make_blobs.

Step 6: Stop the execution of the program.

Program:

```
# importing libraries
fromsklearn.datasets import make_blobs
# matplotlib for ploting
frommatplotlib import pyplot as plt
frommatplotlib import style
# Creating Test DataSets using sklearn.datasets.make_blobs
fromsklearn.datasets import make_blobs
frommatplotlib import pyplot as plt
frommatplotlib 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()
```

Output:



RESULT:

Thus the above program has been executed successfully.

Ex: 2
Date: 04/10/2023

Implement data preprocessing techniques on real time dataset

Aim:

To write a program to implement data preprocessing techniques on real time dataset

Algorithm:

Step1 : Start the program.

Step 2: import the modules we'll need.

Step 3: Function that takes in a data frame and creates a text link to download it (will only work for files < 2mb or so)

Step 4: create a random sample data frame

Step 5: create a link to download the data frame

Step 6: stop the execution of the program.

Program:

```
# import the modules we'll need
fromIPython.display import HTML
import pandas as pd
importnumpy as np
import base64

# function that takes in a dataframe and creates a text link to
# download it (will only work for files < 2MB or so)
defcreate_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)

# create a random sample dataframe
df = pd.DataFrame(np.random.randn(50, 4), columns=list('ABCD'))

# create a link to download the dataframe
create_download_link(df)
```

Output:

Out[1]: Download CSV file

RESULT:

Thus the above program has been implemented successfully.

Ex: 3

Implement Feature Subset Selection Techniques

Date: 18/10/2023

Aim:

To write a program to implement feature subset selection techniques.

Algorithm:

Step 1: Start the Program.

Step 2: Built-in Boston dataset which can be loaded through Sklearn.

Step 3: we will import all the required libraries and load the Dataset.

Step 4: Given the datas in the csv file and before upload on jupyter note book.

Step 5: To print the data frame.

Step 6: Stop the execution of the program.

Program:

```
from pandas import read_csv
fromnumpy import set_printoptions
fromsklearn.feature_selection import SelectKBest
fromsklearn.feature_selection import chi2
path = r'd:\\blogs\\bill.csv'
names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class']
dataframe = read_csv(path, names=names)
array = dataframe.values
print(dataframe)
```

Output:

```
          preg      plas      pres     skin   test  mass  pedi  age  class
0     Variance  Skewness  Curtosis  Entropy  Class   NaN   NaN  NaN    NaN
1       3.6216    8.6661   -2.8073 -0.44699      0   NaN   NaN  NaN    NaN
2       4.5459    8.1674   -2.4586  -1.4621      0   NaN   NaN  NaN    NaN
3        3.866   -2.6383    1.9242  0.10645      0   NaN   NaN  NaN    NaN
4       3.4566    9.5228   -4.0112  -3.5944      0   NaN   NaN  NaN    NaN
...        ...       ...       ...      ...    ...   ...   ...  ...    ...
1368   0.40614    1.3492   -1.4501 -0.55949      1   NaN   NaN  NaN    NaN
1369   -1.3887   -4.8773    6.4774  0.34179      1   NaN   NaN  NaN    NaN
1370   -3.7503  -13.4586   17.5932  -2.7771      1   NaN   NaN  NaN    NaN
1371   -3.5637   -8.3827    12.393  -1.2823      1   NaN   NaN  NaN    NaN
1372   -2.5419  -0.65804    2.6842   1.1952      1   NaN   NaN  NaN    NaN

[1373 rows x 9 columns]
```

RESULT:

Thus the above program has been implemented successfully.

Ex: 4
Date:25/10/2023

Measure the performance of machine learning Model

Aim:

To evaluate and measure the performance of a machine learning model.

Algorithm:

Step1: Start the Program.

Step2: we need to import the *accuracy_score* function of the scikit-learn library.

Step3: Then we need to pass the ground truth and predicted values in the function to calculate the accuracy.

Step4: TPR or true Positive rate is a synonym for Recall, hence can be Calculated

$$TPR = \frac{TP}{TP + FN}$$

Step5: FPR or False Positive Rate can be calculated as:

$$TPR = \frac{FP}{FP + TN}$$

Step6: Stop the execution of the program.

Program:

```
fromsklearn.metrics import confusion_matrix
fromsklearn.metrics import accuracy_score
fromsklearn.metrics import classification_report
fromsklearn.metrics import roc_auc_score
fromsklearn.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))
```

Output:

```
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 13.815750437193334
```

RESULT:

Thus the above program has been demonstrated successfully.

Ex: 5

Naive Bayesian Classifier

Date: 01/11/2023

Aim:

To write a program to implement the naïve Bayesian classifier for a sample training data set stored asa.CSV file. Compute the accuracy of the classifier, considering few tests data sets.

Algorithm:

Step 1 : Start the Program.
Step 2 : Load the iris dataset.
Step 3 : Store the feature matrix (X) and response vector (y).
Step 4 : Training the model on training setand making predictions on the testing set.
Step 5 : Comparing actual response values (y_test) with predicted response values (y_pred)
Step 6 : Stop the execution of the program.

Program:

```
fromsklearn.datasetsimportload_irisiris= load_iris()
#storethefeaturematrix(X)andresponsevector(y)
X = iris.data
y=iris.target

#splittingXandyintotrainingandtestingsetsfromsklearn.model_selectionimporttrain_test_split
X_train,X_test,y_train, y_test=train_test_split(X,y,test_size=0.4,random_state=1)

#trainingthe model on trainingset
fromsklearn.naive_bayes import GaussianNBgnb=GaussianNB()
gnb.fit(X_train, y_train)

#makingpredictionsonthetestingsety_pred=gnb.predict(X_test)
#comparingactualresponsevalues(y_test) withpredictedresponsevalues(y_pred)fromsklearn import
metrics
print("GaussianNaiveBayes model accuracy(in %):",
metrics.accuracy_score(y_test,y_pred)*100)
```

Output:

```
Gaussian Naïve Bayes accuracy(in%):95.0
```

RESULT:

Thus, the naïve Bayesian classifier for a sample training dataset has been implemented and the accuracy of the classifier has been computed.

Ex: 6
Date: 01/11/2023

Bayesian Network Considering Medical data

Aim:

To write a program to construct a Bayesian network considering medical data. Use this model to demonstrate the diagnosis of heart patients using standard Heart Disease Data Set. You can use Java/Python ML library classes/API.

Algorithm:

Step 1: Start the Program.
Step 2: Read Cleveland Heart Disease data.
Step 3: Learning CPDs using Maximum Likelihood Estimators**.**
Step 4: Computing the Probability of Heart Disease given Age.
Step 5: Computing the Probability of Heart Disease given cholesterol**.**
Step 6: Stop the execution of the program.

PROGRAM:-

```
importnumpy as np
importcsv
import pandas as pd
frompgmpy.models import BayesianModel
frompgmpy.estimators import Maximum Likelihood Estimator
frompgmpy.inference import VariableElimination
#read Cleveland Heart Disease data
heartDisease = pd.read_csv('heart.csv')
heartDisease = heartDisease.replace('?',np.nan)
#display the data
print('Few examples from the dataset are given below')
print(heartDisease.head())
#Model Bayesian Network
Model=BayesianModel([('age','trestbps'),('age','fbs'),
('sex','trestbps'),('exang','trestbps'),('trestbps','heartdise
ase'),('fbs','heartdisease'),('heartdisease','restecg'),
('heartdisease','thalach'),('heartdisease','chol')])
#Learning CPDs using Maximum Likelihood Estimators
print('\n Learning CPD using Maximum likelihood estimators')
model.fit(heartDisease,estimator=Maximum Likelihood Estimator)
# Inferencing with Bayesian Network
print('\n Inferencing with Bayesian Network:')
HeartDisease_infer = VariableElimination(model)
#computing the Probability of HeartDisease given Age
print('\n 1. Probability of HeartDisease given Age=30')
q=HeartDisease_infer.query(variables=['heartdisease'],evidence
={'age':28})
print(q['heartdisease'])
```

```
#computing the Probability of HeartDisease given cholesterol
print('\n 2. Probability of HeartDisease given cholesterol=100')
q=HetDisease_infer.query(variables=['heartdisease'],evidence
={'chol':100})
print(q['heartdisease'])
```

Output:

```
Few examples from the dataset are given below
    age  sex cp  trestbps   ...slope  ca  thal  heartdisease
0   63    1   1    145      ...  3    0    6              0
1   67    1   4    160      ...  2    3    3              2
2   67    1   4    120      ...  2    2    7              1
3   37    1   3    130      ...  3    0    3              0
4   41    0   2    130      ...  1    0    3              0

[5 rows x 14 columns]

Learning CPD using Maximum likelihood estimators

Inferencing with Bayesian Network:

1. Probability of HeartDisease given Age=28
```

heartdisease	phi(heartdisease)
heartdisease_0	0.6791
heartdisease_1	0.1212
heartdisease_2	0.0810
heartdisease_3	0.0939
heartdisease_4	0.0247

```
2. Probability of HeartDisease given cholesterol=100
```

heartdisease	phi(heartdisease)
heartdisease_0	0.5400
heartdisease_1	0.1533
heartdisease_2	0.1303
heartdisease_3	0.1259
heartdisease_4	0.0506

RESULT:

Thus, a program to construct a Bayesian network considering medical data has been Written and the diagnosis of heart patients using standard Heart Disease Data Set has been demonstrated

Ex: 7 **Apply EM Algorithm to cluster a set of data stored in .CSV file**

Date: 08/11/2023

Aim:

Apply EM algorithm to cluster a set of data stored in a .CSV file. Use the same data set for clustering using k-Means algorithm. Compare the results of these two algorithms and comment on the quality of clustering. You can add Java/Python ML library classes/API in the program.

Algorithm:

Step 1: Start the Program.

Step 2: Specify number of clusters *K*.

Step 3: Initialize centroids by first shuffling the dataset and then randomly selecting *K* data points for the centroids without replacement.

Step 4: Keep iterating until there is no change to the centroids.i.e. assignment of data Points to clusters isn't changing.

Step 5: Compute the sum of the squared distance between data points and all centroids.

Step 6: Assign each data point to the closest cluster (centroids).

Step 7: Stop the execution of the program.

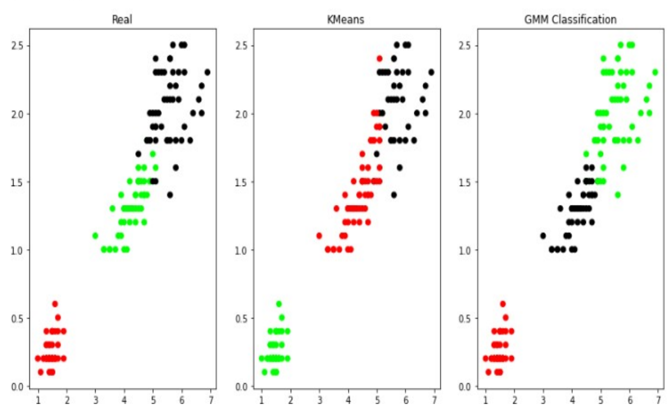
PROGRAM:

```
fromsklearn.cluster import KMeans
fromsklearn.mixture import GaussianMixture
importsklearn.metrics as metrics
import pandas as pd
importnumpy as np
importmatplotlib.pyplot as plt

names = ['Sepal_Length','Sepal_Width','Petal_Length','Petal_Width', 'Class']
dataset = pd.read_csv("8-dataset.csv", names=names)

X = dataset.iloc[:, :-1]

label = {'Iris-setosa': 0,'Iris-versicolor': 1, 'Iris-virginica': 2}

y = [label[c] for c in dataset.iloc[:, -1]]

plt.figure(figsize=(14,7))
colormap=np.array(['red','lime','black'])

# REAL PLOT
plt.subplot(1,3,1)
plt.title('Real')
plt.scatter(X.Petal_Length,X.Petal_Width,c=colormap[y])
# K-PLOT
```

```
model=KMeans(n_clusters=3, random_state=0).fit(X)
plt.subplot(1,3,2)
plt.title('KMeans')

plt.scatter(X.Petal_Length,X.Petal_Width,c=colormap[model.labels_])
print('The accuracy score of K-Mean: ',metrics.accuracy_score(y, model.labels_))
print('The Confusion matrixof K-Mean:\n',metrics.confusion_matrix(y, model.labels_))

# GMM PLOT
gmm=GaussianMixture(n_components=3, random_state=0).fit(X)
y_cluster_gmm=gmm.predict(X)
plt.subplot(1,3,3)
plt.title('GMM Classification')
plt.scatter(X.Petal_Length,X.Petal_Width,c=colormap[y_cluster_gmm])

print('The accuracy score of EM: ',metrics.accuracy_score(y, y_cluster_gmm))
print('The Confusion matrix of EM:\n ',metrics.confusion_matrix(y, y_cluster_gmm))
```

Output

```
The accuracy score of K-Mean:  0.24
The Confusion matrixof K-Mean:
 [[ 0 50  0]
 [48  0  2]
 [14  0 36]]
The accuracy score of EM:  0.36666666666666664
The Confusion matrix of EM:
  [[50  0  0]
 [ 0  5 45]
 [ 0 50  0]]
```



Result:

Thus, a program has been written to implement EM algorithm to classify the datas

Ex: 8
Date: 15/11/2023

k-Nearest Neighbor algorithm to classify the dataset

Aim:

To write a program to implement k-Nearest Neighbor algorithm to classify the iris data set. Print both correct and wrong predictions. Java/Python ML library classes can be used for this problem.

Algorithm:

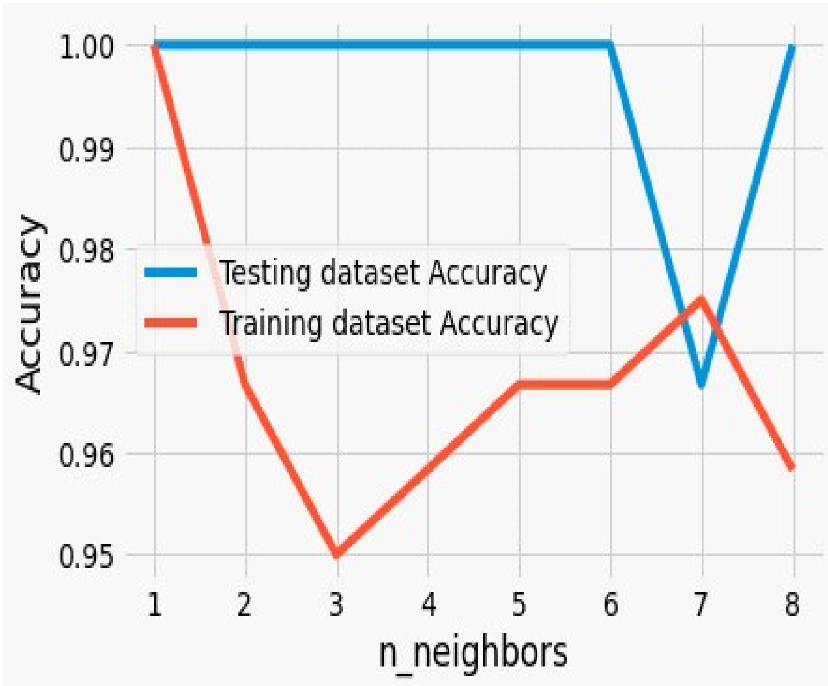
Step 1 : Start the Program.
Step 2 : The k-nearest neighbor algorithm is imported from the scikit-learn package.
Step 3 : Create feature and target variables.
Step 4 : Split data into training and test data.
Step 5 : Generate a k-NN model using neighbors value.
Step 6 : Train or fit the data into the model.
Step 7 : Stop the execution of the program.

PROGRAM

```
fromsklearn.cluster import KMeans
fromsklearn.mixture import GaussianMixture
importsklearn.metrics as metrics
import pandas as pd
importnumpy as np
importmatplotlib.pyplot as plt

names = ['Sepal_Length','Sepal_Width','Petal_Length','Petal_Width',
    'Class']

dataset = pd.read_csv("8-dataset.csv", names=names)

    X = dataset.iloc[:, :-1]

label = {'Iris-setosa': 0,'Iris-versicolor': 1, 'Iris-virginica': 2}

     y = [label[c] for c in dataset.iloc[:, -1]]

plt.figure(figsize=(14,7))
colormap=np.array(['red','lime','black'])

     # REAL PLOT
plt.subplot(1,3,1)
plt.title('Real')
plt.scatter(X.Petal_Length,X.Petal_Width,c=colormap[y])

     # K-PLOT
model=KMeans(n_clusters=3, random_state=0).fit(X)
plt.subplot(1,3,2)
```

```
plt.title('KMeans')
plt.scatter(X.Petal_Length,X.Petal_Width,c=colormap[model.labels_])


print('The accuracy score of K-Mean: ',metrics.accuracy_score(y,
model.labels_))


print('The Confusion matrixof K-Mean:\n',metrics.confusion_matrix(y,
model.labels_))

  # GMM PLOT
gmm=GaussianMixture(n_components=3, random_state=0).fit(X)
y_cluster_gmm=gmm.predict(X)
plt.subplot(1,3,3)
plt.title('GMM Classification')
plt.scatter(X.Petal_Length,X.Petal_Width,c=colormap[y_cluster_gmm])

print('The accuracy score of EM: ',metrics.accuracy_score(y,
y_cluster_gmm))

print('The Confusion matrix of EM:\n ',metrics.confusion_matrix(y,
y_cluster_gmm))
```

Output:



Result:

Thus, a program has been written to implement k-Nearest Neighbor algorithm to classify the dataset.

Ex: 9

Date: 15/11/2023

Decision Tree

Aim:

Apply the technique of pruning for a noisy data monk2 data, and derive the decision tree from this data. Analyze the results by comparing the structure of pruned and un pruned tree.

Algorithm:

Step 1: Start the Program.

Step 2: Splitting data set into train and test sets.

Step 3: The **predict()** method will use the trained model to make predictions on a new set of data (test set).

Step 4: we have 75% accuracy in our predictions.

Step 5: Test_size=.3 means that our test set will be 30% of the train set.

Step 6: Train or fit the data into the model.

Step 7: Stop the execution of the program.

Program:

```
fromsklearn.datasets import make_classification
fromsklearn import tree
fromsklearn.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)):
ift_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):
	ifpredicted_value[i] == t_test[i]:
		match += 1
	else:
		UnMatch += 1

accuracy = match/30
print("Accuracy is: ", accuracy)
```

Output:

```
[0 1 0 1 0 0 0 0 0 0 1 0 0 1 0 0 0 0 1 1 0 1 0 0 0 1 1 1 0 0]
31
39
Gini : 0.4934693877551021
Accuracy is:  0.8333333333333334
```

Result:

Thus, a decision tree has been constructed from the given dataset and a technique of pruning has been applied over decision tree.

Ex: 10

Back Propagation Algorithm

Date: 22/11/2023

Aim:

To Build an Artificial Neural Network by implementing the Back propagation algorithm and test the same using appropriate datasets.

Algorithm:

Step 1: Start the Program.

Step 2: Lets import the required modules and libraries such as numpy,pandas, scikit-learn, and matplotlib.

Step 3: Iris dataset using load_iris() function of scikit-learn library separate them in features and target labels.

Step 4: This data set has three classes Iris-setosa, Iris-versicolor, and Iris-virginica.

Step 5: Create dummy variables for class labels using get_dummies() function.

Step 6: Draws a random range of numbers uniformly of dim x*y.

Step 7: Stop the execution of the program.

Program:

```
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) #maximum of X array longitudinally
y = y/100

#Sigmoid Function
def sigmoid (x):
return 1/(1 + np.exp(-x))

#Derivative of Sigmoid Function
defderivatives_sigmoid(x):
return x * (1 - x)

#Variable initialization
epoch=5 #Setting training iterations
lr=0.1 #Setting learning rate

inputlayer_neurons = 2 #number of features in data set
hiddenlayer_neurons = 3 #number of hidden layers neurons
output_neurons = 1 #number of neurons at output layer
#weight and bias initialization

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))
```

```
#draws a random range of numbers uniformly of dim x*y
for i in range(epoch):

    #Forward Propogation
    hinp1=np.dot(X,wh)
hinp=hinp1 + bh
hlayer_act = sigmoid(hinp)

outinp1=np.dot(hlayer_act,wout)
outinp= outinp1+bout
output = sigmoid(outinp)

#Backpropagation
EO = y-output
outgrad = derivatives_sigmoid(output)
d_output = EO * outgrad
EH = d_output.dot(wout.T)
hiddengrad = derivatives_sigmoid(hlayer_act)#how much hidden layer wts contributed to
error
d_hiddenlayer = EH * hiddengrad

wout += hlayer_act.T.dot(d_output) *lr  # dotproduct of nextlayererror and currentlayerop
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)
```

Output:

```
-----------Epoch- 1 Starts----------
Input:
[[0.66666667 1.        ]
 [0.33333333 0.55555556]
 [1.         0.66666667]]
Actual Output:
[[0.92]
 [0.86]
 [0.89]]
Predicted Output:
 [[0.82981895]
 [0.81138227]
 [0.82802248]]
-----------Epoch- 1 Ends----------

-----------Epoch- 2 Starts----------
Input:
[[0.66666667 1.        ]
 [0.33333333 0.55555556]
 [1.         0.66666667]]
Actual Output:
[[0.92]
 [0.86]
 [0.89]]
Predicted Output:
 [[0.83059341]
 [0.812151  ]
 [0.82879231]]
-----------Epoch- 2 Ends----------
```

```
-----------Epoch- 3 Starts----------
Input:
[[0.66666667 1.        ]
 [0.33333333 0.55555556]
 [1.         0.66666667]]
Actual Output:
[[0.92]
 [0.86]
 [0.89]]
Predicted Output:
 [[0.83135375]
 [0.81290605]
 [0.82954816]]
-----------Epoch- 3 Ends----------

-----------Epoch- 4 Starts----------
Input:
[[0.66666667 1.        ]
 [0.33333333 0.55555556]
 [1.         0.66666667]]
Actual Output:
[[0.92]
 [0.86]
 [0.89]]
Predicted Output:
 [[0.83210036]
 [0.81364776]
 [0.83029041]]
-----------Epoch- 4 Ends----------
```

```
-----------Epoch- 5 Starts----------
Input:
[[0.66666667 1.        ]
 [0.33333333 0.55555556]
 [1.         0.66666667]]
Actual Output:
[[0.92]
 [0.86]
 [0.89]]
Predicted Output:
 [[0.8328336 ]
 [0.8143765 ]
 [0.83101941]]
-----------Epoch- 5 Ends----------

Input:
[[0.66666667 1.        ]
 [0.33333333 0.55555556]
 [1.         0.66666667]]
Actual Output:
[[0.92]
 [0.86]
 [0.89]]
Predicted Output:
 [[0.8328336 ]
 [0.8143765 ]
 [0.83101941]]
```

Result:

Thus, a program has been written to implement k-Nearest Neighbor algorithm to Classify the dataset

Ex: 11
Date: 29/11/2023

Support Vector Classifications

Aim:

To implement Support Vector Classification for linear kernel.

Algorithm:

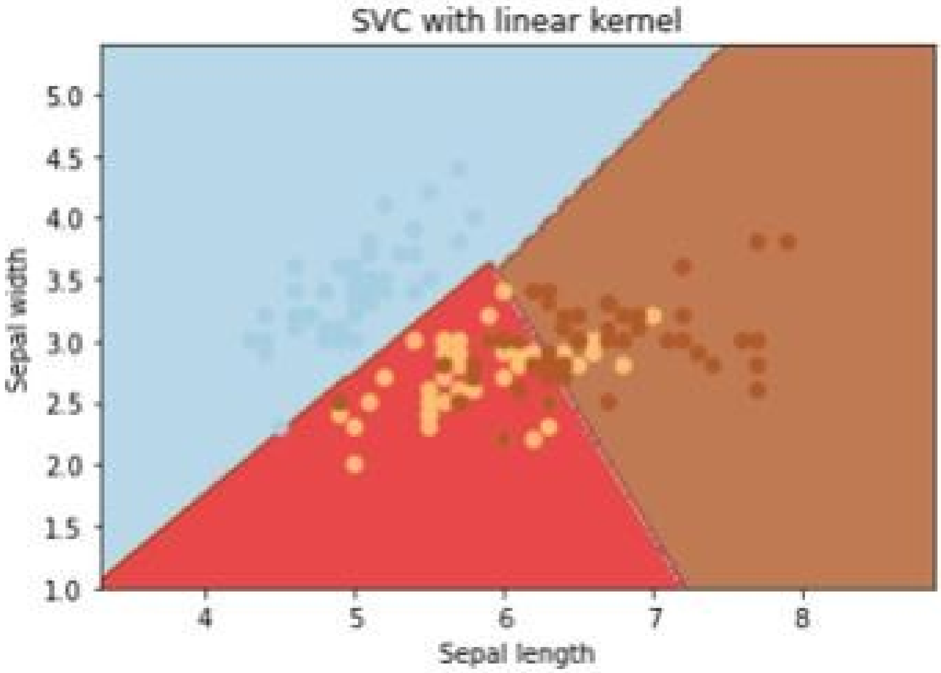
Step 1 : Start the Program.
Step 2 : Import some Data from the iris Data Set
Step 3 : Take only the first two features of Data.
To avoid the slicing, Two-Dim Dataset.
Step 4: Data is not scaled so as to be able to plot the support vectors
Step 5: Plot the data for Proper Visual Representation.
Step 6: Predict the result by giving Data to the model.
Step 7: Stop the execution of the program.

PROGRAM

```
# Import the Libraries
Import numpy as np
Import matplotlib.pyplot as plt
from sklearn import svm, datasets

# Import some Data from the iris Data Set
iris = datasets.load_iris()

# Take only the first two features of Data.
# To avoid the slicing, Two-Dim Dataset can be used

X = iris.data[:, :2]
y = iris.target

# C is the SVM regularization parameter
C = 1.0

# Create an Instance of SVM and Fit out the data.
# Data is not scaled so as to be able to plot the support vectors
svc = svm.SVC(kernel ='linear', C = 1).fit(X, y)

# create a mesh to plot
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))

# Plot the data for Proper Visual Representation
plt.subplot(1, 1, 1)

# Predict the result by giving Data to the model
```

```
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')

# Output the Plot
plt.show()
```

Output:



Result:

Thus, Support Vector Classification has been successfully implemented for linear kernel.

Ex: 12
Date: 22/11/2023

Logistic Regressions

Aim:

Implement Logistic Regression to classify the problems such as spam detection. Diabetes Predictions soon.

Algorithm:

Step 1: Start the Program.
Step 2: To create plots - bar, histogram, box plot etc.
Step 3: Calculate accuracy sure and confusion matrix.
Step 4: we have created an instance of a **Logistic Regression** object.
Step 5: Load CSV file.
Step 6: Split data into training and validation datasets.
Step 7: Stop the execution of the program.

Program:

```
import pandas as pd
import numpy as np
from sklearn.linear_model import Logistic Regression
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("diabetes.csv")
Data.head().transpose()
Data.describe ()
```

Output:

	151	0.038076	0.05068	0.061696	0.021872	-0.044223	-0.034821	-0.043401	-0.002592	0.019908	-0.017646
count	441.000000	441.000000	441.000000	441.000000	441.000000	441.000000	441.000000	441.000000	441.000000	441.000000	441.000000
mean	152.136054	-0.000086	-0.000115	-0.000140	-0.000050	0.000100	0.000079	0.000098	0.000006	-0.000045	0.000040
std	77.180542	0.047638	0.047612	0.047582	0.047662	0.047626	0.047644	0.047628	0.047673	0.047664	0.047666
min	25.000000	-0.107226	-0.044642	-0.090275	-0.112400	-0.126781	-0.115613	-0.102307	-0.076395	-0.126097	-0.137767
25%	87.000000	-0.038207	-0.044642	-0.034229	-0.036656	-0.033216	-0.030124	-0.032356	-0.039493	-0.033249	-0.034215
50%	140.000000	0.005383	-0.044642	-0.007284	-0.005671	-0.004321	-0.003819	-0.006584	-0.002592	-0.002397	-0.001078
75%	212.000000	0.038076	0.050680	0.030440	0.035644	0.028702	0.030001	0.030232	0.034309	0.032433	0.027917
max	346.000000	0.110727	0.050680	0.170555	0.132044	0.153914	0.198788	0.181179	0.185234	0.133599	0.135612

Result:

Thus, Logistic Regression has been implemented to classify the problems such as spam detection and Diabetes predictions.