

SRM VALLIAMMAI ENGINEERING COLLEGE

(An Autonomous Institution)
SRM Nagar, Kattankulathur-603203.

DEPARTMENT OF COMPUTER APPLICATIONS



MC4363- Machine Learning Laboratory

THIRD SEMSTER MCA

R-2024

LAB MANUAL

Prepared By
Mr. M. Asan Nainar, Asst.Prof. / MCA
Ms.D.Revathy,Asst.Prof. / MCA

Syllabus

MC4363

MACHINE LEARNING LABORATORY

L T P C
0 0 4 2

COURSE OBJECTIVES:

- To understand about data cleaning and data preprocessing
- To familiarize with the Supervised Learning algorithms and implement them in practical situations.
- To familiarize with unsupervised Learning algorithms and carry on the implementation part.
- To involve the students to practice ML algorithms and techniques.
- Learn to use algorithms for real time data sets.

LIST OF EXPERIMENTS :

1. Demonstrate how do you structure data in Machine Learning
2. Implement data preprocessing techniques on real time dataset
3. Implement Feature subset selection techniques
4. Demonstrate how will you measure the performance of a machine learning model
5. Write a program to implement the naïve Bayesian classifier for a sample training data set. Compute the accuracy of the classifier, considering few test data sets.
6. Write a program to construct a Bayesian network considering medical data. Use this model to demonstrate the diagnosis of heart patients using the standard Heart Disease Data Set.
7. Apply EM algorithm to cluster a set of data stored in a .CSV file.
8. Write a program to implement k-Nearest Neighbor algorithm to classify the data set.
9. 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 unpruned tree.
10. Build an Artificial Neural Network by implementing the Backpropagation algorithm and test the same using appropriate data sets
11. Implement Support Vector Classification for linear kernels.
12. Implement Logistic Regression to classify problems such as spam detection. Diabetes predictions and so on.

TOTAL: 60 PERIODS

LAB REQUIREMENTS:

Python or any ML tools like R

COURSE OUTCOMES:

On completion of the laboratory course, the student should be able to

CO1: apply data preprocessing technique and explore the structure of data to prepare for predictive modeling

CO2: understand how to select and train a model and measure the performance.

CO3: apply feature selection techniques in Machine Learning

CO4: construct Bayesian Network for appropriate problem

CO5: learn about parametric and non-parametric machine Learning algorithms and implement to practical situations

INTERNAL ASSESSMENT FOR LABORATORY

S.No	Description	Mark
1.	Execution	30
2.	Record	10
3	Model Exam	20
	Total	60

INDEX

E.No.	Experiment Name	Page. No.
1	Demonstrate how do you structure data in Machine Learning	1
2	Implement data preprocessing techniques on real time dataset	2
3	Implement Feature subset selection techniques	6
4	Demonstrate how will you measure the performance of a machine learning model	8
5	Write a program to implement the naïve Bayesian classifier for a sample training data set. Compute the accuracy of the classifier, considering few test data sets.	12
6	Write a program to construct a Bayesian network considering medical data. Use this model to demonstrate the diagnosis of heart patients using the standard Heart Disease Data Set.	17
7	Apply EM algorithm to cluster a set of data stored in a .CSV file.	19
8	Write a program to implement k-Nearest Neighbor algorithm to classify the dataset.	20
9	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 unpruned tree.	24
10	Build an Artificial Neural Network by implementing the Backpropagation algorithm and test the same using appropriate data sets.	26
11	Implement Support Vector Classification for linear kernels.	28
12A	Implement Logistic Regression to classify problems such as spam detection. Diabetes predictions	30
12B	Implement Logistic Regression to classify problems such as Diabetes predictions	32
13	Topic Beyond Syllabus: Customer Segmentation using K – Means Clustering	34
Lab Viva Questions		37

Exercise No. : 01

Date:

Demonstrate how you structure data in Machine Learning

Aim:

Write a program to demonstrate how you structure data in Machine Learning

Procedure:

- Step 1: Start the Jupiter notebook.
- Step 2: Create a new Python 3 (ipykernel) file
- Step 3: import the library pandas
- Step 4: Read the CSV file
- Step 5: Print the Dataset
- Step 6: End the program

Program:

```
In [1]: import pandas as pd
```

```
In [2]: df=pd.read_csv("student.csv")
```

```
In [3]: df
```

```
Out[3]:
```

	Reg. no.	M1	M2	M3	result
0	100	654.0	984.0	45.0	0
1	101	654.0	NaN	2.0	1
2	102	NaN	65.0	45.0	0
3	103	84.0	87.0	67.0	1
4	104	1.0	5.0	78.0	1
5	105	67.0	34.0	NaN	0

```
In [6]: df.head(2)
```

```
Out[6]:
```

	Reg. no.	M1	M2	M3	result
0	100	654.0	984.0	45.0	0
1	101	654.0	NaN	2.0	1

```
In [7]: df.tail(2)
```

```
Out[7]:
```

	Reg. no.	M1	M2	M3	result
4	104	1.0	5.0	78.0	1
5	105	67.0	34.0	NaN	0

Result:

The above program was executed successfully. Hence, the output is verified

Implementing Data Preprocessing techniques on real time dataset

Aim:

Write a program to implementing Data Pre-processing techniques on real time dataset

Procedure:

- Step 1: Start the Jupiter notebook.
- Step 2: Create a new Python 3 (ipykernel) file
- Step 3: import the library pandas
- Step 4: Read the CSV file
- Step 5: Cleaning the missing values with NaN
- Step 6: Filling the missing values by giving number
- Step 7: And Filling the values using forward, backward fill and average value
- Step 8: Print the Dataset
- Step 9: End the program

Program:

```
In [1]: import pandas as pd
In [2]: df=pd.read_csv("student.csv")
In [3]: df
Out[3]:
   Reg. no.    M1     M2     M3  result
0      100  654.0  984.0  45.0       0
1      101  654.0    NaN    2.0       1
2      102    NaN   65.0   45.0       0
3      103   84.0   87.0   67.0       1
4      104    1.0    5.0   78.0       1
5      105   67.0   34.0    NaN       0
```

```
In [6]: df.describe()
Out[6]:
   Reg. no.    M1     M2     M3  result
count  6.000000  5.000000  5.000000  5.000000  6.000000
mean   102.500000 292.000000 235.000000 47.400000 0.500000
std    1.870829 331.910379 419.85295 29.12559 0.547723
min   100.000000  1.000000  5.000000  2.000000 0.000000
25%   101.250000  67.000000  34.000000 45.00000 0.000000
50%   102.500000  84.000000  65.000000 45.00000 0.500000
75%   103.750000  654.000000 87.00000 67.00000 1.000000
max   105.000000 654.000000 984.00000 78.00000 1.000000
```

```
In [7]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6 entries, 0 to 5
Data columns (total 5 columns):
 #   Column   Non-Null Count  Dtype  
--- 
 0   Reg. no.    6 non-null      int64  
 1   M1          5 non-null      float64 
 2   M2          5 non-null      float64 
 3   M3          5 non-null      float64 
 4   result       6 non-null      int64  
dtypes: float64(3), int64(2)
memory usage: 368.0 bytes
```

```
In [8]: df.isnull().sum()
```

```
Out[8]: Reg. no.      0
         M1          1
         M2          1
         M3          1
         result       0
         dtype: int64
```

```
In [9]: df.dtypes
```

```
Out[9]: Reg. no.      int64
         M1          float64
         M2          float64
         M3          float64
         result       int64
         dtype: object
```

```
In [10]: df.shape
```

```
Out[10]: (6, 5)
```

```
In [11]: df1=df.fillna("n")
```

```
In [12]: df1
```

```
Out[12]:
```

	Reg. no.	M1	M2	M3	result
0	100	654.0	984.0	45.0	0
1	101	654.0	n	2.0	1
2	102	n	65.0	45.0	0
3	103	84.0	87.0	67.0	1
4	104	1.0	5.0	78.0	1
5	105	67.0	34.0	n	0

```
In [13]: df2=df.fillna(5)
```

```
In [14]: df2
```

Out[14]:

	Reg. no.	M1	M2	M3	result
0	100	654.0	984.0	45.0	0
1	101	654.0	5.0	2.0	1
2	102	5.0	65.0	45.0	0
3	103	84.0	87.0	67.0	1
4	104	1.0	5.0	78.0	1
5	105	67.0	34.0	5.0	0

dictionary

```
In [15]: df1=df.fillna({'chol':1,'fbs':2})
```

```
In [16]: df1.isnull().sum()  
df1
```

Out[16]:

	Reg. no.	M1	M2	M3	result
0	100	654.0	984.0	45.0	0
1	101	654.0	NaN	2.0	1
2	102	NaN	65.0	45.0	0
3	103	84.0	87.0	67.0	1
4	104	1.0	5.0	78.0	1
5	105	67.0	34.0	NaN	0

Carry forward

```
In [17]: df1=df.fillna(method="ffill")  
df1
```

Out[17]:

	Reg. no.	M1	M2	M3	result
0	100	654.0	984.0	45.0	0
1	101	654.0	984.0	2.0	1
2	102	654.0	65.0	45.0	0
3	103	84.0	87.0	67.0	1
4	104	1.0	5.0	78.0	1
5	105	67.0	34.0	78.0	0

Backward fill

```
In [18]: df1=df.fillna(method="bfill")  
df1
```

Out[18]:

	Reg. no.	M1	M2	M3	result
0	100	654.0	984.0	45.0	0
1	101	654.0	65.0	2.0	1
2	102	84.0	65.0	45.0	0
3	103	84.0	87.0	67.0	1
4	104	1.0	5.0	78.0	1
5	105	67.0	34.0	NaN	0

Fill avg value

In [19]: df1=df.interpolate()
df1

Out[19]:

	Reg. no.	M1	M2	M3	result
0	100	654.0	984.0	45.0	0
1	101	654.0	524.5	2.0	1
2	102	369.0	65.0	45.0	0
3	103	84.0	87.0	67.0	1
4	104	1.0	5.0	78.0	1
5	105	67.0	34.0	78.0	0

Drop NA row/column

In [20]: df1=df.dropna()
df1

Out[20]:

	Reg. no.	M1	M2	M3	result
0	100	654.0	984.0	45.0	0
3	103	84.0	87.0	67.0	1
4	104	1.0	5.0	78.0	1

Result:

The above program was executed successfully. Hence, the output is verified

Implementing Feature subset selection techniques

Aim:

Write a program to implement Feature subset selection techniques

Procedure:

- Step 1:Import pandas to create DataFrame
- Step 2:Make DataFrame of the given data
- Step 3:Variance Threshold feature selector that removes all low-variance features.
- Step 4:It will zero variance features
- Step 5: Drop the error data
- Step 6: Print the DataFrame

Program:

```
In [1]: import pandas as pd

In [2]: data = pd.DataFrame({ "A": [1,2,3,4,5,6],
                           "B": [7,8,9,10,11,12],
                           "C": [0,0,0,0,0,0],
                           "D": [21,54,32,85,35,2]})

In [13]: data
Out[13]:   A  B  C  D
0  1  7  0  21
1  2  8  0  54
2  3  9  0  32
3  4  10  0  85
4  5  11  0  35
5  6  12  0  2

In [4]: from sklearn.feature_selection import VarianceThreshold

In [5]: var_thres = VarianceThreshold(threshold = 0)

In [6]: var_thres.fit(data)
Out[6]: VarianceThreshold(threshold=0)

In [7]: var_thres.get_support()
Out[7]: array([ True,  True, False,  True])

In [8]: data.columns[var_thres.get_support()]
Out[8]: Index(['A', 'B', 'D'], dtype='object')

In [9]: constant_columns = [column for column in data.columns if column not in data.columns[var_thres.get_support()]]
In [10]: print(len(constant_columns))
1
In [11]: for feature in constant_columns: print(feature)
C
```

```
In [12]: data.drop(constant_columns, axis = 1)
```

```
Out[12]:
```

	A	B	D
0	1	7	21
1	2	8	54
2	3	9	32
3	4	10	85
4	5	11	35
5	6	12	2

Result:

The above program was executed successfully. Hence, the output is verified

Exercise No. : 04

Date:

Demonstrate how you will measure the performance of a machine learning model

Aim:

Write a program to demonstrate how you will measure the performance of a machine learning model

Procedure:

- Step 1: Import the Dependencies
- Step 2: Load the csv data to a Pandas DataFrame
- Step 3: Split the Features and Target
- Step 4: Split the Data into Training data & Test Data
- Step 5: Train the LogisticRegression model with Training data
- Step 6: Find accuracy on training data & accuracy on test data
- Step 7: Import the confusion matrix and again find the accuracy of the data
- Step 8: Train and Test the data with Precision
- Step 9: Recall for training and testing data predictions
- Step 10: F1 score for training and testing data predictions

Program:

```
In [1]: import pandas as pd
df=pd.read_csv("diabetes.csv")
df
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome	
0	6	148	72	35	0	33.6		0.627	50	1
1	1	85	66	29	0	26.6		0.351	31	0
2	8	183	64	0	0	23.3		0.672	32	1
3	1	89	66	23	94	28.1		0.167	21	0
4	0	137	40	35	168	43.1		2.288	33	1
...
763	10	101	76	48	180	32.9		0.171	63	0
764	2	122	70	27	0	36.8		0.340	27	0
765	5	121	72	23	112	26.2		0.245	30	0
766	1	126	60	0	0	30.1		0.349	47	1
767	1	93	70	31	0	30.4		0.315	23	0

768 rows × 9 columns

```
In [2]: from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
```

```
In [4]: df['Outcome'].value_counts()
```

```
Out[4]: 0    500
1    268
Name: Outcome, dtype: int64
```

```
In [5]: X = df.drop(columns='Outcome', axis=1)
Y = df['Outcome']
```

```
In [6]: print(X)
```

```
Pregnancies Glucose BloodPressure SkinThickness Insulin BMI \ DiabetesPedigreeFunction Age
0           6      148          72        35       0   33.6    0      0.627  50
1           1       85          66        29       0   26.6    1      0.351  31
2           8      183          64        0       0   23.3    2      0.672  32
3           1       89          66        23      94   28.1    3      0.167  21
4           0      137          40        35     168   43.1    4      2.288  33
..          ...
763         10      101          76        48     180   32.9    763    0.171  63
764         2       122          70        27       0   36.8    764    0.340  27
765         5       121          72        23     112   26.2    765    0.245  30
766         1       126          60        0       0   30.1    766    0.349  47
767         1       93          70        31       0   30.4    [768 rows x 8 columns]
```

```
In [7]: print(Y)
```

```
0      1
1      0
2      1
3      0
4      1
..
763    0
764    0
765    0
766    1
767    0
Name: Outcome, Length: 768, dtype: int64
```

```
In [8]: X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2,
stratify=Y, random_state=2)
```

```
In [9]: print(X.shape, X_train.shape, X_test.shape)
```

```
(768, 8) (614, 8) (154, 8)
```

```
In [10]: model = LogisticRegression(max_iter=1000)
```

```
In [11]: model.fit(X_train, Y_train)
```

```
Out[11]: LogisticRegression(max_iter=1000)
```

```
In [12]: from sklearn.metrics import accuracy_score
```

```
In [13]: X_train_prediction = model.predict(X_train)
training_data_accuracy = accuracy_score(Y_train, X_train_prediction)
print(training_data_accuracy)
```

```
0.7882736156351792
```

```
In [14]: print('Accuracy on Training data :', round(training_data_accuracy*100, 2),
'%')
```

```
Accuracy on Training data : 78.83 %
```

```
In [15]: X_test_prediction = model.predict(X_test)
test_data_accuracy = accuracy_score(Y_test, X_test_prediction)
print(test_data_accuracy)
```

```
0.7597402597402597
```

```
In [16]: print('Accuracy on Training data :', round(test_data_accuracy*100, 2),
'%')
```

```
Accuracy on Training data : 75.97 %
```

```
In [15]: X_test_prediction = model.predict(X_test)
test_data_accuracy = accuracy_score(Y_test, X_test_prediction)
print(test_data_accuracy)

0.7597402597402597

In [16]: print('Accuracy on Training data :', round(test_data_accuracy*100, 2),
        '%')

Accuracy on Training data : 75.97 %
```

```
In [20]: import seaborn as sns
sns.heatmap(cf_matrix, annot=True)

Out[20]: <AxesSubplot:>
```

```
In [21]: from sklearn.metrics import precision_score
precision_train = precision_score(Y_train, X_train_prediction)
print('Training data Precision =', precision_train)

Training data Precision = 0.7530120481927711
```

```
In [22]: precision_test = precision_score(Y_test, X_test_prediction)
print('Test data Precision =', precision_test)

Test data Precision = 0.717948717948718
```

```
In [23]: from sklearn.metrics import recall_score
recall_train = recall_score(Y_train, X_train_prediction)
print('Training data Recall =', recall_train)

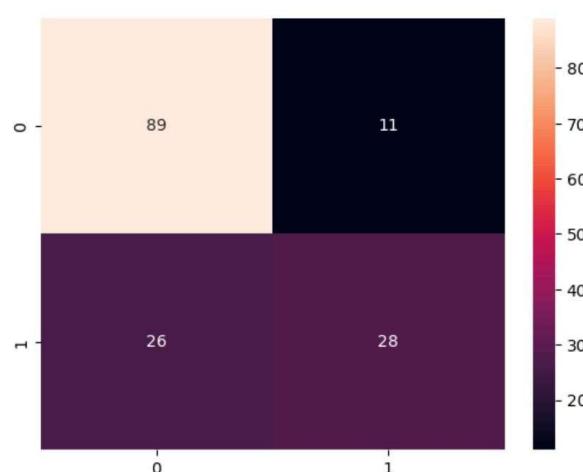
Training data Recall = 0.5841121495327103
```

```
In [24]: recall_test = recall_score(Y_test, X_test_prediction)
print('Test data Recall =', recall_test)

Test data Recall = 0.5185185185185185
```

```
In [25]: from sklearn.metrics import f1_score
f1_score_train = f1_score(Y_train, X_train_prediction)
print('Training data F1 Score =', f1_score_train)

Training data F1 Score = 0.6578947368421052
```



```
In [26]: f1_score_test = recall_score(Y_test, X_test_prediction)
print('Test data F1 Score =', f1_score_test)

Test data F1 Score = 0.5185185185185185

In [27]: def precision_recall_f1_score(true_labels, pred_labels):
    precision_value = precision_score(true_labels, pred_labels)
    recall_value = recall_score(true_labels, pred_labels)
    f1_score_value = f1_score(true_labels, pred_labels)
    print('Precision =',precision_value)
    print('Recall =',recall_value)
    print('F1 Score =',f1_score_value)

In [28]: precision_recall_f1_score(Y_train, X_train_prediction)

Precision = 0.7530120481927711
Recall = 0.5841121495327103
F1 Score = 0.6578947368421052

In [29]: precision_recall_f1_score(Y_test, X_test_prediction)

Precision = 0.717948717948718
Recall = 0.5185185185185185
F1 Score = 0.6021505376344085
```

Result:

The above program executed successfully. Hence output verified

Exercise No. : 05A

Date:

Write a program to implement the naïve Bayesian classification for a sample training data set

Aim:

Write a program to implement the naïve Bayesian classification for a sample training data set. Compute the accuracy of the classifier, with Titanic test data sets.

Procedure:

- Step 1: Import the Dependencies
- Step 2: Load the csv data to a Pandas DataFrame
- Step 3: Split the Features and Target
- Step 4: Split the Data into Training data & Test Data
- Step 5: Train the GaussianNB model with Training data
- Step 6: Find accuracy on training data & accuracy on test data
- Step 7: Use the cross validation and again find the accuracy of the data with training data.

Program:

```
In [1]: import pandas as pd
```

```
In [24]: df = pd.read_csv("titanic.csv")
df.head()
```

```
Out[24]:   PassengerId          Name  Pclass  Sex  Age  SibSp  Parch     Ticket  Fare Cabin Embarked  Survived
0            1    Braund, Mr. Owen Harris    3  male  22.0      1     0  A/5 21171  7.2500   NaN       S       0
1            2  Cumings, Mrs. John Bradley (Florence Briggs Th...    1  female  38.0      1     0        PC 17599  71.2833  C85       C       1
2            3        Heikkinen, Miss. Laina    3  female  26.0      0     0  STON/O2. 3101282  7.9250   NaN       S       1
3            4    Futrelle, Mrs. Jacques Heath (Lily May Peel)    1  female  35.0      1     0        113803  53.1000  C123       S       1
4            5       Allen, Mr. William Henry    3  male  35.0      0     0        373450  8.0500   NaN       S       0
```

```
In [25]: df.drop(['PassengerId', 'Name', 'SibSp', 'Parch', 'Ticket', 'Cabin', 'Embarked'], axis='columns', inplace=True)
df.head()
```

```
Out[25]:   Pclass  Sex  Age  Fare  Survived
0      3  male  22.0  7.2500       0
1      1  female  38.0  71.2833       1
2      3  female  26.0  7.9250       1
3      1  female  35.0  53.1000       1
4      3  male  35.0  8.0500       0
```

```
In [26]: inputs = df.drop('Survived', axis='columns')
target = df.Survived
```

```
In [5]: # inputs.Sex = inputs.Sex.map({'male': 1, 'female': 2})
```

```
In [27]: dummies = pd.get_dummies(inputs.Sex)
dummies.head(3)
```

```
Out[27]:   female  male
0      0      1
1      1      0
2      1      0
```

```
In [28]: inputs = pd.concat([inputs, dummies], axis='columns')
inputs.head(3)
```

```
Out[28]:   Pclass  Sex  Age  Fare  female  male
0      3  male  22.0  7.2500       0      1
1      1  female  38.0  71.2833      1      0
2      3  female  26.0  7.9250      1      0
```

```
In [29]: inputs.drop(['Sex','male'],axis='columns',inplace=True)
inputs.head(3)
```

```
Out[29]:   Pclass  Age    Fare  female
0         3  22.0  7.2500      0
1         1  38.0  71.2833      1
2         3  26.0  7.9250      1
```

```
In [30]: inputs.columns[inputs.isna().any()]
```

```
Out[30]: Index(['Age'], dtype='object')
```

```
In [10]: inputs.Age[:10]
```

```
Out[10]: 0    22.0
1    38.0
2    26.0
3    35.0
4    35.0
5    NaN
6    54.0
7    2.0
8    27.0
9    14.0
Name: Age, dtype: float64
```

```
In [31]: inputs.Age = inputs.Age.fillna(inputs.Age.mean())
inputs.head()
```

```
Out[31]:   Pclass  Age    Fare  female
0         3  22.0  7.2500      0
1         1  38.0  71.2833      1
2         3  26.0  7.9250      1
3         1  35.0  53.1000      1
4         3  35.0  8.0500      0
```

```
In [12]: from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(inputs,target,test_size=0.3)
```

```
In [13]: from sklearn.naive_bayes import GaussianNB
model = GaussianNB()
```

```
In [14]: model.fit(X_train,y_train)
```

```
Out[14]: GaussianNB(priors=None, var_smoothing=1e-09)
```

```
In [15]: model.score(X_test,y_test)
```

```
Out[15]: 0.7835820895522388
```

```
In [16]: X_test[0:10]
```

```
Out[16]:   Pclass      Age    Fare  female  male
309       1  30.000000  56.9292      1      0
839       1  29.699118  29.7000      0      1
110       1  47.000000  52.0000      0      1
872       1  33.000000  5.0000      0      1
235       3  29.699118  7.5500      1      0
411       3  29.699118  6.8583      0      1
32        3  29.699118  7.7500      1      0
562       2  28.000000  13.5000      0      1
542       3  11.000000  31.2750      1      0
250       3  29.699118  7.2500      0      1
```

```
In [17]: y_test[0:10]
```

```
Out[17]: 309    1
839    1
110    0
872    0
235    0
411    0
32     1
562    0
542    0
250    0
Name: Survived, dtype: int64

In [18]: model.predict(X_test[0:10])

Out[18]: array([1, 0, 0, 0, 1, 0, 1, 0, 1, 0], dtype=int64)

In [19]: model.predict_proba(X_test[:10])

Out[19]: array([[0.00455992, 0.99544008],
   [0.91382024, 0.08617976],
   [0.88164575, 0.11835425],
   [0.92347978, 0.07652022],
   [0.09084386, 0.90915614],
   [0.99093305, 0.00906695],
   [0.09094857, 0.90905143],
   [0.97923786, 0.02076214],
   [0.0516967 , 0.9483033 ],
   [0.9909573 , 0.0090427 ]])

In [34]: from sklearn.model_selection import cross_val_score
cross_val_score(GaussianNB(), X_train, y_train, cv=5)

Out[34]: array([0.75396825, 0.784      , 0.76612903, 0.82258065, 0.77419355])
```

Result:

The above program executed successfully. Hence output verified

Write a program to implement the naïve Bayesian classification for a sample training data set

Aim:

Write a program to implement the naïve Bayesian classification for a sample training data set. Compute the accuracy of the classifier, with Spam test data sets.

Procedure:

- Step 1: Import the Dependencies
- Step 2: Load the csv data to a Pandas DataFrame
- Step 3: Categorize the target of data
- Step 4: Splitting the Data into Training data & Test Data
- Step 5: Import CountVectorizer to vectorize the data
- Step 6: Import the MultinomialNB model and find the accuracy of the data
- Step 7: Import the Pipeline model and find the accuracy of the data by cross validating the vectorized data with trained data.

Program:

```
In [1]: import pandas as pd
df = pd.read_csv("spam.csv")
df.head()
```

	Category	Message
0	ham	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...
3	ham	U dun say so early hor... U c already then say...
4	ham	Nah I don't think he goes to usf, he lives aro...

```
In [2]: df.groupby('Category').describe()
```

Category	count		unique		Message
	count	unique	top	freq	
ham	4825	4516	Sorry, I'll call later	30	
spam	747	641	Please call our customer service representativ...	4	

```
In [3]: df['spam']=df['Category'].apply(lambda x: 1 if x=='spam' else 0)
df.head()
```

	Category	Message	spam
0	ham	Go until jurong point, crazy.. Available only ...	0
1	ham	Ok lar... Joking wif u oni...	0
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	1
3	ham	U dun say so early hor... U c already then say...	0
4	ham	Nah I don't think he goes to usf, he lives aro...	0

```
In [4]: from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(df.Message,df.spam)
```

```
In [5]: from sklearn.feature_extraction.text import CountVectorizer
v = CountVectorizer()
X_train_count = v.fit_transform(X_train.values)
X_train_count.toarray()[:2]

Out[5]: array([[0, 0, 0, ..., 0, 0, 0],
   [0, 0, 0, ..., 0, 0, 0]], dtype=int64)

In [6]: from sklearn.naive_bayes import MultinomialNB
model = MultinomialNB()
model.fit(X_train_count,y_train)

Out[6]: MultinomialNB()

In [8]: X_test_count = v.transform(X_test)
model.score(X_test_count, y_test)

Out[8]: 0.9877961234745154

In [9]: from sklearn.pipeline import Pipeline
clf = Pipeline([
    ('vectorizer', CountVectorizer()),
    ('nb', MultinomialNB())
])

In [10]: clf.fit(X_train, y_train)

Out[10]: Pipeline(steps=[('vectorizer', CountVectorizer()), ('nb', MultinomialNB())])

In [11]: clf.score(X_test,y_test)

Out[11]: 0.9877961234745154

In [12]: clf.predict(emails)

Out[12]: array([0, 1], dtype=int64)
```

Result:

The above program executed successfully. Hence output verified

Exercise No. : 06

Date:

Using Bayesian model to demonstrate the diagnosis a heart patients using the standard heart disease data set.

Aim:

Write a program to construct a Bayesian network considering medical data. Use this model to demonstrate the diagnosis a heart patients using the standard heart disease data set.

Procedure:

Step 1: Import the dependencies

Step 2: Load the csv data to a Pandas Data Frame

Step 3: Import BayesianModel and train the data

Step 4: Import MaximumLikelihoodEstimator and estimate the maximum likelihood of the data

Step 5: Use the ValidationElimination method on the model to inferencing the data.

Step 6: Display the maximum likelihood probability of the target data.

Program:

```
In [1]: import numpy as np
import pandas as pd
import csv
```

```
In [2]: from pgmpy.estimators import MaximumLikelihoodEstimator
from pgmpy.models import BayesianModel
from pgmpy.inference import VariableElimination
```

```
In [3]: hD = pd.read_csv('heart.csv')
hD = hD.replace('?', np.nan)
hD
```

Out[3]:

	age	sex	cp	trestbps	chol	fbfs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3.0	145.0	233.0	1.0	0.0	150.0	0.0	2.3	0.0	0.0	1	1
1	37	1	2.0	130.0	250.0	0.0	1.0	187.0	0.0	3.5	0.0	0.0	2	1
2	41	0	1.0	130.0	204.0	0.0	0.0	172.0	0.0	1.4	2.0	0.0	2	1
3	56	1	1.0	120.0	236.0	0.0	1.0	178.0	0.0	0.8	2.0	0.0	2	1
4	57	0	0.0	120.0	NaN	0.0	1.0	163.0	1.0	0.6	2.0	0.0	2	1
...
298	57	0	0.0	140.0	241.0	0.0	1.0	123.0	1.0	0.2	1.0	0.0	3	0
299	45	1	3.0	110.0	264.0	0.0	1.0	132.0	0.0	1.2	1.0	0.0	3	0
300	68	1	0.0	144.0	193.0	1.0	1.0	141.0	0.0	3.4	1.0	2.0	3	0
301	57	1	0.0	130.0	131.0	0.0	1.0	115.0	1.0	1.2	1.0	1.0	3	0
302	57	0	1.0	130.0	236.0	0.0	0.0	174.0	0.0	0.0	1.0	1.0	2	0

303 rows × 14 columns

```
In [9]: model = BayesianModel([('age','target'),('sex','target'),('trestbps','target'),('cp','target'),('target','restecg'),
                           ('target','chol'),('target','fbfs'),('target','thalach'),('target','exang'),('target','oldpeak'),('target','slope'),
                           ('target','ca'),('target','thal')])
```

```
In [5]: print('\n Learning CPD using Maximum Likelihood estimators')
model.fit(hD,estimator = MaximumLikelihoodEstimator)
```

Learning CPD using Maximum Likelihood estimators

```
In [6]: print('\nInferencing with Bayesian Network')
hD_infer = VariableElimination(model)
```

Inferencing with Bayesian Network

```
In [10]: print('\n 1. Probability of heatdesease given evidence = restecg :1')
q1 = hD_infer.query(variables = ['target'], evidence = {'restecg':1})
print(q1)
```

```

1. Probability of heatdesease given evidence = restecg :1
+-----+
| target | phi(target) |
+=====+
| target(0) | 0.4031 |
+-----+
| target(1) | 0.5969 |
+-----+


In [8]: print('\n 2. Probability of heatdesease given evidence = cp:2')
q2 = hD_infer.query(variables = ['target'], evidence = {'cp':2})
print(q2)

2. Probability of heatdesease given evidence = cp:2
+-----+
| target | phi(target) |
+=====+
| target(0) | 0.4862 |
+-----+
| target(1) | 0.5138 |
+-----+

```

Result:

The above program executed successfully. Hence output verified

Exercise No. : 07

Date:

Apply EM algorithm to cluster a set of data stored in a .csv file

Aim:

Write a program to apply EM algorithm to cluster a set of data stored in a .csv file

Procedure:

- Step 1:Imported libraries and dataset
- Step 2:loading data-set for EM algorithm
- Step 3:Defining EM Model
- Step 4:Training of the model
- Step 5:Predicting classes for our data
- Step 6:Accuracy of EM Model

Program:

```
In [1]: #Imported libraries and dataset
from sklearn import datasets
from sklearn.cluster import KMeans
from sklearn.utils import shuffle
import numpy as np
import pandas as pd
```

```
In [2]: #Loading data-set for EM algorithm
iris = datasets.load_iris()
X = pd.DataFrame(iris.data)
Y = pd.DataFrame(iris.target)
```

```
In [3]: #Defining EM Model
from sklearn.mixture import GaussianMixture
model2=GaussianMixture(n_components=3,random_state=3425)

#Training of the model
model2.fit(X)
```

```
Out[3]: GaussianMixture(n_components=3, random_state=3425)
```

```
In [4]: #Predicting classes for our data
uu= model2.predict(X)

#Accuracy of EM Model
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(Y,uu)

print(cm)

from sklearn.metrics import accuracy_score

print(accuracy_score(Y,uu))

[[ 0  0 50]
 [45  5  0]
 [ 0 50  0]]
0.03333333333333333
```

Result:

The above program executed successfully. Hence output verified

Exercise No. : 08

Date:

Implement k-Nearest Neighbour algorithm to classify the dataset

Aim:

Write a program to implement k-Nearest Neighbour algorithm to classify the dataset

Procedure:

- Step 1:
- Step 2:
- Step 3:
- Step 4:
- Step 5:
- Step 6:

Program:

```
In [1]: import pandas as pd
from sklearn.datasets import load_iris
iris=load_iris()

In [3]: iris.feature_names

Out[3]: ['sepal length (cm)',
 'sepal width (cm)',
 'petal length (cm)',
 'petal width (cm)']

In [4]: iris.target_names

Out[4]: array(['setosa', 'versicolor', 'virginica'], dtype='<U10')
```

```
In [5]: df=pd.DataFrame(iris.data,columns=iris.feature_names)
```

```
In [6]: df.head()
```

```
Out[6]:   sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)
0            5.1            3.5            1.4            0.2
1            4.9            3.0            1.4            0.2
2            4.7            3.2            1.3            0.2
3            4.6            3.1            1.5            0.2
4            5.0            3.6            1.4            0.2
```

```
In [7]: df['target']=iris.target
df.head()
```

```
Out[7]:   sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)  target
0            5.1            3.5            1.4            0.2            0
1            4.9            3.0            1.4            0.2            0
2            4.7            3.2            1.3            0.2            0
3            4.6            3.1            1.5            0.2            0
4            5.0            3.6            1.4            0.2            0
```

```
In [8]: df[df.target==1].head()
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
50	7.0	3.2	4.7	1.4	1
51	6.4	3.2	4.5	1.5	1
52	6.9	3.1	4.9	1.5	1
53	5.5	2.3	4.0	1.3	1
54	6.5	2.8	4.6	1.5	1

```
In [9]: df[df.target==2].head()
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
100	6.3	3.3	6.0	2.5	2
101	5.8	2.7	5.1	1.9	2
102	7.1	3.0	5.9	2.1	2
103	6.3	2.9	5.6	1.8	2
104	6.5	3.0	5.8	2.2	2

```
In [18]: df['flower_name']=df.target.apply(lambda x:iris.target_names[x])
```

```
In [19]: df.head()
```

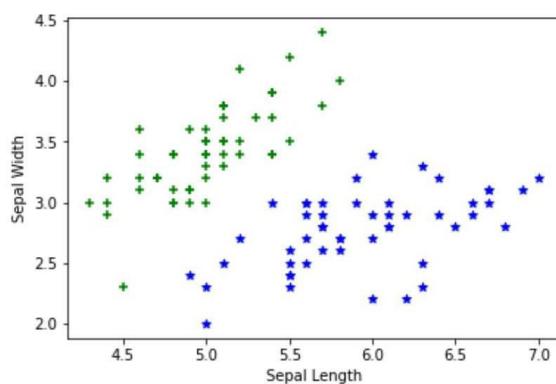
	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target	flower_name
0	5.1	3.5	1.4	0.2	0	setosa
1	4.9	3.0	1.4	0.2	0	setosa
2	4.7	3.2	1.3	0.2	0	setosa
3	4.6	3.1	1.5	0.2	0	setosa
4	5.0	3.6	1.4	0.2	0	setosa

```
In [20]: df0=df[:50]
df1=df[50:100]
df2=df[100:]
```

```
In [21]: import matplotlib.pyplot as plt
%matplotlib inline
```

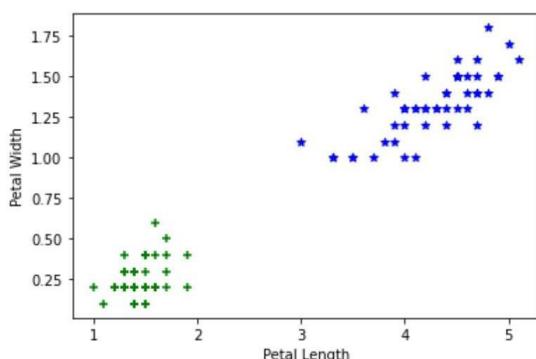
```
In [31]: plt.xlabel('Sepal Length')
plt.ylabel('Sepal Width')
plt.scatter(df0['sepal length (cm)'],df0['sepal width (cm)'],color="green",marker='+')
plt.scatter(df1['sepal length (cm)'],df1['sepal width (cm)'],color="blue",marker='*')
```

```
Out[31]: <matplotlib.collections.PathCollection at 0x2388bc49cd0>
```



```
In [32]: plt.xlabel('Petal Length')
plt.ylabel('Petal Width')
plt.scatter(df0['petal length (cm)'],df0['petal width (cm)'],color="green",marker='+')
plt.scatter(df1['petal length (cm)'],df1['petal width (cm)'],color="blue",marker='*')
```

```
Out[32]: <matplotlib.collections.PathCollection at 0x2388c9cf1f0>
```



```
In [23]: from sklearn.model_selection import train_test_split
x=df.drop(['target','flower_name'],axis='columns')
y=df.target
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=1)
```

```
In [24]: len(x_train)
```

```
Out[24]: 120
```

```
In [26]: len(x_test)
```

```
Out[26]: 30
```

```
In [34]: from sklearn.neighbors import KNeighborsClassifier
knn=KNeighborsClassifier(n_neighbors=10)
```

```
In [35]: knn.fit(x_train,y_train)
```

```
Out[35]: KNeighborsClassifier(n_neighbors=10)
```

```
In [36]: knn.score(x_test,y_test)
```

```
Out[36]: 0.9666666666666667
```

```
In [37]: knn.predict([[4.8,3.0,1.5,0.3]])
```

```
Out[37]: array([0])
```

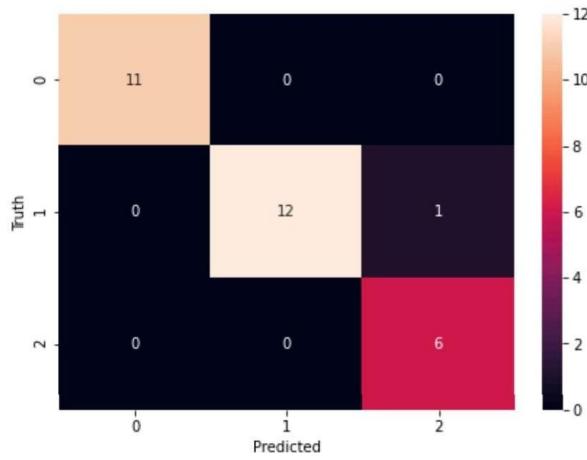
```
In [38]: from sklearn.metrics import confusion_matrix
y_pred=knn.predict(x_test)
cm=confusion_matrix(y_test,y_pred)
```

```
In [39]: cm
```

```
Out[39]: array([[11,  0,  0],
 [ 0, 12,  1],
 [ 0,  0,  6]], dtype=int64)
```

```
In [40]: %matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sn
plt.figure(figsize=(7,5))
sn.heatmap(cm,annot=True)
plt.xlabel('Predicted')
plt.ylabel('Truth')
```

```
Out[40]: Text(42.0, 0.5, 'Truth')
```



```
In [41]: from sklearn.metrics import classification_report
```

```
In [42]: print(classification_report(y_test,y_pred))
```

	precision	recall	f1-score	support
0	1.00	1.00	1.00	11
1	1.00	0.92	0.96	13
2	0.86	1.00	0.92	6
accuracy			0.97	30
macro avg	0.95	0.97	0.96	30
weighted avg	0.97	0.97	0.97	30

Result:

The above program executed successfully. Hence output verified

Derive a decision tree from the dataset

Aim:

Write a program to drive a decision tree from the dataset

Procedure:

- Step 1:
- Step 2:
- Step 3:
- Step 4:
- Step 5:
- Step 6:

Program:

```
In [1]: import pandas as pd
```

```
In [2]: df = pd.read_csv("salaries.csv")
df.head()
```

```
Out[2]:   company          job    degree salary_more_then_100k
0    google  sales executive  bachelors            0
1    google  sales executive    masters            0
2    google  business manager  bachelors            1
3    google  business manager    masters            1
4    google  computer programmer  bachelors            0
```

```
In [3]: inputs = df.drop('salary_more_then_100k',axis='columns')
```

```
In [4]: target = df['salary_more_then_100k']
```

```
In [5]: from sklearn.preprocessing import LabelEncoder
le_company = LabelEncoder()
le_job = LabelEncoder()
le_degree = LabelEncoder()
```

```
In [6]: inputs['company_n'] = le_company.fit_transform(inputs['company'])
inputs['job_n'] = le_job.fit_transform(inputs['job'])
inputs['degree_n'] = le_degree.fit_transform(inputs['degree'])
```

```
In [7]: inputs
```

```
Out[7]:   company          job    degree  company_n  job_n  degree_n
0    google  sales executive  bachelors        2       2       0
1    google  sales executive    masters        2       2       1
2    google  business manager  bachelors        2       0       0
3    google  business manager    masters        2       0       1
4    google  computer programmer  bachelors        2       1       0
5    google  computer programmer    masters        2       1       1
6  abc pharma  sales executive    masters        0       2       1
7  abc pharma  computer programmer  bachelors        0       1       0
8  abc pharma  business manager  bachelors        0       0       0
9  abc pharma  business manager    masters        0       0       1
10   facebook  sales executive  bachelors        1       2       0
11   facebook  sales executive    masters        1       2       1
12   facebook  business manager  bachelors        1       0       0
```

```
In [8]: inputs_n = inputs.drop(['company','job','degree'],axis='columns')
```

```
In [9]: inputs_n
```

```
Out[9]:   company_n  job_n  degree_n
```

	company_n	job_n	degree_n
0	2	2	0
1	2	2	1
2	2	0	0
3	2	0	1
4	2	1	0
5	2	1	1
6	0	2	1
7	0	1	0
8	0	0	0
9	0	0	1
10	1	2	0
11	1	2	1
12	1	0	0
13	1	0	1
14	1	1	0

```
In [10]: target
```

```
Out[10]: 0    0
1    0
2    1
3    1
4    0
5    1
6    0
7    0
8    0
9    1
10   1
11   1
12   1
13   1
14   1
15   1
Name: salary_more_then_100k, dtype: int64
```

```
In [11]: from sklearn import tree
model = tree.DecisionTreeClassifier()
```

```
In [12]: model.fit(inputs_n, target)
```

```
Out[12]: DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
                                 max_features=None, max_leaf_nodes=None,
                                 min_impurity_decrease=0.0, min_impurity_split=None,
                                 min_samples_leaf=1, min_samples_split=2,
                                 min_weight_fraction_leaf=0.0, presort=False, random_state=None,
                                 splitter='best')
```

```
In [13]: model.score(inputs_n,target)
```

```
Out[13]: 1.0
```

Is salary of Google, Computer Engineer, Bachelors degree > 100 k ?

```
In [14]: model.predict([[2,1,0]])
```

```
Out[14]: array([0], dtype=int64)
```

Is salary of Google, Computer Engineer, Masters degree > 100 k ?

```
In [15]: model.predict([[2,1,1]])
```

```
Out[15]: array([1], dtype=int64)
```

Result:

The above program executed successfully. Hence output verified

Exercise No. : 10

Date:

Implementing the back propagation algorithm and test the same using appropriate dataset

Aim:

Write a program to build an Artificial Neural Network by implementing the back propagation algorithm and test the same using appropriate dataset

Procedure:

Step 1:

Step 2:

Step 3:

Step 4:

Step 5:

Step 6:

Program:

```
In [1]: import numpy as np
X = np.array(([2, 9], [1, 5], [3, 6]), dtype=float)
y = np.array(([92], [86], [89]), dtype=float) # scale units
X = X/np.amax(X, axis=0) #maximum of X array
y = y/100 # maximum test score is 100

class NeuralNetwork(object):
    def __init__(self): #parameters
        self.inputSize = 2
        self.outputSize = 1
        self.hiddenSize = 3
        self.W1 = np.random.randn(self.inputSize, self.hiddenSize)
        self.W2 = np.random.randn(self.hiddenSize, self.outputSize)

    def feedForward(self, X):
        self.z = np.dot(X, self.W1)
        self.z2 = self.sigmoid(self.z)
        self.z3 = np.dot(self.z2, self.W2)
        output = self.sigmoid(self.z3)
        return output

    def sigmoid(self, s, deriv=False):
        if (deriv == True):
            return s * (1 - s)
        return 1/(1 + np.exp(-s))

    def backward(self, X, y, output): #backward propogate through the network
        self.output_error = y - output # error in output
        self.output_delta = self.output_error * self.sigmoid(output, deriv=True)
        self.z2_error = self.output_delta.dot(self.W2.T) #z2 error: how much our hidden layer weights
        self.z2_delta = self.z2_error * self.sigmoid(self.z2, deriv=True) #applying derivative of sigmoid
        self.W1 += X.T.dot(self.z2_delta) # adjusting first set (input -> hidden) weights
        self.W2 += self.z2.T.dot(self.output_delta) # adjusting second set (hidden -> output) weights
    def train(self, X, y):
        output = self.feedForward(X)
        self.backward(X, y, output)

NN = NeuralNetwork()
for i in range(1000): #trains the NN 1000 times
    if (i % 100 == 0):
        print("Loss: " + str(np.mean(np.square(y - NN.feedForward(X)))))
        NN.train(X, y)
print("Input: " + str(X))
print("Actual Output: " + str(y))
print("Loss: " + str(np.mean(np.square(y - NN.feedForward(X)))))
print("\n")
print("Predicted Output: " + str(NN.feedForward(X)))
```

```
Loss: 0.024912562840868836
Loss: 0.020568438885881255
Loss: 0.017167958121637485
Loss: 0.014471137378264548
Loss: 0.012306361746843379
Loss: 0.010549272872305953
Loss: 0.009108544221234353
Loss: 0.007916223149021984
Loss: 0.006921107548652352
Loss: 0.006084151847431011
Input: [[0.66666667 1.        ]
 [0.33333333 0.55555556]
 [1.          0.66666667]]
Actual Output: [[0.92]
 [0.86]
 [0.89]]
Loss: 0.005375242340215718

Predicted Output: [[0.85079209]
 [0.80977176]
 [0.7961218 ]]
```

Result:

The above program executed successfully. Hence output verified

Exercise No. : 11**Date:****Implement Support Vector Classification
for linear kernels****Aim:**

Write a program to implement Support Vector Classification for linear kernels

Procedure:

- Step 1: Import necessary dataset from sklearn datasets
- Step 2: Import the dependencies
- Step 3: Separate the target from the features
- Step 4: Split the data into training and test data
- Step 5: Import the Standard Scaler and implement it on training and test data
- Step 6: Import the SVM model and implement it on the scaled data by setting the kernel as linear
- Step 7: Display the accuracy on both training and test data

Program:

```
In [1]: from sklearn.datasets import load_breast_cancer

In [2]: from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler, StandardScaler
from sklearn.svm import SVC
import pandas as pd

In [3]: a = load_breast_cancer()
a.target_names

Out[3]: array(['malignant', 'benign'], dtype='|<U9')

In [4]: df=pd.DataFrame(a.data, columns = list(a.feature_names))
df['diagnosis'] = a.target
df.head(4)

Out[4]:
   mean radius  mean texture  mean perimeter  mean area  mean smoothness  mean compactness  mean concavity  mean concave points  mean symmetry  mean fractal dimension ...  worst texture  worst perimeter  worst area  worst smoothness  worst compa
0    17.99     10.38      122.80    1001.0    0.11840      0.27760      0.3001    0.14710      0.2419      0.07871 ...     17.33     184.60    2019.0      0.1622
1    20.57     17.77      132.90    1326.0    0.08474      0.07864      0.0869    0.07017      0.1812      0.05667 ...     23.41     158.80    1956.0      0.1238
2    19.69     21.25      130.00    1203.0    0.10960      0.15990      0.1974    0.12790      0.2069      0.05999 ...     25.53     152.50    1709.0      0.1444
3    11.42     20.38      77.58     386.1    0.14250      0.28390      0.2414    0.10520      0.2597      0.09744 ...     26.50      98.87     567.7      0.2098

In [5]: X_train, X_test, y_train, y_test = train_test_split(a.data, a.target, stratify=a.target, random_state=42)
print(f"X_train shape: {X_train.shape}")
print(f"X_test shape: {X_test.shape}")
print(f"y_train shape: {y_train.shape}")
print(f"y_test shape: {y_test.shape}")

X_train shape: (426, 30)
X_test shape: (143, 30)
y_train shape: (426,)
y_test shape: (143,)

In [6]: svm = SVC(kernel='linear')
svm.fit(X_train, y_train)

Out[6]: SVC(kernel='linear')

In [7]: print(f'Accuracy on training subset is: {svm.score(X_train, y_train):.3f}')
print(f'Accuracy on test subset is: {svm.score(X_test, y_test):.3f}')

Accuracy on training subset is: 0.962
Accuracy on test subset is: 0.951

In [8]: scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

```
In [9]: svm = SVC(kernel='linear')
svm.fit(X_train_scaled, y_train)

Out[9]: SVC(kernel='linear')

In [10]: print(f'Accuracy on training subset is: {svm.score(X_train_scaled, y_train):.3f}')
print(f'Accuracy on test subset is: {svm.score(X_test_scaled, y_test):.3f}')

Accuracy on training subset is: 0.990610
Accuracy on test subset is: 0.986
```

Result:

The above program executed successfully. Hence output verified

Aim:

Write a program to implement Logistic Regression to classify problems such as Spam detection

Procedure:

- Step 1: Import the Dependencies
- Step 2: Load the csv data to a Pandas DataFrame
- Step 3: Split the Features and Target
- Step 4: Split the Data into Training data & Test Data
- Step 5: Train the LogisticRegression model with Training data
- Step 6: Find accuracy on training data & accuracy on test data

Program:

```
In [ ]: import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression

In [ ]: data = pd.read_csv('spam.csv')

In [22]: data.head()

Out[22]:   Category          Message
0      ham  Go until jurong point, crazy.. Available only ...
1      ham           Ok lar... Joking wif u oni...
2     spam  Free entry in 2 a wkly comp to win FA Cup fina...
3      ham  U dun say so early hor... U c already then say...
4      ham  Nah I don't think he goes to usf, he lives aro...
```



```
In [23]: data['Message'].value_counts()

Out[23]: Sorry, I'll call later
30
I cant pick the phone right now. Pls send a message
12
Ok...
10
Ok.

4
Say this slowly.? GOD,I LOVE YOU & I NEED YOU,CLEAN MY HEART WITH YOUR BLOOD.Send this to Ten special people & u c mire
cle tomorrow, do it,pls,pls do it...        4

..
Haha, my friend tyler literally just asked if you could get him a dubsack
1
Try neva mate!!
1
Ur cash-balance is currently 500 pounds - to maximize ur cash-in now send GO to 86688 only 150p/msg. CC: 08718720201 PO BOX 11
4/14 TCR/W1
1
Its just the effect of irritation. Just ignore it
1
Booked ticket for pongal?
1
Name: Message, Length: 5157, dtype: int64
```



```
In [39]: from sklearn import preprocessing
c1=preprocessing.LabelEncoder()
data['Category']=c1.fit_transform(data['Category'])
data['Message']=c1.fit_transform(data['Message'])

In [40]: x = data.drop(columns='Message',axis=1)

In [41]: y=data['Message']

In [42]: x
```

```
Out[42]:
```

	Category
0	0
1	0
2	1
3	0
4	0
...	...
5567	1
5568	0
5569	0
5570	0
5571	0

5572 rows × 1 columns

```
In [43]:
```

```
y
```

```
Out[43]:
```

0	1080
1	3126
2	999
3	4121
4	2781
...	...
5567	4025
5568	4596
5569	3313
5570	3932
5571	3437

Name: Message, Length: 5572, dtype: int32

```
In [49]:
```

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2)
```

```
In [50]:
```

```
print(x.shape,x_train.shape,x_test.shape)
```

(5572, 1) (4457, 1) (1115, 1)

```
In [61]:
```

```
model=LogisticRegression(max_iter=1000)
```

```
In [62]:
```

```
model.fit(x_train,y_train)
```

```
Out[62]:
```

```
LogisticRegression(max_iter=1000)
```

```
In [63]:
```

```
from sklearn.metrics import accuracy_score
```

```
In [64]:
```

```
x_train_prediction=model.predict(x_train)
trained_data_accuracy=accuracy_score(y_train,x_train_prediction)
print(trained_data_accuracy)
```

0.006057886470720216

```
In [65]:
```

```
print('Accuracy on training data:',round(trained_data_accuracy*100,2),'%)')
Accuracy on training data: 0.61 %
```

```
In [67]:
```

```
x_test_prediction=model.predict(x_test)
test_data_accuracy=accuracy_score(y_test,x_test_prediction)
print(test_data_accuracy)
```

0.006278026905829596

```
In [68]:
```

```
print('Accuracy on test data:',round(test_data_accuracy*100,2),'%)')
Accuracy on test data: 0.63 %
```

Result:

The above program executed successfully. Hence output verified

Exercise No. : 12B

Date:

Implement Logistic Regression to classify problems such as Diabetics detection

Aim:

Write a program to implement Logistic Regression to classify problems such as Spam detection

Procedure:

- Step 1: Import the Dependencies
- Step 2: Load the CSV data to a Pandas DataFrame
- Step 3: Split the Features and Target
- Step 4: Split the Data into Training data & Test Data
- Step 5: Train the LogisticRegression model with Training data
- Step 6: Find accuracy on training data & accuracy on test data

Program:

```
In [1]: import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression

In [2]: data = pd.read_csv('diabetes.csv')

In [3]: data.head()

Out[3]:    Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin  BMI  DiabetesPedigreeFunction  Age  Outcome
0            6      148             72          35        0   33.6           0.627   50       1
1            1       85             66          29        0   26.6           0.351   31       0
2            8      183             64          0        0   23.3           0.672   32       1
3            1       89             66          23       94   28.1           0.167   21       0
4            0      137             40          35      168   43.1           2.288   33       1

In [6]: data['Outcome'].value_counts()

Out[6]: 0    500
1    268
Name: Outcome, dtype: int64

In [7]: x=data.drop(columns='Outcome',axis=1)

In [9]: y=data['Outcome']

In [10]: x

Out[10]:    Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin  BMI  DiabetesPedigreeFunction  Age
0            6      148             72          35        0   33.6           0.627   50
1            1       85             66          29        0   26.6           0.351   31
2            8      183             64          0        0   23.3           0.672   32
3            1       89             66          23       94   28.1           0.167   21
4            0      137             40          35      168   43.1           2.288   33
...
763           10     101             76          48      180   32.9           0.171   63
764           2      122             70          27        0   36.8           0.340   27
765           5      121             72          23      112   26.2           0.245   30
766           1      126             60          0        0   30.1           0.349   47
767           1      93              70          31        0   30.4           0.315   23

768 rows × 8 columns
```

```
In [11]: y
```

```

Out[11]: 0      1
         1      0
         2      1
         3      0
         4      1
        ..
763     0
764     0
765     0
766     1
767     0
Name: Outcome, Length: 768, dtype: int64

In [12]: x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,stratify=y,random_state=2)

In [17]: print(x.shape,x_train.shape,x_test.shape)
(768, 8) (614, 8) (154, 8)

In [18]: from sklearn.metrics import accuracy_score

In [20]: model=LogisticRegression(max_iter=1000)
model.fit(x_train,y_train)
x_train_prediction=model.predict(x_train)
trained_data_accuracy=accuracy_score(y_train,x_train_prediction)
print(trained_data_accuracy)

0.7882736156351792

In [21]: print('Accuracy on training data:',round(trained_data_accuracy*100,2),'%')
Accuracy on training data: 78.83 %

In [22]: x_test_prediction=model.predict(x_test)
test_data_accuracy=accuracy_score(y_test,x_test_prediction)
print(test_data_accuracy)

0.7597402597402597

In [23]: print('Accuracy on test data:',round(test_data_accuracy*100,2),'%')
Accuracy on test data: 75.97 %

```

Result:

The above program executed successfully. Hence output verified

Exercise No. : 13 Customer Segmentation using K-Means Clustering

Date:

Aim:

Write a program Customer Segmentation using K-Means Clustering

Procedure:

- Step 1: Loading the data from csv file to a Pandas DataFrame
- Step 2: First 5 rows in the dataframe
- Step 3: Finding the number of rows and columns
- Step 4: Getting some informations about the dataset
- Step 5: Checking for missing values
- Step 6: Finding wcss value for different number of clusters
- Step 7: Plot an elbow graph
- Step 8: Optimum Number of Clusters = 5
- Step 9: Training the k-Means Clustering Model
- Step 10: Return a label for each data point based on their cluster
- Step 11: Visualizing all the Clusters

Program:

```
In [1]: import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.cluster import KMeans
```

```
In [2]: # Loading the data from csv file to a Pandas DataFrame
customer_data = pd.read_csv('Mall_Customers.csv')
```

```
In [3]: # first 5 rows in the dataframe
customer_data.head()
```

```
Out[3]:   CustomerID  Gender  Age  Annual Income (k$)  Spending Score (1-100)
0            1    Male   19             15                39
1            2    Male   21             15                81
2            3  Female   20             16                 6
3            4  Female   23             16                77
4            5  Female   31             17                40
```

```
In [4]: # finding the number of rows and columns
customer_data.shape
```

```
Out[4]: (200, 5)
```

```
In [5]: # getting some informations about the dataset
customer_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 5 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   CustomerID      200 non-null    int64  
 1   Gender          200 non-null    object  
 2   Age              200 non-null    int64  
 3   Annual Income (k$) 200 non-null    int64  
 4   Spending Score (1-100) 200 non-null    int64  
dtypes: int64(4), object(1)
memory usage: 7.9+ KB
```

```
In [6]: # checking for missing values  
customer_data.isnull().sum()
```

```
Out[6]: CustomerID      0  
Gender          0  
Age            0  
Annual Income (k$)  0  
Spending Score (1-100) 0  
dtype: int64
```

```
In [7]: X = customer_data.iloc[:,[3,4]].values
```

```
In [8]: print(X)
```

```
[[ 15  39]  
 [ 15  81]  
 [ 16   6]  
 [ 16  77]  
 [ 17  40]  
 [ 17  76]  
 [ 18   6]  
 [ 18  94]  
 [ 19   3]  
 [ 19  72]  
 [ 19  14]  
 [ 19  99]  
 [ 20  15]  
 [ 20  77]  
 [ 20  13]  
 [ 20  79]  
 [ 21  35]  
 [ 21  66]  
 [ 23  29]  
 [ 23  29]
```

Choosing the number of clusters

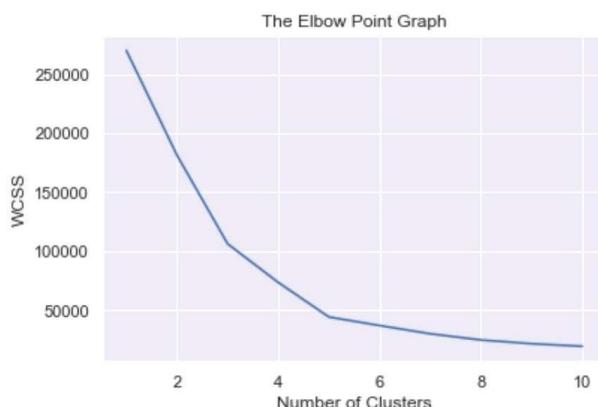
WCSS -> Within Clusters Sum of Squares

```
In [9]: # finding wcss value for different number of clusters
```

```
wcss = []  
  
for i in range(1,11):  
    kmeans = KMeans(n_clusters=i, init='k-means++', random_state=42)  
    kmeans.fit(X)  
  
    wcss.append(kmeans.inertia_)
```

```
In [10]: # plot an elbow graph
```

```
sns.set()  
plt.plot(range(1,11), wcss)  
plt.title('The Elbow Point Graph')  
plt.xlabel('Number of Clusters')  
plt.ylabel('WCSS')  
plt.show()
```

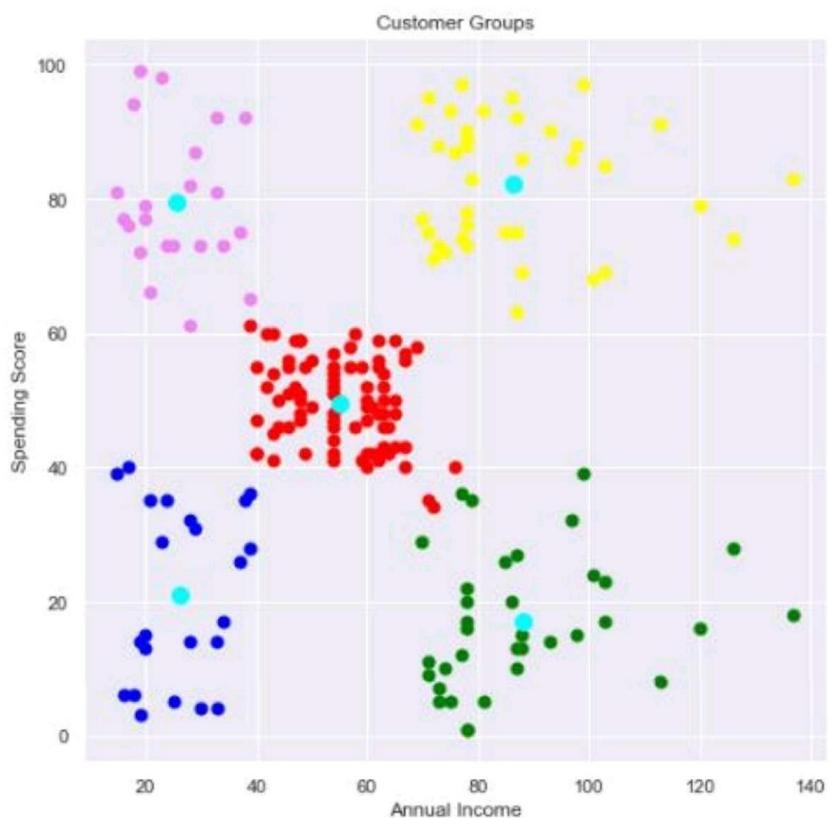


```
In [12]: # 5 Clusters - 0, 1, 2, 3, 4
# Visualizing all the Clusters
# plotting all the clusters and their Centroids

plt.figure(figsize=(8,8))
plt.scatter(X[Y==0,0], X[Y==0,1], s=50, c='green', label='Cluster 1')
plt.scatter(X[Y==1,0], X[Y==1,1], s=50, c='red', label='Cluster 2')
plt.scatter(X[Y==2,0], X[Y==2,1], s=50, c='yellow', label='Cluster 3')
plt.scatter(X[Y==3,0], X[Y==3,1], s=50, c='violet', label='Cluster 4')
plt.scatter(X[Y==4,0], X[Y==4,1], s=50, c='blue', label='Cluster 5')

# plot the centroids
plt.scatter(kmeans.cluster_centers_[:,0], kmeans.cluster_centers_[:,1], s=100, c='cyan', label='Centroids')

plt.title('Customer Groups')
plt.xlabel('Annual Income')
plt.ylabel('Spending Score')
plt.show()
```



Result:

The above program executed successfully. Hence output verified

Lab Viva Voce Questions

No.	Question	Answer
1	What is overfitting?	Model performs well on training data but poorly on unseen data.
2	What is underfitting?	Model is too simple, can't capture the data pattern.
3	Define accuracy.	Ratio of correctly predicted observations to total.
4	What is precision?	$\frac{\text{TP}}{\text{TP} + \text{FP}}$
5	What is recall?	$\frac{\text{TP}}{\text{TP} + \text{FN}}$
6	What is F1 Score?	Harmonic mean of precision and recall.
7	What is confusion matrix?	Table for evaluating classifier performance.
8	Define entropy in decision trees.	Measure of impurity in a dataset.
9	What is Gini index?	Another impurity measure used in trees.
10	Difference between supervised and unsupervised learning?	Labeled vs. unlabeled data.
11	What is a hyperparameter?	Parameters set before training (e.g., learning rate).
12	What is cross-validation?	Splitting data for better evaluation.
13	What is Naïve Bayes based on?	Bayes Theorem with independence assumption.
14	What is a kernel in SVM?	Function to transform input space.
15	When to use logistic regression?	For binary classification problems.
16	Why use feature scaling?	Ensures features contribute equally.
17	What is PCA?	Technique for dimensionality reduction.
18	What is a ROC curve?	Plot of TPR vs. FPR.
19	What is AUC?	Area under ROC curve.
20	What is EM algorithm?	Iterative optimization for clustering.
21	Define KNN.	Classifies based on nearest neighbors.
22	What is distance metric in KNN?	Usually Euclidean.
23	What is a Bayesian Network?	Graphical model for probabilistic relationships.
24	What is missing value imputation?	Replacing missing data using strategy.

No.	Question	Answer
25	What is standardization?	Mean = 0, SD = 1.
26	What is normalization?	Scaling to [0,1] range.
27	Define epoch.	One pass over the training data.
28	Define learning rate.	Controls step size in weight update.
29	What is pruning in decision tree?	Reducing size to avoid overfitting.
30	What is early stopping?	Stop training when validation error increases.
31	Why use backpropagation?	To minimize error in neural nets.
32	What is weight decay?	Regularization technique.
33	What is stratified sampling?	Maintains class proportions.
34	Difference between generative and discriminative models?	Generative models learn data distribution.
35	What is dropout?	Regularization in neural nets.
36	What is the sigmoid function?	Activation function (0,1).
37	What is ReLU?	$f(x)=\max(0,x)$
38	What is softmax?	Converts logits to probabilities.
39	What is one-hot encoding?	Binary representation of categorical data.
40	What is label encoding?	Converts categories to integers.
41	What is class imbalance?	Unequal class distribution.
42	How to handle class imbalance?	Use SMOTE, class weights, etc.
43	What is ensemble learning?	Combining multiple models.
44	What is bagging?	Parallel ensemble (e.g., Random Forest).
45	What is boosting?	Sequential ensemble (e.g., XGBoost).
46	What is a decision boundary?	Surface separating classes.
47	What is curse of dimensionality?	High-dimensional data becomes sparse.
48	What is bias?	Error due to simplistic assumptions.
49	What is variance?	Error due to sensitivity to small changes.
50	What is grid search?	Tuning hyperparameters using exhaustive search.