MEENAKSHI COLLEGE OF ENGINEERING

WEST K.K.NAGAR, CHENNAI-600 078.

(Approved by AICTE and Affiliated to ANNA UNIVERSITY)

MC4311 - MACHINE LEARNING LABORATORY

NAME : ______________________

REG. NO. : ______________________

BRANCH : ______________________

YEAR : ______________________

SEMESTER : ______________________

MEENAKSHI COLLEGE OF ENGINEERING

(Approved by AICTE and affiliated to Anna University)

West K.K Nagar, Chennai-600 078.

BONAFIDE CERTIFICATE

Certified that this is a bonafide record of the work done by **Selvan/Selvi**…………………………………**Reg.No**……………..……………of **II** year **MCA** in **MC4311 MACHINE LEARNING** Laboratory during the academic year 2024-2025.

STAFF-IN-CHARGE **HEAD OF THE DEPARTMENT**

(Mrs. Irine Priya J) (Mr. K. Ram Dev)

Submitted for the Practical Examination held on _______________

INTERNAL EXAMINER **EXTERNAL EXAMINER**

INDEX

INDEX

Ex.No. :1	Demonstrate how do you structure data in Machine Learning
Date:	

Aim:

Write a python program to demonstrate how to structure data in MachineLearning.

Algorithm:

Step 1:Determine the type of problem:

- Is it a supervised learning, unsupervised learning or reinforcement learning problem?
- What is the target variable and what are the input variables?

Step 2: Gather and collect data:

- Get a dataset or create one if needed
- Clean and pre-process the data to handle missing values, outliers, and convert categorical variables to numerical values.

Step 3: Split the data into training and testing sets:

- Allocate a portion of the data for training and another portionfor testing the model.
- Common split ratios are 80:20 or 70:30.

Step 4: Feature engineering:

- Select the relevant features to be used in the model.

- Transform or normalize the features to improve model performance.

Step 5: Train the model:

- Select an appropriate machine learning algorithm based on theproblem type and data
- Train the model using the training set and evaluate its performance using the testing set.

Step 6: Hyperparameter tuning:

- Optimize the performance of the model by tuning the hyperparameters.
- Use techniques like grid search or random search to find theoptimal hyperparameters.

Step 7: Evaluate and refine the model:

- Evaluate the performance of the model using evaluation metricslike accuracy, precision, recall, F1 score, etc.
- Refine the model based on the evaluation results and repeat theprocess until satisfactory performance is achieved.

Step 8: Deploy the model:

- Deploy the model in a production environment and monitor its performance.
- Continuously update the model as new data is collected.

Program:

```
import pandas as pd

from sklearn.model_selection import train_test_split from

sklearn.linear_model import LogisticRegressionfrom

sklearn.metrics import accuracy_score

from sklearn import datasets#

Load the iris dataset

iris = datasets.load_iris()

X = iris.data y

= iris.target

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random
_state=42)

# Initialize and fit the modelclf =

LogisticRegression()

clf = LogisticRegression(max_iter=1000)

clf.fit(X_train, y_train)

# Make predictions on the test set

y_pred = clf.predict(X_test)

# Evaluate the model's performance score =

accuracy_score(y_test, y_pred)

print(f'Accuracy: {score:.2f}')
```

Output:

```
Accuracy: 1.00
```

Result:

Thus, the python program to demonstrate how to structure data in MachineLearning has been successfully implemented and executed.

Ex.No. :2	**Implement data preprocessing techniques on real time dataset**
Date:	

Aim:

Write a python program to implement data preprocessing techniques on realtime dataset.

Algorithm:

Step 1: Load the dataset:

- Read the dataset into a pandas dataframe or numpy array.

Step 2: Handle missing values:

- Identify missing values and decide on an appropriate strategyfor handling them.
- Common strategies include dropping the missing values, imputing the missing values with mean, median, mode or usingmachine learning algorithms like KNN.

Step 3: Handle outliers:

- Identify outliers and decide on an appropriate strategy forhandling them.
- Common strategies include dropping the outliers, imputing the outliers with mean, median, mode or using statistical methods like winsorization.

Step 4: Convert categorical variables to numerical variables:

- Identify categorical variables and convert them to numericalvalues.
- Common strategies include one-hot encoding or ordinal encoding.

Step 5: Split the data into training and testing sets:

- Allocate a portion of the data for training and another portionfor testing the model.
- Common split ratios are 80:20 or 70:30.

Step 6: Normalize the data:

- Normalize the data to bring all the variables to th same scale.
- Common normalization techniques include min-max scaling or z-score normalization.

Step 7: Store the pre-processed data:

- Store the pre-processed data in a new file or data structure foruse in the next step of the machine learning pipeline.

Step 8: Validate the pre-processed data:

- Validate the pre-processed data by comparing it to the original data to ensure that the pre-processing steps have been correctlyimplemented and that the data has been correctly transformed.

diabetes.csv

num_preg	glucose_conc	diastolic_bp	thickness	insulin	bmi	diab_pred	age	diabetes
6	148	72	35	0	33.6	0.627	50	1
1	85	66	29	0	26.6	0.351	31	0
8	183	64	0	0	23.3	0.672	32	1
1	89	66	23	94	28.1	0.167	21	0
0	137	40	35	168	43.1	2.288	33	1
5	116	74	0	0	25.6	0.201	30	0
3	78	50	32	88	31	0.248	26	1
10	115	0	0	0	35.3	0.134	29	0
2	197	70	45	543	30.5	0.158	53	1
8	125	96	0	0	0	0.232	54	1
4	110	92	0	0	37.6	0.191	30	0
10	168	74	0	0	38	0.537	34	1
10	139	80	0	0	27.1	1.441	57	0
1	189	60	23	846	30.1	0.398	59	1
5	166	72	19	175	25.8	0.587	51	1
7	100	0	0	0	30	0.484	32	1
0	118	84	47	230	45.8	0.551	31	1
7	107	74	0	0	29.6	0.254	31	1
1	103	30	38	83	43.3	0.183	33	0
1	115	70	30	96	34.6	0.529	32	1
3	126	88	41	235	39.3	0.704	27	0
8	99	84	0	0	35.4	0.388	50	0
7	196	90	0	0	39.8	0.451	41	1
9	119	80	35	0	29	0.263	29	1

Program:

```
# importing libraries
import pandas import
scipy import numpy
from sklearn.preprocessing import MinMaxScaler from
sklearn.preprocessing import StandardScaler from
sklearn.preprocessing import Binarizer

# data set link
# data parameters
names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class']

# preparating of dataset using the data at given link and defined columns listdataset =
pandas.read_csv('diabetes.csv')
array = dataset.values

# separate array into input and output components X =
array[:,0:8]
Y = array[:,8]

# initialising the MinMaxScaler
scaler = MinMaxScaler(feature_range=(0, 1))
# learning the statistical parameters for each of the data and transformingrescaledX =
scaler.fit_transform(X)
# summarize transformed data
numpy.set_printoptions(precision=3)
print("Rescaled Data") print(rescaledX[0:5,:])

#standardize
scaler = StandardScaler().fit(X)
```

```
rescaledX = scaler.transform(X)

# summarize transformed data
numpy.set_printoptions(precision = 3)
print("Standardized Data")
print(rescaledX[0:5,:])
#binarize
binarizer = Binarizer(threshold = 0.0).fit(X)
binaryX = binarizer.transform(X)

# summarize transformed data
numpy.set_printoptions(precision = 3)
print("Binarized Data") print(binaryX[0:5,:])
```

Output:

```
Rescaled Data
[[0.353 0.744 0.59  0.354 0.        0.501 0.234 0.483]
 [0.059 0.427 0.541 0.293 0.        0.396 0.117 0.167]
 [0.471 0.92  0.525 0.       0.     0.347 0.254 0.183]
 [0.059 0.447 0.541 0.232 0.111 0.419 0.038 0.      ]
 [0.     0.688 0.328 0.354 0.199 0.642 0.944 0.2  ]]
Standardized Data
[[ 0.64      0.848  0.15     0.907 -0.693  0.204  0.468  1.426]
 [-0.845 -1.123 -0.161  0.531 -0.693 -0.684 -0.365 -0.191]
 [ 1.234  1.944 -0.264 -1.288 -0.693 -1.103  0.604 -0.106]
 [-0.845 -0.998 -0.161  0.155  0.123 -0.494 -0.921 -1.042]
 [-1.142  0.504 -1.505  0.907  0.766  1.41      5.485 -0.02 ]]
Binarized Data
[[1. 1. 1. 1. 0. 1. 1. 1.]
 [1. 1. 1. 1. 0. 1. 1. 1.]
 [1. 1. 1. 0. 0. 1. 1. 1.]
 [1. 1. 1. 1. 1. 1. 1. 1.]
 [0. 1. 1. 1. 1. 1. 1. 1.]]
```

Result:

Thus, the python program to implement data preprocessing techniques onreal time datasethas been successfully implemented and executed.

Ex.No. :3	Implement Feature subset selection techniques
Date:	

Aim:

Write a python program toimplement Feature subset selection techniques.

Algorithm:

Step 1: Load the dataset:

- Read the dataset into a pandas dataframe or numpy array.

Step 2: Split the data into training and testing sets:

- Allocate a portion of the data for training and another portionfor testing the model.
- Common split ratios are 80:20 or 70:30.

Step 3: Perform feature scaling:

- Normalize the data to bring all the variables to the same scale.
- Common normalization techniques include min-max scaling orz-score normalization.

Step 4: Choose the feature selection technique:

- Select a feature selection technique based on the problem typeand data.
- Common feature selection techniques include filter methods,wrapper methods, and embedded methods.

Step 5: Apply the feature selection technique:

- Apply the selected feature selection technique to the data andobtain the subset of features.

Step 6: Train the model:

- Train the machine learning model using the selec ed subset offeatures.

Step 7: Evaluate the model:

- Evaluate the performance of the model using evaluation metricslike accuracy, precision, recall, F1 score, etc.

Step 8: Refine the feature subset:

- Refine the feature subset by repeating the feature selectionprocess with different techniques or different subsets of features.

Step 9: Choose the final feature subset:

- Choose the final feature subset based on the modand l performance feature interpretation.

Step 10: Store the final feature subset:

- Store the final feature subset for use in the next step of the machinelearning pipeline.

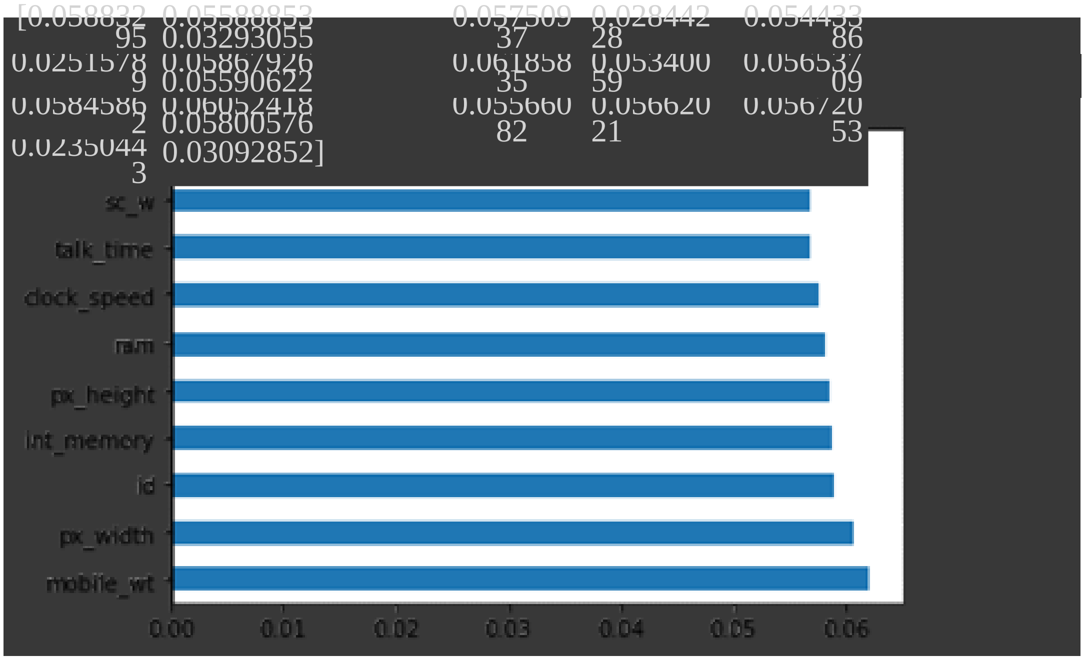
test.csv

id	battery	blue	clock_speec	dual_sim	fc	four_g	int_memc	m_dep
1	1043	1	1.8	1	14	0	5	0.1
2	841	1	0.5	1	4	1	61	0.8
3	1807	1	2.8	0	1	0	27	0.9
4	1546	0	0.5	1	18	1	25	0.5
5	1434	0	1.4	0	11	1	49	0.5
6	1464	1	2.9	1	5	1	50	0.8
7	1718	0	2.4	0	1	0	47	1
8	833	0	2.4	1	0	0	62	0.8
9	1111	0	2.9	1	9	1	25	0.6
10	1520	1	0.5	0	1	0	35	0.5

Program:

```
import pandas as pd

import numpy as np

data = pd.read_csv("test.csv") #train.csv can also be usedX =

data.iloc[:,0:20] #independent variable columns

y = data.iloc[:,-1]         #target variable column (price range)from

sklearn.ensemble import ExtraTreesClassifier import

matplotlib.pyplot as plt

model = ExtraTreesClassifier()

model.fit(X,y)

print(model.feature_importances_) #plot the

graph of feature importances

feat_importances = pd.Series(model.feature_importances_, index=X.columns
)

feat_importances.nlargest(10).plot(kind='barh')plt.show()
```

Output:



Result:

Thus, the python program to implement Feature subset selectiontechniques has been successfully implemented and executed.

Ex.No. :4	**Demonstrate how will you measure the performance of a machine learning model**
Date:	

Aim:

Write a python program todemonstrate how will you measure the performance of a machine learning model.

Algorithm:

Step 1: Load the dataset:

- Read the dataset into a pandas dataframe or numpy array.

Step 2: Split the data into training and testing sets:

- Allocate a portion of the data for training and another portionfor testing the model.
- Common split ratios are 80:20 or 70:30.

Step 3: Train the model:

- Train the machine learning model using the training set.

Step 4: Make predictions:

- Use the trained model to make predictions on the testing set.

Step 5: Choose the evaluation metric:

- Choose the appropriate evaluation metric based on the problemtype and the nature of the target variable.
- Common evaluation metrics for classification problems includeaccuracy, precision, recall, F1 score, etc.

- Common evaluation metrics for regression problems includemean absolute error, mean squared error, R-squared, etc.

Step 6: Calculate the evaluation metric:

- Calculate the chosen evaluation metric using the predictionsand the actual values.

Step 7: Evaluate the model:

- Evaluate the performance of the model based on the calculatedevaluation metric.

Step 8: Refine the model:

- Refine the model based on the evaluation results and repeat theprocess until satisfactory performance is achieved.

Step 9: Store the final model:

- Store the final model for use in the next step of the machinelearning pipeline

diabetes.csv

num_preg	glucose_conc	diastolic_bp	thickness	insulin	bmi	diab_pred	age	diabetes
6	148	72	35	0	33.6	0.627	50	1
1	85	66	29	0	26.6	0.351	31	0
8	183	64	0	0	23.3	0.672	32	1
1	89	66	23	94	28.1	0.167	21	0
0	137	40	35	168	43.1	2.288	33	1
5	116	74	0	0	25.6	0.201	30	0
3	78	50	32	88	31	0.248	26	1
10	115	0	0	0	35.3	0.134	29	0
2	197	70	45	543	30.5	0.158	53	1
8	125	96	0	0	0	0.232	54	1
4	110	92	0	0	37.6	0.191	30	0
10	168	74	0	0	38	0.537	34	1
10	139	80	0	0	27.1	1.441	57	0
1	189	60	23	846	30.1	0.398	59	1
5	166	72	19	175	25.8	0.587	51	1
7	100	0	0	0	30	0.484	32	1
0	118	84	47	230	45.8	0.551	31	1
7	107	74	0	0	29.6	0.254	31	1
1	103	30	38	83	43.3	0.183	33	0
1	115	70	30	96	34.6	0.529	32	1
3	126	88	41	235	39.3	0.704	27	0
8	99	84	0	0	35.4	0.388	50	0
7	196	90	0	0	39.8	0.451	41	1
9	119	80	35	0	29	0.263	29	1

Program:

```
import pandas

from sklearn import model_selection

from sklearn.linear_model import LogisticRegression

dataframe = pandas.read_csv("diabetes.csv")

array = dataframe.valuesX =

array[:,0:8]

Y = array[:,8]

# Evaluate using a train and a test set

test_size = 0.33

seed = 7

X_train, X_test, Y_train, Y_test = model_selection.train_test_split(X, Y, test_size=test_size,
random_state=seed)

model = LogisticRegression()

model.fit(X_train, Y_train)

result = model.score(X_test, Y_test)

print("Evaluating using Train and Test sets")

print("Accuracy: %.3f%%" % (result*100.0))

# Evaluate using Leave One Out Cross Validation

num_folds = 10

num_instances = len(X)

loocv = model_selection.LeaveOneOut()model

= LogisticRegression()
```

```
results = model_selection.cross_val_score(model, X, Y, cv=loocv)

print("Evaluating using Leave One Out Cross Validation")

print("Accuracy: %.3f%% (%.3f%%)" % (results.mean()*100.0, results.std()*100.0))
```

Output:

```
Evaluating using Train and Test setsAccuracy:
78.740%
Evaluating using Leave One Out Cross ValidationAccuracy:
77.865% (41.516%)
```

Result:

Thus, the python program to demonstrate how will you measure the performance of a machine learning modelhas been successfully implementedand executed.

Ex.No. :5	**Implement the naïve Bayesian classifier for a sample training data set. Compute the accuracy of the classifier, considering few test data sets.**
Date:	

Aim:

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

Algorithm:

Step 1: Load the dataset:

- Read the dataset into a pandas dataframe or numpy array.

Step 2: Split the data into training and testing sets:

- Allocate a portion of the data for training and another portionfor testing the model.
- Common split ratios are 80:20 or 70:30.

Step 3: Prepare the data:

- Prepare the data by converting categorical variables to numerical variables and normalizing the data if needed.

Step 4: Compute class probabilities:

- Compute the prior class probabilities using the formula:P(class) = count(class) / total_examples

Step 5: Compute class-conditional probabilities:

- Compute the class-conditional probabilities for each feature using the formula: P(feature|class) = count(feature and class) /count(class)

Step 6: Predict class for test examples:

- For each test example, compute the likelihood of the example belonging to each class and choose the class with the maximumlikelihood.

Step 7: Compute accuracy:

- Compute the accuracy of the classifier by comparing the predicted class labels with the actual class labels in the testingset.
- Accuracy can be calculated as the ratio of the number of correct predictions to the total number of predictions.

Step 8: Visualize the accuracy:

- Visualize the accuracy using appropriate plots like bar plots,confusion matrices, etc.

Step 9: Store the evaluation results:

- Store the evaluation results for future reference and comparison.

Program:

```
from sklearn.naive_bayes import MultinomialNBfrom
sklearn.metrics import accuracy_score
from sklearn.feature_extraction.text import CountVectorizer

#Sample training data
#Assume the data is in the format [features, label]
training_data = [['Chinese Beijing Chinese', 'china'],
['Chinese Chinese Shanghai', 'china'],
['Chinese Macao', 'china'], ['Tokyo
Japan Chinese', 'japan']]

#Prepare the data for training X =
[i[0] for i in training_data]y = [i[1]
for i in training_data]

#Initialize the vectorizer vectorizer =
CountVectorizer()

#Transform the training data using the vectorizerX =
vectorizer.fit_transform(X)

#Initialize the classifierclf
= MultinomialNB()

#Train the classifier
clf.fit(X, y)
```

```
#Sample test data

test_data = ['Chinese Chinese Chinese Tokyo Japan', 'Beijing China']test_labels
= ['japan', 'china']


#Transform the test data using the vectorizer

test_data = vectorizer.transform(test_data)


#Make predictions

predictions = clf.predict(test_data)

#Compute the accuracy

accuracy = accuracy_score(test_labels, predictions)

print("Accuracy: ", accuracy)
```

Output:

```
Accuracy:  0.5
```

Result:

Thus, the python program to implement the naïve Bayesian classifier for a sample training data set. Compute the accuracy of the classifier, consideringfew test data setshas been successfully implemented and executed.

Ex.No. :6	Construct a Bayesian network considering medical data. Use this model to demonstrate the diagnosis of heart patients using the standard Heart Disease Data Set.
Date:	

Aim:

Write a python program to Construct a Bayesian network considering medical data. Use this model to demonstrate the diagnosis of heart patientsusing the standard Heart Disease Data Set.

Algorithm:

Step 1: Load the dataset:

- Read the standard Heart Disease Data Set into a pandas dataframe or numpy array.

Step 2: Prepare the data:

- Prepare the data by converting categorical variables to numerical variables and normalizing the data if needed.

Step 3: Define the structure of the Bayesian network:

- Determine the variables involved in the network and their relationships based on domain knowledge and/or previousresearch.
- For example, age, gender, chest pain type, blood pressure, cholesterol, etc.

Step 4: Compute class probabilities:

- Compute the prior class probabilities using the formula:P(class) = count(class) / total_examples

Step 5: Compute class-conditional probabilities:

- Compute the class-conditional probabilities for each variablegiven the class using the formula: P(variable|class) = count(variable and class) / count(class)

Step 6: Build the Bayesian network:

- Using the structure and probabilities computed, build the Bayesian network using appropriate software tools or libraries.

Step 7: Predict the class for a new patient:

- For a new patient, compute the likelihood of the patient havingheart disease based on the Bayesian network and the patient's characteristics.
- The class with the highest likelihood can be considered as thediagnosis for the patient.

Step 8: Validate the model:

- Validate the model by testing it on a test set and computing itsaccuracy, precision, recall, and F1 score.

Step 9: Refine the model:

- Refine the model based on the validation results and repeat theprocess until satisfactory performance is achieved.

Step 10: Store the evaluation results:

- Store the evaluation results for future reference and comparison.heart.csv

age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	heartdisease
63	1	1	145	233	1	2	150	0	2.3	3	0	6	0
67	1	4	160	286	0	2	108	1	1.5	2	3	3	2
67	1	4	120	229	0	2	129	1	2.6	2	2	7	1
37	1	3	130	250	0	0	187	0	3.5	3	0	3	0
41	0	2	130	204	0	2	172	0	1.4	1	0	3	0
56	1	2	120	236	0	0	178	0	0.8	1	0	3	0
62	0	4	140	268	0	2	160	0	3.6	3	2	3	3
57	0	4	120	354	0	0	163	1	0.6	1	0	3	0
63	1	4	130	254	0	2	147	0	1.4	2	1	7	2
53	1	4	140	203	1	2	155	1	3.1	3	0	7	1
57	1	4	140	192	0	0	148	0	0.4	2	0	6	0
56	0	2	140	294	0	2	153	0	1.3	2	0	3	0
56	1	3	130	256	1	2	142	1	0.6	2	1	6	2
44	1	2	120	263	0	0	173	0	0	1	0	7	0
52	1	3	172	199	1	0	162	0	0.5	1	0	7	0
57	1	3	150	168	0	0	174	0	1.6	1	0	3	0
48	1	2	110	229	0	0	168	0	1	3	0	7	1
54	1	4	140	239	0	0	160	0	1.2	1	0	3	0
48	0	3	130	275	0	0	139	0	0.2	1	0	3	0
49	1	2	130	266	0	0	171	0	0.6	1	0	3	0
64	1	1	110	211	0	2	144	1	1.8	2	0	3	0
58	0	1	150	283	1	2	162	0	1	1	0	3	0
58	1	2	120	284	0	2	160	0	1.8	2	0	3	1
58	1	3	132	224	0	2	173	0	3.2	1	2	7	3

Program:

```
import   numpy   as   np
import   pandas   as   pd
import csv
from pgmpy.estimators import MaximumLikelihoodEstimatorfrom
pgmpy.models import BayesianModel
from pgmpy.inference import VariableElimination#read
Cleveland Heart Disease data
heartDisease = pd.read_csv('heart.csv') heartDisease =
heartDisease.replace('?',np.nan)#display the data
print('Sample instances from the dataset are given below')
print(heartDisease.head())
#display the Attributes names and datatyes
print('\n Attributes and datatypes')
print(heartDisease.dtypes)
#Creat Model- Bayesian Network
model = BayesianModel([('age','heartdisease'),('sex','heartdisease'),(
'exang','heartdisease'),('cp','heartdisease'),('heartdisease', 'restecg'),('heartdisease','chol')])
#Learning CPDs using Maximum Likelihood Estimators print('\n
Learning CPD using Maximum likelihood estimators')
model.fit(heartDisease,estimator=MaximumLikelihoodEstimator)

# Inferencing with Bayesian Network
print('\n Inferencing with Bayesian Network:')
HeartDiseasetest_infer = VariableElimination(model) #computing
the Probability of HeartDisease given restecg
print('\n 1.Probability of HeartDisease given evidence= restecg :1')
```

q1=HeartDiseasetest_infer.query(variables=['heartdisease'],evidence={'restec g':1})

print(q1)

#computing the Probability of HeartDisease given cp

print('\n 2.Probability of HeartDisease given evidence= cp:2 ')

q2=HeartDiseasetest_infer.query(variables=['heartdisease'],evidence={'cp':2}) print(q2)

Output:

```
================ RESTART: E:\ML Lab - [illegible]\MLLab-7\ML7.py ================
Few examples from the dataset are given below
   age  sex  cp  trestbps  chol  ...  oldpeak  slope  ca  thal  heartdisease
0   63    1   1       145   233  ...      2.3      3   0     6             0
1   67    1   4       160   286  ...      1.5      2   3     3             2
2   67    1   4       120   229  ...      2.6      2   2     7             1
3   37    1   3       130   250  ...      3.5      3   0     3             0
4   41    0   2       130   204  ...      1.4      1   0     3             0

[5 rows x 14 columns]

 Attributes and datatypes
age               int64
sex               int64
cp                int64
trestbps          int64
chol              int64
fbs               int64
restecg           int64
thalach           int64
exang             int64
oldpeak         float64
slope             int64
ca               object
thal             object
heartdisease      int64
dtype: object
```

```
Learning CPD using Maximum likelihood estimators

 Inferencing with Bayesian Network:

 1. Probability of HeartDisease given evidence= restecg

    +-----------------+---------------------+
    | heartdisease    |   phi(heartdisease) |
    +=================+=====================+
    | heartdisease(0) |              0.1012 |
    +-----------------+---------------------+
    | heartdisease(1) |              0.0000 |
    +-----------------+---------------------+
    | heartdisease(2) |              0.2392 |
    +-----------------+---------------------+
    | heartdisease(3) |              0.2015 |
    +-----------------+---------------------+
    | heartdisease(4) |              0.4581 |
    +-----------------+---------------------+
```

2. Probability of HeartDisease given evidence= cp

```
+-----------------+---------------------+
| heartdisease    |   phi(heartdisease) |
+=================+=====================+
| heartdisease(0) |              0.3610 |
+-----------------+---------------------+
| heartdisease(1) |              0.2159 |
+-----------------+---------------------+
| heartdisease(2) |              0.1373 |
+-----------------+---------------------+
| heartdisease(3) |              0.1537 |
+-----------------+---------------------+
| heartdisease(4) |              0.1321 |
+-----------------+---------------------+
```

Result:

Thus, the python 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 Sethas been successfully implementedand executed.

Ex.No. :7	**Apply EM algorithm to cluster a set of data stored in a .CSV file**
Date:	

Aim:

Write a python program to apply EM algorithm to cluster a set of data storedin a .CSV file.

Algorithm:

Step 1: Load the dataset:

- Read the .CSV file into a pandas dataframe or numpy array.

Step 2: Prepare the data:

- Prepare the data by converting categorical variables to numerical variables and normalizing the data if needed.

Step 3: Choose the number of clusters:

- Choose the number of clusters for the data based on domain knowledge and/or previous research.

Step 4: Initialize the model parameters:

- Initialize the mean and covariance matrices for each clusterbased on random or heuristic methods.

Step 5: Perform the E-step:

- Compute the responsibilities (weights) of each data point toeach cluster using the formula: r_ij = P(z_j | x_i)
- where r_ij is the responsibility of the j-th cluster for the i-th data point, z_j is the j-th cluster, and x_i is the i-th data point.

Step 6: Perform the M-step:

- Re-estimate the mean and covariance matrices for each clusterusing the formula: mu_j = (1/N_j) * sum(r_ij * x_i) sigma_j =(1/N_j) * sum(r_ij * (x_i - mu_j) * (x_i - mu_j)^T)
- where mu_j is the mean of the j-th cluster, sigma_j is the covariance matrix of the j-th cluster, N_j is the number of datapoints assigned to the j-th cluster, and x_i is the i-th data point.

Step 7: Repeat steps 5 and 6 until convergence:

- Repeat the E-step and M-step until the model parameters nolonger change or the change is smaller than a predefined threshold.

Step 8: Assign cluster labels to the data points:

- Assign the cluster label to each data point based on the maximum responsibility of the data point to a cluster.

Step 9: Evaluate the model:

- Evaluate the model by computing the silhouette score or theDavies-Bouldin index.

Step 10: Visualize the results:

- Visualize the results by plotting the data points in a 2D or 3Dspace with different colors representing different clusters.

Step 11: Store the evaluation results:

- Store the evaluation results for future reference and comparison.

Program:

```
import pandas as pd

from sklearn.mixture import GaussianMixturefrom

sklearn.datasets import load_iris


# Load iris datasetiris

= load_iris()

iris_data = pd.DataFrame(data=iris.data, columns=iris.feature_names)


# Initialize the Gaussian Mixture Model gmm =

GaussianMixture(n_components=3)


# Fit the model to the data

gmm.fit(iris_data)


# Predict the cluster labels for the datalabels

= gmm.predict(iris_data)


# Print the cluster labels

print(labels)
```

Output:

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

Result:

Thus, the python program to apply EM algorithm to cluster a set of datastored in a .CSV file has been successfully implemented and executed.

Ex.No. :8	Implement k-Nearest Neighbor algorithm to classify the data set
Date:	

Aim:

Write a python program to implement k-Nearest Neighbor algorithm toclassify the data set.

Algorithm:

Step 1: Load the dataset:

- Read the .CSV file or dataset into a pandas dataframe or numpyarray.

Step 2: Prepare the data:

- Prepare the data by converting categorical variables to numerical variables and normalizing the data if needed.

Step 3: Choose the number of neighbors:

- Choose the number of neighbors (k) for the k-NN algorithm. This value can be determined through trial and error or using techniques such as cross-validation.

Step 4: Split the dataset into training and test data:

- Split the dataset into two parts, training data and test data. The training data is used to build the model and the test data is usedto evaluate the model.

Step 5: Train the model:

- The k-NN algorithm does not have a training phase, so there isno need to train the model.

Step 6: Predict the class labels for test data:

- For each test data point, compute the Euclidean distancebetween the test data point and each training data point.
- Sort the distances in ascending order and choose the k nearest neighbors.
- Predict the class label for the test data point as the mostcommon class label among the k nearest neighbors.

Step 7: Evaluate the model:

- Evaluate the model by computing the accuracy, precision,recall, and F1 score on the test data.

Step 8: Visualize the results:

- Visualize the results by plotting the test data points in a 2D or3D space with different colors representing different classes.

Step 9: Store the evaluation results:

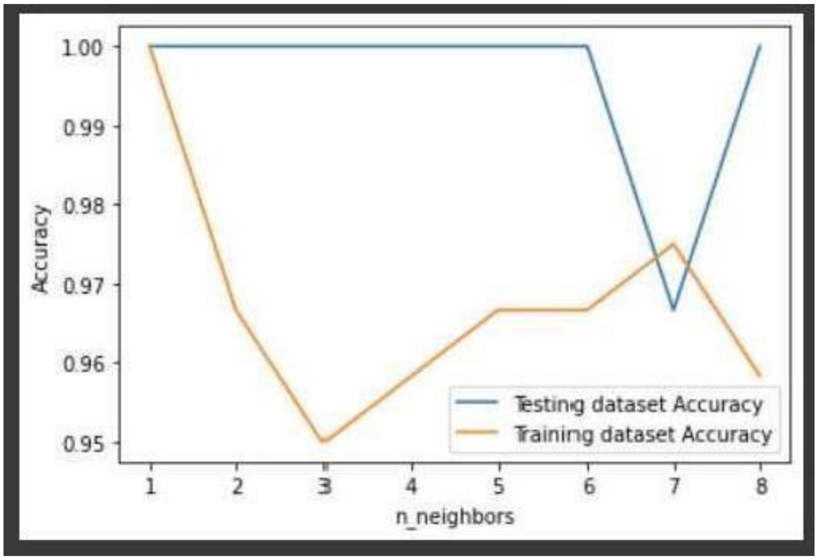
- Store the evaluation results for future reference and comparison.

Program:

```
# Import necessary modules
from sklearn.neighbors import KNeighborsClassifier from
sklearn.model_selection import train_test_split from
sklearn.datasets import load_iris
import numpy as np
import matplotlib.pyplot as plt
irisData = load_iris()
# Create feature and target arraysX =
irisData.data
y = irisData.target
# Split into training and test set
X_train, X_test, y_train, y_test = train_test_split(X, y,
    test_size = 0.2, random_state=42)
neighbors = np.arange(1, 9) train_accuracy =
np.empty(len(neighbors))test_accuracy =
np.empty(len(neighbors)) # Loop over K values
for i, k in enumerate(neighbors):
  knn = KNeighborsClassifier(n_neighbors=k)
  knn.fit(X_train, y_train)
  # Compute training and test data accuracy
  train_accuracy[i] = knn.score(X_train, y_train)
  test_accuracy[i] = knn.score(X_test, y_test)#
Generate plot
plt.plot(neighbors, test_accuracy, label = 'Testing dataset Accuracy')
plt.plot(neighbors, train_accuracy, label = 'Training dataset Accuracy')plt.legend()
```

```
plt.xlabel('n_neighbors')
plt.ylabel('Accuracy')
plt.show()
```

Output:



Result:

Thus, thepython program to implement k-Nearest Neighbor algorithm toclassify the data sethas been successfully implemented and executed.

Ex.No. :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**
Date:	

Aim:

Write a python program to apply the technique of pruning for a noisy data monk2 data, and derive the decision tree from this data. Analyze the results bycomparing the structure of pruned and unpruned tree.

Algorithm:

Step 1: Load the dataset:

- Read the monk2 data into a pandas dataframe or numpy array.

Step 2: Prepare the data:

- Prepare the data by converting categorical variables to numerical variables and normalizing the data if needed.

Step 3: Split the dataset into training and validation data:

- Split the dataset into two parts, training data and validationdata. The training data is used to build the model and the validation data is used to evaluate the model.

Step 4: Build the decision tree:

- Build the decision tree using the training data and an algorithmsuch as ID3 or C4.5.

Step 5: Prune the tree:

- Prune the tree by removing branches with low accuracy on thevalidation data. The pruning process can be done using algorithms such as reduced error pruning or cost complexitypruning.

Step 6: Evaluate the pruned tree:

- Evaluate the pruned tree by computing the accuracy, precision,recall, and F1 score on the validation data.

Step 7: Visualize the pruned tree:

- Visualize the pruned tree by plotting the tree structure in agraphical format.

Step 8: Compare the structure of the pruned and unpruned tree:

- Compare the structure of the pruned and unpruned tree by analyzing the accuracy, precision, recall, and F1 score on the validation data and the size and complexity of the tree structure.

Step 9: Store the results:

- Store the results for future reference and comparison.

Program:

```
from sklearn.tree import DecisionTreeClassifier from
sklearn.datasets import make_classification from
sklearn.model_selection import train_test_splitfrom
sklearn.metrics import accuracy_score
# Generate a noisy dataset
X, y = make_classification(n_samples=1000, n_features=10, n_classes=2,
                           n_informative=5, n_clusters_per_class=1, random_state=42,
                           class_sep=2, flip_y=0.05)
# Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
# Train a decision tree on the training data
tree = DecisionTreeClassifier(random_state=42)
tree.fit(X_train, y_train)
# Make predictions on the test data
y_pred = tree.predict(X_test)
# Calculate the accuracy of the unpruned tree accuracy_unpruned
= accuracy_score(y_test, y_pred) print("Accuracy of unpruned
tree:", accuracy_unpruned)


# Prune the tree by setting a minimum number of samples required at a leaf node

tree_pruned = DecisionTreeClassifier(min_samples_leaf=20, random_state=42)
tree_pruned.fit(X_train, y_train)

# Make predictions on the test data using the pruned tree
y_pred_pruned = tree_pruned.predict(X_test)
```

Calculate the accuracy of the pruned tree accuracy_pruned =

accuracy_score(y_test, y_pred_pruned)print("Accuracy of pruned

tree:", accuracy_pruned)

Output:

```
Accuracy of unpruned tree: 0.925
Accuracy of pruned tree: 0.965
```

Result:

Thus, the python program to apply the technique of pruning for a noisy datamonk2 data, and derive the decision tree from this data. Analyze the results by comparing the structure of pruned and unpruned treehas beeimplemented and executed .successfully

Ex.No. :10	**Build an Artificial Neural Network by implementing the Backpropagation algorithm and test the same using appropriate data sets**
Date:	

Aim:

Write a python program to build an Artificial Neural Network by implementing the Backpropagation algorithm and test the same usingappropriate data sets.

Algorithm:

Step 1: Load the dataset:

- Read the dataset into a pandas dataframe or numpy array.

Step 2: Prepare the data:

- Prepare the data by converting categorical variables to numerical variables, normalizing the data, and splitting the datainto training, validation, and test sets.

Step 3: Define the neural network architecture:

- Define the architecture of the neural network, including thenumber of input nodes, hidden layers, and output nodes.

Step 4: Initialize the weights and biases:

- Initialize the weights and biases of the neural network randomly.

Step 5: Train the neural network:

- Use the backpropagation algorithm to train the neural networkon the training data. The algorithm involves computing the forward pass, computing the error gradient, and updating theweights and biases.

Step 6: Evaluate the neural network:

- Evaluate the neural network by computing the accuracy, precision, recall, and F1 score on the validation data.

Step 7: Test the neural network:

- Test the neural network by computing the accuracy, precision,recall, and F1 score on the test data.

Step 8: Visualize the results:

- Visualize the results by plotting the training loss and validationaccuracy over time.

Step 9: Store the results:

- Store the results for future reference and comparison.

Program:

```
import numpy as np class

NeuralNetwork:

    def___init___(self, x, y):

        self.input = x

        self.weights1 = np.random.rand(x.shape[1],4)

        self.weights2 = np.random.rand(4,1)

        self.y = y

        self.output = np.zeros(y.shape)def

    feedforward(self):

        self.layer1 = 1 / (1 + np.exp(-np.dot(self.input, self.weights1))) self.output = 1 /

        (1 + np.exp(-np.dot(self.layer1, self.weights2)))return self.output

    def backprop(self):

        d_weights2 = np.dot(self.layer1.T, (self.output -self.y) *
 self.output * (1 - self.output))

        d_weights1 = np.dot(self.input.T,  np.dot((self.output -
 self.y) * self.output * (1 - self.output), self.weights2.T) * self.layer1 * (1 -self.layer1))

        self.weights1 -= d_weights1

        self.weights2 -= d_weights2

X = np.array([[0,0,1],

           [0,1,1],

           [1,0,1],
```

```
                [1,1,1]])
y = np.array([[0],[1],[1],[0]])

nn = NeuralNetwork(X,y)for i
in range(1500):
    nn.feedforward()
    nn.backprop()
print(nn.output)
```

Output:

```
[[0.01717184]
 [0.94984504]
 [0.94981877]
 [0.05929842]]
```

Result:

Thus, the python program to build an Artificial Neural Network by implementing the Backpropagation algorithm and test the same usingappropriate data setshas been successfully implemented and executed.

Ex.No. :11	Implement Support Vector Classification for linear kernels
Date:	

Aim:

Write a python program to implement Support Vector Classification forlinear kernels.

Algorithm:

Step 1: Load the dataset:

- Read the dataset into a pandas dataframe or numpy array.

Step 2: Prepare the data:

- Prepare the data by converting categorical variables to numerical variables, normalizing the data, and splitting the datainto training and test sets.

Step 3: Define the SVM model:

- Define the support vector machine model using a linear kernel.

Step 4: Train the SVM model:

- Train the SVM model on the training data by finding the optimal hyperplane that maximizes the margin between theclasses.

Step 5: Evaluate the SVM model:

- Evaluate the SVM model by computing the accuracy, precision,recall, and F1 score on the test data.

Step 6: Visualize the results:

- Visualize the results by plotting the data points and the decisionboundary of the SVM model.

Step 7: Store the results:

- Store the results for future reference and comparison.

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 Setiris =
datasets.load_iris()
X = iris.data[:, :2]y
= iris.target
# C is the SVM regularization parameterC =
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() + 1h =
(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 modelZ =
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')

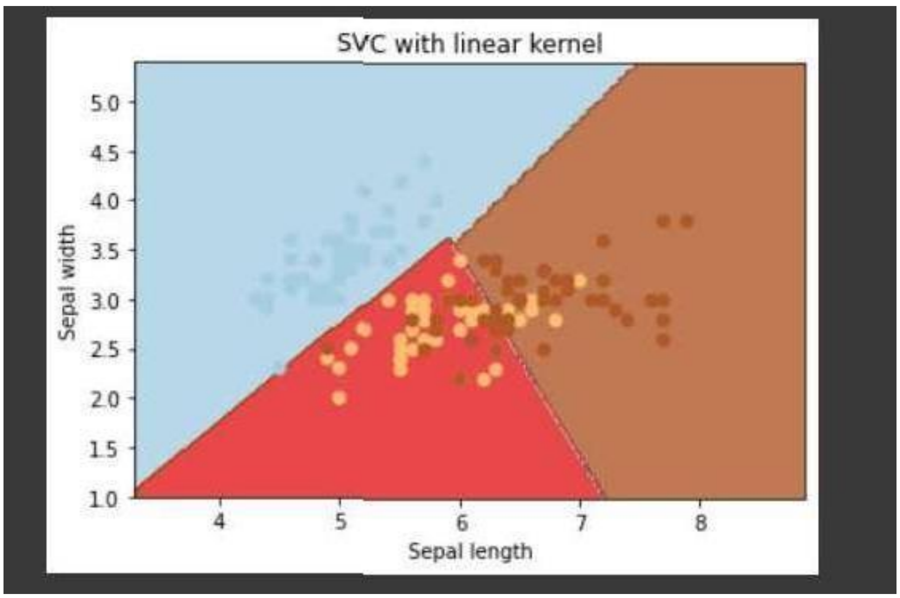
plt.xlim(xx.min(), xx.max())

plt.title('SVC with linear kernel')#

Output the Plot

plt.show()

Output:



Result:

Thus, the python program to implement Support Vector Classification forlinear kernelshas been successfully implemented and executed.

Ex.No. :12	**Demonstrate how will you measure the performance of a machine learning model**
Date:	

Aim:

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

Algorithm:

Step 1: Load the dataset:

- Read the dataset into a pandas dataframe or numpy array.

Step 2: Prepare the data:

- Prepare the data by converting categorical variables to numerical variables, normalizing the data, and splitting the datainto training, validation, and test sets.

Step 3: Train the model:

- Train the machine learning model on the training data.

Step 4: Make predictions:

- Make predictions on the test data using the trained model.

Step 5: Compute performance metrics:

- Compute performance metrics such as accuracy, precision,recall, F1 score, AUC-ROC, confusion matrix, and others, depending on the type of problem and the type of model.

Step 6: Visualize the results:

- Visualize the results by plotting the performance metrics overtime or by creating a confusion matrix.

Step 7: Compare the results:

- Compare the results with the results from other models or withthe baseline.

Step 8: Store the results:

- Store the results for future reference and comparison.

diabetes.csv

num_preg	glucose_conc	diastolic_bp	thickness	insulin	bmi	diab_pred	age	diabetes
6	148	72	35	0	33.6	0.627	50	1
1	85	66	29	0	26.6	0.351	31	0
8	183	64	0	0	23.3	0.672	32	1
1	89	66	23	94	28.1	0.167	21	0
0	137	40	35	168	43.1	2.288	33	1
5	116	74	0	0	25.6	0.201	30	0
3	78	50	32	88	31	0.248	26	1
10	115	0	0	0	35.3	0.134	29	0
2	197	70	45	543	30.5	0.158	53	1
8	125	96	0	0	0	0.232	54	1
4	110	92	0	0	37.6	0.191	30	0
10	168	74	0	0	38	0.537	34	1
10	139	80	0	0	27.1	1.441	57	0
1	189	60	23	846	30.1	0.398	59	1
5	166	72	19	175	25.8	0.587	51	1
7	100	0	0	0	30	0.484	32	1
0	118	84	47	230	45.8	0.551	31	1
7	107	74	0	0	29.6	0.254	31	1
1	103	30	38	83	43.3	0.183	33	0
1	115	70	30	96	34.6	0.529	32	1
3	126	88	41	235	39.3	0.704	27	0
8	99	84	0	0	35.4	0.388	50	0
7	196	90	0	0	39.8	0.451	41	1
9	119	80	35	0	29	0.263	29	1

Program:

```
#import libraries

import pandas as pd

import numpy as np

from sklearn.linear_model import LogisticRegressionfrom

sklearn.model_selection import train_test_split import

warnings

warnings.filterwarnings("ignore")

# to create plots - bar, histogram, boxplot etcimport

seaborn as sns

import matplotlib.pyplot as plt

#calculate accuracy measure and confusion matrixfrom

sklearn import metrics

#Load CSV file

Data= pd. read_csv ("diabetes.csv")

from sklearn.model_selection import train_test_splitX=

Data.drop ("Outcome" ,axis=1)

y= Data[ [ "Outcome"]]

X_train, X_test, y_train, y_test= train_test_split( X,y,test_size=0.30,random_state=7)

#fit model on 30% data model

=LogisticRegression ()model.fit

(X_train, y_train)


y_predict=model.predict (X_test) model_score=

model.score (X_test, y_test)print (model_score)

print (metrics.confusion_matrix (y_test, y_predict))
```

Output:

```
0.7489177489177489
[[127  20]
 [ 38  46]]
```

Result:

Thus, the python program to implement Support Vector Classification forlinear kernels has been successfully implemented and exected.