EX.NO:1A	IMPLEMENTATION OF LINEAR REGRESSION USING GRADIENT DESCENT
DATE:	

AIM:

To implement the linear regression using gradient descent

PROCEDURE:

Step1: Import the necessary libraries

Step2: Define the gradient function using mx+c straight line formula.

Step3: Create and initialize the data points.

Step4: Call the Gradient descent function.

Step5: Interpret the result.

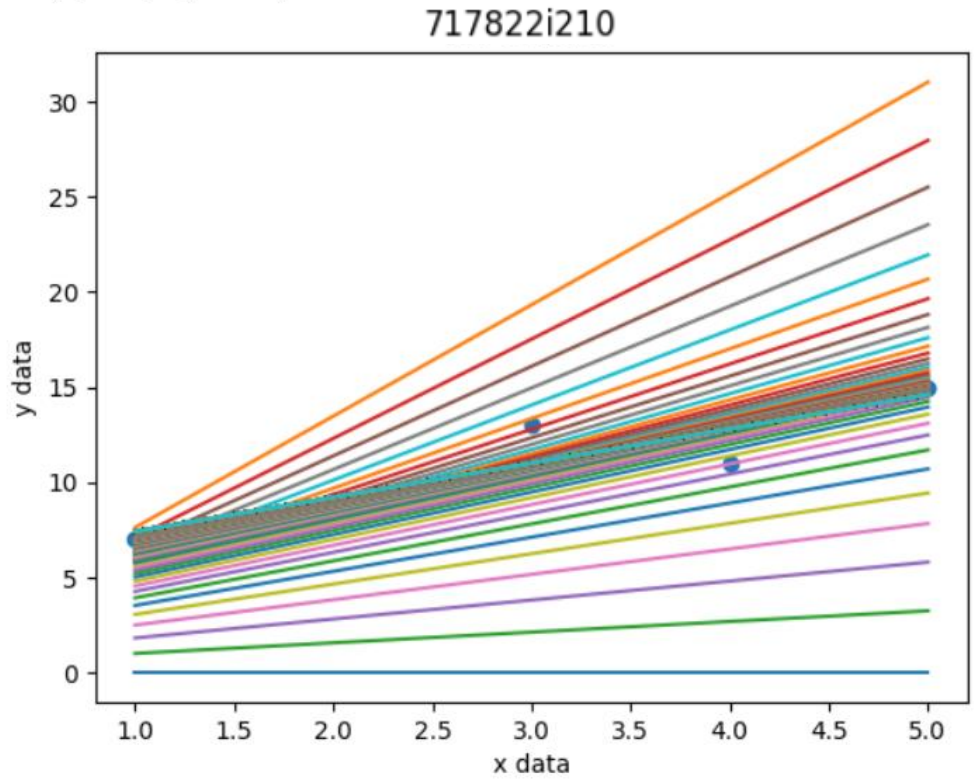
Step6: Visualize the result.

CODE:717822I210

```
import numpy as np
def gradient_descent(x,y):
    m_curr=b_curr=0
    iteration=1000
   learning_rate=0.08
   n=len(x)
   y_p=[]
   for i in range(iteration):
      y_predicted = m_curr * x +b_curr
      y_p.append(y_predicted)
      cost=(1/n)*sum(val**2 for val in (y-y_predicted))
      md=-(2/n)*sum(x*(y-y_predicted))
      bd=-(2/n)*sum(y-y_predicted)
      m_curr=m_curr-learning_rate*md
      b_curr=b_curr-learning_rate*bd
      print('m{},b{},cost{},iteration{}'.format(m_curr,b_curr,cost,i))
   return y_p
x=np.array([1,2,3,4,5])
y=np.array([7,9,13,11,15])
y_p=gradient_descent(x,y)

#Visualization
import matplotlib.pyplot as plt
plt.scatter(x,y)
for i in y_p:
  plt.plot(x,i)
plt.title("717822i210")
plt.xlabel("x data")
plt.ylabel("y data")
```

OUTPUT:

```
Text(0, 0.5, 'y data')
```



PREPARATION	**30**	
LAB PERFORMANCE	**30**	
REPORT	**40**	
TOTAL	**100**	
INITIAL OF THE FACULTY		

RESULT:

The implementation of Linear Regression using gradient descent is completed successfully.

Ex.No: 3 A	SUPPORT VECTOR REGRESSION
Date:	

Aim:

To implement the support vector machines for regression.

Procedure:

Step1: Import the necessary libraries

Step2: Import the dataset and preprocess it.

Step3: Split the data set into training and testing.

Step4: Find the correlation among the data.

Step5: Create and Train the model.

Step6: Interpret the result.

Step7: Visualize the result.

Code:

```
#Importing necessary libraries and dataset
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap
from sklearn import svm
data = pd.read_csv("/content/Student_Performance.csv")

#This dataset has no missing values and outlier, hence directly proceeding with Encoding
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
```

```
[30] data.head(5)
```

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	...	radius_
0	842302	1	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	...	
1	842517	1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	...	
2	84300903	1	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	...	
3	84348301	1	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	...	
4	84358402	1	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	...	

5 rows × 32 columns

```
x = data.iloc[:,:-1].values
y = data.iloc[:, -1].values

labelencoder = LabelEncoder()
x[:, 2] = labelencoder.fit_transform(x[:, 2])

from sklearn.model_selection import train_test_split
xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size = 0.2, random_state = 1)

#Support Vector Machines
from sklearn.svm import SVR
```

```
svr=SVR()
svr.fit(xtrain,ytrain)

#Using evalutation metrics
# Importing libraries for evaluation metrics
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
#Predicting the model
y_pred=svr.predict(xtest)

mse = mean_squared_error(ytest, y_pred)
mae = mean_absolute_error(ytest, y_pred)
r2 = r2_score(ytest, y_pred)
print("Mean Squared Error:", mse)
print("Mean Absolute Error:", mae)
print("R-squared Score:", r2)
```

Output:

```
Mean Squared Error: 5.382488776042803
Mean Absolute Error: 1.8505116449243701
R-squared Score: 0.9855371110854331
```

PREPARATION	**30**	
LAB PERFORMANCE	**30**	
REPORT	**40**	
TOTAL	**100**	
INITIAL OF THE FACULTY		

Result:

The implementation of Support Vector Machines for regression is completed successfully.

Ex.No: 3 B	SUPPORT VECTOR CLASSIFICATION
Date:	

Aim:

To implement the support vector machines for classification.

Procedure:

Step1: Import the necessary libraries

Step2: Import the dataset and preprocess it.

Step3: Split the data set into training and testing.

Step4: Find the correlation among the data.

Step5: Create and Train the model.

Step6: Interpret the result.

Step7: Visualize the result.

Code:

```
#Importing necessary libraries and dataset
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap
from sklearn import svm
data = pd.read_csv("/content/cars_dataset.csv")

#This dataset has no missing values and outlier, hence directly proceeding with Encoding
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
```

```
[30] data.head(5)
```

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	...	radius_
0	842302	1	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	...	
1	842517	1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	...	
2	84300903	1	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	...	
3	84348301	1	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	...	
4	84358402	1	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	...	

5 rows × 32 columns

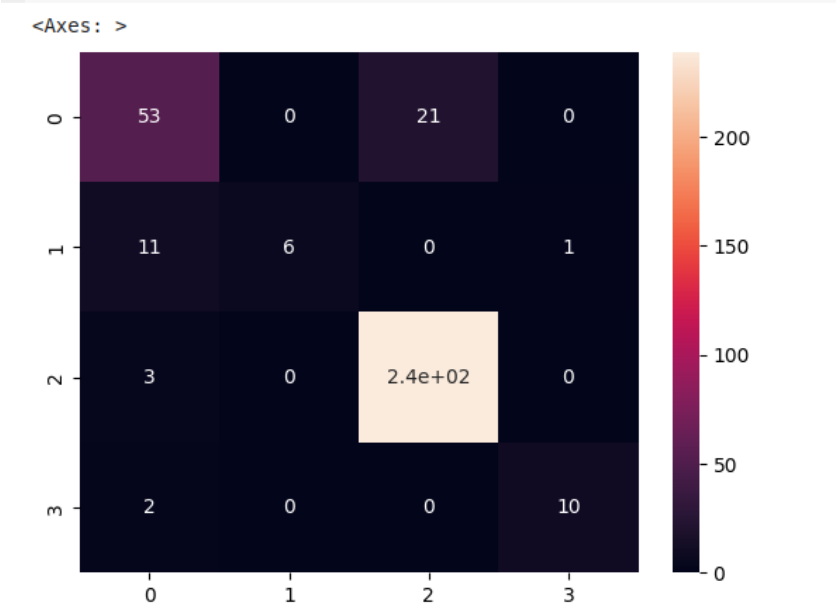
```
x = data.iloc[:,:-1].values
y = data.iloc[:, -1].values

labelencoder = LabelEncoder()
x[:, 0] = labelencoder.fit_transform(x[:, 0])
x[:, 1] = labelencoder.fit_transform(x[:, 1])
x[:, 2] = labelencoder.fit_transform(x[:, 2])
x[:, 3] = labelencoder.fit_transform(x[:, 3])
x[:, 4] = labelencoder.fit_transform(x[:, 4])
x[:, 5] = labelencoder.fit_transform(x[:, 5])
```

```
from sklearn.model_selection import train_test_split
xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size = 0.2, random_state = 1)

#Support Vector Machines
from sklearn.svm import SVC
svc=SVC()
svc.fit(xtrain,ytrain)

#Using evalutation metrics
# Importing libraries for evaluation metrics
from sklearn.metrics import accuracy_score, confusion_matrix, precision_score, recall_score, f1_score,
roc_auc_score, log_loss
#Predicting the model
y_pred=svc.predict(xtest)

#Checking Accuracy
accuracy = accuracy_score(ytest, y_pred)
conf_matrix = confusion_matrix(ytest, y_pred)
#Printing Accuracy
print("Accuracy : ",accuracy)
print("Confusion Matrix : ",conf_matrix)
```

```
Accuracy :  0.8901734104046243
Confusion Matrix :  [[ 53   0  21   0]
 [ 11   6   0   1]
 [  3   0 239   0]
 [  2   0   0  10]]
```

```
import seaborn as sns
sns.heatmap(conf_matrix,annot = True)
```

Output:



PREPARATION	**30**	
LAB PERFORMANCE	**30**	
REPORT	**40**	
TOTAL	**100**	
INITIAL OF THE FACULTY		

Result:

The implementation of Support Vector Machines for Classification is completed successfully.

Ex.No: 4 A	DECISION TREE REGRESSION
Date:	

Aim:

To implement the decision tree for regression.

Procedure:

Step1: Import the necessary libraries

Step2: Import the dataset and preprocess it.

Step3: Split the data set into training and testing.

Step4: Find the correlation among the data.

Step5: Create and Train the model.

Step6: Interpret the result.

Step7: Visualize the result.

Code:

```
#Importing necessary libraries and dataset
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier,DecisionTreeRegressor
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
data = pd.read_csv("/content/Real estate.csv")

#Splitting into X and Y
x = data.iloc[:,:-1]
y = data.iloc[:,-1]
```

```
[30] data.head(5)
```

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	...	radius_
0	842302	1	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	...	
1	842517	1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	...	
2	84300903	1	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	...	
3	84348301	1	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	...	
4	84358402	1	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	...	

5 rows × 32 columns

```
#Splitting into training and testing dataset
xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size=0.2, random_state=42)

#Implementing Decision Tree
d = DecisionTreeRegressor()
d.fit(xtrain, ytrain)
y_pred = d.predict(xtest)

#Evaluation Metrics
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
```

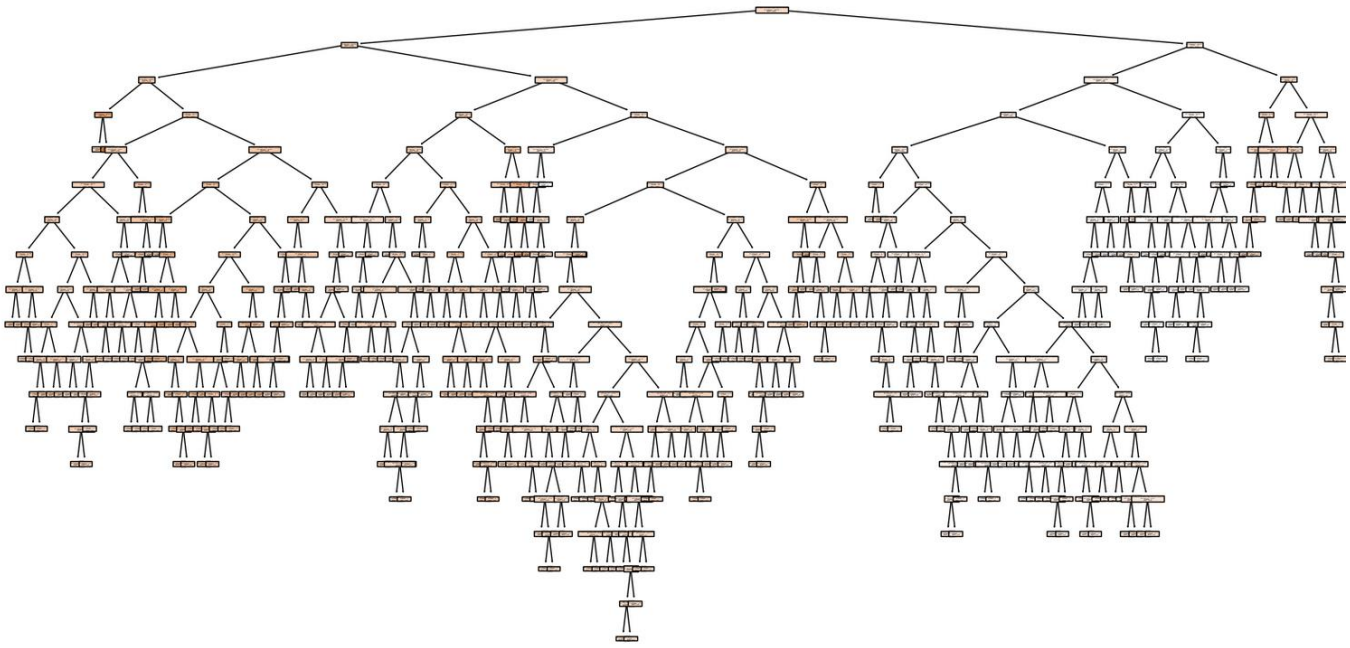
```
mse = mean_squared_error(ytest, y_pred)
mae = mean_absolute_error(ytest, y_pred)
r2 = r2_score(ytest, y_pred)

# Print the regression metrics
print("Mean Squared Error (MSE):", mse)
print("Mean Absolute Error (MAE):", mae)
print("R-squared Score (R2):", r2)
```

Output:

```
Mean Squared Error (MSE): 62.14746987951808
Mean Absolute Error (MAE): 5.84578313253012
R-squared Score (R2): 0.6295447989787772
```

```
#Plotting the tree
from sklearn.tree import plot_tree
plt.figure(figsize=(20, 10))
plot_tree(d, filled=True, feature_names = data.columns)
plt.show()
```

Output:



PREPARATION	**30**	
LAB PERFORMANCE	**30**	
REPORT	**40**	
TOTAL	**100**	
INITIAL OF THE FACULTY		

Result:

The implementation of decision tree for regression is completed successfully.

Ex.No: 4 B	DECISION TREE CLASSIFIER
Date:	

Aim:

To implement the decision tree for classifier.

Procedure:

Step1: Import the necessary libraries

Step2: Import the dataset and preprocess it.

Step3: Split the data set into training and testing.

Step4: Find the correlation among the data.

Step5: Create and Train the model.

Step6: Interpret the result.

Step7: Visualize the result.

Code:

```
#Importing necessary libraries and dataset
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier,DecisionTreeRegressor
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
data = pd.read_csv("/content/heart_statlog_cleveland_hungary_final.csv")

#Splitting into X and Y
x = data.iloc[:,:-1]
y = data.iloc[:,-1]
```

```
data.head(2)
```

	age	sex	chest pain type	resting bp s	cholesterol	fasting blood sugar	resting ecg	max heart rate	exercise angina	oldpeak	ST slope	target
0	40	1	2	140	289	0	0	172	0	0.0	1	0
1	49	0	3	160	180	0	0	156	0	1.0	2	1

```
#Splitting into training and testing dataset
xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size=0.2, random_state=42)

#Implementing Decision Tree
d = DecisionTreeClassifier()
d.fit(xtrain, ytrain)
y_pred = d.predict(xtest)

#Evaluation Metrics
accuracy = accuracy_score(ytest, y_pred)
```

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

conf_matrix = confusion_matrix(ytest, y_pred)
print("Confusion Matrix:\n", conf_matrix)

report = classification_report(ytest, y_pred)
print("Classification Report:\n", report)
```

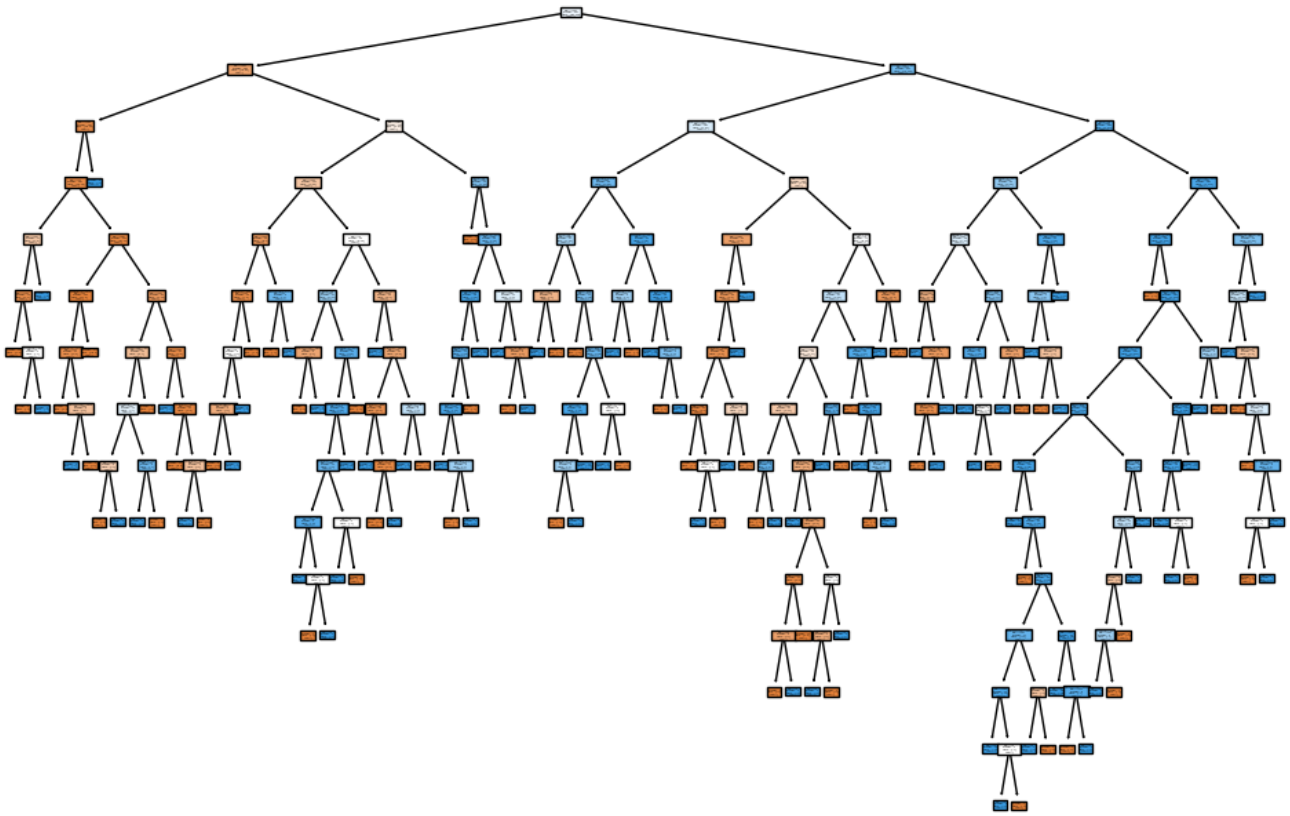
Output:

```
Accuracy: 0.8991596638655462
Confusion Matrix:
 [[101   6]
 [ 18 113]]
Classification Report:
              precision    recall  f1-score   support

           0       0.85      0.94      0.89       107
           1       0.95      0.86      0.90       131

    accuracy                           0.90       238
   macro avg       0.90      0.90      0.90       238
weighted avg       0.90      0.90      0.90       238
```

Plotting decision tree:

```
from sklearn.tree import plot_tree
plt.figure(figsize=(12, 8))
plot_tree(d, filled=True, feature_names=x.columns, class_names=str(data['target'].unique()))
plt.show()
```

Output:



PREPARATION	**30**	
LAB PERFORMANCE	**30**	
REPORT	**40**	
TOTAL	**100**	
INITIAL OF THE FACULTY		

Result:

The implementation of decision tree for regression is completed successfully.

Ex.No: 5 A	RANDOM FOREST REGRESSION
Date:	

Aim:

To implement the random forest algorithm for regression.

Procedure:

Step1: Import the necessary libraries

Step2: Import the dataset and preprocess it.

Step3: Split the data set into training and testing.

Step4: Find the correlation among the data.

Step5: Create and Train the model.

Step6: Interpret the result.

Step7: Visualize the result.

Code:

```
#Importing necessary libraries and dataset
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
data = pd.read_csv("/content/bike data.csv")

#Splitting into X and Y
x = data.iloc[:,:-1]
y = data.iloc[:,-1]

#Splitting into training and testing dataset
xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size=0.2, random_state=42)

#Implementing Random Forest
r = RandomForestRegressor(n_estimators=13, random_state=42)
r.fit(xtrain, ytrain)

#Evaluation Metrics
mse = mean_squared_error(ytest, y_pred)
mae = mean_absolute_error(ytest, y_pred)
r2 = r2_score(ytest, y_pred)

print("Mean Squared Error:", mse)
print("Mean Absolute Error:", mae)
print("R-squared Score:", r2)
```

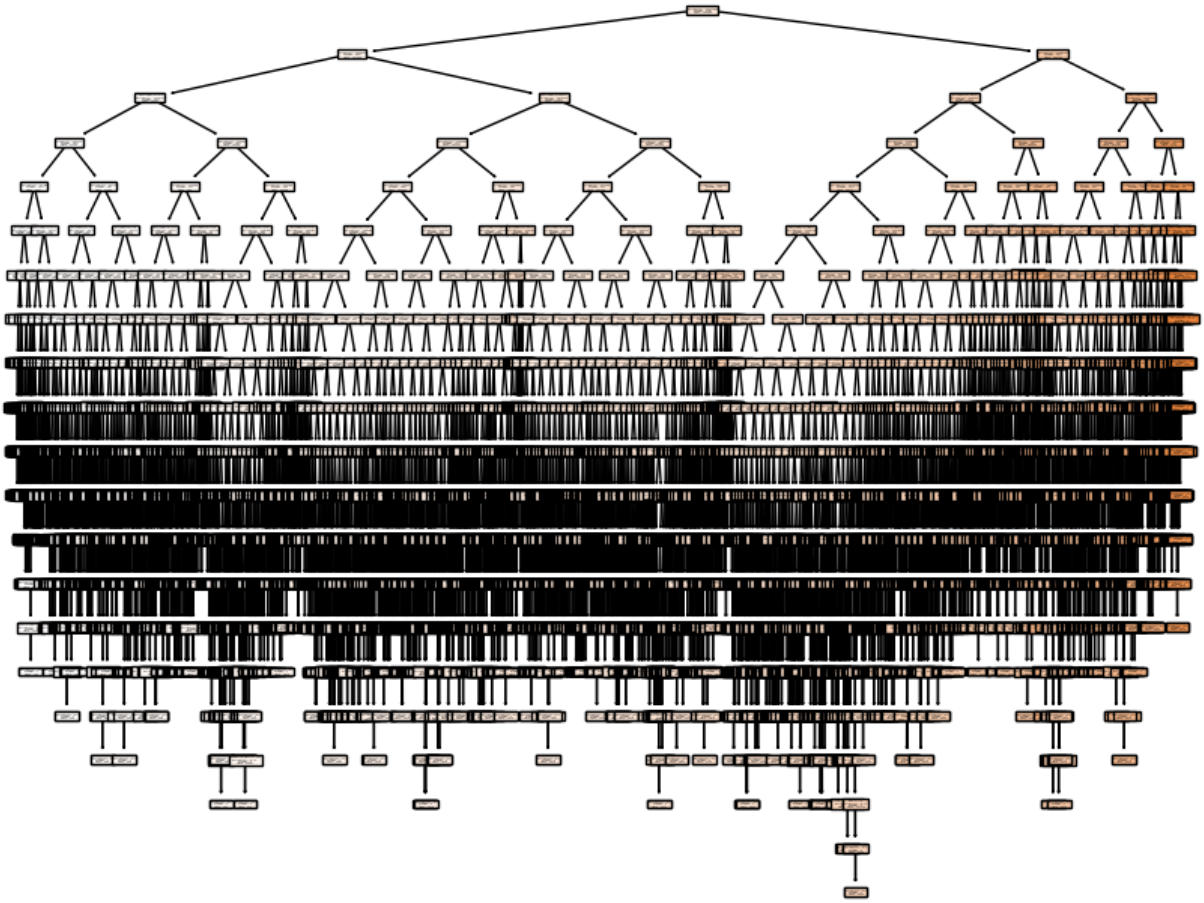
Output:

```
Mean Squared Error: 8.55324252184038
Mean Absolute Error: 1.1493980702841464
R-squared Score: 0.999729886896748
```

#Plotting the tree
from sklearn.tree import plot_tree

```
plt.figure(figsize=(10, 8))
plot_tree(r.estimators_[0], feature_names=data.columns, filled=True)
plt.show()
```

Output:



PREPARATION	**30**	
LAB PERFORMANCE	**30**	
REPORT	**40**	
TOTAL	**100**	
INITIAL OF THE FACULTY		

Result:

The implementation of random forest for regression is completed successfully.

Ex.No: 5 B	RANDOM FOREST CLASSIFICATION
Date:	

Aim:

To implement the random forest algorithm for classification.

Procedure:

Step1: Import the necessary libraries

Step2: Import the dataset and preprocess it.

Step3: Split the data set into training and testing.

Step4: Find the correlation among the data.

Step5: Create and Train the model.

Step6: Interpret the result.

Step7: Visualize the result.

Code:

```
#Importing necessary libraries and dataset
from sklearn.ensemble import RandomForestClassifier
from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split
from sklearn.tree import plot_tree
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
data = pd.read_csv("/content/Pulsar_cleaned.csv")

#Splitting into X and Y
x = data.iloc[:,:-1]
y = data.iloc[:,-1]

#Splitting into training and testing dataset
xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size=0.2, random_state=42)

#Implementing Random Forest
r = RandomForestRegressor(n_estimators=13, random_state=42)
r.fit(xtrain, ytrain)
y_pred = r.predict(xtest)
#Evaluation Metrics
from sklearn.metrics import accuracy_score, confusion_matrix, precision_score, recall_score, f1_score,
roc_auc_score, log_loss
accuracy = accuracy_score(ytest, y_pred)
conf_matrix = confusion_matrix(ytest, y_pred)
precision = precision_score(ytest, y_pred)
recall = recall_score(ytest, y_pred)
f1 = f1_score(ytest, y_pred)

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

```
print("Confusion Matrix:\n", conf_matrix)
print("Precision:", precision)
print("Recall:", recall)
print("F1 Score:", f1)
```

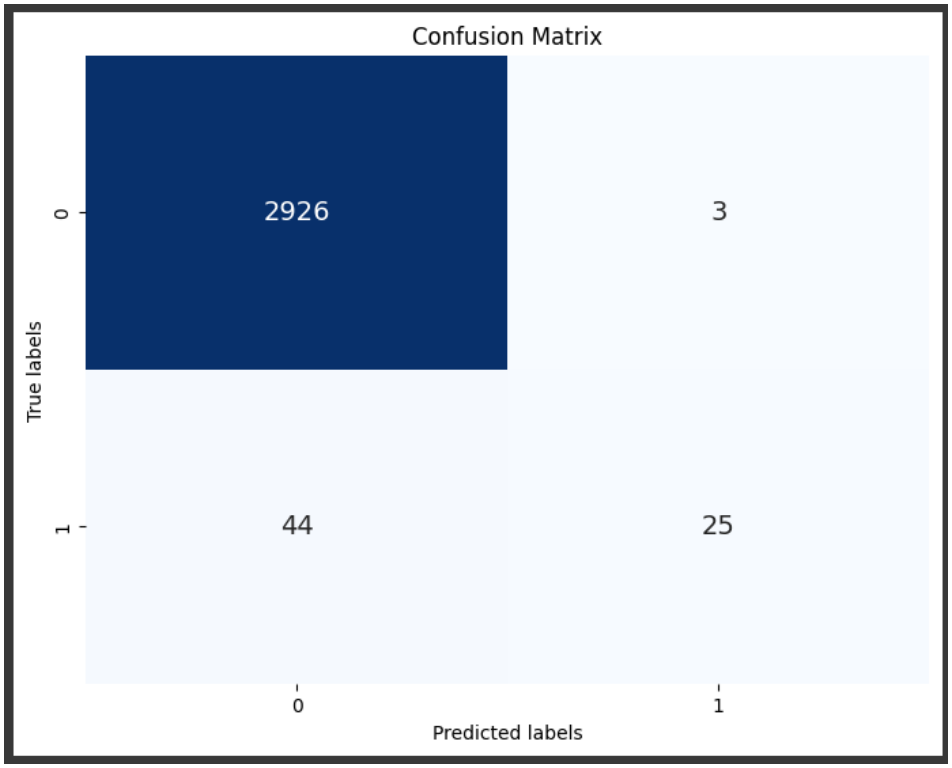
Output:

```
Accuracy: 0.9843228819212808
Confusion Matrix:
 [[2926    3]
 [  44   25]]
Precision: 0.8928571428571429
Recall: 0.36231884057971014
F1 Score: 0.5154639175257733
```

#Plotting the confusion matrix

```
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, cmap='Blues', fmt='d', cbar=False, annot_kws={"size": 14})
plt.title('Confusion Matrix')
plt.xlabel('Predicted labels')
plt.ylabel('True labels')
plt.show()
```

Output:



PREPARATION	**30**	
LAB PERFORMANCE	**30**	
REPORT	**40**	
TOTAL	**100**	
INITIAL OF THE FACULTY		

Result:

The implementation of random forest for classification is completed successfully.

EX.NO:7A	K MEANS CLUSTERING
DATE:	

AIM:

To implement the K Means Clustering Algorithm.

PROCEDURE:

Step 1: Start the program by importing required libraries.

Step 2: Read and preprocess the dataset.

Step 3: Get the X(independent) and Y(dependent) features.

Step 4: Split the dataset into training and testing sets.

Step 5: Find the correlation among the data.

Step 6: Interpret the result.

Step 7: Visualize the result.

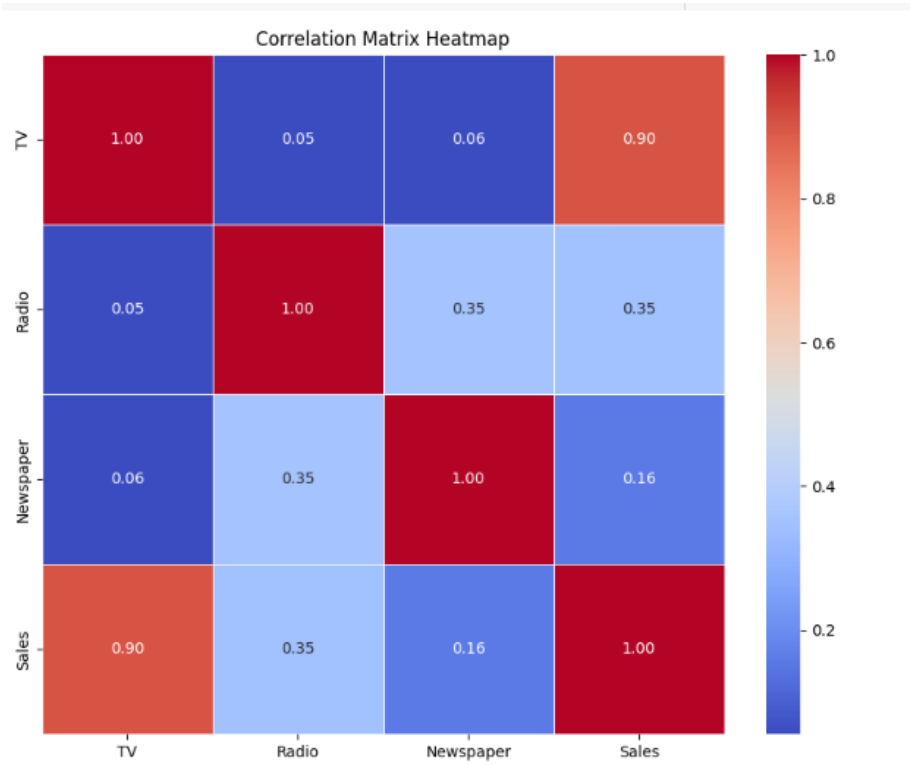
PROGRAM:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
data = pd.read_csv('/content/drive/MyDrive/Walmart_sales.csv')
data.head()
```

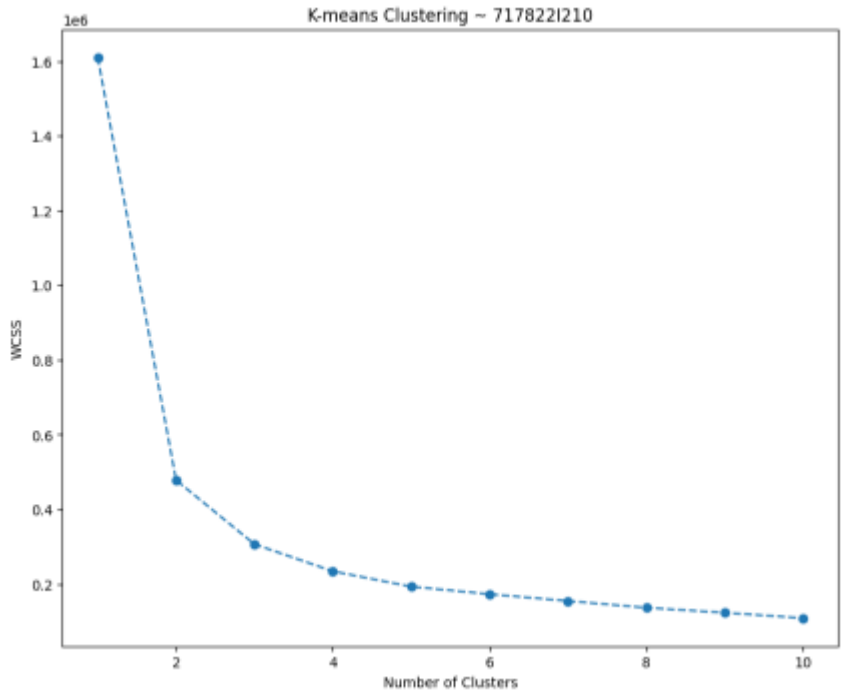
	TV	Radio	Newspaper	Sales
0	230.1	37.8	69.2	22.1
1	44.5	39.3	45.1	10.4
2	17.2	45.9	69.3	12.0
3	151.5	41.3	58.5	16.5
4	180.8	10.8	58.4	17.9

```
from sklearn.impute import SimpleImputer
se=SimpleImputer(missing_values=np.NaN,strategy='mean')
```

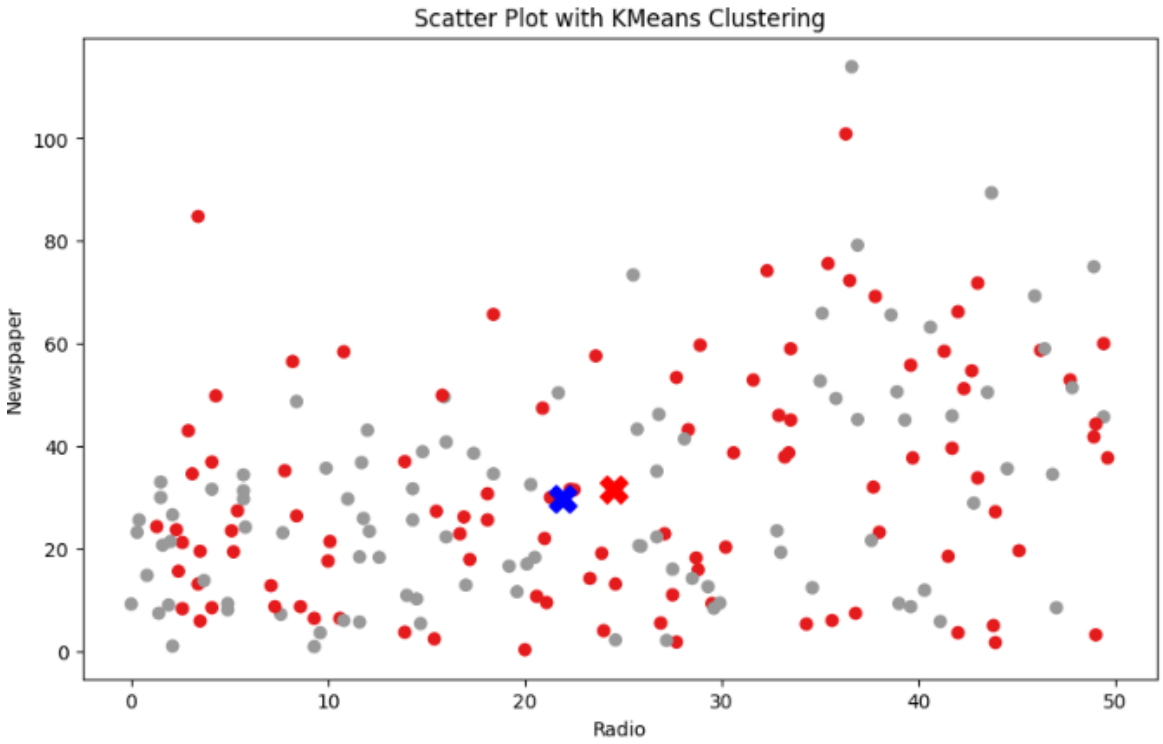
```
import seaborn as sns
correlation_matrix = data.corr()
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f", linewidths=.5)
plt.title('Correlation Matrix Heatmap')
plt.show()
```




```
from sklearn.cluster import KMeans
wcss = []
for i in range(1,11):
    kmeans = KMeans(n_clusters = i, init = 'k-means++', random_state = 4)
    kmeans.fit(data)
    wcss.append(kmeans.inertia_)
plt.figure(figsize = (10,8))
plt.plot(range(1, 11), wcss, marker = 'o', linestyle = '--')
plt.xlabel('Number of Clusters')
plt.ylabel('WCSS')
plt.title('K-means Clustering ~ 717822I210')
plt.show()
```




```
kmeans = KMeans(n_clusters = 2, init = 'k-means++', random_state = 4)
pred = kmeans.fit_predict(data)

plt.figure(figsize=(10, 6))
plt.scatter(data['Radio'], data['Newspaper'], c=pred, cmap='Set1')
plt.scatter(kmeans.cluster_centers_[:,1], kmeans.cluster_centers_[:,2], c=['red','blue'], s=200,
marker='X', label='Centroids')
plt.title('Scatter Plot with KMeans Clustering')
plt.xlabel('Radio')
plt.ylabel('Newspaper')
plt.show()
```



PREPARATION	**30**	
LAB PERFORMANCE	**30**	
REPORT	**40**	
TOTAL	**100**	
INITIAL OF THE FACULTY		

RESULT:

The implementation of K Means clustering is completed successfully.

EX.NO:7B	HIERARCHICAL CLUSTERING
DATE:	

AIM:

To implement the hierarchical clustering Algorithm

PROCEDURE:

Step1: Import the necessary libraries

Step2: Import the dataset and preprocess it.

Step3: Split the data set into training and testing.

Step4: Find the correlation among the data.

Step5: Create and Train the model.

Step6: Interpret the result.

Step7: Visualize the result.

PROGRAM:

```
import numpy as np
import pandas as pd
data=pd.read_csv("/Mall_Customers (1) - Mall_Customers (1).csv")
data
```

	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
...	...	...	...	...	...
195	196	Female	35	120	79
196	197	Female	45	126	28
197	198	Male	32	126	74
198	199	Male	32	137	18
199	200	Male	30	137	83

```
data.isnull().sum()
```

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

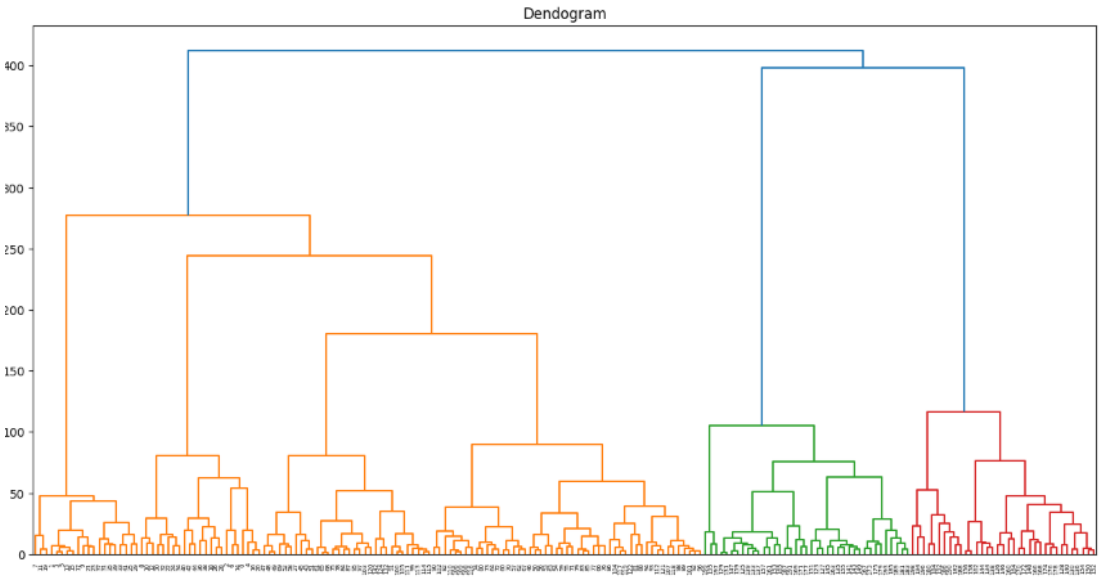
```
data=data.iloc[:,1:5]
data.head()
```

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

```
from sklearn.preprocessing import LabelEncoder
l=LabelEncoder()
data['Gender']=l.fit_transform(data['Gender'])
data.head()
```

	Gender	Age	Annual Income (k$)	Spending Score (1-100)
0	1	19	15	39
1	1	21	15	81
2	0	20	16	6
3	0	23	16	77
4	0	31	17	40

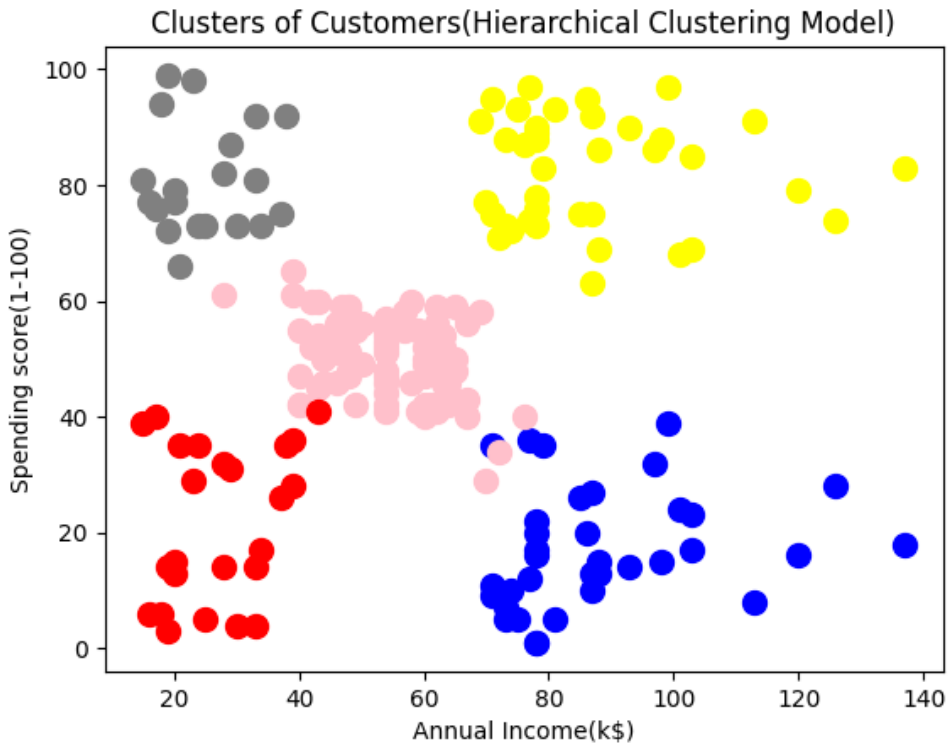
```
import matplotlib.pyplot as plt
import scipy.cluster.hierarchy as sch
plt.figure(1,figsize=(16,8))
dendrogram=sch.dendrogram(sch.linkage(data,method="ward"))
plt.title('Dendogram')
plt.show()
```




```
from sklearn.cluster import AgglomerativeClustering
h=AgglomerativeClustering(n_clusters=5,affinity='euclidean',linkage='average')
y_h=h.fit_predict(data)
y_h
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_agglomerative.py:9
  warnings.warn(
array([4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3,
       4, 3, 4, 3, 4, 1, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 1,
       4, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 4, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 2, 0, 2, 0, 2, 0, 2,
       1, 2, 0, 2, 0, 2, 0, 2, 0, 2, 1, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2,
       0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2,
       0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2,
       0, 2])
```

```
x=data.iloc[:,[2,3]].values
plt.scatter(x[y_hc==0,0],x[y_hc==0,1],s=100 ,c='red',label='cluster 1')
plt.scatter(x[y_hc==1,0],x[y_hc==1,1],s=100 ,c='blue',label='cluster 2')
plt.scatter(x[y_hc==2,0],x[y_hc==2,1],s=100 ,c='green',label='cluster 3')
plt.scatter(x[y_hc==3,0],x[y_hc==3,1],s=100 ,c='purple',label='cluster 4')
plt.scatter(x[y_hc==4,0],x[y_hc==4,1],s=100 ,c='orange',label='cluster 5')
plt.title('cluster of customers (Hierarchial clustering Model)')
plt.xlabel('Annual Income(k$)')
plt.ylabel('spending score(1-100)')
plt.show()
```



y_h==0

```
array([False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
        True, False,  True, False,  True, False, False, False,  True,
       False,  True, False,  True, False,  True, False, False, False,
        True, False,  True, False,  True, False,  True, False,  True,
       False,  True, False,  True, False,  True, False,  True, False,
        True, False,  True, False,  True, False,  True, False,  True,
       False,  True, False,  True, False,  True, False,  True, False,
        True, False,  True, False,  True, False,  True, False,  True,
       False,  True, False,  True, False,  True, False,  True, False,
        True, False])
```

PREPARATION	**30**	
LAB PERFORMANCE	**30**	
REPORT	**40**	
TOTAL	**100**	
INITIAL OF THE FACULTY		

RESULT:

The implementation of hierarchical clustering is completed successfully.

Ex.No: 8	ASSOCIATION RULE MINING
Date:	

AIM:

To implement the Apriori Algorithm

PROCEDURE:

Step1: Import the necessary libraries

Step2: Import the dataset and preprocess it.

Step3: Split the data set into training and testing.

Step4: Find the correlation among the data.

Step5: Create and Train the model.

Step6: Interpret the result.

Step7: Visualize the result.

PROGRAM:

```
#Install Apriori
!pip install apyori
from apyori import apriori

import numpy as np
import pandas as pd
data=pd.read_csv("/content/Market_Basket_Optimisation - Market_Basket_Optimisation.csv")
data
```

	shrimp	almonds	avocado	vegetables mix	green grapes	whole weat flour	yams	cottage cheese	energy drink	tomato juice	low fat yogurt	green tea	honey	salad	mineral water	salmon	antioxydant juice	frozen smoothie
0	burgers	meatballs	eggs	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	chutney	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	turkey	avocado	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	mineral water	milk	energy bar	whole wheat rice	green tea	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	low fat yogurt	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
7495	butter	light mayo	fresh bread	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
7496	burgers	frozen vegetables	eggs	french fries	magazines	green tea	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

```
data.fillna(0,inplace=True)
```

```
data.head()
```

	shrimp	almonds	avocado	vegetables mix	green grapes	whole weat flour	yams	cottage cheese	energy drink	tomato juice	low fat yogurt	green tea	honey	salad	mineral water	salmon	antioxydant juice	frozen smoothie
0	burgers	meatballs	eggs	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	chutney	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	turkey	avocado	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	mineral water	milk	energy bar	whole wheat rice	green tea	0	0	0	0	0	0	0	0	0	0	0	0	0
4	low fat yogurt	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

```
transactions=[]
for i in range(0,len(data)):
  transactions.append([str(data.values[i,j]) for j in range(0,20) if str(data.values[i,j])!='0'])
```

```
transactions[0]
```

```
['burgers', 'meatballs', 'eggs', '0.0']
```

```
from apyori import apriori
r=apriori(transactions,min_support=0.003,min_confidence=0.2,min_lift=3,min_length=2)
R=list(r)
data_results=pd.DataFrame(R)
data_results.head()
```

	items	support	ordered_statistics
0	(cottage cheese, brownies)	0.003467	[((brownies), (cottage cheese), 0.102766798418...
1	(chicken, light cream)	0.004533	[((chicken), (light cream), 0.0755555555555555...
2	(escalope, mushroom cream sauce)	0.005733	[((escalope), (mushroom cream sauce), 0.072268...
3	(escalope, pasta)	0.005867	[((escalope), (pasta), 0.07394957983193277, 4....
4	(tomato juice, fresh bread)	0.004267	[((fresh bread), (tomato juice), 0.09907120743...

```
support=data_results.support
first_values=[]
second_values=[]
third_values=[]
fourth_values=[]
for i in range(data_results.shape[0]):
  single_list=data_results['ordered_statistics'][i][0]
  first_values.append(list(single_list[0]))
  second_values.append(list(single_list[1]))
  third_values.append(single_list[2])
  fourth_values.append(single_list[3])
```

```
l=pd.DataFrame(first_values)
r=pd.DataFrame(second_values)
confidance=pd.DataFrame(third_values,columns=['confidance'])
```

```
lift=pd.DataFrame(fourth_values,columns=['lift'])
```

	0	1	0	1	2	3	support	Confidance	lift
0	brownies	None	cottage cheese	None	None	None	0.003467	0.102767	3.238450
1	chicken	None	light cream	None	None	None	0.004533	0.075556	4.843305
2	escalope	None	mushroom cream sauce	None	None	None	0.005733	0.072269	3.790327
3	escalope	None	pasta	None	None	None	0.005867	0.073950	4.700185
4	fresh bread	None	tomato juice	None	None	None	0.004267	0.099071	3.273278
...	...	...	...	...	...	...	...	...	...
183	ground beef	pancakes	mineral water	0.0	spaghetti	None	0.003067	0.211009	3.532520
184	ground beef	None	mineral water	0.0	tomatoes	spaghetti	0.003067	0.031208	3.343671
185	olive oil	None	mineral water	milk	0.0	spaghetti	0.003333	0.050710	3.223089
186	mineral water	milk	shrimp	0.0	spaghetti	None	0.003067	0.063889	3.013627
187	tomatoes	None	mineral water	milk	0.0	spaghetti	0.003333	0.048733	3.097433

```
data_final.fillna(value=' ',inplace=True)
data_final.columns=['l','l2','l3','r','r2','r3','support','confidance','lift']
data_final['l']=data_final['l']+str(", ")+data_final['l2']
data_final['r']=data_final['r']+str(', ')+data_final['l2']+str(', ')+data_final['l3']
data_final
```

	lhs	rhs	support	confidance	lift
0	brownies,	cottage cheese, , ,	0.003467	0.102767	3.238450
1	chicken,	light cream, , ,	0.004533	0.075556	4.843305
2	escalope,	mushroom cream sauce, , ,	0.005733	0.072269	3.790327
3	escalope,	pasta, , ,	0.005867	0.073950	4.700185
4	fresh bread,	tomato juice, , ,	0.004267	0.099071	3.273278
...	...	...	...	...	...
183	ground beef, pancakes	mineral water, 0.0, spaghetti,	0.003067	0.211009	3.532520
184	ground beef,	mineral water, 0.0, tomatoes,spaghetti	0.003067	0.031208	3.343671
185	olive oil,	mineral water, milk, 0.0,spaghetti	0.003333	0.050710	3.223089
186	mineral water, milk	shrimp, 0.0, spaghetti,	0.003067	0.063889	3.013627
187	tomatoes,	mineral water, milk, 0.0,spaghetti	0.003333	0.048733	3.097433

PREPARATION	**30**	
LAB PERFORMANCE	**30**	
REPORT	**40**	
TOTAL	**100**	
INITIAL OF THE FACULTY		

RESULT:

The implementation of association rule mining is completed successfully.

Ex.No: 9A	PRINCIPAL COMPONENT ANALYSIS
Date:	

AIM:

To implement principal component analysis using digits data

PROCEDURE:

Step1: Import the necessary libraries

Step2: Import the dataset and preprocess it.

Step3: Split the data set into training and testing.

Step4: Find the correlation among the data.

Step5: Create and Train the model.

Step6: Interpret the result.

Step7: Visualize the result.

PROGRAM:

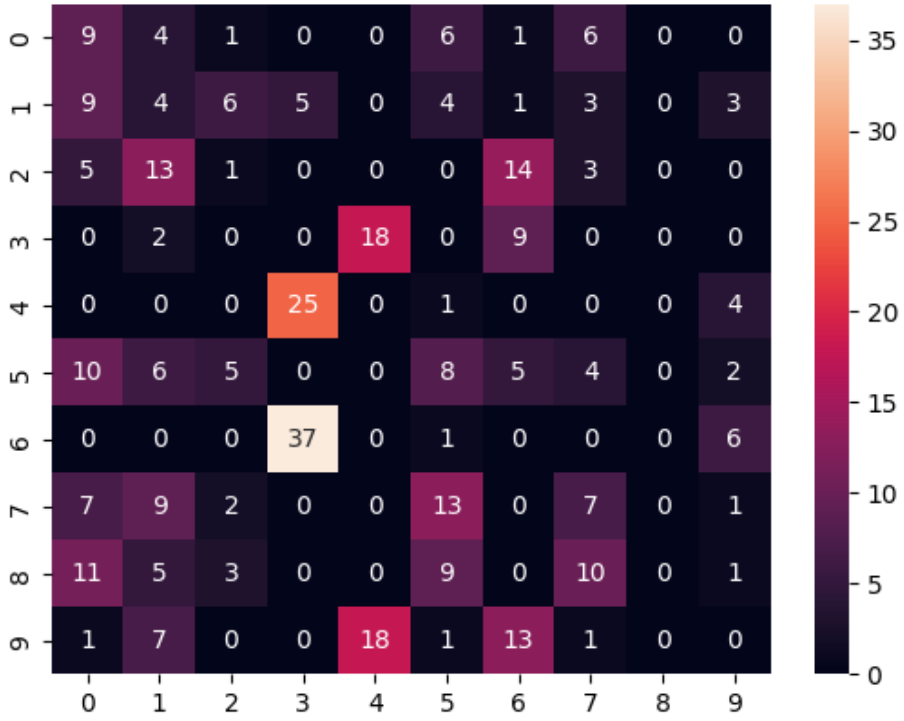
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.datasets import load_digits
digits = load_digits()

x=digits.data
y=digits.target

from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=0)

from sklearn.decomposition import PCA
pca = PCA(n_components = 2)

pcaxt = pca.fit_transform(x_train)
pcaxs = pca.fit_transform(x_test)

from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(pcaxt, y_train)

y_pred=model.predict(pcaxs)

from sklearn.metrics import accuracy_score
accuracy = accuracy_score(y_test, y_pred)
print(accuracy)
```

```
0.9902574808629089
```

```
from sklearn.metrics import confusion_matrix as cm
a = cm(y_test,y_pred)
```

```
import seaborn as sns
sns.heatmap(a,annot = True)
```

```
<Axes: >
```



PREPARATION	**30**	
LAB PERFORMANCE	**30**	
REPORT	**40**	
TOTAL	**100**	
INITIAL OF THE FACULTY		

RESULT:

The implementation of principal component analysis is completed successfully.

Ex.No: 9B	PRINCIPAL COMPONENT ANALYSIS
Date:	

AIM:

To implement principal component analysis using wine data

PROCEDURE:

Step1: Import the necessary libraries

Step2: Import the dataset and preprocess it.

Step3: Split the data set into training and testing.

Step4: Find the correlation among the data.

Step5: Create and Train the model.

Step6: Interpret the result.

Step7: Visualize the result.

PROGRAM:

```
import pandas as pd
import numpy as np
data=pd.read_csv('/content/ wine1.csv')
data.head()
```

	class	Alcohol	Malic acid	Ash	Alcalinityofash	Magnesium	Total phenols	Flavanoids	Nonflavanoid phenols
0	1	14.23	1.71	2.43	15.6	127	2.80	3.06	0.28
1	1	13.20	1.78	2.14	11.2	100	2.65	2.76	0.26
2	1	13.16	2.36	2.67	18.6	101	2.80	3.24	0.30
3	1	14.37	1.95	2.50	16.8	113	3.85	3.49	0.24
4	1	13.24	2.59	2.87	21.0	118	2.80	2.69	0.39

```
data.isna().sum()
```

```
class                         0
Alcohol                       0
Malic acid                    0
Ash                           0
Alcalinityofash               0
Magnesium                     0
Total phenols                 0
Flavanoids                    0
Nonflavanoid phenols          0
Proanthocyanins               0
Color intensity               0
Hue                           0
OD280/OD315 of diluted wines  0
Proline                       0
dtype: int64
```

```
x=data.iloc[:,1:]
y=data.iloc[:,[0]]
```

```
from sklearn.model_selection import train_test_split
xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.2,random_state=1)
```

```
from sklearn.decomposition import PCA
pca=PCA(n_components=6)
#pca=PCA(0.99)
xtrain_pca=pca.fit_transform(xtrain)
xtest_pca=pca.fit_transform(xtest)
```

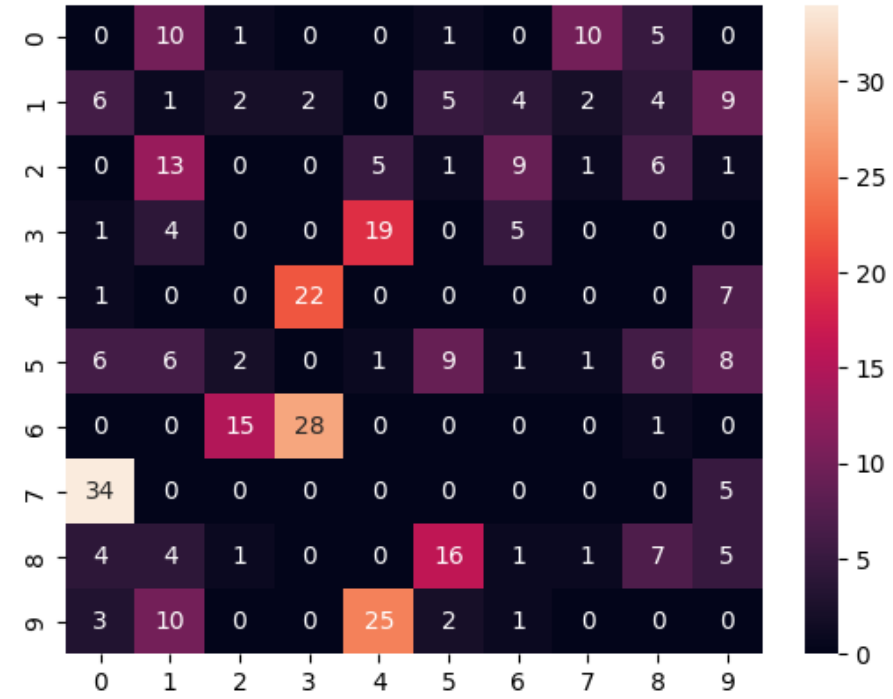
```
from sklearn.linear_model import LogisticRegression
l=LogisticRegression()
l.fit(xtrain_pca,ytrain)
```

```
ypred=l.predict(xtrain_pca)
from sklearn.metrics import accuracy_score
print(accuracy_score(ytrain,ypred))
```

```
0.9929577464788732
```

```
import seaborn as sns
sns.heatmap(a,annot = True)
```

```
<Axes: >
```



PREPARATION	**30**	
LAB PERFORMANCE	**30**	
REPORT	**40**	
TOTAL	**100**	
INITIAL OF THE FACULTY		

RESULT:

The implementation of principal component analysis is completed successfully.

Ex.No: 10A	LINEAR DISCRIMINANT ANALYSIS
Date:	

AIM:

To implement Linear Discriminant analysis using digits data

PROCEDURE:

Step1: Import the necessary libraries

Step2: Import the dataset and preprocess it.

Step3: Split the data set into training and testing.

Step4: Find the correlation among the data.

Step5: Create and Train the model.

Step6: Interpret the result.

Step7: Visualize the result.

PROGRAM:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.datasets import load_digits
digits = load_digits()
```

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=0)
```

```
ldatr = lda.fit_transform(x_train,y_train)
ldate = lda.transform(x_test)
```

```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(ldatr, y_train)
```

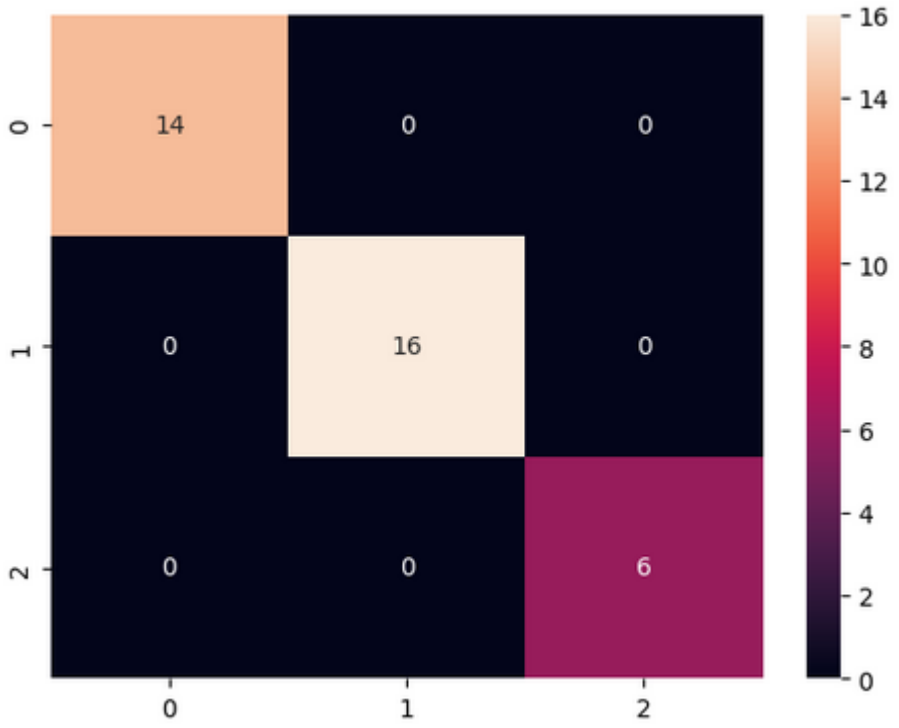
```
y_pred=model.predict(ldate)
```

```
from sklearn.metrics import accuracy_score
accuracy = accuracy_score(y_test, y_pred)
print(accuracy)
```

```
from sklearn.metrics import confusion_matrix as cm
a = cm(y_test,y_pred)
```

```
import seaborn as sns
sns.heatmap(a,annot = True)
```

```
<Axes: >
```



PREPARATION	**30**	
LAB PERFORMANCE	**30**	
REPORT	**40**	
TOTAL	**100**	
INITIAL OF THE FACULTY		

RESULT:

The implementation of Linear Discriminant analysis is completed successfully.

717822I210 – DHARANISH M

Ex.No: 10B	LINEAR DISCRIMINANT ANALYSIS
Date:	

AIM:

To implement Linear Discriminant Analysis using wine data

PROCEDURE:

Step1: Import the necessary libraries

Step2: Import the dataset and preprocess it.

Step3: Split the data set into training and testing.

Step4: Find the correlation among the data.

Step5: Create and Train the model.

Step6: Interpret the result.

Step7: Visualize the result.

PROGRAM:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
d=pd.read_csv('/content/wine1.csv')
d.head()
```

	class	Alcohol	Malic acid	Ash	Alcalinityofash	Magnesium	Total phenols	Flavanoids	Nonflavanoid phenols	Proanthocyanins
0	1	14.23	1.71	2.43	15.6	127	2.80	3.06	0.28	2.29
1	1	13.20	1.78	2.14	11.2	100	2.65	2.76	0.26	1.28
2	1	13.16	2.36	2.67	18.6	101	2.80	3.24	0.30	2.81
3	1	14.37	1.95	2.50	16.8	113	3.85	3.49	0.24	2.18
4	1	13.24	2.59	2.87	21.0	118	2.80	2.69	0.39	1.82

```
d.columns = ["class","Alcohol","Malic acid","Ash","Alcalinity of ash","Magnesium","Total phenols","Flavanoids","Nonflavanoid phenols","Proanthocyanins","Color intensity","Hue","OD280/OD315 of diluted wines","Proline"]
```

```
x = d.drop(["class"],axis = 1)
y = d.iloc[:,0]
```

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=0)
```

```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as LDA
lda = LDA(n_components = 2)
ldatr = lda.fit_transform(x_train,y_train)
ldate = lda.transform(x_test)
```

```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(ldatr, y_train)
```

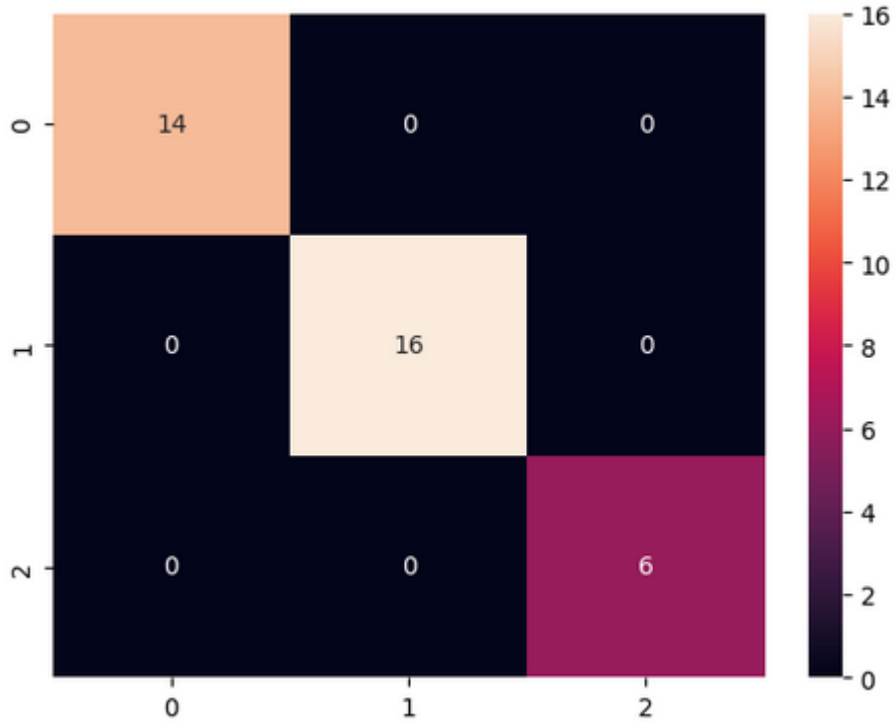
```
y_pred=model.predict(ldate)
```

```
from sklearn.metrics import accuracy_score
accuracy = accuracy_score(y_test, y_pred)
print(accuracy)
```

```
array([[14,  0,  0],
       [ 0, 14,  1],
       [ 0,  5,  2]])
```

```
import seaborn as sns
sns.heatmap(a,annot = True)
```

```
<Axes: >
```



PREPARATION	**30**	
LAB PERFORMANCE	**30**	
REPORT	**40**	
TOTAL	**100**	
INITIAL OF THE FACULTY		

RESULT:

The implementation of Linear Discriminant Analysis is completed successfully.

Ex.No: 2B	IMPLEMENTATION OF MULTIPLE POLYNOMIAL REGRESSION
Date:	

Aim:

To implement the Multiple Linear Regression

Procedure:

Step1: Import the necessary libraries

Step2: Import the dataset and preprocess it.

Step3: Split the data set into training and testing.

Step4: Find the correlation among the data.

Step5: Create and Train the model.

Step6: Interpret the result.

Step7: Visualize the result.

Code:

```
#Importing package and files
import numpy as np
import pandas as pd
d=pd.read_csv("/content/houseprice - houseprice.csv")
d.head()
```

Output:

```
[4]  d.head()
```

Unnamed: 0	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	...	grade	sqft_above	sqft_basement	yr_built	yr_ren
0	7129300520	20141013T000000	221900.0	3.0	1	1180.0	5650	1.0	0	...	7	1180	0	1955	
1	6414100192	20141209T000000	NaN	NaN	?	NaN	7242	2.0	0	...	7	2170	400	1951	
2	5631500400	20150225T000000	180000.0	2.0	1	770.0	10000	1.0	0	...	6	770	0	1933	
3	2487200875	20141209T000000	604000.0	4.0	3	1960.0	5000	1.0	0	...	7	1050	910	1965	
4	1954400510	20150218T000000	510000.0	3.0	2	1680.0	8080	1.0	0	...	8	1680	0	1987	

rows × 22 columns

```
#checking the null values
d.isna().sum()
```

Output:

```
#checking the null values
d.isna().sum()
```

```
Unnamed: 0       0
id               0
date             0
price            1
bedrooms         1
bathrooms        0
sqft_living      1
sqft_lot         0
floors           0
waterfront       0
view             0
condition        0
grade            0
sqft_above       0
sqft_basement    0
yr_built         0
yr_renovated     0
zipcode          0
lat              0
long             0
sqft_living15    0
sqft_lot15       0
dtype: int64
```

```
#Filtering data
d=d.replace('?',np.nan)
d=d.fillna(d.mean())
```

Output:

```
[7] d=d.fillna(d.mean())
```

```
<ipython-input-7-fb2c2214c851>:1: FutureWarning: The default value of numeric_only in DataFrame.mean is deprecated.
  d=d.fillna(d.mean())
```

```
d=d.dropna()
d.corr()
```

Output:

	Unnamed: 0	id	price	bedrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	grade	sqft_above	sqft_basement	yr_built	yr_renovated
Unnamed: 0	1.000000	0.006770	0.027374	0.010762	0.044763	-0.026894	0.179229	-0.007619	-0.013776	-0.095477	0.082011	0.072000	-0.041825	0.199511	-0.025235
id	0.006770	1.000000	-0.016762	0.001298	-0.012274	-0.132104	0.018498	-0.002719	0.011602	-0.023764	0.008147	-0.010856	-0.005158	0.021401	-0.017056
price	0.027374	-0.016762	1.000000	0.308351	0.702040	0.089661	0.256799	0.266369	0.397294	0.036362	0.667439	0.605570	0.323817	0.054012	0.126501
bedrooms	0.010762	0.001298	0.308351	1.000000	0.576686	0.031700	0.175450	-0.006584	0.079527	0.028461	0.356960	0.477613	0.303099	0.154168	0.018938
sqft_living	0.044763	-0.012274	0.702040	0.576686	1.000000	0.172832	0.353936	0.103821	0.284621	-0.058738	0.762729	0.876595	0.435040	0.318071	0.055275
sqft_lot	-0.026894	-0.132104	0.089661	0.031700	0.172832	1.000000	-0.005193	0.021603	0.074708	-0.008964	0.113617	0.183517	0.015288	0.053075	0.007689
floors	0.179229	0.018498	0.256799	0.175450	0.353936	-0.005193	1.000000	0.023703	0.029458	-0.263748	0.458219	0.523878	-0.245720	0.489364	0.006136
waterfront	-0.007619	-0.002719	0.266369	-0.006584	0.103821	0.021603	0.023703	1.000000	0.401857	0.016651	0.082773	0.072077	0.080589	-0.026164	0.092952
view	-0.013776	0.011602	0.397294	0.079527	0.284621	0.074708	0.029458	0.401857	1.000000	0.045981	0.251315	0.167657	0.276951	-0.053450	0.104039
condition	-0.095477	-0.023764	0.036362	0.028461	-0.058738	-0.008964	-0.263748	0.016651	0.045981	1.000000	-0.144692	-0.158202	0.174114	-0.361444	-0.060512

```
#feature
x=d[['sqft_living','grade']]
x
```

Output:

```
[10] #feature
     x=d[['sqft_living','grade']]
     x
```

	sqft_living	grade
0	1180.0	7
2	770.0	6
3	1960.0	7
4	1680.0	8
5	5420.0	11
...	...	...
21608	1530.0	8
21609	2310.0	8
21610	1020.0	7
21611	1600.0	8
21612	1020.0	7

21612 rows × 2 columns

```
#target
y=d[['price']]
y
```

Output:

```
[11] #target
     y=d[['price']]
     y
```

	price
0	221900.0
2	180000.0
3	604000.0
4	510000.0
5	1225000.0
...	...
21608	360000.0
21609	400000.0
21610	402101.0
21611	400000.0
21612	325000.0

21612 rows × 1 columns

```
from sklearn.preprocessing import PolynomialFeatures
p=PolynomialFeatures(degree=3)
x_poly=p.fit_transform(x)
```

```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_poly,y)
```

Output:

```
[13] from sklearn.linear_model import LinearRegression
     lr=LinearRegression()
     lr.fit(x_poly,y)
```

```
LinearRegression
LinearRegression()
```

```
from sklearn.metrics import r2_score,mean_squared_error
print(mean_squared_error(y,lr.predict(x_poly)))
r2_score(y,lr.predict(x_poly))
```

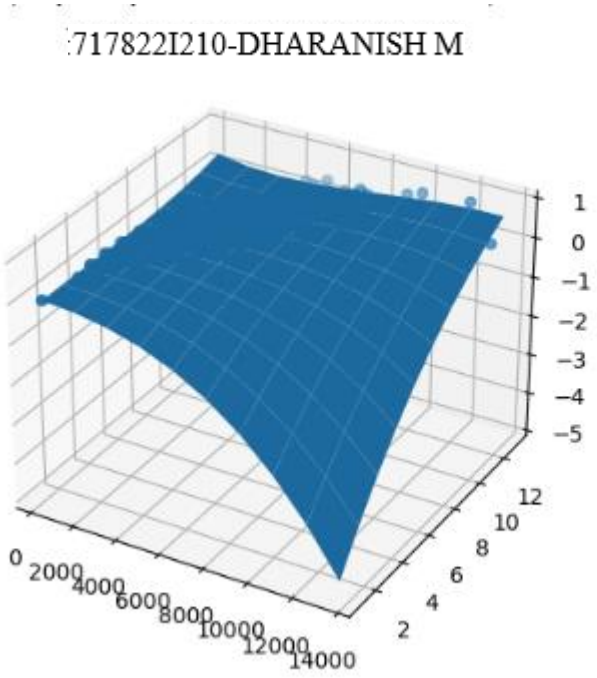
Output:

```
[14] from sklearn.metrics import r2_score,mean_squared_error
     print(mean_squared_error(y,lr.predict(x_poly)))
     r2_score(y,lr.predict(x_poly))
```

```
53711504888.63661
0.601494604801112
```

```
import matplotlib.pyplot as plt
x1_range=np.linspace(min(x.iloc[:,0]),max(x.iloc[:,0]),10)
x2_range=np.linspace(min(x.iloc[:,1]),max(x.iloc[:,1]),10)
x1_grid,x2_grid=np.meshgrid(x1_range,x2_range)
x_grid_poly=p.transform(np.c_[x1_grid.ravel(),x2_grid.ravel()])
y_grid=lr.predict(x_grid_poly).reshape(x1_grid.shape)
fig=plt.figure()
ax=fig.add_subplot(111,projection='3d')
ax.scatter(x.iloc[:,0],x.iloc[:,1],y)
ax.plot_surface(x1_grid,x2_grid,y_grid)
plt.title("717822I210-DHARANISH M ")
```

Output:



PREPARATION	**30**	
LAB PERFORMANCE	**30**	
REPORT	**40**	
TOTAL	**100**	
INITIAL OF THE FACULTY		

Result:

The implementation of Multiple Polynomial Regression is completed successfully.

EX. NO: 2A	**POLYNOMIAL REGRESSION**
DATE:	

AIM:

To process the dataset using polynomial regression

PROCEDURE:

Step 1. Import necessary libraries
Step 2. Import the dataset
Step 3. Handling the null values using the imputing technique
Step 4. Split the dataset into features and target
Step 5. Find the correlation among the data.
Step 6. Implement the feature scaling and train the model
Step 7.After the model train fit into the polynomial regression.
Step 8. Interpret the results
Step 9. Visualize the results

PROGRAM:

CODE 1:

```
# importing libraries and dataset
import pandas as pd
d=pd.read_csv("/content/drive/MyDrive/datasets/Position_Salaries - Position_Salaries.csv")
d.head()
```

OUTPUT:

	Position	Level	Salary
0	Business Analyst	1	45000
1	Junior Consultant	2	50000
2	Senior Consultant	3	60000
3	Manager	4	80000
4	Country Manager	5	110000

CODE 2:

```
#Check for null values
d.isna().sum()
```

OUTPUT:

```
Position    0
Level       0
Salary      0
dtype: int64
```

CODE 3:

```
#To split the dataset into target and features
x=d.iloc[:,[1]]
x
y=d.iloc[:,[2]]
y
```

OUTPUT:

	Level
0	1
1	2
2	3
3	4
4	5

	Salary
0	45000
1	50000
2	60000
3	80000
4	110000

CODE 4:

```
# To split the dataset
from sklearn.model_selection import train_test_split
xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.2,random_state=1)
```

OUTPUT:

```
(   Level
 6      7
 4      5
 0      1
 3      4
 1      2
 7      8
 8      9
 5      6,
    Salary
 6  200000
 4  110000
 0   45000
 3   80000
 1   50000
 7  300000
 8  500000
 5  150000)
```

CODE 5:

```
#To train the model on linear regression
from sklearn.linear_model import LinearRegression
r=LinearRegression()
r.fit(xtrain,ytrain)
```

OUTPUT:

```
▾ LinearRegression
LinearRegression()
```

CODE 6:

```
#To find coefficient and intercept
r.coef_,r.intercept_
```

OUTPUT:

```
(array([[48310.81081081]]), array([-74256.75675676]))
```

CODE 8:

```
y_pred1=r.predict(xtest)
y_pred2=r.predict(xtrain)
y_pred1,y_pred2
```

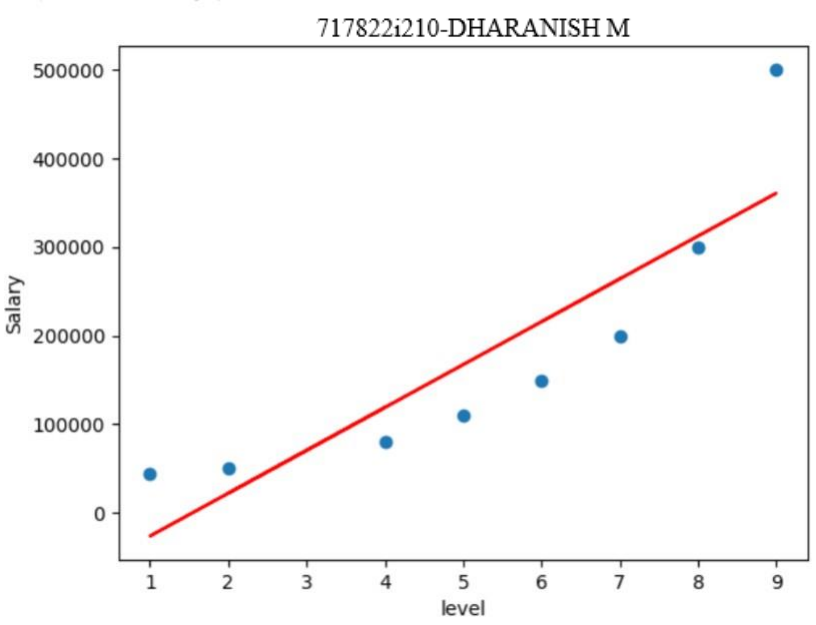
OUTPUT:

```
(array([[ 70675.67567568],
        [408851.35135135]]),
 array([[263918.91891892],
        [167297.2972973 ],
        [-25945.94594595],
        [118986.48648649],
        [ 22364.86486486],
        [312229.72972973],
        [360540.54054054],
        [215608.10810811]]))
```

CODE 8:

```
import numpy as np
import matplotlib.pyplot as plt
plt.scatter(xtrain,ytrain)
plt.plot(np.array(xtrain),y_pred2,color="red")
plt.title("717822i210-DHARANISH M")
plt.xlabel("level")
plt.ylabel("Salary")
```

OUTPUT:



CODE 9:

```
#R2_score
from sklearn.metrics import r2_score,mean_squared_error as mse
print("testdata r2_score:",r2_score(ytest,y_pred1))
print("traindata r2_score:",r2_score(ytrain,y_pred2))
print("testdata mse:",mse(ytest,y_pred1))
print("traindata mse:",mse(ytrain,y_pred2))
```

OUTPUT:

```
testdata r2_score: 0.20875804696637434
traindata r2_score: 0.7704729767407513
testdata mse: 174785347425.1279
traindata mse: 4823564189.189188
```

CODE 10 :

```
#Train the model in polynomial regression
from sklearn.linear_model import LinearRegression
r=LinearRegression()
r.fit(xtrain,ytrain)
from sklearn.preprocessing import PolynomialFeatures
py=PolynomialFeatures(degree=3)
x_poly=py.fit_transform(x)
r.fit(x_poly,y)
```

OUTPUT:

```
▾ LinearRegression
LinearRegression()
```

CODE 10:

```
r.coef_,r.intercept_
```

OUTPUT:

```
(array([[     0.        , 180664.33566432, -48548.95104895,
          4120.04662005]]),
 array([-121333.3333333]))
```

CODE 11:

```
p=x_poly
k=y
p,k
y_pred_poly=r.predict(p)
y_pred_poly
```

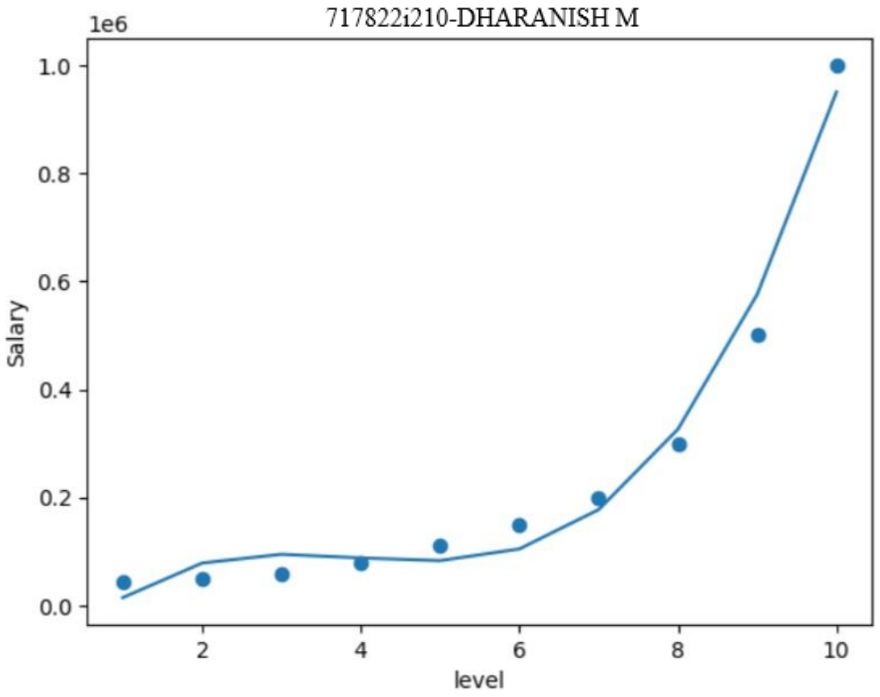
OUTPUT:

```
(array([[   1.,    1.,    1.,    1.],
        [   1.,    2.,    4.,    8.],
        [   1.,    3.,    9.,   27.],
        [   1.,    4.,   16.,   64.],
        [   1.,    5.,   25.,  125.],
        [   1.,    6.,   36.,  216.],
        [   1.,    7.,   49.,  343.],
        [   1.,    8.,   64.,  512.],
        [   1.,    9.,   81.,  729.],
        [   1.,   10.,  100., 1000.]]),
    Salary
0    45000
1    50000
2    60000
3    80000
4   110000
5   150000
6   200000
7   300000
8   500000
9  1000000)

array([[ 14902.09790211],
       [ 78759.90675991],
       [ 94960.37296037],
       [ 88223.77622378],
       [ 83270.3962704 ],
       [104820.51282052],
       [177594.40559441],
       [326312.35431236],
       [575694.63869463],
       [950461.53846152]])
```

CODE 12:

```
#Plotting
import matplotlib.pyplot as plt
plt.scatter(x,y)
plt.plot(x,y_pred_poly)
plt.title("717822i210-
DHARANISH M")
plt.xlabel("level")
plt.ylabel("Salary")
```

OUTPUT:



CODE 13:

```
#R2_score
from sklearn.metrics import r2_score,mean_squared_error as mse
print("traindata r2_score:",r2_score(y,y_pred_poly))
print("traindata mse:",mse(y,y_pred_poly))
```

OUTPUT:

```
traindata r2_score: 0.9812097727913366
traindata mse: 1515662004.6620114
```

PREPARATION	30	
LAB PERFORMANCE	30	
REPORT	40	
TOTAL	100	
INITIAL OF THE FACULTY		

RESULT:

Thus the polynomial regression using position salary dataset has been successfully completed.

EX. NO: 6A	IMPLEMENTATION OF LOGISTIC REGRESSION
DATE:	

AIM:

To implement the logistic regression

PROCEDURE:

Step 1. Import necessary libraries
Step 2. Import the dataset
Step 3. Handling the null values using the imputing technique
Step 4. Split the dataset into features and target
Step 5. Implement the feature scaling and train the model
Step 6.After the model train fit into the logistic regression.
Step 7. Interpret the results
Step 8. Visualize the results

PROGRAM:

CODE 1:

```
#Importing the necessary libraries and dataset
import pandas as pd
d=pd.read_csv("/content/drive/MyDrive/datasets/insurance_data - insurance_data (1).csv")
d.head()
```

OUTPUT:

	age	bought_insurance
0	22	0
1	25	0
2	47	1
3	52	0
4	46	1

CODE 2:

```
#Check for null values
d.isna().sum()
```

OUTPUT:

```
age                0
bought_insurance   0
dtype: int64
```

CODE 3:

```
#To split the dataset into features and target
x=d.iloc[:,[0]]
y=d.iloc[:,[-1]]
```

CODE 4:

```
#Split the data into train and test
from sklearn.model_selection import train_test_split
xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.2,random_state=1)
```

CODE 5:

```
#To fit the model
from sklearn.linear_model import LogisticRegression
l=LogisticRegression()
l.fit(xtrain,ytrain)
```

OUTPUT:

```
LogisticRegression
LogisticRegression()
```

CODE 6:

```
#To find the model test data accuracy score
ypred=l.predict(xtest)
from sklearn.metrics import accuracy_score
print(accuracy_score(ytest,ypred))
```

OUTPUT:

```
0.8333333333333334
```

CODE 8:

```
#Confusion_matrix shows the model prediction output in Logistic way 0 and 1
from sklearn.metrics import confusion_matrix as cm
cm(ytest,ypred)
```

OUTPUT:

CODE 8:

```
#Train the model
from sklearn.linear_model import LinearRegression
r=LinearRegression()
r.fit(xtrain,ytrain)
```

OUTPUT:

```
array([[2, 1],
       [0, 3]])
```

CODE 9:

```
#It gives our classification model reports
from sklearn.metrics import classification_report as cr
print(cr(ytest,ypred))
```

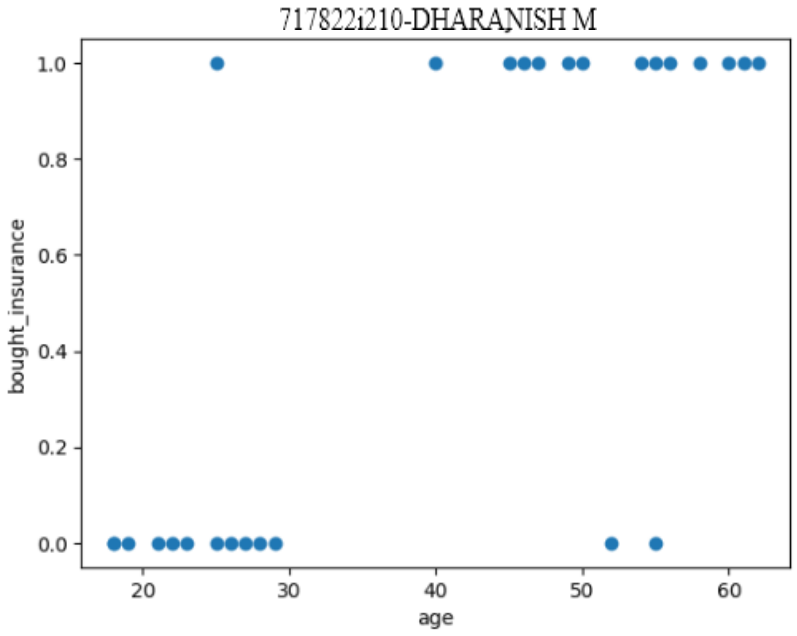
OUTPUT:

	precision	recall	f1-score	support
0	1.00	0.67	0.80	3
1	0.75	1.00	0.86	3
accuracy			0.83	6
macro avg	0.88	0.83	0.83	6
weighted avg	0.88	0.83	0.83	6

CODE 10 :

```
#Plotting of the dataset
import matplotlib.pyplot as plt
plt.scatter(x,y)
plt.title("717822i210-DHARANISH M")
plt.xlabel("age")
plt.ylabel("bought_insurance")
```

OUTPUT:



PREPARATION	30	
LAB PERFORMANCE	30	
REPORT	40	
TOTAL	100	
INITIAL OF THE FACULTY		

RESULT:

The implementation of logistic regression using insurance dataset has been successfully completed

EX. NO: 6B	IMAGE ANALYSIS USING LOGISTIC REGRESSION
DATE:	

AIM:

To image analysis using logistic regression

PROCEDURE:

Step 1. Import necessary libraries
Step 2. Get the image dataset from sklearn library import load_digits dataset
Step 4. Get the features and target from the digits dataset
Step 5. To train the model
Step 6.After the model train fit into the logistic regression.
Step 7. Interpret the results
Step 8. Visualize the results

PROGRAM:

CODE 1:

```
#It is used to get the dataset
from sklearn.datasets import load_digits
digits=load_digits()
digits
```

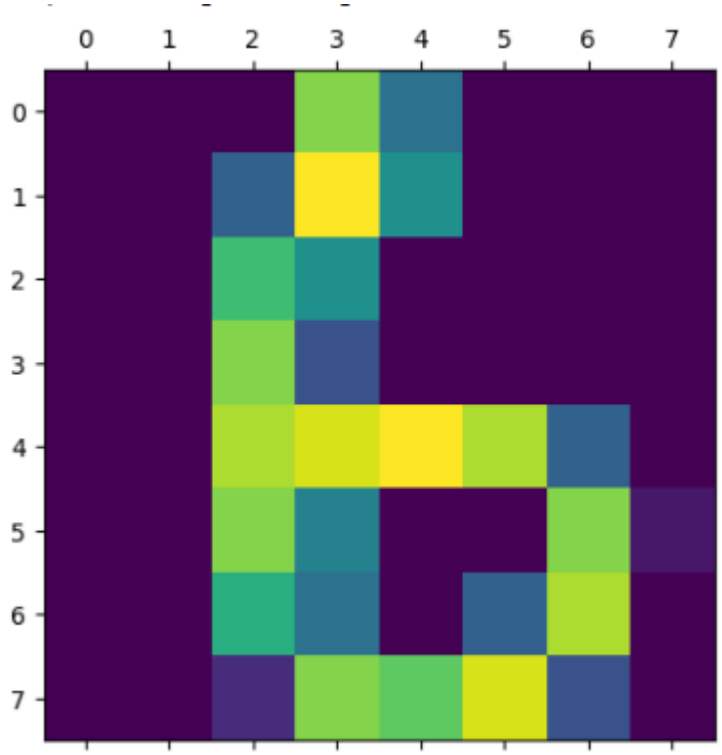
OUTPUT:

```
{'data': array([[ 0.,  0.,  5., ...,  0.,  0.,  0.],
        [ 0.,  0.,  0., ..., 10.,  0.,  0.],
        [ 0.,  0.,  0., ..., 16.,  9.,  0.],
        ...,
        [ 0.,  0.,  1., ...,  6.,  0.,  0.],
        [ 0.,  0.,  2., ..., 12.,  0.,  0.],
        [ 0.,  0., 10., ..., 12.,  1.,  0.]]),
 'target': array([0, 1, 2, ..., 8, 9, 8]),
 'frame': None,
 'feature_names': ['pixel_0_0',
  'pixel_0_1',
  'pixel_0_2',
  'pixel_0_3',
  'pixel_0_4',
  'pixel_0_5',
  'pixel_0_6',
```

CODE 2:

```
#plotting the  certain image in the dataset
import matplotlib.pyplot as plt
plt.matshow(digits.images[1122])
#plt.gray() Its for getting black and white image only
```

OUTPUT:



CODE 3:

```
#To split the dataset into train and test data
from sklearn.model_selection import train_test_split
xtrain,xtest,ytrain,ytest=train_test_split(digits.data,digits.target,test_size=0.2,random_state=1)
```

CODE 4:

```
#To fit the LogisticRegression model
from sklearn.linear_model import LogisticRegression
l=LogisticRegression()
l.fit(xtrain,ytrain)
```

OUTPUT:

```
LogisticRegression
LogisticRegression()
```

CODE 6:

```
#To find the model test data accuracy score
ypred=l.predict(xtest)
from sklearn.metrics import accuracy_score
print(accuracy_score(ytest,ypred))
```

OUTPUT:

```
0.9694444444444444
```

CODE 8:

```
#Confusion_matrix shows the model prediction output in Logistic way 0 and 1
from sklearn.metrics import confusion_matrix as cm
cm(ytest,ypred)
```

OUTPUT:

```
array([[42,  0,  0,  0,  1,  0,  0,  0,  0,  0],
       [ 0, 34,  0,  0,  1,  0,  0,  0,  0,  0],
       [ 0,  0, 36,  0,  0,  0,  0,  0,  0,  0],
       [ 0,  0,  0, 40,  0,  0,  0,  0,  1,  0],
       [ 0,  0,  0,  0, 38,  0,  0,  0,  0,  0],
       [ 0,  1,  0,  1,  0, 28,  0,  0,  0,  0],
       [ 0,  0,  0,  0,  0,  0, 37,  0,  0,  0],
       [ 0,  0,  0,  1,  1,  1,  0, 33,  0,  1],
       [ 0,  0,  0,  0,  0,  1,  0,  0, 28,  0],
       [ 0,  0,  0,  0,  0,  1,  0,  0,  0, 33]])
```

CODE 8:

```
#It gives our classification model reports
from sklearn.metrics import classification_report as cr
print(cr(ytest,ypred))
```

OUTPUT:

```
              precision    recall  f1-score   support

           0       1.00      0.98      0.99        43
           1       0.97      0.97      0.97        35
           2       1.00      1.00      1.00        36
           3       0.95      0.98      0.96        41
           4       0.93      1.00      0.96        38
           5       0.90      0.93      0.92        30
           6       1.00      1.00      1.00        37
           7       1.00      0.89      0.94        37
           8       0.97      0.97      0.97        29
           9       0.97      0.97      0.97        34

    accuracy                           0.97       360
   macro avg       0.97      0.97      0.97       360
weighted avg       0.97      0.97      0.97       360
```

PREPARATION	30	
LAB PERFORMANCE	30	
REPORT	40	
TOTAL	100	
INITIAL OF THE FACULTY		

RESULT:

The image analysis using logistic regression using sklearn inbuild dataset has been successfully completed

Ex.No: 1C	IMPLEMENTATION OF MULTIPLELINEAR REGRESSION
Date:	

Aim:

To implement the Multiple Linear Regression

Procedure:

Step1: Import the necessary libraries

Step2: Import the dataset and preprocess it.

Step3: Split the data set into training and testing.

Step4: Find the correlation among the data.

Step5: Create and Train the model.

Step6: Interpret the result.

Step7: Visualize the result.

Code:

```
#dharanish-717822i210
import pandas as pd
import numpy as np
#dharanish-717822i210
d=pd.read_csv("/content/startup-2 - startup-2.csv")
#dharanish-717822i210
d.head()
```

Output:

```
d.head()
```

	Unnamed: 0	R&D Spend	Administration	Marketing Spend	State	Profit
0	0	165349.20	136897.80	471784.10	New York	192261.83
1	1	162597.70	151377.59	443898.53	California	191792.06
2	2	NaN	NaN	407934.54	Florida	191050.39
3	3	144372.41	118671.85	383199.62	New York	182901.99
4	4	142107.34	91391.77	366168.42	Florida	166187.94

```
#dharanish-717822i210
d=d.drop("Unnamed: 0",axis=1)
d.head()
```

Output:

```
d.head()
```

	R&D Spend	Administration	Marketing Spend	State	Profit
0	165349.20	136897.80	471784.10	New York	192261.83
1	162597.70	151377.59	443898.53	California	191792.06
2	NaN	NaN	407934.54	Florida	191050.39
3	144372.41	118671.85	383199.62	New York	182901.99
4	142107.34	91391.77	366168.42	Florida	166187.94

```
d.isnull().sum()
```

Output:

```
[60] d.isnull().sum()

R&D Spend          4
Administration     4
Marketing Spend    0
State              3
Profit             3
dtype: int64
```

```
d["R&D Spend"]=d["R&D Spend"].fillna(d["R&D Spend"].mean())
d["Administration"]=d["Administration"].fillna(d["Administration"].mean())
d["Profit"]=d["Profit"].fillna(d["Profit"].mean())
from sklearn.impute import SimpleImputer
a=SimpleImputer(missing_values=np.nan,strategy="most_frequent")
d[["State"]]=a.fit_transform(d[["State"]])
#dharanish-717822i210
d=pd.get_dummies(d,columns=["State"])
d.head()
```

Output:

```
[68] d.head()
```

	R&D Spend	Administration	Marketing Spend	Profit	State_California	State_Florida	State_New York
0	165349.200000	136897.800000	471784.10	192261.83	0	0	1
1	162597.700000	151377.590000	443898.53	191792.06	1	0	0
2	70148.693261	120580.946957	407934.54	191050.39	0	1	0
3	144372.410000	118671.850000	383199.62	182901.99	0	0	1
4	142107.340000	91391.770000	366168.42	166187.94	0	1	0

```
x=d.iloc[:,[0,1,2,4,5,6]]
x
```

Output:

```
x=d.iloc[:,[0,1,2,4,5,6]]
x
```

	R&D Spend	Administration	Marketing Spend	State_California	State_Florida	State_New York
0	165349.200000	136897.800000	471784.10	0	0	1
1	162597.700000	151377.590000	443898.53	1	0	0
2	70148.693261	120580.946957	407934.54	0	1	0
3	144372.410000	118671.850000	383199.62	0	0	1
4	142107.340000	91391.770000	366168.42	0	1	0

```
y=d.iloc[:,[3]]
y
```

Output:

```
y=d.iloc[:,[3]]
y
```

	Profit
0	192261.830000
1	191792.060000
2	191050.390000
3	182901.990000
4	166187.940000

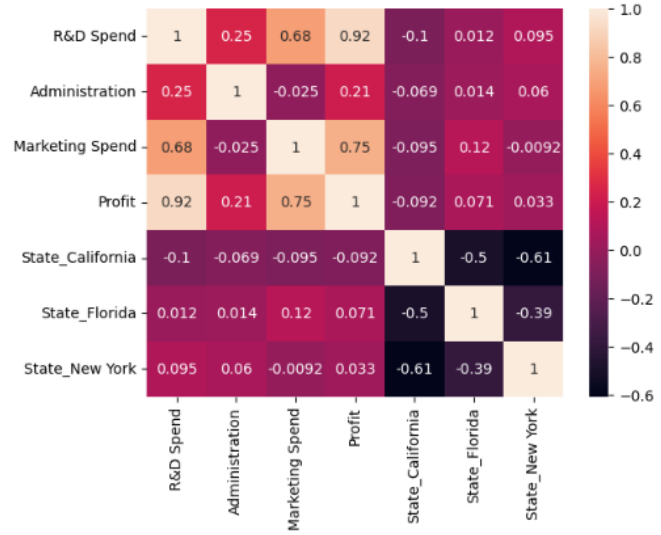
```
#dharanish-717822i210
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
c=ColumnTransformer([("Encoder",OneHotEncoder(),[3])],remainder="passthrough")
```

```
#dharanish-717822i210
import seaborn as sns
sns.heatmap(d.corr(),annot=True)
```

Output:

```
import seaborn as sns
sns.heatmap(d.corr(),annot=True)
```

```
<Axes: >
```




```
#dharanish-717822i210
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
lg=LinearRegression()
a=lg.fit(x,y)
a
```

Output:

```
LinearRegression
LinearRegression()
```

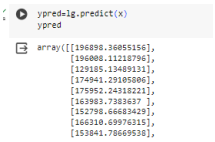
lg.coef_

```
[76] lg.coef_

array([[ 7.06309388e-01,  3.35557337e-02,  7.37079527e-02,
         1.54603373e+02,  2.31346347e+03, -2.46806684e+03]])
```

lg.intercept_

```
lg.intercept_

array([43210.7888911])
```

```
ypred=lg.predict(x)
ypred
```



r2_score(y,ypred)

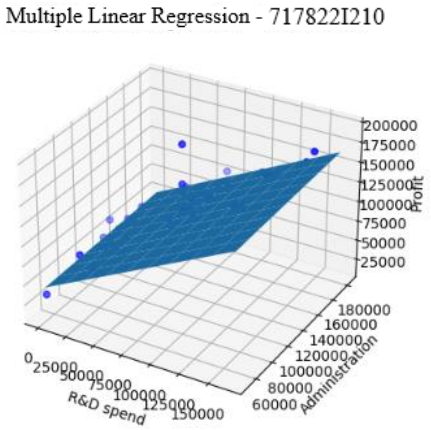
Output:

```
[79] r2_score(y,ypred)

0.8778631247742841
```

```
#dharanish-717822i210
import matplotlib.pyplot as plt
x1_range = np.linspace(min(x.iloc[:, 0]), max(x.iloc[:, 0]), 10)
x2_range = np.linspace(min(x.iloc[:, 1]), max(x.iloc[:, 1]), 10)
X1_grid, X2_grid = np.meshgrid(x1_range, x2_range)
X_grid = np.column_stack((X1_grid.ravel(),X2_grid.ravel(),
                          np.zeros_like(X1_grid.ravel()),
                          np.zeros_like(X1_grid.ravel()),
                          np.zeros_like(X1_grid.ravel()),
                          np.zeros_like(X1_grid.ravel())))
Y_grid=lg.predict(X_grid).reshape(X1_grid.shape)
# Creating the 3D plot
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
# Plotting the original data points
ax.scatter(x.iloc[:, 0], x.iloc[:, 1], y, color='blue', label='Data')
# Plotting the plane of best fit
ax.plot_surface(X1_grid, X2_grid, Y_grid)
# Adding labels and legend
```

```
ax.set_xlabel('R&D spend')
ax.set_ylabel('Administration')
ax.set_zlabel('Profit')
ax.set_title('Multiple Linear Regression - 717822I210')
```

Output:



PREPARATION	**30**	
LAB PERFORMANCE	**30**	
REPORT	**40**	
TOTAL	**100**	
INITIAL OF THE FACULTY		

Result:

The implementation of Multiple Linear Regression is completed successfully.