EXPERIMENT 1

Write a program to open Data Sets in Python.

- Importing CSV Files
- Importing Excel Files
- Importing Text Files

CSV Files:-

- A CSV (comma-separated values) file is a simple text file in which information is separated by commas.
- CSV file is imported in python pandas you will need to use **read.csv** function

- **Importing CSV Files**

```
# Import Libraries
 import pandas as pd
#Load Dataset
df=pd.read_csv("C:/Users/test/Desktop/nba.csv")
df
# To retrieve specific column in Data Frame
   df["Name"]
#To retrieve more than one column in Data Frame
  df[["Name","College","Salary"]]
```

- **Importing Excel Files**

  - Excel file is imported in python using pandas you will need to use **read_excel** function.

```
# import libraries
import pandas as pd
#Load Dataset
df = pd.read_excel ("C:/Users/test/Desktop/nba.xls")
print (df)
#select the 5th row of the DataFrame
df.iloc[[5]]
```

❖ **Importing text Files**

- Text file is imported in python using pandas you will need to use **read_table** function.

```
# import libraries
import pandas as pd
df=pd.read_table("C:/Users/test/Desktop/nba.txt")
df
#select the 3rd, 4th, and 5th rows of the DataFrame
df.iloc[[2, 3, 4]]
```

Output:-

nba.csv

	Name	College	Salary
0	Avery Bradley	Texas	7730337.0
1	Jae Crowder	Marquette	6796117.0
2	John Holland	Boston University	NaN
3	R.J. Hunter	Georgia State	1148640.0
4	Jonas Jerebko	NaN	5000000.0

 # Select the 5th row of the Data Frame

	Name	Team	Number	Position	Age	Height	Weight	College	Salary
5	Amir Johnson	Boston Celtics	90.0	PF	29.0	6-9	240.0	NaN	12000000.0

#select the 3rd, 4th, and 5th rows of the DataFrame

	Name	Team	Number	Position	Age	Height	Weight	College	Salary
2	John Holland	Boston Celtics	30.0	SG	27.0	6-5	205.0	Boston University	NaN
3	R.J. Hunter	Boston Celtics	28.0	SG	22.0	6-5	185.0	Georgia State	1148640.0
4	Jonas Jerebko	Boston Celtics	8.0	PF	29.0	6-10	231.0	NaN	5000000.0

EXPERIMENT 2

Explain various Plotting Techniques of Python

(a) Line Graph OR Line Plot:-

- The line chart is used to display the information as a series of the line. It is easy to plot

```
# Import libraries
import matplotlib.pyplot as plt
import numpy as np
x = np.array([1, 2, 3, 4])
y = x*2
plt.plot(x, y)
plt.xlabel("X-axis")
plt.ylabel("Y-axis")
plt.title("Any suitable title")
plt.show()  # show first chart
# The figure() function helps in creating a new figure that can hold a new chart in it.
plt.figure()
x1 = [2, 4, 6, 8]
y1 = [3, 5, 7, 9]
plt.plot(x1, y1, '-.')
```

(b) Bar chart:-

- Bar chart is a chart that represents the **categorical data** in a rectangular format.
- You can create both horizontal and vertical bar charts

```
# Import Library
import matplotlib.pyplot as plt
import pandas as pd
# Define Data
 data = {'Girls': [15, 20, 25, 30, 35],
       'Boys': [25, 30, 28, 19, 40] }
df = pd.DataFrame(data,columns=['Girls','Boys'],
```

```
                    index = ['Team-1','Team-2','Team-3','Team-4','Team-5'])
# Multiple horizontal bar chart
df.plot.barh()
# Display
plt.show()
```

(c) Histogram:-

- Histogram is used where the data is been distributed.
- Histograms are preferred during the arrays or data containing the long list.
- In histogram the data will equally distribute into bins.
- Each bin represents data intervals

```
import matplotlib.pyplot as plt
import pandas as pd
df = pd.read_csv("https://media.geeksforgeeks.org/wp-content/uploads/nba.csv")
df
plt.hist(df['Age'],bins=20)
plt.xlabel('Bins', fontsize = 15, color = 'green')
plt.ylabel('Age', fontsize = 15, color = 'blue')
plt.xticks(fontsize = 12)
plt.yticks(fontsize = 12)
plt.show()
```

(d) Pie chart:-

Pie chart is a circular graph which is divided into the sub-part or segment.

```
import matplotlib.pyplot as plt
import numpy as np
y = np.array([35, 25, 25, 15])
mylabels = ["Apples", "Bananas", "Cherries", "Dates"]
plt.pie(y, labels = mylabels, startangle = 90)
plt.show()
```

(e) Scatter Plot:-

A scatter plot is a diagram where each value in the data set is represented by a dot.

```
import matplotlib.pyplot as plt
x = [5,7,8,7,2,17,2,9,4,11,12,9,6]
y = [99,86,87,88,111,86,103,87,94,78,77,85,86]
plt.scatter(x, y)
plt.show()
```

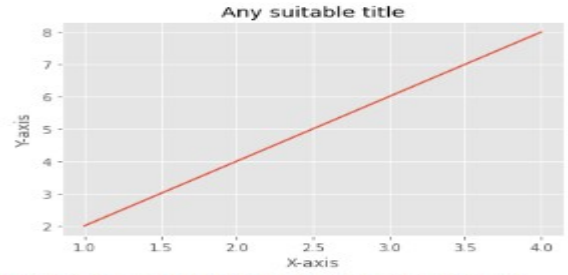
(f) Box Plot:-

- Boxplots are **a measure of how well data is distributed across a data set**.
- This divides the data set into three quartiles.
- This graph represents the minimum, maximum, average, first quartile, and the third quartile in the data set.

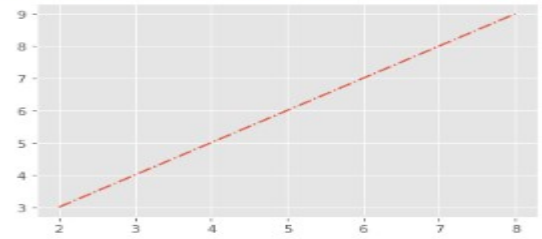
```
# Import libraries
import matplotlib.pyplot as plt
import numpy as np
# Creating dataset
np.random.seed(10)
data = np.random.normal(100, 20, 200)
fig = plt.figure(figsize =(10, 7))
# Creating plot
plt.boxplot(data)
# show plot
plt.show()
```

Output:-

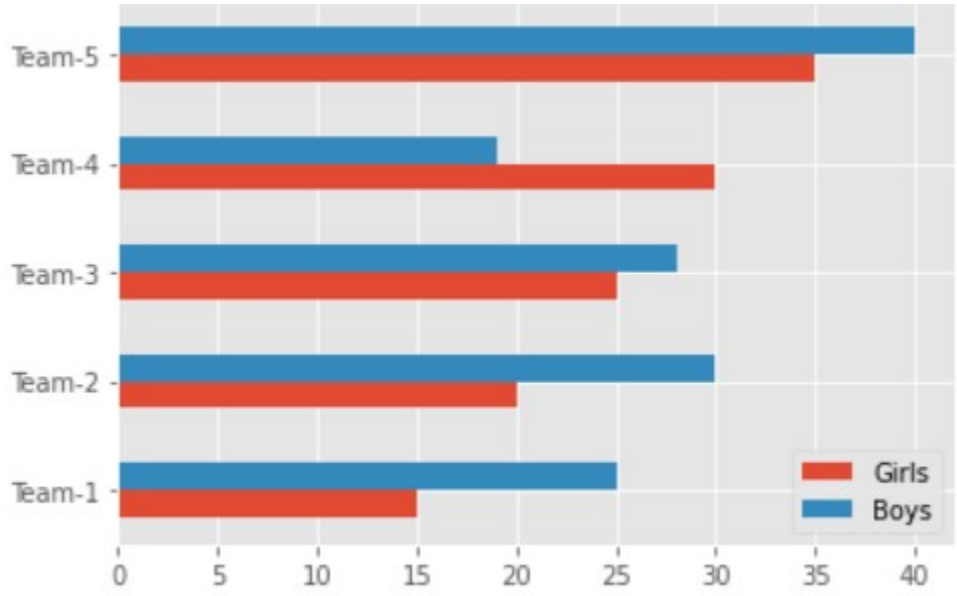
Line Graph:-



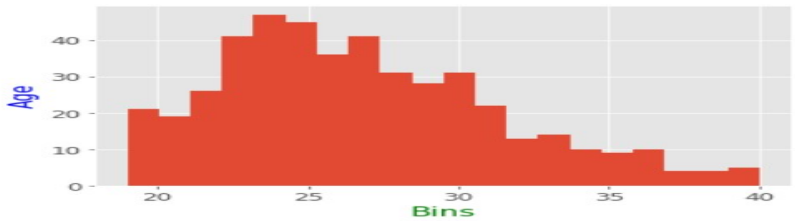

```
[<matplotlib.lines.Line2D at 0x7f41fefcfe50>]
```



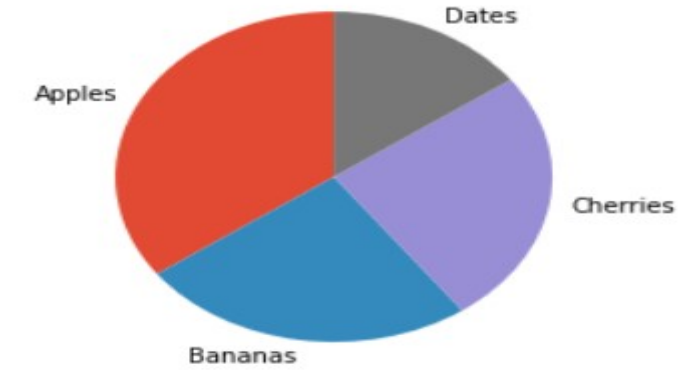
Bar chart:-



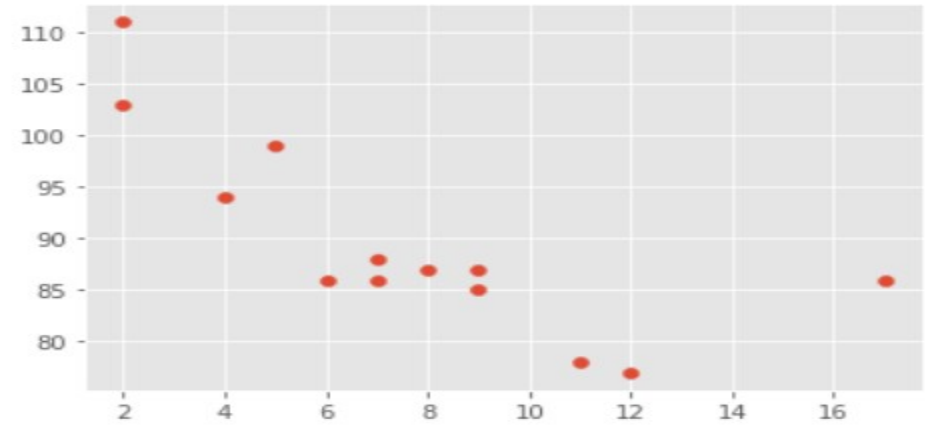
Histogram:-



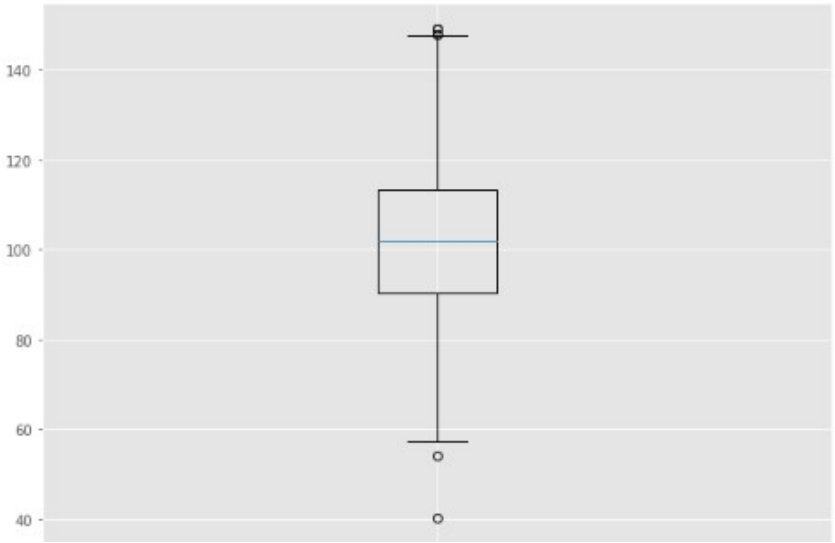
Pie chart:-



Scatter Plot:-



Box plot:-



EXPERIMENT 3

Demonstrate Simple Linear Regression in Python with Sample Data Sets.

```
# import libraries
import numpy as np
import matplotlib.pyplot as mtp
import pandas as pd
data_set= pd.read_csv('Salary_Data.csv')
data_set.head()
# Remove last column  or extract independent variable
x= data_set.iloc[:,:-1].values
x
# Get second column of data frame   or extract dependent variable
y= data_set.iloc[:, 1].values
y
# Splitting the dataset into training and test set.
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test= train_test_split(x, y, test_size= 1/3, random_state=0)
print(x_train.shape)
print(x_train)
print(x_test.shape)
print(x_test)
print(y_train.shape)
print(y_train)
#Fitting the Simple Linear Regression model to the training dataset
from sklearn.linear_model import LinearRegression
regressor= LinearRegression()
regressor.fit(x_train, y_train)
#Prediction of Test and Training set result
y_pred= regressor.predict(x_test)
y_pred
```

```
x_pred= regressor.predict(x_train)
#visualizing the Training set results
mtp.scatter(x_train,y_train, color="green")
mtp.plot(x_train,x_pred, color="red")
mtp.title("Salary vs Experience (Training Dataset)")
mtp.xlabel("Years of Experience")
mtp.ylabel("Salary(In Rupees)")
mtp.show()
#visualizing the Test set results
mtp.scatter(x_test, y_test, color="blue")
mtp.plot(x_train, x_pred, color="red")
mtp.title("Salary vs Experience (Test Dataset)")
mtp.xlabel("Years of Experience")
mtp.ylabel("Salary(In Rupees)")
mtp.show()
# Compute and print R^2, MAE, MSE and RMSE
from sklearn import metrics
print("R^2: {}".format(regressor.score(x_test, y_test)))
print("Mean Absolute Error:",metrics.mean_absolute_error(y_test,y_pred))
print("Mean Squared Error:",metrics.mean_squared_error(y_test,y_pred))
print('Root Mean Squared Error:',np.sqrt(metrics.mean_squared_error(y_test,y_pred)))
```

Output:-


```
R^2: 0.9749154407708353
Mean Absolute Error: 3426.4269374307078
Mean Squared Error: 21026037.329511296
Root Mean Squared Error: 4585.4157204675885
```

EXPERIMENT 4

Demonstrate Multiple Linear Regression in Python with Sample Data Sets

```
#importing libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
#Load dataset
df=pd.read_csv("insurance.csv")
df.head()
df['sex']=df['sex'].astype('category')
df['sex']=df['sex'].cat.codes
df
df['smoker']=df['smoker'].astype('category')
df['smoker']=df['smoker'].cat.codes
df
df['region']=df['region'].astype('category')
df['region']=df['region'].cat.codes
df
df.isnull().sum()
#Extracting independent variables
X=df.iloc[:,:-1].values
X
#Extracting  dependent  variables
y=df.iloc[:,6]
y
# Splitting the dataset into training and test set.
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test= train_test_split(X,y,test_size= 0.3,random_state=0)
#Fitting the MLR model to the training set:
from sklearn.linear_model import LinearRegression
```

```
Lr=LinearRegression()
Lr.fit(x_train,y_train)
#Predicting the Test set result
y_pred= Lr.predict(x_test)
print('Train Score: ',Lr.score(x_train, y_train))
print('Test Score: ',Lr.score(x_test, y_test))
```

Output:-

Train Score: 0.7306860719626136

Test Score: 0.791151983192236

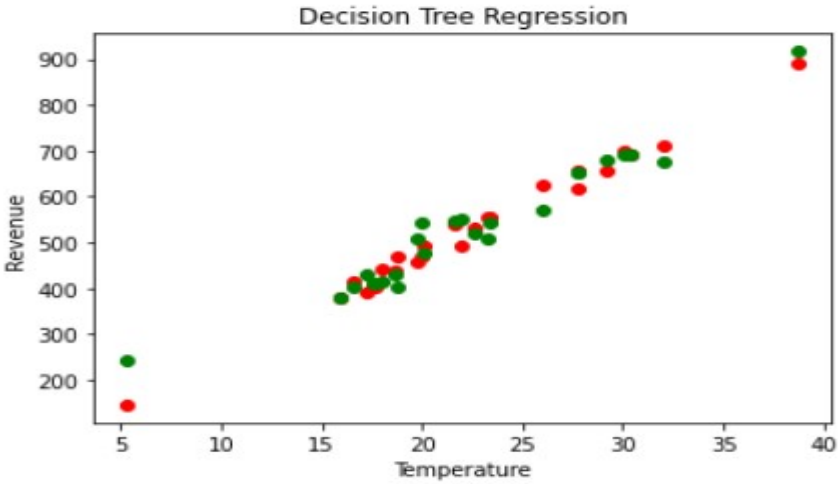
EXPERIMENT 5

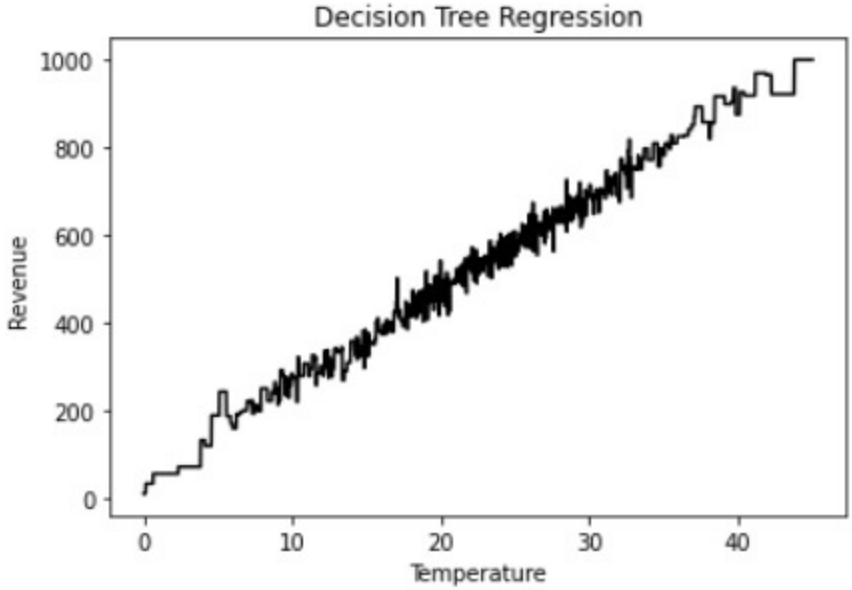
Demonstrate Decision Tree Regression in Python with Sample Data Sets.

```
#import libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
#load dataset
dataset = pd.read_csv('https://raw.githubusercontent.com/mk-gurucharan/Regression/master/IceCreamData.csv')
dataset
#Extracting Independent and dependent Variables
X = dataset['Temperature'].values
y = dataset['Revenue'].values
dataset.head(5)
#Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.05)
print(X_train.shape)
print(X_test.shape)
#Fitting Decision Tree Regression to the dataset
from sklearn.tree import DecisionTreeRegressor
regressor = DecisionTreeRegressor()
regressor.fit(X_train.reshape(-1,1), y_train.reshape(-1,1))
#Predict the results of the test set with the model trained on the training set
y_pred = regressor.predict(X_test.reshape(-1,1))
y_pred
# Comparing the Real Values with Predicted Values
df = pd.DataFrame({'Real Values':y_test.reshape(-1), 'Predicted Values':y_pred.reshape(-1)})
df
```

```
# Visualising the Decision Tree Regression Results
# arange for creating a range of values  from min value of X to #max value of X  with a
 difference of 0.01 between two #consecutive values
X_grid = np.arange(min(X), max(X), 0.01)
X_grid = X_grid.reshape((len(X_grid), 1))
plt.scatter(X_test, y_test, color = 'red')
plt.scatter(X_test, y_pred, color = 'green')
plt.title('Decision Tree Regression')
plt.xlabel('Temperature')
plt.ylabel('Revenue')
plt.show()
plt.plot(X_grid, regressor.predict(X_grid), color = 'black')
plt.title('Decision Tree Regression')
plt.xlabel('Temperature')
plt.ylabel('Revenue')
plt.show()
```

Output:-


Decision Tree Regression
Revenue
Temperature
900
800
700
600
500
400
300
200
5
10
15
20
25
30
35
40


Decision Tree Regression
Revenue
Temperature
1000
800
600
400
200
0
0
10
20
30
40

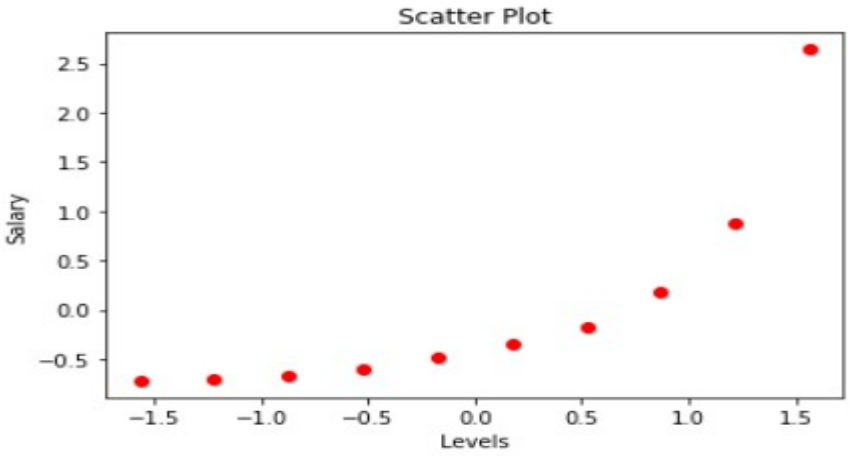
EXPERIMENT 6

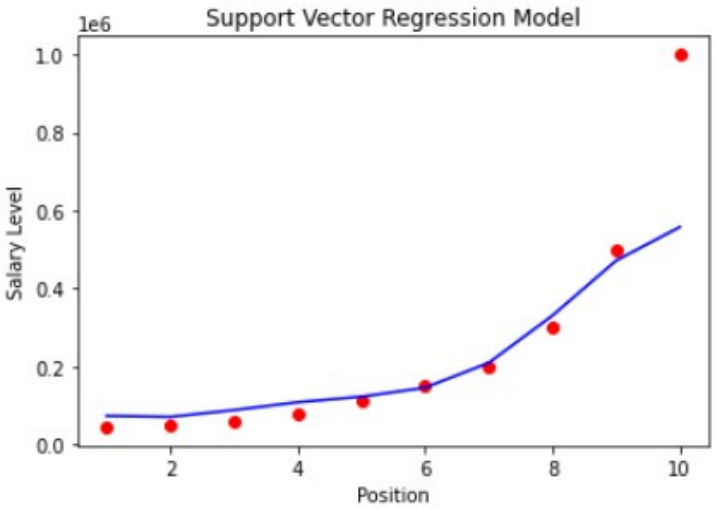
Demonstrate Support Vector Regression in Python with Sample Data Sets..

```
# import libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
# get the dataset
dataset = pd.read_csv('Position_Salaries.csv')
print(dataset)
# split the data into featutes and target variable seperately
X_l = dataset.iloc[:, 1:-1].values # features set
y_p = dataset.iloc[:, -1].values # set of study variable
print(X_l)
print(y_p)
y_p = y_p.reshape(-1,1)
y_p
#feature Scaling
from sklearn.preprocessing import StandardScaler
StdS_X = StandardScaler()
StdS_y = StandardScaler()
X = StdS_X.fit_transform(X_l)
y = StdS_y.fit_transform(y_p)
print("Scaled X:")
print(X)
print("Scaled y:")
print(y)
plt.scatter(X, y, color = 'red') # plotting the training set
plt.title('Scatter Plot') # adding a tittle to our plot
plt.xlabel('Levels') # adds a label to the x-axis
```

```
plt.ylabel('Salary') # adds a label to the y-axis
plt.show() # prints
# import the model
#Radial Basis function
from sklearn.svm import SVR
# create the model object
regressor = SVR(kernel = 'rbf')
# fit the model on the data
regressor.fit(X, y)
#Prediction result
A=regressor.predict(StdS_X.transform([[6.5]]))
print(A)
#visualization
# inverse the transformation to go back to the initial scale
plt.scatter(StdS_X.inverse_transform(X), StdS_y.inverse_transform(y), color = 'red')
plt.plot(StdS_X.inverse_transform(X),
StdS_y.inverse_transform(regressor.predict(X).reshape(-1,1)), color = 'blue')
# add the title to the plot
plt.title('Support Vector Regression Model')
# label x axis
plt.xlabel('Position')
# label y axis
plt.ylabel('Salary Level')
# print the plot
plt.show()
```

Output:-


Scatter Plot
Salary
Levels
2.5
2.0
1.5
1.0
0.5
0.0
−0.5
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5


Support Vector Regression Model
1e6
Salary Level
Position
1.0
0.8
0.6
0.4
0.2
0.0
2
4
6
8
10

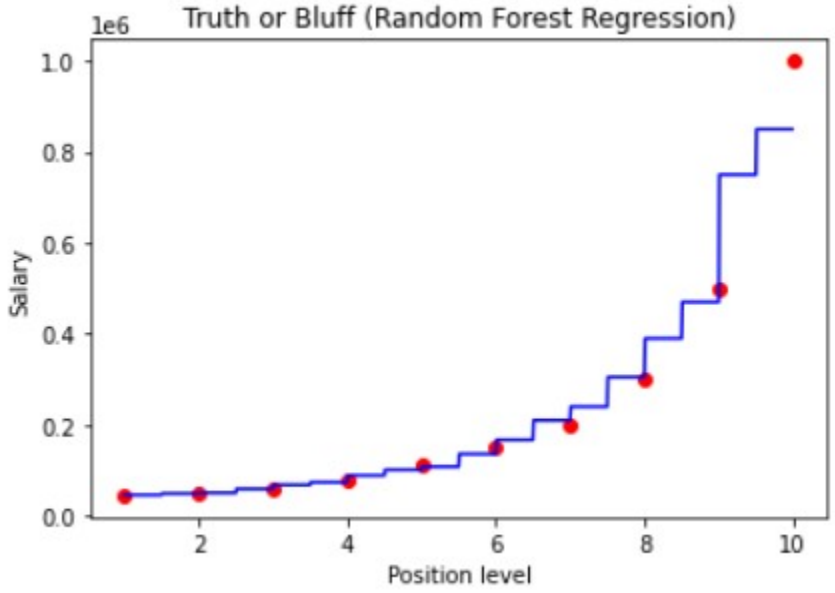
EXPERIMENT 7

Demonstrate Random Forest Regression in Python with Sample Data Sets

```
# Importing the Essential Libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
# Importing the Dataset
dataset = pd.read_csv("Position_Salaries.csv")
dataset
# Independent Variables
X = dataset.iloc[:, 1:2].values
X
# Dependent Variables
y = dataset.iloc[:, 2].values
y
# Creating A Random Forest Regression Model And Fitting It To The Training Data
#n_estimators is a parameter that sets the number of decision trees created for a ran
dom data point
#random_state = 0 is used to produce the same output
from sklearn.ensemble import RandomForestRegressor
 # create regressor object
regressor = RandomForestRegressor(n_estimators = 10, random_state = 0)
# fit the regressor with x and y data
regressor.fit(X, y)
# Predicting a New Value
y_pred = regressor.predict([[6.5]])
y_pred
# Visualizing the Training Set
X_grid = np.arange(min(X), max(X), 0.01)
X_grid = X_grid.reshape((len(X_grid), 1))
```

```
plt.scatter(X, y, color = 'red')
plt.plot(X_grid, regressor.predict(X_grid), color = 'blue')
plt.title('Truth or Bluff (Random Forest Regression)')
plt.xlabel('Position level')
plt.ylabel('Salary')
plt.show()
```

Output:-


Truth or Bluff (Random Forest Regression)
1e6
1.0
0.8
0.6
0.4
0.2
0.0
Salary
2
4
6
8
10
Position level

EXPERIMENT 8

Demonstrate Logistic Regression in Python with Sample Data Sets.

```
# importing libraries

import numpy as nm

import matplotlib.pyplot as mtp

import pandas as pd

#importing datasets

data_set= pd.read_csv('suv_data.csv')

data_set.head()

#Extracting Independent  Variables

x= data_set.iloc[:, [2,3]].values

x

#Extracting dependent Variables

y= data_set.iloc[:, 4].values

y

# Splitting the dataset into training and test set.

from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test= train_test_split(x, y, test_size= 0.25, random_state=0)

#feature Scaling

from sklearn.preprocessing import StandardScaler

st_x= StandardScaler()

x_train= st_x.fit_transform(x_train)

x_test= st_x.transform(x_test)
```

#Fitting Logistic Regression to the training set

```
from sklearn.linear_model import LogisticRegression
classifier= LogisticRegression(random_state=0)
classifier.fit(x_train, y_train)
```

#Predicting the test set result

```
y_pred= classifier.predict(x_test)
y_pred
```

#Creating the Confusion matrix

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

#Visualizing the training set result

```
from matplotlib.colors import ListedColormap
x_set, y_set = x_train, y_train
x1, x2 = nm.meshgrid(nm.arange(start =
                x_set[:, 0].min()
 - 1, stop = x_set[:, 0].max() + 1, step  =0.01),
nm.arange(start = x_set[:, 1].min() - 1,
        stop = x_set[:, 1].max() + 1, step = 0.01))
mtp.contourf(x1, x2, classifier.predict(nm.array([x1.ravel(), x2.ravel()]).T).reshape(x1.shap
e),
alpha = 0.75, cmap = ListedColormap(('purple','green' )))
mtp.xlim(x1.min(), x1.max())
```

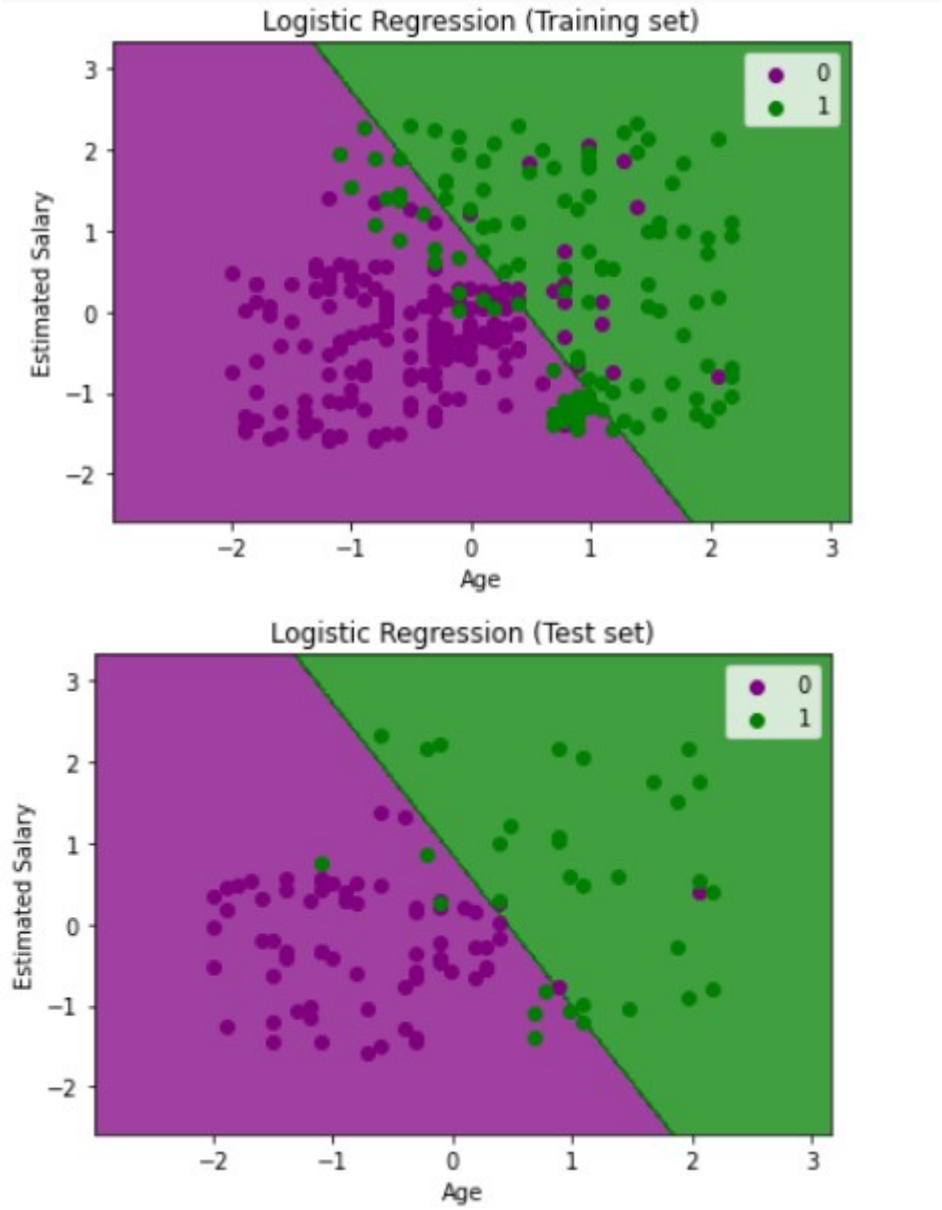
```
mtp.ylim(x2.min(), x2.max())
for i, j in enumerate(nm.unique(y_set)):
   mtp.scatter(x_set[y_set == j, 0], x_set[y_set == j, 1],
      c = ListedColormap(('purple', 'green'))(i), label = j)
mtp.title('Logistic Regression (Training set)')
mtp.xlabel('Age')
mtp.ylabel('Estimated Salary')
mtp.legend()
mtp.show()
```

#Visulaizing the test set result

```
from matplotlib.colors import ListedColormap
x_set, y_set = x_test, y_test
x1, x2 = nm.meshgrid(nm.arange(start = x_set[:, 0].min() -
 1, stop = x_set[:, 0].max() + 1, step  =0.01),
nm.arange(start = x_set[:, 1].min() - 1, stop = x_set[:, 1].max() + 1, step = 0.01))
mtp.contourf(x1, x2, classifier.predict(nm.array([x1.ravel(), x2.ravel()]).T).reshape(x1.shap
e),
alpha = 0.75, cmap = ListedColormap(('purple','green' )))
mtp.xlim(x1.min(), x1.max())
mtp.ylim(x2.min(), x2.max())
for i, j in enumerate(nm.unique(y_set)):
   mtp.scatter(x_set[y_set == j, 0], x_set[y_set == j, 1],
      c = ListedColormap(('purple', 'green'))(i), label = j)
```

```
mtp.title('Logistic Regression (Test set)')
mtp.xlabel('Age')
mtp.ylabel('Estimated Salary')
mtp.legend()
mtp.show()
```

Output:-


Logistic Regression (Training set)
Estimated Salary
Age
0
1
Logistic Regression (Test set)
Estimated Salary
Age
0
1

EXPERIMENT 9

Demonstrate Support Vector Classification in Python with Sample Data Set

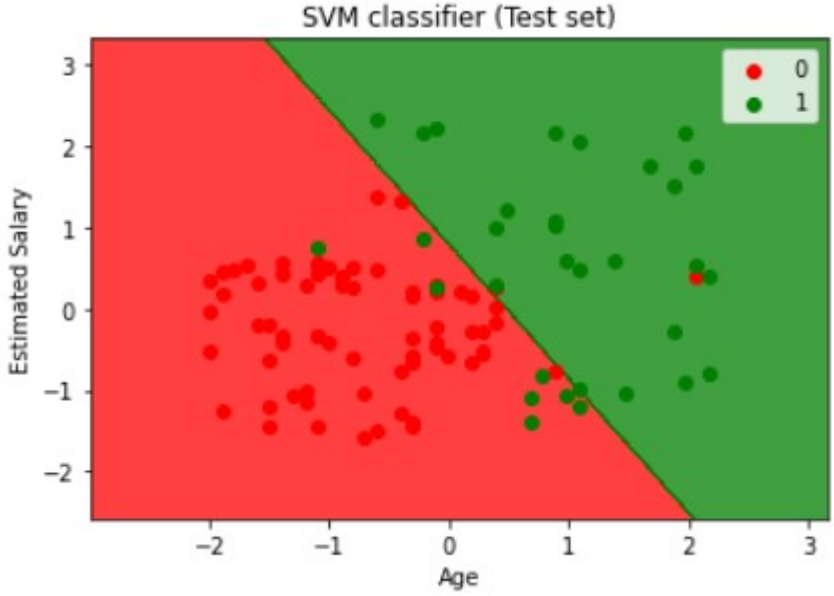
```
# importing libraries
import numpy as nm
import matplotlib.pyplot as mtp
import pandas as pd
#importing datasets
data_set= pd.read_csv('suv_data.csv')
#Extracting Independent and dependent Variable
x= data_set.iloc[:, [2,3]].values
y= data_set.iloc[:, 4].values
# Splitting the dataset into training and test set.
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test= train_test_split(x, y, test_size= 0.35, random_state=0)
#feature Scaling
from sklearn.preprocessing import StandardScaler
st_x= StandardScaler()
x_train= st_x.fit_transform(x_train)
x_test= st_x.transform(x_test)
#Fitting the SVM classifier to the training set
# import SVC class from Sklearn.svm library.
#SVC means Support vector classifier
#we have used kernel='linear'we are creating SVM for linearly separable data
from sklearn.svm import SVC
classifier = SVC(kernel='linear', random_state=0)
classifier.fit(x_train, y_train)
#Predicting the test set result
y_pred= classifier.predict(x_test)
y_pred
#Creating the Confusion matrix
```

```
from sklearn.metrics import confusion_matrix
cm= confusion_matrix(y_test, y_pred)
cm
```
#visualizing the training set result:
```
from matplotlib.colors import ListedColormap
x_set, y_set = x_train, y_train
x1, x2 = nm.meshgrid(nm.arange(start = x_set[:, 0].min() - 1, stop = x_set[:, 0].max() + 1,
step  =0.01),
nm.arange(start = x_set[:, 1].min() - 1, stop = x_set[:, 1].max() + 1, step = 0.01))
mtp.contourf(x1, x2, classifier.predict(nm.array([x1.ravel(),
x2.ravel()]).T).reshape(x1.shape),
alpha = 0.75, cmap = ListedColormap(('red', 'green')))
mtp.xlim(x1.min(), x1.max())
mtp.ylim(x2.min(), x2.max())
for i, j in enumerate(nm.unique(y_set)):
   mtp.scatter(x_set[y_set == j, 0], x_set[y_set == j, 1],
      c = ListedColormap(('red', 'green'))(i), label = j)
mtp.title('SVM classifier (Training set)')
mtp.xlabel('Age')
mtp.ylabel('Estimated Salary')
mtp.legend()
mtp.show()
```
#Visualizing the testing set result
```
from matplotlib.colors import ListedColormap
x_set, y_set = x_test, y_test
x1, x2 = nm.meshgrid(nm.arange(start = x_set[:, 0].min() - 1, stop = x_set[:, 0].max() + 1,
step  =0.01),
nm.arange(start = x_set[:, 1].min() - 1, stop = x_set[:, 1].max() + 1, step = 0.01))
mtp.contourf(x1, x2, classifier.predict(nm.array([x1.ravel(),
x2.ravel()]).T).reshape(x1.shape),
```

```
alpha = 0.75, cmap = ListedColormap(('red','green' )))
mtp.xlim(x1.min(), x1.max())
mtp.ylim(x2.min(), x2.max())
for i, j in enumerate(nm.unique(y_set)):
   mtp.scatter(x_set[y_set == j, 0], x_set[y_set == j, 1],
      c = ListedColormap(('red', 'green'))(i), label = j)
mtp.title('SVM classifier (Test set)')
mtp.xlabel('Age')
mtp.ylabel('Estimated Salary')
mtp.legend()
mtp.show()
```

Output:-


SVM classifier (Training set)
Estimated Salary
Age
0
1
3
2
1
0
−1
−2
−2
−1
0
1
2
3


SVM classifier (Test set)
Estimated Salary
Age
0
1
3
2
1
0
−1
−2
−2
−1
0
1
2
3

EXPERIMENT 10

Demonstrate Random Forest Classification in Python with Sample Data Set.

```
# importing libraries
import numpy as nm
import matplotlib.pyplot as mtp
import pandas as pd
#importing datasets
data_set= pd.read_csv('suv_data.csv')
data_set
#Extracting Independent and dependent Variable
x= data_set[['Age','EstimatedSalary']]
y= data_set['Purchased']
# Splitting the dataset into training and test set.
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test= train_test_split(x, y, test_size= 0.25, random_state=44)
#feature Scaling
from sklearn.preprocessing import StandardScaler
st_x= StandardScaler()
x_train= st_x.fit_transform(x_train)
x_test= st_x.transform(x_test)
#Fitting Random Forest  classifier to the training set
from sklearn.ensemble import RandomForestClassifier
classifier= RandomForestClassifier(n_estimators=
10,criterion="entropy",random_state=44)
classifier.fit(x_train, y_train)
#n_estimators=
# The required number of trees in the Random Forest.
#The default value is 10.
 # We can choose any number but need to take care of the overfitting issue.
#Predicting the test set result
```

```
y_pred= classifier.predict(x_test)
y_pred
#Creating Confusion matrix
from sklearn.metrics import confusion_matrix
cm= confusion_matrix(y_test,y_pred)
cm
from sklearn import metrics
import numpy as np
print('Mean Absolute Error:',
    metrics.mean_absolute_error(y_test, y_pred))
print('Mean Squared Error:',
    metrics.mean_squared_error(y_test, y_pred))
print('Root Mean Squared Error:',
  np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
#plotting tree
import matplotlib.pyplot as plt
from sklearn import tree
plt.figure(figsize=(15,15))
plt.rcParams.update({'font.size':10})
tree.plot_tree(classifier.estimators_[5],
 class_names=["0","1"],filled = True,fontsize=5)
plt.show()
# Model Accuracy
from sklearn import metrics
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
#comparing Real values and Actual values
import pandas as pd
dataset = pd.DataFrame({'Real Values':y_test,
                        'Predicted Values':y_pred})
dataset
```

#visualizing the training set result

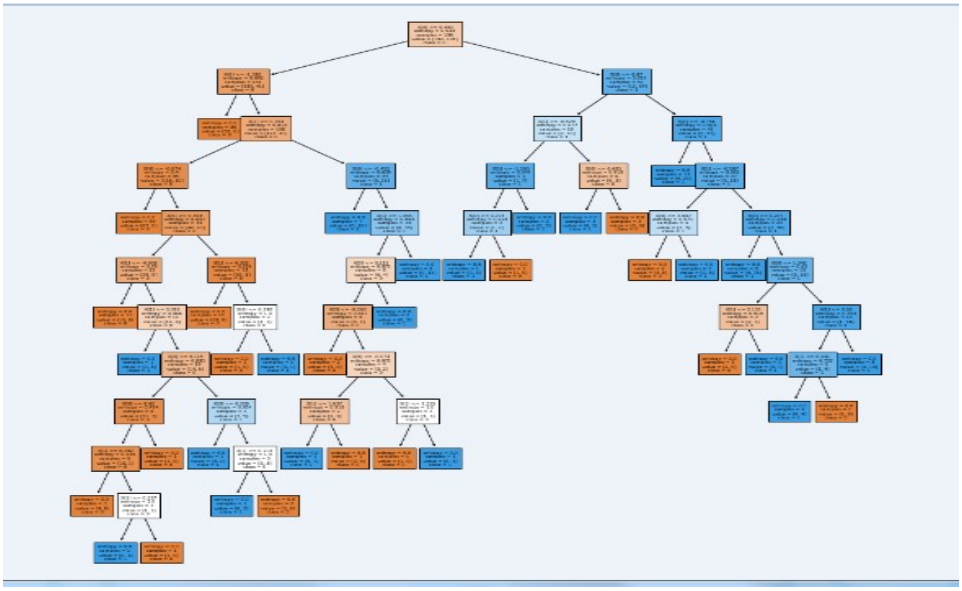
```
from matplotlib.colors import ListedColormap
x_set, y_set = x_train, y_train
x1, x2 = nm.meshgrid(nm.arange(start = x_set[:, 0].min() - 1, stop = x_set[:, 0].max() + 1,
step  =0.01),
nm.arange(start = x_set[:, 1].min() - 1, stop = x_set[:, 1].max() + 1, step = 0.01))
mtp.contourf(x1, x2, classifier.predict(nm.array([x1.ravel(),
x2.ravel()]).T).reshape(x1.shape),
alpha = 0.75, cmap = ListedColormap(('purple','green' )))
mtp.xlim(x1.min(), x1.max())
mtp.ylim(x2.min(), x2.max())
for i, j in enumerate(nm.unique(y_set)):
   mtp.scatter(x_set[y_set == j, 0], x_set[y_set == j, 1],
      c = ListedColormap(('purple', 'green'))(i), label = j)
mtp.title('Random Forest Algorithm (Training set)')
mtp.xlabel('Age')
mtp.ylabel('Estimated Salary')
mtp.legend()
mtp.show()
```

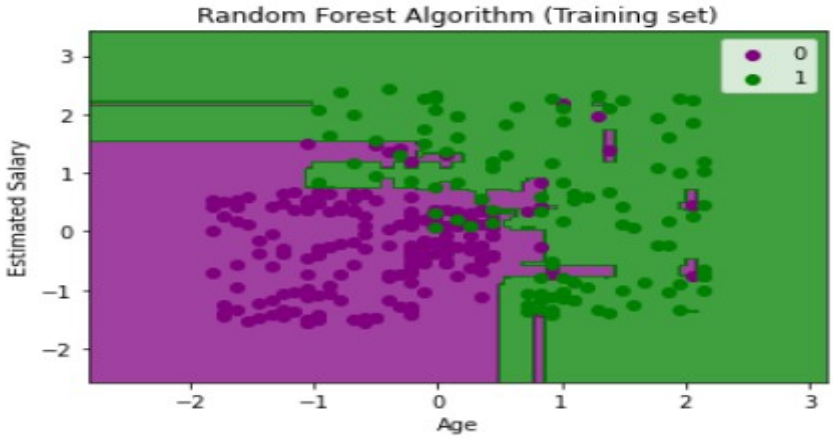
#visualizing the test set result

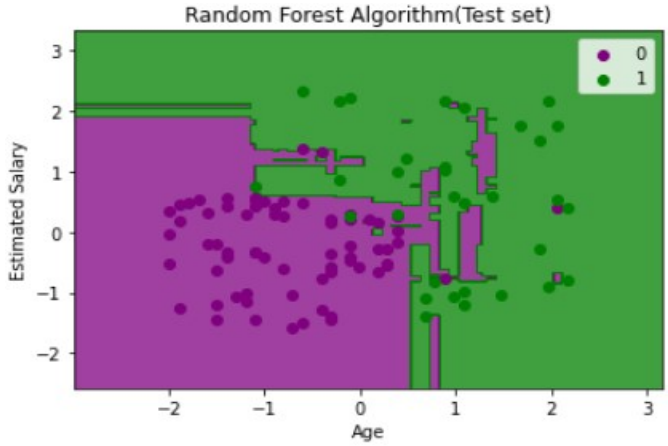
```
from matplotlib.colors import ListedColormap
x_set, y_set = x_test, y_test
x1, x2 = nm.meshgrid(nm.arange(start = x_set[:, 0].min() - 1, stop = x_set[:, 0].max() + 1,
step  =0.01),
nm.arange(start = x_set[:, 1].min() - 1, stop = x_set[:, 1].max() + 1, step = 0.01))
mtp.contourf(x1, x2, classifier.predict(nm.array([x1.ravel(),
x2.ravel()]).T).reshape(x1.shape),
alpha = 0.75, cmap = ListedColormap(('purple','green' )))
mtp.xlim(x1.min(), x1.max())
mtp.ylim(x2.min(), x2.max())
```

```
for i, j in enumerate(nm.unique(y_set)):
   mtp.scatter(x_set[y_set == j, 0], x_set[y_set == j, 1],
      c = ListedColormap(('purple', 'green'))(i), label = j)
mtp.title('Random Forest Algorithm(Test set)')
mtp.xlabel('Age')
mtp.ylabel('Estimated Salary')
mtp.legend()
mtp.show()
```

Output:-




Random Forest Algorithm (Training set)
Estimated Salary
Age
0
1


Random Forest Algorithm(Test set)
Estimated Salary
Age
0
1

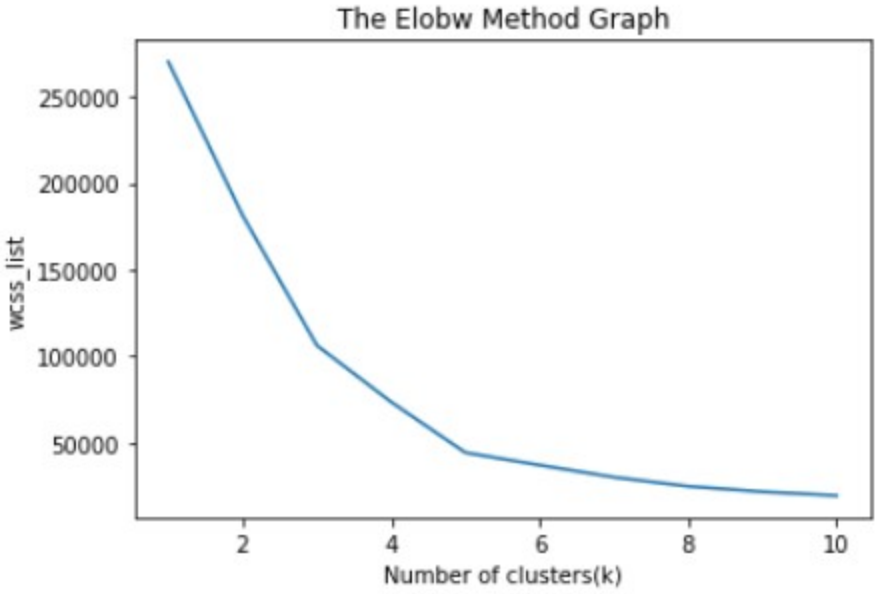
EXPERIMENT 11

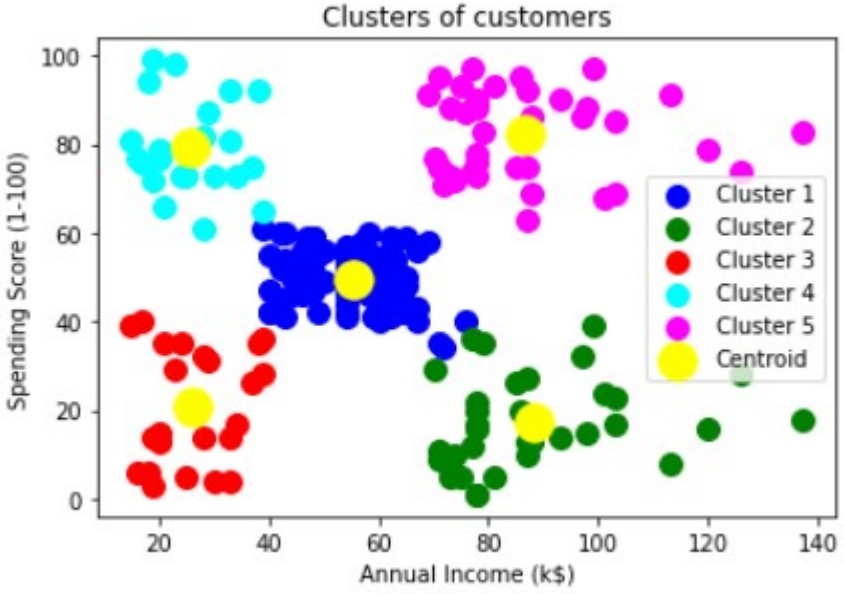
Demonstrate K-Means Clustering with Sample Data Set.

```
# Importing the dataset
import pandas as pd
import matplotlib.pyplot as mtp
dataset = pd.read_csv('Mall_Customers.csv')
#Extract independent variables
x = dataset.iloc[:, [3, 4]].values
#Finding optimal number of clusters using the elbow method
from sklearn.cluster import KMeans
wcss_list= []  #Initializing the list for the values of WCSS
#Using for loop for iterations from 1 to 10.
for i in range(1, 11):
   kmeans = KMeans(n_clusters=i, init='k-means++', random_state= 42)
   kmeans.fit(x)
   wcss_list.append(kmeans.inertia_)
mtp.plot(range(1, 11), wcss_list)
mtp.title('The Elobw Method Graph')
mtp.xlabel('Number of clusters(k)')
mtp.ylabel('wcss_list')
mtp.show()
#training the K-means model on a dataset
kmeans = KMeans(n_clusters=5, init='k-means++', random_state= 42)
y_predict= kmeans.fit_predict(x)
#visualizing the clusters
mtp.scatter(x[y_predict == 0, 0], x[y_predict == 0, 1], s = 100, c = 'blue', label = 'Cluster 1') #
for first cluster
mtp.scatter(x[y_predict == 1, 0], x[y_predict == 1, 1], s = 100, c = 'green', label = 'Cluster 2')
#for second cluster
```

```
mtp.scatter(x[y_predict== 2, 0], x[y_predict == 2, 1], s = 100, c = 'red', label = 'Cluster 3') #f
or third cluster
mtp.scatter(x[y_predict == 3, 0], x[y_predict == 3, 1], s = 100, c = 'cyan', label = 'Cluster 4') #
for fourth cluster
mtp.scatter(x[y_predict == 4, 0], x[y_predict == 4, 1], s = 100, c = 'magenta', label = 'Cluster
5') #for fifth cluster
mtp.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s = 300, c = 'yellow'
, label = 'Centroid')
mtp.title('Clusters of customers')
mtp.xlabel('Annual Income (k$)')
mtp.ylabel('Spending Score (1-100)')
mtp.legend()
mtp.show()
```

Output:-


The Elobw Method Graph
wcss_list
250000
200000
150000
100000
50000
2
4
6
8
10
Number of clusters(k)


Clusters of customers
Spending Score (1-100)
100
80
60
40
20
0
20
40
60
80
100
120
140
Annual Income (k$)
Cluster 1
Cluster 2
Cluster 3
Cluster 4
Cluster 5
Centroid

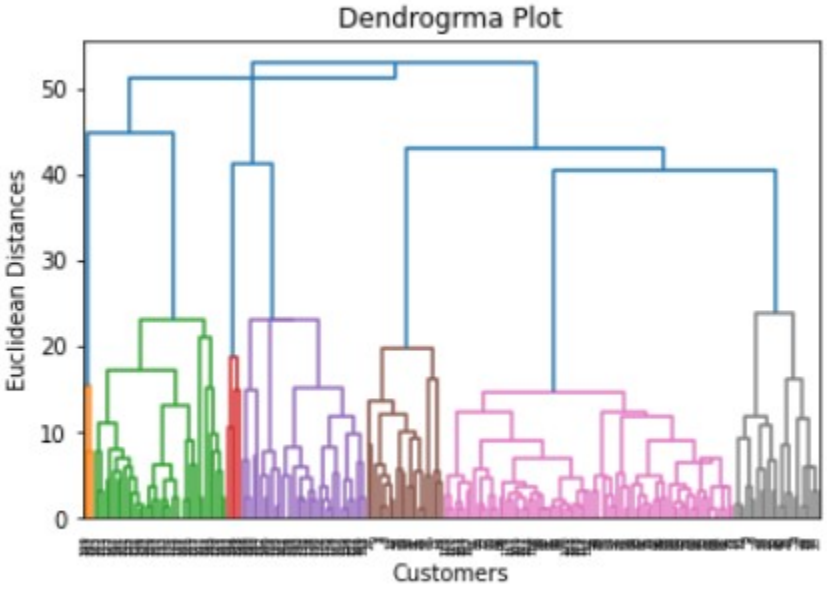
EXPERIMENT 12

Demonstrate Hierarchical Clustering with Sample Data Set.

```
# Importing the libraries
import numpy as nm
import matplotlib.pyplot as mtp
import pandas as pd
# Importing the dataset
dataset = pd.read_csv('Mall_Customers.csv')
dataset
x = dataset.iloc[:, [3, 4]].values
x
#Finding the optimal number of clusters using the dendrogram
import scipy.cluster.hierarchy as shc
dendro = shc.dendrogram(shc.linkage(x,
                method="centroid"))
mtp.title("Dendrogrma Plot")
mtp.ylabel("Euclidean Distances")
mtp.xlabel("Customers")
mtp.show()
#training the hierarchical model on dataset
from sklearn.cluster import AgglomerativeClustering
hc= AgglomerativeClustering(n_clusters=5,
          affinity='euclidean', linkage='complete')
y_pred= hc.fit_predict(x)
y_pred
#visualizing the clusters
mtp.scatter(x[y_pred == 0, 0], x[y_pred == 0, 1],
        s = 100, c = 'blue', label = 'Cluster 1')
mtp.scatter(x[y_pred == 1, 0], x[y_pred == 1, 1], s = 100,
        c = 'green', label = 'Cluster 2')
```

```
mtp.scatter(x[y_pred== 2, 0], x[y_pred == 2, 1], s = 100,
        c = 'red', label = 'Cluster 3')
mtp.scatter(x[y_pred == 3, 0], x[y_pred == 3, 1], s = 100,
        c = 'cyan', label = 'Cluster 4')
mtp.scatter(x[y_pred == 4, 0], x[y_pred == 4, 1], s = 100,
        c = 'magenta', label = 'Cluster 5')
mtp.title('Clusters of customers')
mtp.xlabel('Annual Income (k$)')
mtp.ylabel('Spending Score (1-100)')
mtp.legend()
mtp.show()
```

Output:-


Dendrogrma Plot
Euclidean Distances
0
10
20
30
40
50
Customers


Clusters of customers
Spending Score (1-100)
0
20
40
60
80
100
20
40
60
80
100
120
140
Annual Income (k$)
Cluster 1
Cluster 2
Cluster 3
Cluster 4
Cluster 5

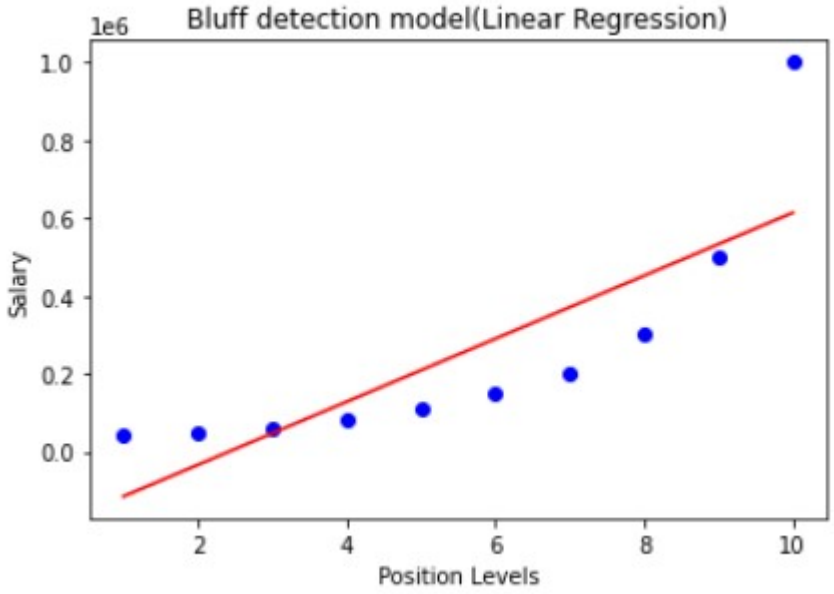
EXPERIMENT 13

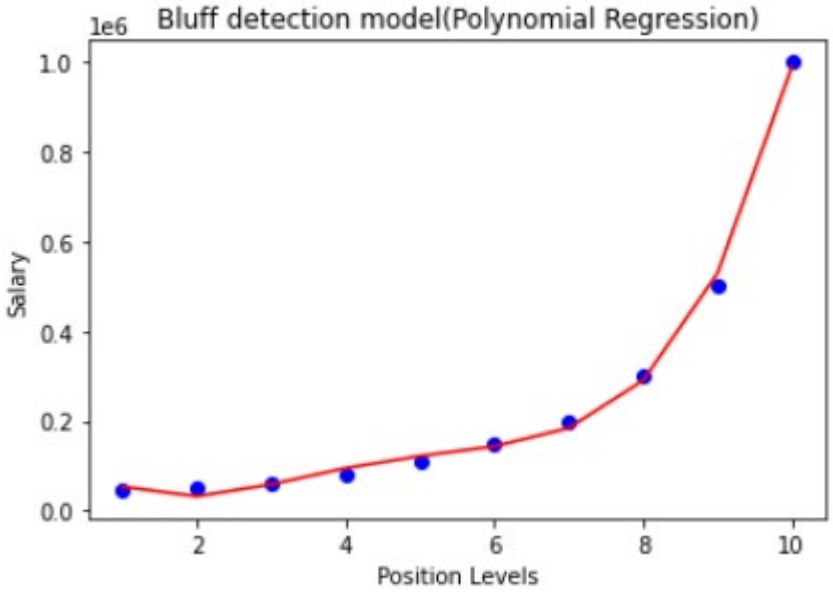
Demonstrate Polynomial Regression with Sample Data Set.

```
# importing libraries
import numpy as nm
import matplotlib.pyplot as mtp
import pandas as pd
#importing datasets
data_set= pd.read_csv('Position_Salaries.csv')
#Extracting Independent and dependent Variable
x= data_set.iloc[:, 1:2].values
y= data_set.iloc[:, 2].values
#Fitting the Linear Regression to the dataset
from sklearn.linear_model import LinearRegression
lin_regs= LinearRegression()
lin_regs.fit(x,y)
#Fitting the Polynomial regression to the dataset
from sklearn.preprocessing import PolynomialFeatures
poly_regs= PolynomialFeatures(degree= 4)
x_poly= poly_regs.fit_transform(x)
lin_reg_2 =LinearRegression()
lin_reg_2.fit(x_poly, y)
#Visulaizing the result for Linear Regression model
mtp.scatter(x,y,color="blue")
mtp.plot(x,lin_regs.predict(x), color="red")
mtp.title("Bluff detection model(Linear Regression)")
mtp.xlabel("Position Levels")
mtp.ylabel("Salary")
mtp.show()
#Visulaizing the result for Polynomial Regression
mtp.scatter(x,y,color="blue")
```

```
mtp.plot(x, lin_reg_2.predict(poly_regs.fit_transform(x)), color="red")
mtp.title("Bluff detection model(Polynomial Regression)")
mtp.xlabel("Position Levels")
mtp.ylabel("Salary")
mtp.show()
```

Output:-


Bluff detection model(Linear Regression)
1e6
1.0
0.8
0.6
0.4
0.2
0.0
Salary
2
4
6
8
10
Position Levels


Bluff detection model(Polynomial Regression)
1e6
1.0
0.8
0.6
0.4
0.2
0.0
Salary
2
4
6
8
10
Position Levels

EXPERIMENT 14

Implement different Activation Functions in neural Network

1. Sigmoid Function:-

```
#declare Sigmoid Function
def sigmoid_array(x):
   return 1/(1 +numpy.exp(-x))
#Generating numbers from -10 to 10
import numpy
numbers =numpy.arange(-10,10)
sigmoids=sigmoid_array(numbers)
sigmoids
# plotting graph
import matplotlib.pyplot as plt
fig,ax=plt.subplots()
ax.plot(numbers,sigmoids)
ax.grid(True)
plt.show()
```

2. Tanh Function:-

```
# Declare tanh function
def tanh_array(x):
   return numpy.tanh(x)
#Generate  numbers from -10 to 10
import numpy
numbers= numpy.arange(-10,10)
tanh=tanh_array(numbers)
#generate Graph
import matplotlib.pyplot as plt
fig,ax=plt.subplots()
ax.plot(numbers,tanh)
```

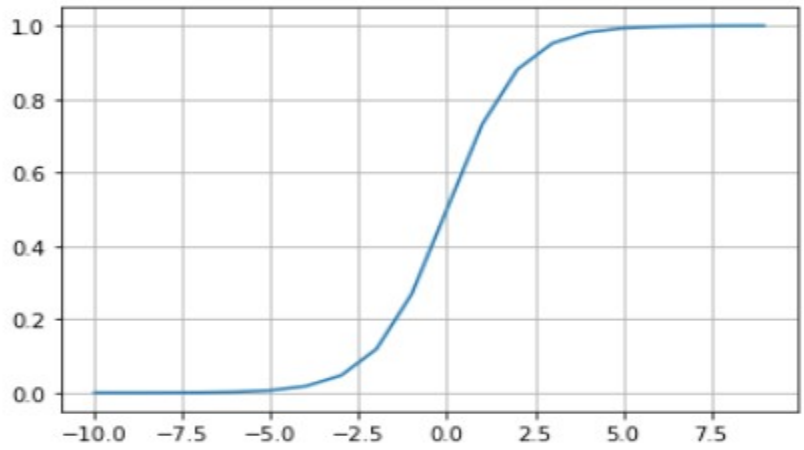
```
ax.grid(True)
plt.show()
```

3. Relu Function (Rectified Linear Unit):-

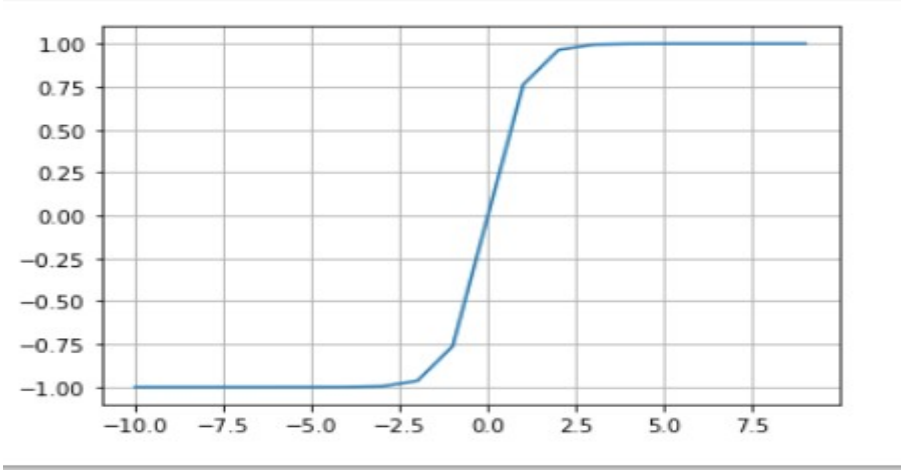
```
# Declare ReLu function
   def   relu_array(x):
   return numpy.maximum(0,x)
#Generate  numbers from -10 to 10
import numpy
numbers= numpy.arange(-10,10)
relu=relu_array(numbers)
#Generate graph
import matplotlib.pyplot as plt
fig,ax=plt.subplots()
ax.plot(numbers,relu)
ax.grid(True)
plt.show()
```

Output:-

Sigmoid Function



Tanh function:-



Relu Function:-