**Suryadatta College of Management Information Research & Technology (SCMIRT)**

**Machine learning**

**JOURNAL**

**Name: PRATEEK GANGURDE**

**CLASS: SY MSc (COMP. SCI.)**

Machine Learning Practical’s

1. **Write a python program to Prepare Scatter Plot (Use Forge Dataset / Iris Dataset)**

import pandas as pd

import matplotlib.pyplot as plt

from sklearn import preprocessing

iris = pd.read\_csv("iris.csv")

# Drop id column

iris = iris.drop('Id', axis=1)

# Convert Species columns in a numerical column of the iris dataframe

# creating labelEncoder

le = preprocessing.LabelEncoder()

# Converting string labels into numbers.

iris.Species = le.fit\_transform(iris.Species)

x = iris.iloc[:, :-1].values

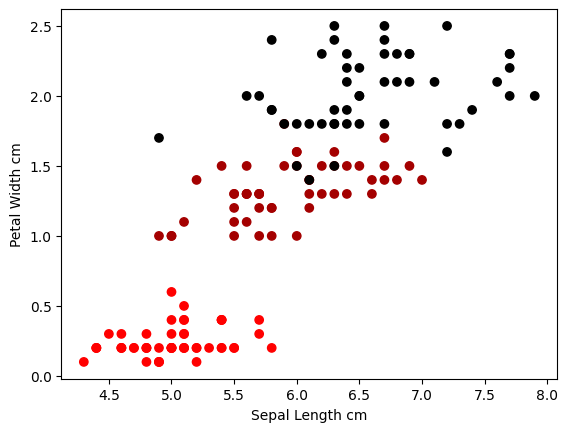
y = iris.iloc[:, 4].values

plt.scatter(x[:, 0], x[:, 3], c=y, cmap='flag')

plt.xlabel('Sepal Length cm')

plt.ylabel('Petal Width cm')

plt.show()



1. **Write a python program to find all null values in a given data set and remove them**.

import pandas as pd

# making data frame from csv file

data = pd.read\_csv("employees.csv")

# creating bool series True for NaN values

bool\_series = pd.isnull(data["Gender"])

# filtering data

# displaying data only with Gender = NaN

print(data[bool\_series],

data[10:25])

data["Gender"].fillna("No Gender", inplace=True)

**OUTPUT**

First Name Gender Start Date Last Login Time Salary Bonus % \

20 Lois NaN 4/22/1995 7:18 PM 64714 4.934

22 Joshua NaN 3/8/2012 1:58 AM 90816 18.816

27 Scott NaN 7/11/1991 6:58 PM 122367 5.218

31 Joyce NaN 2/20/2005 2:40 PM 88657 12.752

41 Christine NaN 6/28/2015 1:08 AM 66582 11.308

.. ... ... ... ... ... ...

961 Antonio NaN 6/18/1989 9:37 PM 103050 3.050

972 Victor NaN 7/28/2006 2:49 PM 76381 11.159

985 Stephen NaN 7/10/1983 8:10 PM 85668 1.909

989 Justin NaN 2/10/1991 4:58 PM 38344 3.794

995 Henry NaN 11/23/2014 6:09 AM 132483 16.655

Senior Management Team

20 True Legal

22 True Client Services

27 False Legal

31 False Product

41 True Business Development

.. ... ...

961 False Legal

972 True Sales

985 False Legal

989 False Legal

995 False Distribution

[145 rows x 8 columns] First Name Gender Start Date Last Login Time Salary Bonus % \

10 Louise Female 8/12/1980 9:01 AM 63241 15.132

11 Julie Female 10/26/1997 3:19 PM 102508 12.637

12 Brandon Male 12/1/1980 1:08 AM 112807 17.492

13 Gary Male 1/27/2008 11:40 PM 109831 5.831

14 Kimberly Female 1/14/1999 7:13 AM 41426 14.543

15 Lillian Female 6/5/2016 6:09 AM 59414 1.256

16 Jeremy Male 9/21/2010 5:56 AM 90370 7.369

17 Shawn Male 12/7/1986 7:45 PM 111737 6.414

18 Diana Female 10/23/1981 10:27 AM 132940 19.082

19 Donna Female 7/22/2010 3:48 AM 81014 1.894

20 Lois NaN 4/22/1995 7:18 PM 64714 4.934

21 Matthew Male 9/5/1995 2:12 AM 100612 13.645

22 Joshua NaN 3/8/2012 1:58 AM 90816 18.816

23 NaN Male 6/14/2012 4:19 PM 125792 5.042

24 John Male 7/1/1992 10:08 PM 97950 13.873

Senior Management Team

10 True NaN

11 True Legal

12 True Human Resources

13 False Sales

14 True Finance

15 False Product

16 False Human Resources

17 False Product

18 False Client Services

19 False Product

20 True Legal

21 False Marketing

22 True Client Services

23 NaN NaN

24 False Client Services

1. **Write a python program the Categorical values in numeric format for a given dataset.**

import pandas as pd

df = pd.read\_csv('data.csv')

# printing DataFrame

print(df)

df1 = pd.get\_dummies(df['Purchased'])

# using pd.concat to concatenate the dataframes

# df and df1 and storing the concatenated

# dataFrame in df.

df = pd.concat([df, df1], axis=1).reindex(df.index)

# removing the column 'Purchased' from df

# as it is of no use now.

df.drop('Purchased', axis=1, inplace=True)

# printing df

print(df)

**OUTPUT**

Country Age Salary Purchased

0 France 44.0 72000.0 No

1 Spain 27.0 48000.0 Yes

2 Germany 30.0 54000.0 No

3 Spain 38.0 61000.0 No

4 Germany 40.0 NaN Yes

5 France 35.0 58000.0 Yes

6 Spain NaN 52000.0 No

7 France 48.0 79000.0 Yes

8 Germany 50.0 83000.0 No

9 France 37.0 67000.0 Yes

Country Age Salary No Yes

0 France 44.0 72000.0 1 0

1 Spain 27.0 48000.0 0 1

2 Germany 30.0 54000.0 1 0

3 Spain 38.0 61000.0 1 0

4 Germany 40.0 NaN 0 1

5 France 35.0 58000.0 0 1

6 Spain NaN 52000.0 1 0

7 France 48.0 79000.0 0 1

8 Germany 50.0 83000.0 1 0

9 France 37.0 67000.0 0 1

1. **Write a python program to implement simple Linear Regression for predicting house price.**

import numpy as np

import matplotlib.pyplot as plt

def estimate\_coef(x, y):

# number of observations/points

n = np.size(x)

# mean of x and y vector

m\_x = np.mean(x)

m\_y = np.mean(y)

# calculating cross-deviation and deviation about x

ss\_xy = np.sum(y \* x) - n \* m\_y \* m\_x

ss\_xx = np.sum(x \* x) - n \* m\_x \* m\_x

# calculating regression coefficients

b\_1 = ss\_xy / ss\_xx

b\_0 = m\_y - b\_1 \* m\_x

return b\_0, b\_1

def plot\_regression\_line(x, y, b):

# plotting the actual points as scatter plot

plt.scatter(x, y, color="m",

marker="o", s=30)

# predicted response vector

y\_pared = b[0] + b[1] \* x

# plotting the regression line

plt.plot(x, y\_pared, color="g")

# putting labels

plt.xlabel('x')

plt.ylabel('y')

# function to show plot

plt.show()

def main():

# observations / data

x = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])

y = np.array([1, 3, 2, 5, 7, 8, 8, 9, 10, 12])

# estimating coefficients

b = estimate\_coef(x, y)

print('Estimated coefficients:\nb\_0 = {} \

\nb\_1 = {}'.format(b[0], b[1]))

# plotting regression line

plot\_regression\_line(x, y, b)

if \_\_name\_\_ == "\_\_main\_\_":

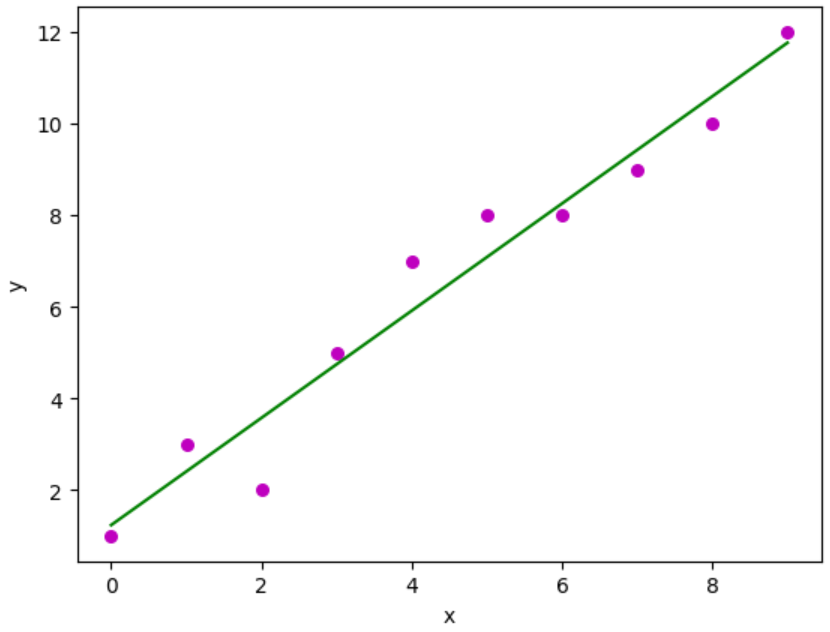
main()

**OUTPUT**

Estimated coefficients:

b\_0 = 1.2363636363636363

b\_1 = 1.1696969696969697



1. **Write a python program to implement multiple Linear Regression for a given dataset**

import pandas as pd

from sklearn import linear\_model

import statsmodels.api as sm

data = {'year': [2017,2017,2017,2017,2017,2017,2017,2017,2017,2017,2017,2017,2016,2016,2016,2016,2016,2016,2016,2016,2016,2016,2016,2016],

'month': [12,11,10,9,8,7,6,5,4,3,2,1,12,11,10,9,8,7,6,5,4,3,2,1],

'interest\_rate': [2.75,2.5,2.5,2.5,2.5,2.5,2.5,2.25,2.25,2.25,2,2,2,1.75,1.75,1.75,1.75,1.75,1.75,1.75,1.75,1.75,1.75,1.75],

'unemployment\_rate': [5.3,5.3,5.3,5.3,5.4,5.6,5.5,5.5,5.5,5.6,5.7,5.9,6,5.9,5.8,6.1,6.2,6.1,6.1,6.1,5.9,6.2,6.2,6.1],

'index\_price': [1464,1394,1357,1293,1256,1254,1234,1195,1159,1167,1130,1075,1047,965,943,958,971,949,884,866,876,822,704,719]

}

df = pd.DataFrame(data)

x = df[['interest\_rate','unemployment\_rate']]

y = df['index\_price']

# with sklearn

regr = linear\_model.LinearRegression()

regr.fit(x, y)

print('Intercept: \n', regr.intercept\_)

print('Coefficients: \n', regr.coef\_)

# with statsmodels

x = sm.add\_constant(x) # adding a constant

model = sm.OLS(y, x).fit()

predictions = model.predict(x)

print\_model = model.summary()

print(print\_model)

**OUTPUT**

Intercept:

1798.4039776258544

Coefficients:

[345.54008701 -250.14657137]

OLS Regression Results

================================================================

Dep. Variable: index\_price R-squared: 0.898

Model: OLS Adj. R-squared: 0.888

Method: Least Squares F-statistic: 92.07

Date: Mon, 26 Dec 2022 Prob (F-statistic): 4.04e-11

Time: 09:57:56 Log-Likelihood: -134.61

No. Observations: 24 AIC: 275.2

Df Residuals: 21 BIC: 278.8

Df Model: 2

Covariance Type: nonrobust

================================================================

coef std err t P>|t| [0.025 0.975]

-------------------------------------------------------------------------------------

const 1798.4040 899.248 2.000 0.059 -71.685 3668.493

interest\_rate 345.5401 111.367 3.103 0.005 113.940 577.140

unemployment\_rate -250.1466 117.950 -2.121 0.046 -495.437 -4.856

================================================================

Omnibus: 2.691 Durbin-Watson: 0.530

Prob(Omnibus): 0.260 Jarque-Bera (JB): 1.551

Skew: -0.612 Prob(JB): 0.461

Kurtosis: 3.226 Cond. No. 394.

1. **Write a python program to implement Polynomial Regression for given dataset**

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

# Importing the dataset

dataset = pd.read\_csv('position\_salaries.csv')

X = dataset.iloc[:, 1:2].values

y = dataset.iloc[:, 2].values

# 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=0.2, random\_state=0)

# Fitting Linear Regression to the dataset

from sklearn.linear\_model import LinearRegression

lin\_reg = LinearRegression()

lin\_reg.fit(X, y)

# Visualizing the Linear Regression results

def viz\_linear():

plt.scatter(X, y, color='red')

plt.plot(X, lin\_reg.predict(X), color='blue')

plt.title('Truth or Bluff (Linear Regression)')

plt.xlabel('Position level')

plt.ylabel('Salary')

plt.show()

return

viz\_linear()

# Fitting Polynomial Regression to the dataset

from sklearn.preprocessing import PolynomialFeatures

poly\_reg = PolynomialFeatures(degree=4)

X\_poly = poly\_reg.fit\_transform(X)

pol\_reg = LinearRegression()

pol\_reg.fit(X\_poly, y)

# Visualizing the Polymonial Regression results

def viz\_polymonial():

plt.scatter(X, y, color='red')

plt.plot(X, pol\_reg.predict(poly\_reg.fit\_transform(X)), color='blue')

plt.title('Truth or Bluff (Linear Regression)')

plt.xlabel('Position level')

plt.ylabel('Salary')

plt.show()

return

viz\_polymonial()

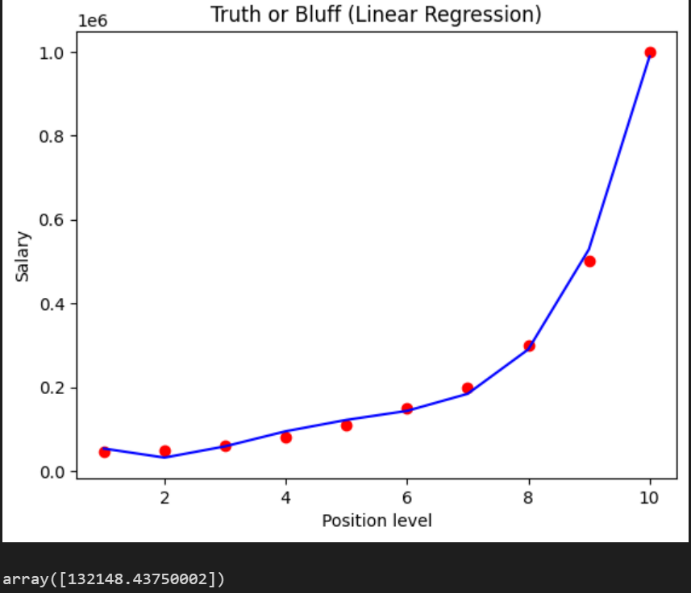
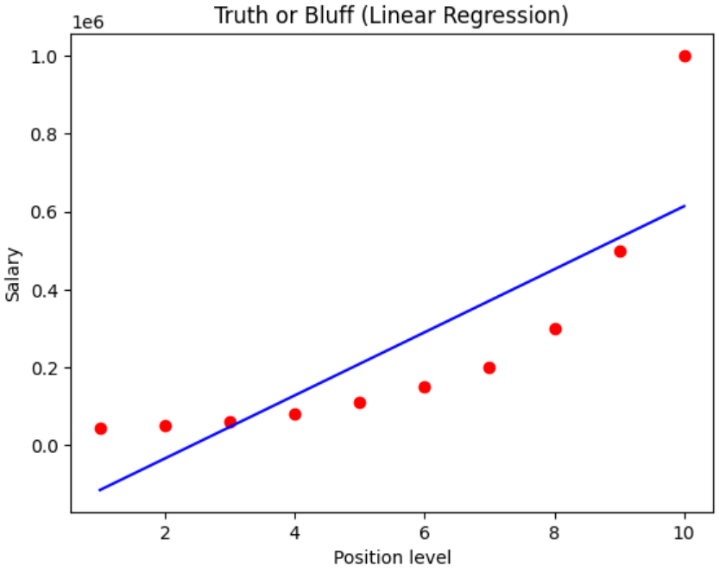
# Predicting a new result with Linear Regression

lin\_reg.predict([[5.5]])

# Predicting a new result with Polymonial Regression

pol\_reg.predict(poly\_reg.fit\_transform([[5.5]]))

**OUTPUT**



1. **Write a python program to Implement Naïve Bayes**

import pandas as pd

from sklearn.metrics import confusion\_matrix

from sklearn.model\_selection import train\_test\_split

from sklearn.naive\_bayes import GaussianNB

from sklearn.preprocessing import StandardScaler

# Importing the dataset

dataset = pd.read\_csv('user\_data.csv')

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

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

# Splitting the dataset into the Training set and Test set

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size=0.25, random\_state=0)

# Feature Scaling

sc = StandardScaler()

x\_train = sc.fit\_transform(x\_train)

x\_test = sc.transform(x\_test)

# Fitting Naive Bayes to the Training set

classifier = GaussianNB()

classifier.fit(x\_train, y\_train)

# Predicting the Test set results

y\_pred = classifier.predict(x\_test)

# Making the Confusion Matrix

cm = confusion\_matrix(y\_test, y\_pred)

print(cm)

**OUTPUT**

[[65 3]

[ 7 25]]

1. **Write a python program to Implement Decision Tree whether or not to play tennis**

import numpy as np

import pandas as pd

PlayTennis = pd.read\_csv("PlayTennis.csv")

from sklearn.preprocessing import LabelEncoder

Le = LabelEncoder()

PlayTennis['Outlook'] = Le.fit\_transform(PlayTennis['Outlook'])

PlayTennis['Temperature'] = Le.fit\_transform(PlayTennis['Temperature'])

PlayTennis['Humidity'] = Le.fit\_transform(PlayTennis['Humidity'])

PlayTennis['Wind'] = Le.fit\_transform(PlayTennis['Wind'])

PlayTennis['Play Tennis'] = Le.fit\_transform(PlayTennis['Play Tennis'])

y = PlayTennis['Play Tennis']

X = PlayTennis.drop(['Play Tennis'],axis=1)

# Fitting the model

from sklearn import tree

clf = tree.DecisionTreeClassifier(criterion = 'entropy')

clf = clf.fit(X, y)

# We can visualize the tree using tree.plot\_tree

tree.plot\_tree(clf)

import graphviz

dot\_data = tree.export\_graphviz(clf, out\_file=None)

graph = graphviz.Source(dot\_data)

graph

# The predictions are stored in X\_pred

X\_pred = clf.predict(X)

# verifying if the model has predicted it all right.

X\_pred == y

**OUTPUT**

0 True

1 True

2 True

3 True

4 True

5 True

6 True

7 True

8 True

9 True

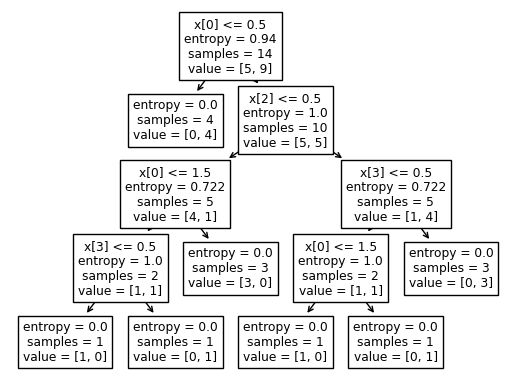
10 True

11 True

12 True

13 True

Name: Play Tennis, dtype: bool



1. **Write a python program to implement linear SVM.**

import numpy as nm

import matplotlib.pyplot as mtp

import pandas as pd

data\_set=pd.read\_csv('suv\_data.csv')

x=data\_set.iloc[:, [2,3]].values

y=data\_set.iloc[:, 4].values

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)

from sklearn.preprocessing import StandardScaler

st\_x=StandardScaler()

X\_train= st\_x.fit\_transform(X\_train)

X\_test= st\_x.transform(X\_test)

from sklearn.svm import SVC

classifier = SVC(kernel='linear',random\_state=0)

classifier.fit(X\_train, y\_train)

SVC(C=1.0, cache\_size=200, class\_weight=None, coef0=0.0, decision\_function\_shape='ovr', degree = 3, gamma = 'auto\_deprecated', kernel='linear', max\_iter = -1, probability=False, random\_state=0,shrinking=True, tol=0.001, verbose=False)

y\_pred = classifier.predict(X\_test)

from sklearn.metrics import confusion\_matrix

cm = confusion\_matrix(y\_test, y\_pred)

print(cm)

from sklearn.metrics import accuracy\_score

print("Accuracy=",accuracy\_score(y\_test, y\_pred))

#visualizing the training set results

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, cmp = ListedColormap(('white','gray')))

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 test set results

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, cmp = ListedColormap(('white','gray')))

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)')

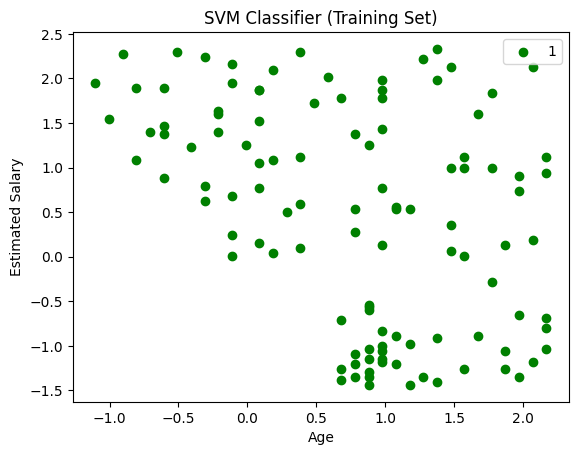
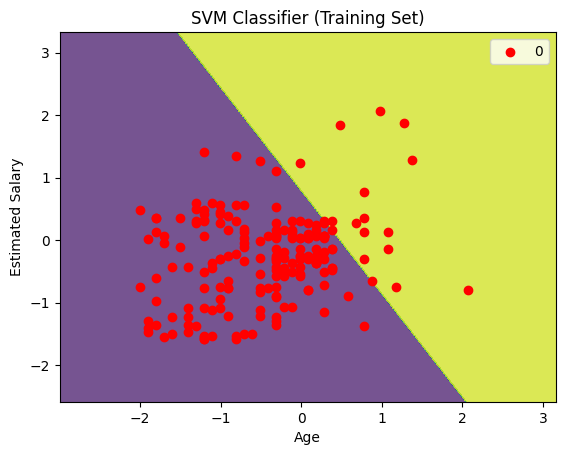
mtp.xlabel('Age')

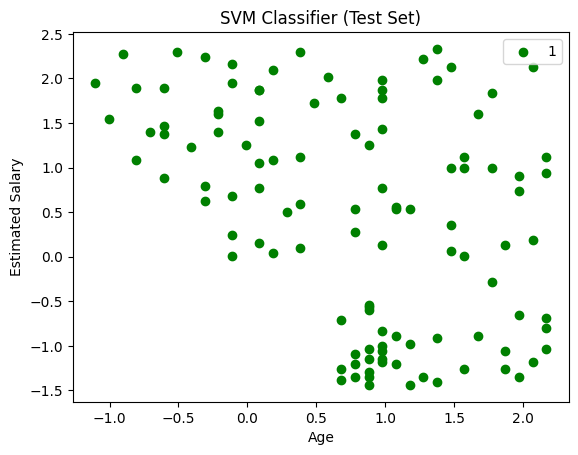
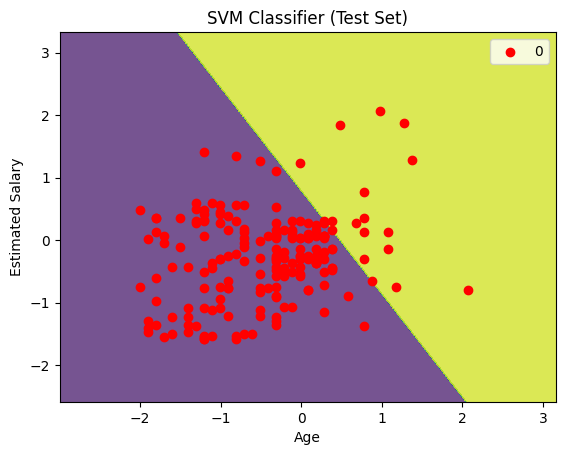
mtp.ylabel('Estimated Salary')

mtp.legend()

mtp.show()

**OUTPUT**





**11 Write a python program to transform data with Principal Component Analysis (PCA)**

# importing required libraries

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

# importing or loading the dataset

dataset = pd.read\_csv('wine.csv')

# distributing the dataset into two components X and Y

X = dataset.iloc[:, 0:13].values

y = dataset.iloc[:, 13].values

# Splitting the X and Y into the

# Training set and Testing set

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.2, random\_state = 0)

# performing preprocessing part

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

X\_train = sc.fit\_transform(X\_train)

X\_test = sc.transform(X\_test)

# Applying PCA function on training

# and testing set of X component

from sklearn.decomposition import PCA

pca = PCA(n\_components = 2)

X\_train = pca.fit\_transform(X\_train)

X\_test = pca.transform(X\_test)

explained\_variance = pca.explained\_variance\_ratio\_

# 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 using

# predict function under LogisticRegression

y\_pred = classifier.predict(X\_test)

# making confusion matrix between

# test set of Y and predicted value.

from sklearn.metrics import confusion\_matrix

cm = confusion\_matrix(y\_test, y\_pred)

# Predicting the training set

# result through scatter plot

from matplotlib.colors import ListedColormap

X\_set, y\_set = X\_train, y\_train

X1, X2 = np.meshgrid(np.arange(start = X\_set[:, 0].min() - 1,

stop = X\_set[:, 0].max() + 1, step = 0.01),

np.arange(start = X\_set[:, 1].min() - 1,

stop = X\_set[:, 1].max() + 1, step = 0.01))

plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(),

X2.ravel()]).T).reshape(X1.shape), alpha = 0.75,

cmap = ListedColormap(('yellow', 'white', 'aquamarine')))

plt.xlim(X1.min(), X1.max())

plt.ylim(X2.min(), X2.max())

for i, j in enumerate(np.unique(y\_set)):

plt.scatter(X\_set[y\_set == j, 0], X\_set[y\_set == j, 1],

c = ListedColormap(('red', 'green', 'blue'))(i), label = j)

plt.title('Logistic Regression (Training set)')

plt.xlabel('PC1') # for Xlabel

plt.ylabel('PC2') # for Ylabel

plt.legend() # to show legend

# show scatter plot

plt.show()

# Visualising the Test set results through scatter plot

from matplotlib.colors import ListedColormap

X\_set, y\_set = X\_test, y\_test

X1, X2 = np.meshgrid(np.arange(start = X\_set[:, 0].min() - 1,

stop = X\_set[:, 0].max() + 1, step = 0.01),

np.arange(start = X\_set[:, 1].min() - 1,

stop = X\_set[:, 1].max() + 1, step = 0.01))

plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(),

X2.ravel()]).T).reshape(X1.shape), alpha = 0.75,

cmap = ListedColormap(('yellow', 'white', 'aquamarine')))

plt.xlim(X1.min(), X1.max())

plt.ylim(X2.min(), X2.max())

for i, j in enumerate(np.unique(y\_set)):

plt.scatter(X\_set[y\_set == j, 0], X\_set[y\_set == j, 1],

c = ListedColormap(('red', 'green', 'blue'))(i), label = j)

# title for scatter plot

plt.title('Logistic Regression (Test set)')

plt.xlabel('PC1') # for Xlabel

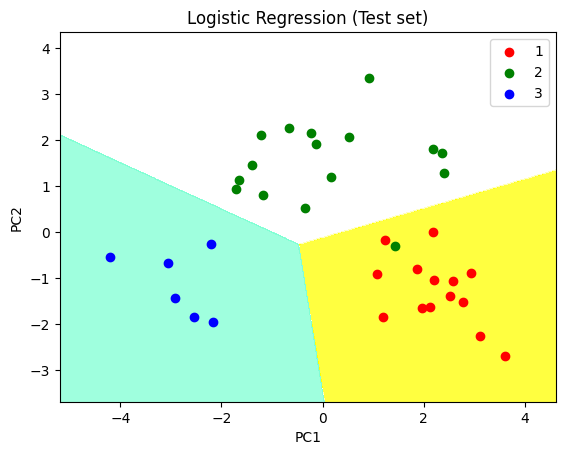
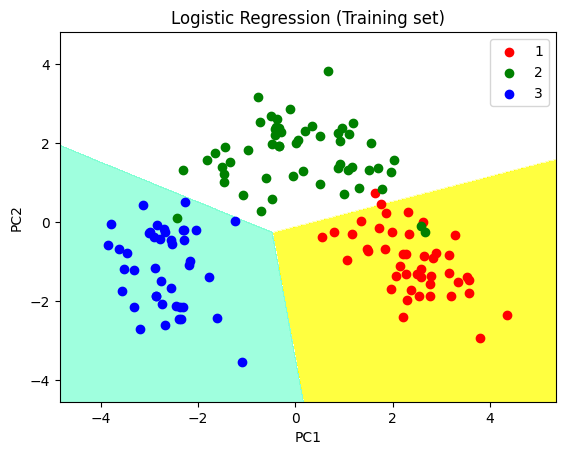
plt.ylabel('PC2') # for Ylabel

plt.legend()

# show scatter plot

plt.show()

**OUTPUT**



**12 Write a python program to implement k-nearest Neighbors ML algorithm to build prediction model (Use Forge Dataset)**

# importing libraries

import numpy as nm

import matplotlib.pyplot as mtp

import pandas as pd

from sklearn.neighbors import KNeighborsClassifier

from sklearn.preprocessing import StandardScaler

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import confusion\_matrix

from matplotlib.colors import ListedColormap

# importing datasets

data\_set = pd.read\_csv('user\_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.

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size=0.25, random\_state=0)

# feature Scaling

st\_x = StandardScaler()

x\_train = st\_x.fit\_transform(x\_train)

x\_test = st\_x.transform(x\_test)

# Fitting K-NN classifier to the training set

classifier = KNeighborsClassifier(n\_neighbors=5, metric='minkowski', p=2)

classifier.fit(x\_train, y\_train)

# Predicting the test set result

y\_pred = classifier.predict(x\_test)

# Creating the Confusion matrix

cm = confusion\_matrix(y\_test, y\_pred)

# Visualizing the training set result

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('K-NN Algorithm (Training set)')

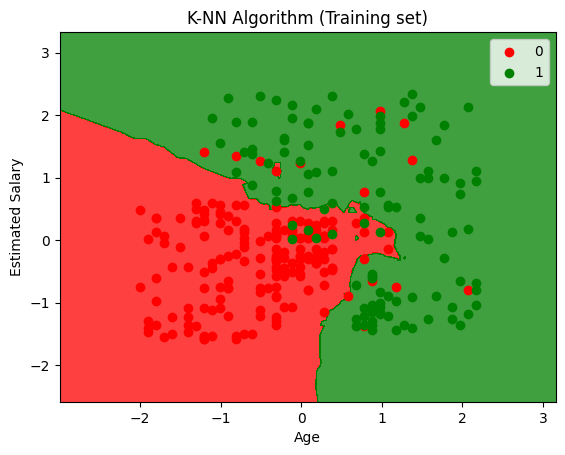
mtp.xlabel('Age')

mtp.ylabel('Estimated Salary')

mtp.legend()

mtp.show()

**OUTPUT**



**13 Write a python program to implement k-means algorithm on a synthetic dataset.**

import numpy as nm

import matplotlib.pyplot as mtp

import pandas as pd

from sklearn.cluster import KMeans

# Importing the dataset

dataset = pd.read\_csv('Mall\_Customers.csv')

x = dataset.iloc[:, [3, 4]].values

# finding optimal number of clusters using the elbow method

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 Elbow 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') # for 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')

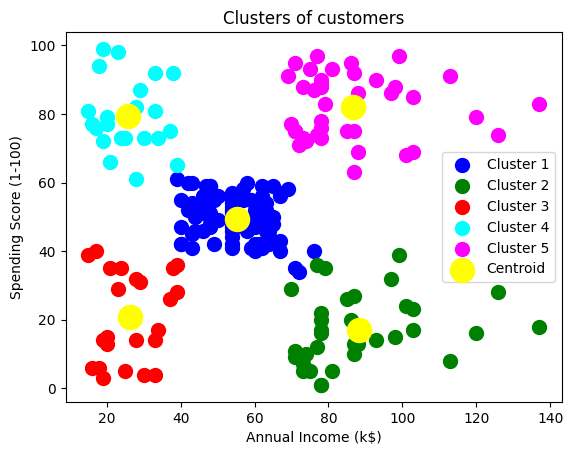
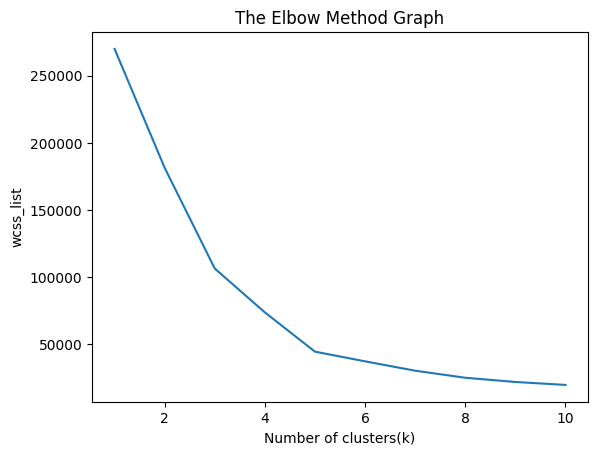
mtp.xlabel('Annual Income (k$)')

mtp.ylabel('Spending Score (1-100)')

mtp.legend()

mtp.show()

**OUTPUT**



**14 Write a python program to implement Agglomerative clustering on a synthetic dataset**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.cluster import KMeans

df = pd.read\_csv("Mall\_Customers.csv")

df.head()

df.shape

df.describe()

x =df.iloc[:, [3,4]].values

print(x[1:6])

import scipy.cluster.hierarchy as shc

dendro = shc.dendrogram(shc.linkage(x,method="ward"))

plt.title("Dendrogram Plot")

plt.ylabel("Euclidean Distance")

plt.xlabel("Data Points")

plt.show()

from sklearn.cluster import AgglomerativeClustering

hc = AgglomerativeClustering(n\_clusters=5, affinity='euclidean',linkage='ward')

y\_predict = hc.fit\_predict(x)

plt.scatter(x[y\_predict ==0,0], x[y\_predict ==0,1], s=100, c='blue',label='Cluster1')

plt.scatter(x[y\_predict ==1,0], x[y\_predict ==1,1], s=100, c='green',label='Cluster2')

plt.scatter(x[y\_predict ==2,0], x[y\_predict ==2,1], s=100, c='magenta',label='Cluster3')

plt.scatter(x[y\_predict ==3,0], x[y\_predict ==3,1], s=100, c='lightgreen',label='Cluster4')

plt.scatter(x[y\_predict ==4,0], x[y\_predict ==4,1], s=100, c='lightblue',label='Cluster5')

plt.title('Clusters of customers')

plt.xlabel('Annual Income')

plt.ylabel('Spending Score')

plt.legend()

plt.show()

**OUTPUT**

[[15 81]

[16 6]

[16 77]

[17 40]

[17 76]]

