**Practical 1:- Data Preprocessing In Machine Learning.**

To run the code install scikit learn library using following command:

pip install scikit-learn

Code:

from sklearn import preprocessing

label\_encoder = preprocessing.LabelEncoder()

input\_classes=['Suzuki','ford','Suzuki','toyota','ford','bmw']

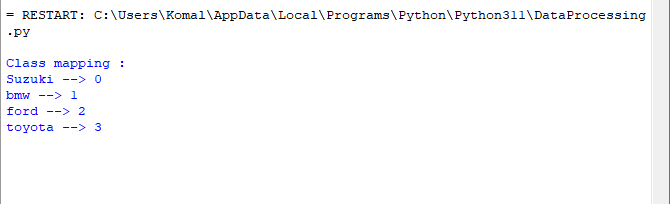
label\_encoder.fit(input\_classes)

print ("\nClass mapping :")

for i, item in enumerate(label\_encoder.classes\_):

print (item,'-->', i)

Output:



**Practical 2:- Implementing One Hot Encoder in Python.**

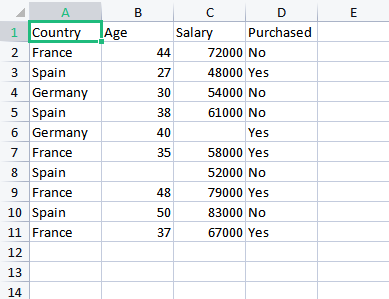
To run following code install libraries:

pip install numpy

pip install matplotlib

pip install pandas

Use following Data



Code:

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

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

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

y=dataset.iloc[:, -1].values

print(X)

print(y)

from sklearn.impute import SimpleImputer

imputer = SimpleImputer(missing\_values=np.nan,strategy='mean')

imputer.fit(X[:,1:3])

print(X)

from sklearn.compose import ColumnTransformer

from sklearn.preprocessing import OneHotEncoder

ct = ColumnTransformer(transformers=[('encoder',OneHotEncoder(),[0])],remainder='passthrough')

X = np.array(ct.fit\_transform(X))

print(X)

from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

y = le.fit\_transform(y)

print(y)

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=1)

print(X\_train)

print(X\_test)

print(y\_train)

print(y\_test)

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

X\_train[:, 3:] = sc.fit\_transform(X\_train[:, 3:])

X\_test[:, 3:] = sc.transform(X\_test[:, 3:])

print(X\_train)

print(X\_test)

Output:



**Practical 3:-Perform Linear Regression Algorithm on data.**

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

data\_set=pd.read\_csv('Salary\_Data.csv')

x=data\_set.iloc[:,:-1].values

y=data\_set.iloc[:,-1].values

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)

from sklearn.linear\_model import LinearRegression

regressor=LinearRegression()

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

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

plt.scatter(x\_train,y\_train,color='red')

plt.plot(x\_train,regressor.predict(x\_train),color='blue')

plt.title('Salary vs Experience(Training Set)')

plt.xlabel('year of Experience')

plt.ylabel('Salary')

plt.show()

plt.scatter(x\_test,y\_test,color='red')

plt.plot(x\_train,regressor.predict(x\_train),color='blue')

plt.title('Salary vs Experience(Test Set)')

plt.xlabel('year of Experience')

plt.ylabel('Salary')

plt.show()

**Output:**



**Practical 4: Python Implementation for Multiple Linear Regression.**

import numpy as nm

import matplotlib.pyplot as mtp

import pandas as pd

data\_set= pd.read\_csv('50\_comp.csv')

x= data\_set.iloc[:, :-1].values

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

from sklearn.preprocessing import LabelEncoder, OneHotEncoder

labelencoder\_x= LabelEncoder()

x[:, 3]= labelencoder\_x.fit\_transform(x[:,3])

onehotencoder= OneHotEncoder()

x= onehotencoder.fit\_transform(x).toarray()

x = x[:, 1:]

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.linear\_model import LinearRegression

regressor= LinearRegression()

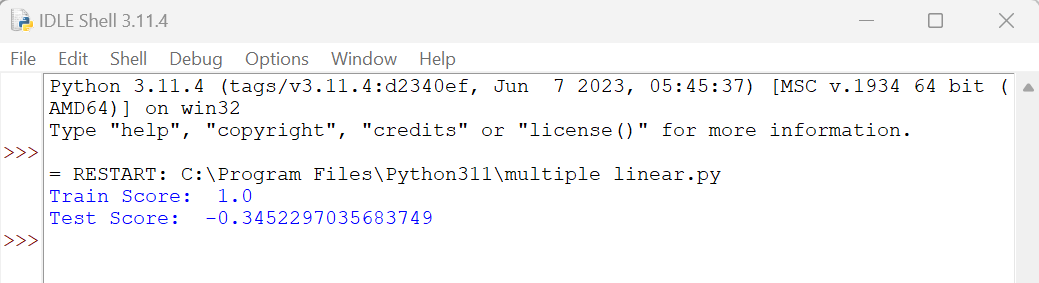
regressor.fit(x\_train, y\_train)

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

print('Train Score: ', regressor.score(x\_train, y\_train))

print('Test Score: ', regressor.score(x\_test, y\_test))

OUTPUT:



**Practical 5:-Perform Decision Tree Classification algorithm on data.**

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

dataset=pd.read\_csv('C:/Users/LAB2-PC34/Desktop/Social\_Network\_Ads.csv')

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

y=dataset.iloc[:,-1].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)

print(x\_train)

print(y\_train)

print(x\_test)

print(y\_test)

from sklearn.preprocessing import StandardScaler

sc=StandardScaler()

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

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

print(x\_train)

print(x\_test)

from sklearn.tree import DecisionTreeClassifier

classifier=DecisionTreeClassifier(criterion='entropy',random\_state=0)

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

print(classifier.predict(sc.transform([[30,87000]])))

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

print(np.concatenate((y\_pred.reshape(len(y\_pred),1),y\_test.reshape(len(y\_test),1)),1))

from sklearn.metrics import confusion\_matrix,accuracy\_score

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

print(cm)

accuracy\_score(y\_test,y\_pred)

from matplotlib.colors import ListedColormap

x\_set,y\_set=sc.inverse\_transform(x\_train),y\_train

x1,x2=np.meshgrid(np.arange(start=x\_set[:,0].min()-10,stop = x\_set[:,0].max()+ 10,step=0.25),

np.arange(start=x\_set[:,1].min()-1000,stop = x\_set[:,1].max()+1000,step=0.25))

plt.contourf(x1,x2,classifier.predict(sc.transform(np.array([x1.ravel(),x2.ravel()]).T)).reshape(x1.shape),

alpha =0.75,cmap=ListedColormap(('red','green')))

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'))(i),label=j)

plt.title('Decision Tree Classification (Training set)')

plt.xlabel('Age')

plt.ylabel('Estimated Salary')

plt.legend()

plt.show()

from matplotlib.colors import ListedColormap

x\_set,y\_set=sc.inverse\_transform(x\_test),y\_test

x1,x2=np.meshgrid(np.arange(start = x\_set[:,0].min()-10,stop=x\_set[:,0].max()+10,step=0.25),

np.arange(start = x\_set[:,1].min()-1000,stop=x\_set[:,1].max()+1000,step=0.25))

plt.contourf(x1,x2,classifier.predict(sc.transform(np.array([x1.ravel(),x2.ravel()]).T)).reshape(x1.shape),

alpha = 0.75,cmap=ListedColormap(('red','green')))

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'))(i),label=j)

plt.title('decision Tree Classification(Training set)')

plt.xlabel('Age')

plt.ylabel('Estimated salary')

plt.legend()

plt.show()

from matplotlib.colors import ListedColormap

X\_set,y\_set=sc.inverse\_transform(X\_test),y\_test

X1,X2=np.meshgrid(np.arange(start=X\_set[:,0].min()-10,stop=X\_set[:,0].max()+10,step=0.25),np.arrange(start=X\_set[:,1].min()-1000,stop=X\_set[:,1].max()+1000,step=0.25))

plt.contourf(X1,X2,classifier.predict(sc.tranform(np.array([X1.ravel(),X2.ravel()]).T)).reshape(X1.shape),

alpha=0.75,cmap=ListedColormap(('red','green')))

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

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

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

plt.scatter(X\_set[y\_set==j,0],X\_set[y\_set==j,1],c=ListedColormap(('red','green'))(i),label=j)

plt.title('Decision Tree Classification(test set)')

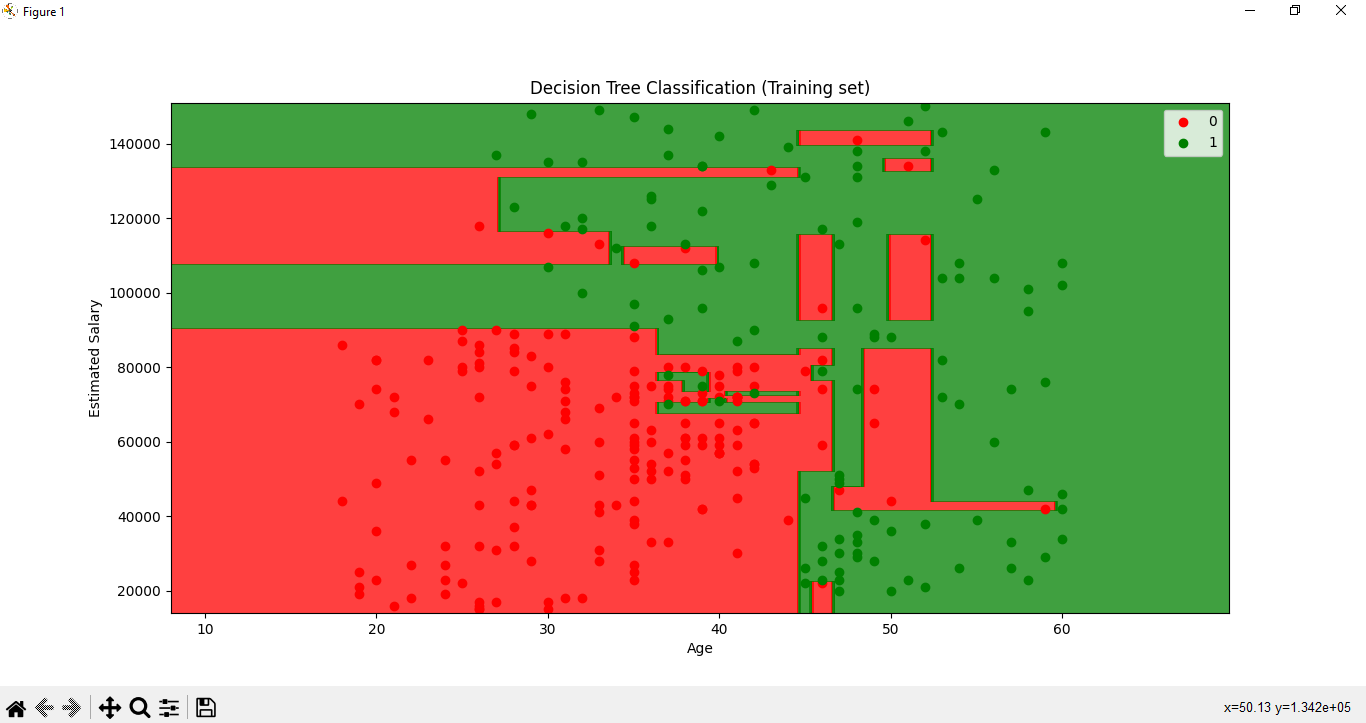
plt.xlabel('Age')

plt.ylabel('Estimated Salary')

plt.legend()

plt.show()

**OUTPUT:**



**Practical 6:Implementation of the Random Forest Algorithm of python**

import numpy as nm

import matplotlib.pyplot as mtp

import pandas as pd

#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.

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)

from sklearn.ensemble import RandomForestClassifier

classifier= RandomForestClassifier(n\_estimators= 10, criterion="entropy")

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

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

from sklearn.metrics import confusion\_matrix

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

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()

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)')

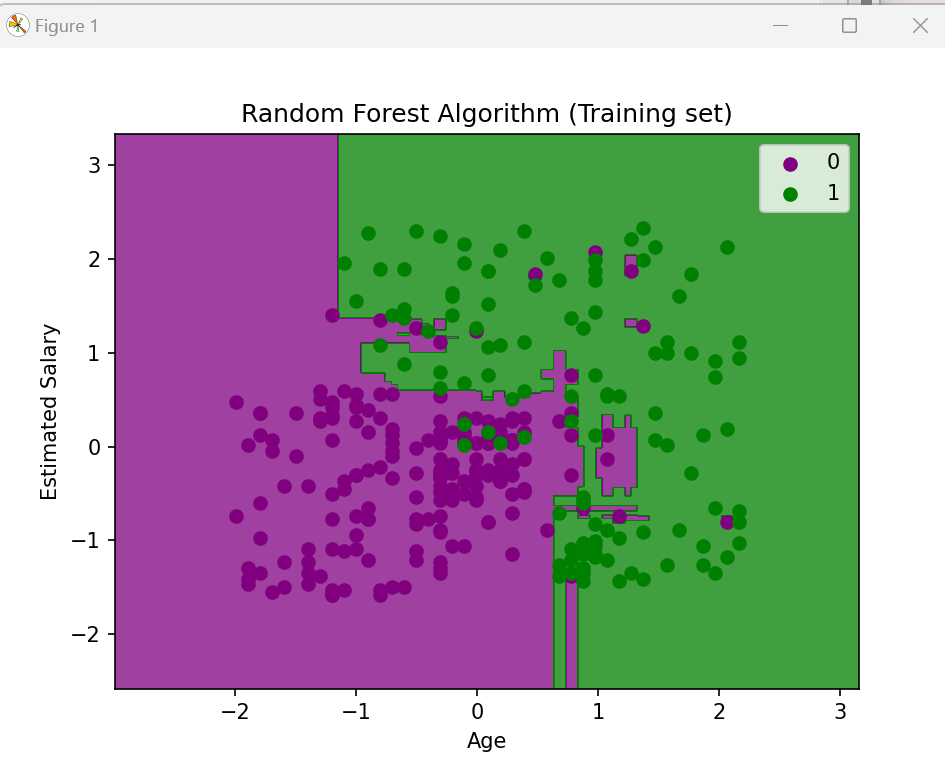
mtp.xlabel('Age')

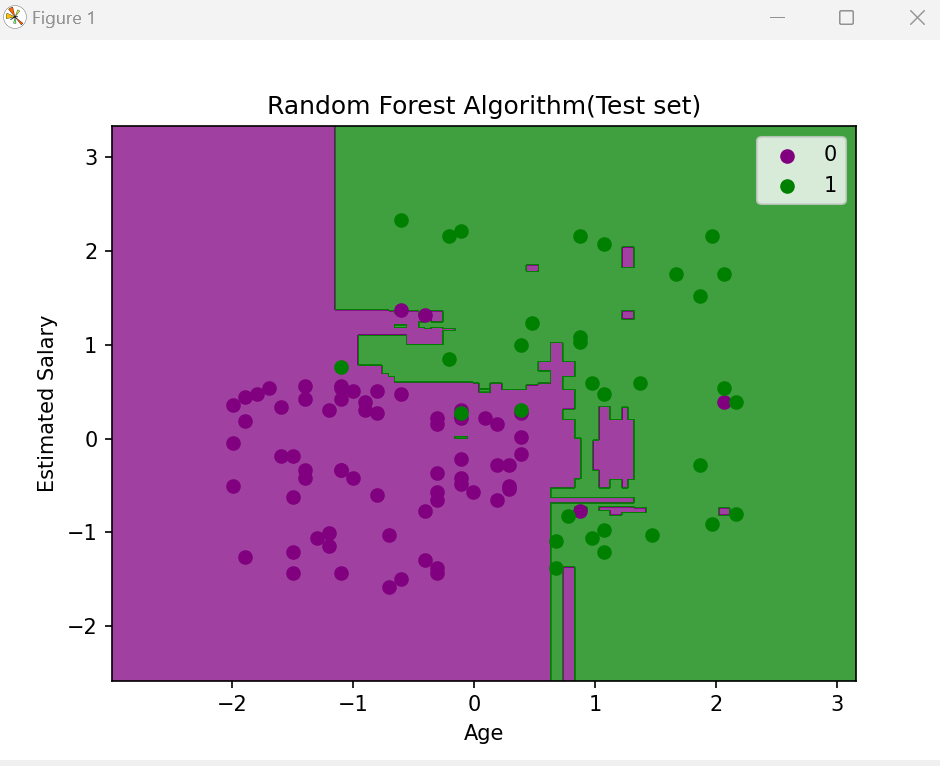
mtp.ylabel('Estimated Salary')

mtp.legend()

mtp.show()

OUTPUT:





**Practical 7-Perform K-Nearest Neighbor(KNN) Algorithm on user data.**

import numpy as nm

import matplotlib.pyplot as mtp

import pandas as pd

data\_set= pd.read\_csv('C:\Program Files\Python310/User\_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.neighbors import KNeighborsClassifier

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

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

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

from sklearn.metrics import confusion\_matrix

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

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],

color = ListedColormap(('red', 'green'))(i), label = j)

mtp.title('K-NN Algorithm (Training set)')

mtp.xlabel('Age')

mtp.ylabel('Estimated Salary')

mtp.legend()

mtp.show()

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],

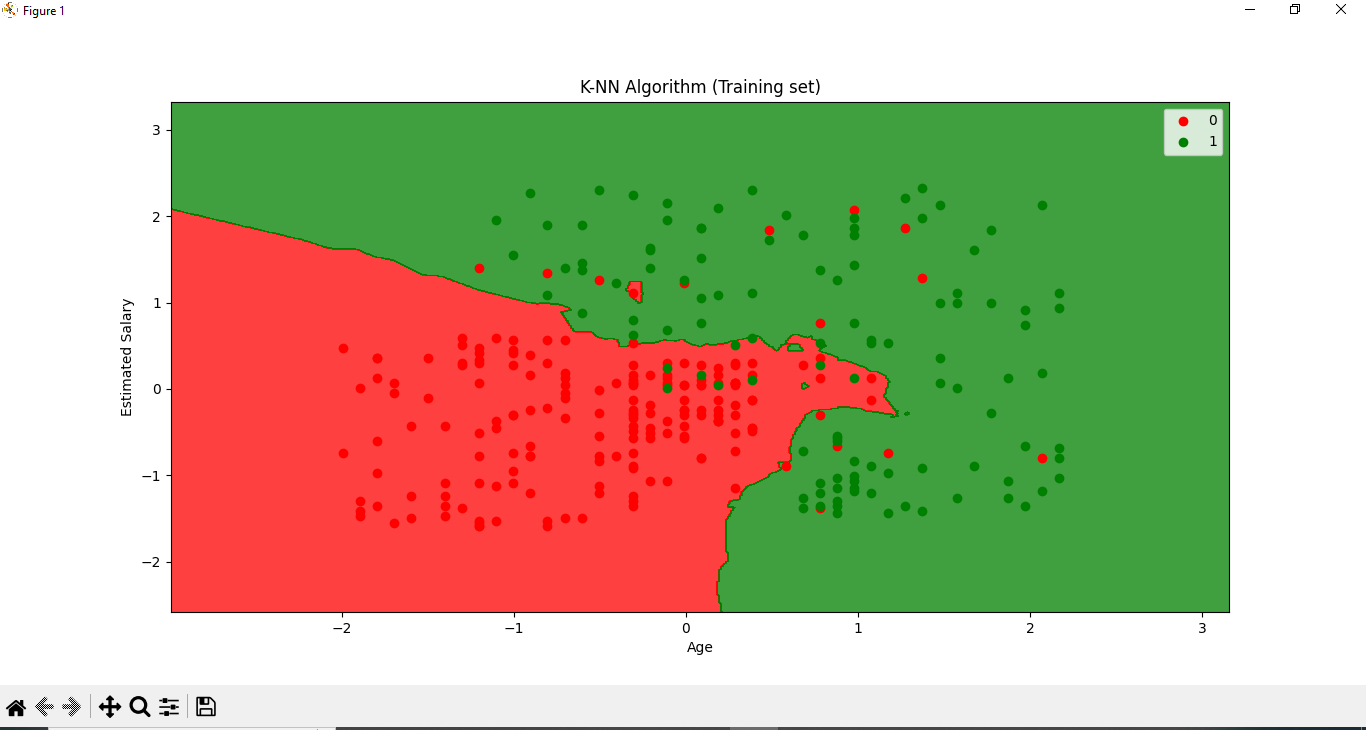
color = ListedColormap(('red', 'green'))(i), label = j)

mtp.title('K-NN algorithm(Test set)')

mtp.xlabel('Age')

mtp.ylabel('Estimated Salary')

mtp.legend()

mtp.show() 

**Practical 8:- Perform Naïve Bayes classifier algorithm on user data.**

import numpy as nm

import matplotlib.pyplot as mtp

import pandas as pd

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

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

y = dataset.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

sc = StandardScaler()

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

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

from sklearn.naive\_bayes import GaussianNB

classifier = GaussianNB()

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

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

from sklearn.metrics import confusion\_matrix

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

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],

color = ListedColormap(('purple', 'green'))(i), label = j)

mtp.title('Naive Bayes (Training set)')

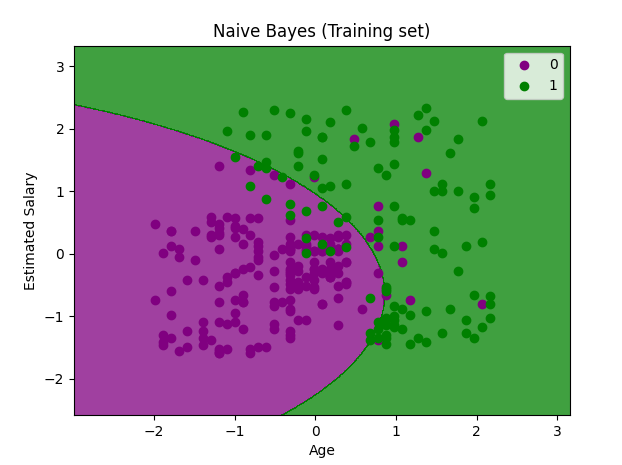
mtp.xlabel('Age')

mtp.ylabel('Estimated Salary')

mtp.legend()

mtp.show()

**Output:-**



**Practical 9:- Perform Support Vector Machine Learning algorithm on user data.**

import numpy as nm

import matplotlib.pyplot as mtp

import pandas as pd

#importing datasets

data\_set= pd.read\_csv('C:/Users/LAB2-PC04/Downloads/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.

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)

from sklearn.svm import SVC # "Support vector classifier"

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

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

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

#Creating the Confusion matrix

from sklearn.metrics import confusion\_matrix

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

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()

**Output:**



**Practical 10:- K-means clustering using machine learning.**

# importing libraries

import numpy as nm

import matplotlib.pyplot as mtp

import pandas as pd

# Importing the dataset

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

#extract dependent and independent variable

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

#Step-2: Finding the optimal number of clusters using the elbow method #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()

#Step- 3: Training the K-means algorithm on the training dataset

#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)

#Step-4: Visualizing the Clusters #visulaizing 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:

