Ex.no:. 1 Date: Reg.no: 210701062	**Implementation of Data Preprocessing Techniques**

Aim:
To preprocess the given dataset to proceed with machine learning.

Dataset Description:
For this experiment we will go with the ‘health’ dataset which has the details of patients about their temperature, chronic disease etc.

Sample Dataset:

Health.csv

age	temperature	chronic_disease	breathing_issue	O2_level	needed_hospital
10	Normal	no	no	97	No
12	Normal	no	no	97	No
15	Normal	no	no	94	No
10	Normal	no	no	97	No
13	Moderate	no	no	94	No

Program:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

#reading dataset
read_file = pd.read_excel ("health.xlsx") read_file.to_csv
("health.csv",
index = None, header=True)
df = pd.DataFrame(pd.read_csv("health.csv")) data_set=pd.read_csv("health.csv")
x=data_set[['age','temperature','chronic_disease','breathing_issue','O2_level ']].values
y=data_set[['needed_hospitalization']].values

#filling missing data
from sklearn.impute import SimpleImputer
imputer=SimpleImputer(missing_values=np.nan,strategy='mean')
imputer=imputer.fit(x[:,0:5:4]) x[:,0:5:4]=imputer.transform(x[:,0:5:4])

#Encoding
```

```
from sklearn.compose import ColumnTransformer from
sklearn.preprocessing import OneHotEncoder
ct=ColumnTransformer(transformers=[('encoder',OneHotEncoder(),
[1])],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)

#test train data split
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) from
sklearn.preprocessing import StandardScaler sc=StandardScaler()
x_train[:,6:]=sc.fit_transform(x_train[:,6:])
x_test[:,6:]=sc.fit_transform(x_test[:,6:])
```

Output:

```
X:
[[0.0 0.0 1.0 10.0 'no' 'no' 97.0]
[0.0 0.0 1.0 12.0 'no' 'no' 97.0]
[0.0 0.0 1.0 15.0 'no' 'no' 94.0]
[0.0 0.0 1.0 10.0 'no' 'no' 97.0]
[0.0 1.0 0.0 13.0 'no' 'no' 94.0]]

Y:
[0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0
1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 0 1 1 1 1 1 0 1 1 1]
```

Result:
Thus the given health dataset was successfully preprocessed and split into test and training datasets.

Ex.no:. 2 Date: Reg.no: 210701062	**Implementation of Simple Linear Regression**

Aim:
To implement Linear Regression to predict values based on the given dataset.

Dataset Description:
The given dataset has two columns namely ‘area’ and ‘price’ respectively which represents the area in square feet and its price in dollars.

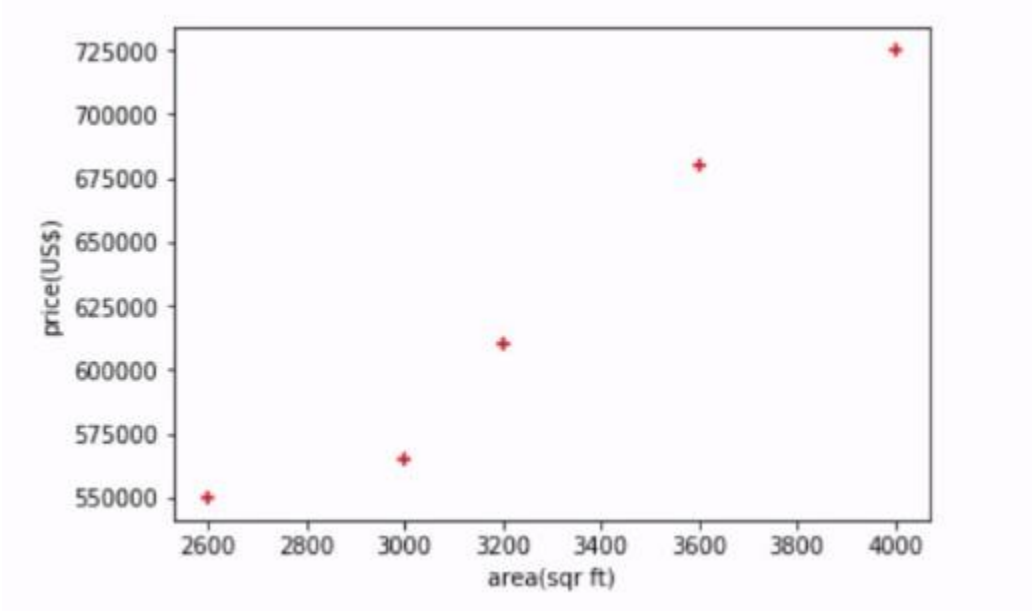
Sample Dataset:
area.csv

Area	Price
2600	550000
3000	565000
3200	610000
3600	680000
4000	725000

Program:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection
import train_test_split from pandas.core.common
import random_state
from sklearn.linear_model import LinearRegression

df = pd.read_csv('area.csv')

plt.xlabel('area(sq.ft)')
plt.ylabel('prices')
plt.scatter(df.area, df.price, color='red', marker='+')

X = df[['area']] y
= df['price']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42) reg =
LinearRegression()
reg.fit(X_train, y_train)\ y_pred =
reg.predict(X_test) new_area =
[[3300]]
predicted_price = reg.predict(new_area)
print("Predicted price for 3300 sq.ft. area:", predicted_price)
```

Output:



Predicted price for 3300 sq.ft.area: 628715.75342

Result:

Thus linear regression was successfully implemented on the ‘area’ dataset to predict prices for given areas.

Ex.no:. 3 Date: Reg.no: 210701062	**Implementation of Multiple Linear Regression**

Aim:
To implement Multivariate Regression to predict values based on the given dataset.

Dataset Description:
The given dataset has columns namely ‘area’ and ‘bedroom’, ‘age’ and ‘price’ which represents the area in square feet and the amount of bedrooms along with the tenants age and the rooms' respective prices..

Sample Dataset:
homeprices.csv

area	bedrooms	age	price
2600	3.0	20	550000
3000	4.0	15	565000
3200	NaN	18	610000
3600	3.0	30	595000
4000	5.0	8	760000
4100	6.0	8	810000

Program:

```
import pandas as pd import
numpy as np
from sklearn import linear_model

# Read the CSV file
df = pd.read_csv('homeprices.csv')

# Fill the NaN values in the 'bedrooms' column with the median df.bedrooms
= df.bedrooms.fillna(df.bedrooms.median())

# Training the linear regression model reg =
linear_model.LinearRegression()
reg.fit(df.drop('price', axis='columns'), df.price)

# Display coefficients and intercept
print("Coefficients:", reg.coef_)
print("Intercept:", reg.intercept_)

# Make predictions
prediction1 = reg.predict([[3000, 3, 40]])
prediction2 = reg.predict([[2500, 4, 5]])

# Display predictions
```

```
print("Prediction for [3000, 3, 40]:", prediction1)
print("Prediction for [2500, 4, 5]:", prediction2)
```

Output:

```
Coefficients: [  112.06244194, 23388.88007794, -3231.71790863]
Intercept: 221323.001865
Prediction for [3000, 3, 40]: 498408.25158
Prediction for [2500, 4, 5]: 578876.03748933
```

Result:

Thus Multivariate regression was successfully implemented on the ‘homeprices.csv’ dataset for prediction.

Ex.no:. 4 Date: Reg.no: 210701062	**Implementation of Polynomial Regression**

Aim:
To implement Polynomial Regression to predict values based on the given dataset.

Dataset Description:
The given dataset has columns ‘age’ and ‘height’ to represent the relation between people's ages with their heights.

Sample Dataset:
age.csv

Age	Height
45	155.43
31	147.56
51	156.27
27	155.11
38	163.54

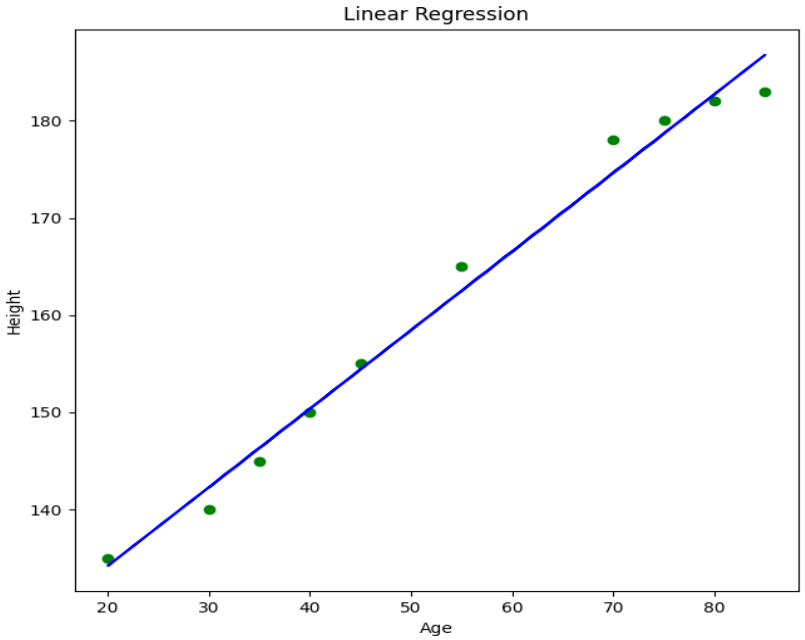
Program:

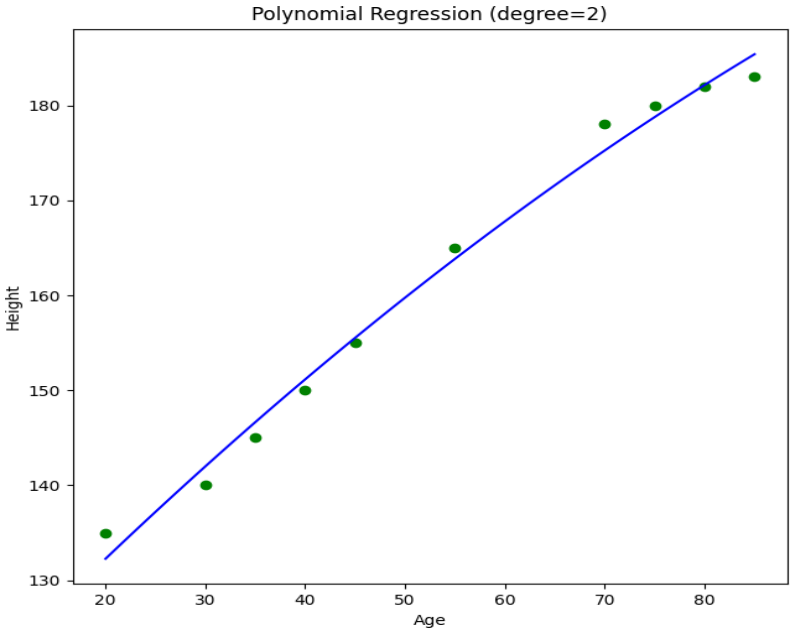
```
import pandas as pd import
numpy as np
import matplotlib.pyplot as plt from
sklearn import linear_model
from sklearn.model_selection import train_test_split from
sklearn.linear_model import LinearRegression

# Read the CSV file
df = pd.read_csv('your_dataset.csv')  # Replace 'your_dataset.csv' with the actual filename

x = df.iloc[:, 0:1].values y =
df.iloc[:, 1].values

# Splitting the data into training and testing sets
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=0)

# Linear Regression Model
linreg = LinearRegression()
linreg.fit(x_train, y_train)

# Plotting Linear Regression plt.scatter(x_train,
y_train, color='green')
plt.plot(x_train, linreg.predict(x_train), color='blue')
plt.title('Linear Regression')
```

```
plt.xlabel('Age')
plt.ylabel('Height')
plt.show()

# Polynomial Regression Model
poly = PolynomialFeatures(degree=2) x_poly =
poly.fit_transform(x_train) polyreg =
LinearRegression() polyreg.fit(x_poly, y_train)

# Plotting Polynomial Regression
x_val = np.linspace(min(x_train), max(x_train), 100).reshape(-1, 1) x_val_poly =
poly.transform(x_val)
y_pred = polyreg.predict(x_val_poly) plt.scatter(x_train,
y_train, color='green') plt.plot(x_val, y_pred, color='blue')
plt.title('Polynomial Regression (degree=2)') plt.xlabel('Age')
plt.ylabel('Height')
plt.show()

# Calculating R^2 for Simple Linear Regression y_predict_slr
= linreg.predict(x_test)
r_square_slr = metrics.r2_score(y_test, y_predict_slr) print("R^2
for Simple Linear Regression:", r_square_slr)

# Calculating R^2 for Polynomial Regression y_predict_pr =
polyreg.predict(poly.fit_transform(x_test)) r_square_pr =
metrics.r2_score(y_test, y_predict_pr) print("R^2 for Polynomial
Regression:", r_square_pr)

# Making predictions for a given value using both models slr_prediction =
linreg.predict([[53]])
print("Simple Linear Regression prediction for age 53:", slr_prediction)
pr_prediction = polyreg.predict(poly.transform([[53]])) print("Polynomial
Regression prediction for age 53:", pr_prediction)
```

Output:

```
R^2 for Simple Linear Regression: 0.9352650699314159 R^2 for
Polynomial Regression: 0.9698989897886501
Simple Linear Regression prediction for age 53: [160.89548113] Polynomial
Regression prediction for age 53: [162.22590826]
```





Result:

Thus Polynomial regression was successfully implemented on the ‘homeprices.csv’ dataset for prediction.

Ex.no:. 5 Date: Reg.no: 210701062	**Implementation of Logistic Regression**

Aim:
To implement Logistic Regression to predict values based on the given dataset.

Dataset Description:
The given dataset has columns namely ‘age’ and ‘bought_insurance’, representing the age of the person and whether they have bought insurance or not.

Sample Dataset:
insurance_data.csv

age	bought_insurance
22	0
25	0
47	1
52	0
46	1

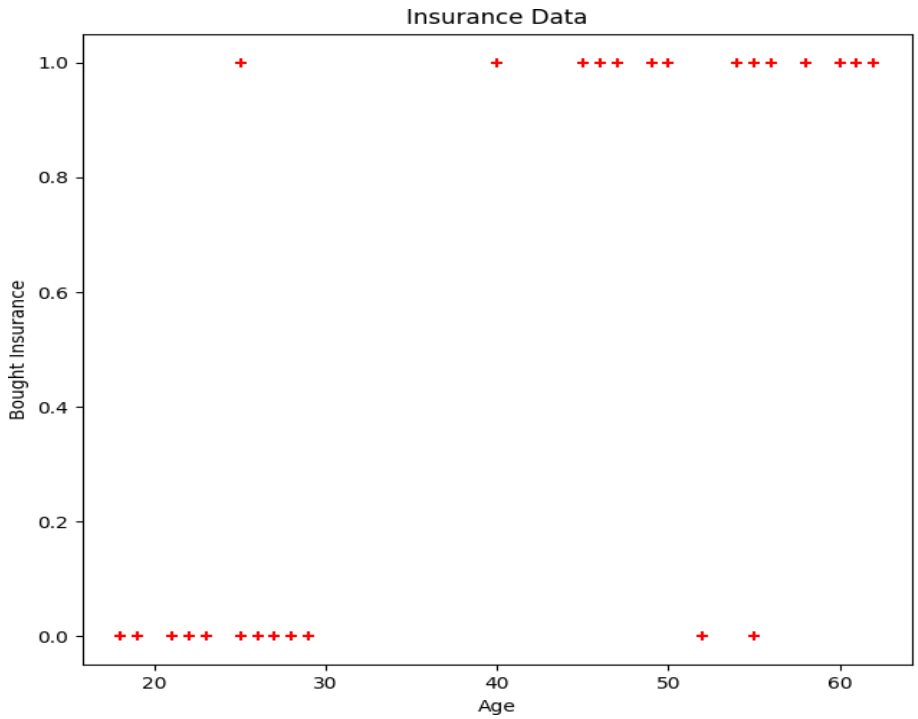
Program:

```
import pandas as pd
from matplotlib import pyplot as plt
from sklearn.model_selection import train_test_split from
sklearn.linear_model import LogisticRegression

# Read the CSV file
df = pd.read_csv("insurance_data.csv")

# Visualize the data
plt.scatter(df.age, df.bought_insurance, marker='+', color='red') plt.xlabel('Age')
plt.ylabel('Bought Insurance')
plt.title('Insurance Data') plt.show()

# Splitting the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(df[['age']], df.bought_insurance, train_size=0.8)

# Logistic Regression model model
= LogisticRegression()
model.fit(X_train, y_train)

# Predictions and model evaluation
```

y_predicted = model.predict(X_test) predicted_probabilities = model.predict_proba(X_test) accuracy = model.score(X_test, y_test)

print("Predicted probabilities:\n", predicted_probabilities) print("Model accuracy:", accuracy)

Output:



Predicted probabilities:
[[0.14887204 0.85112796]
[0.27593066 0.72406934]
[0.07431556 0.92568444]
[0.95625715 0.04374285]
[0.45363364 0.54636636]
[0.03056973 0.96943027]]
Model accuracy: 0.8333333333333334

Result:
Thus Logistic regression was successfully implemented on the 'homeprices.csv' dataset for prediction.

Ex.no:. 6 Date: Reg.no: 210701062	Implementation of K-means Clustering

Aim:
To prepare the machine learning algorithm to perform k means clustering using the appropriate dataset.

Dataset Description:
The dataset contains name, age, salary of the employees which is used to do k means clustering.
Sample Dataset:
Income.csv

Name	Age	Income($)
Rob	27	70000
Michael	29	90000
Mohan	29	61000
Ismail	28	60000
Kory	42	150000

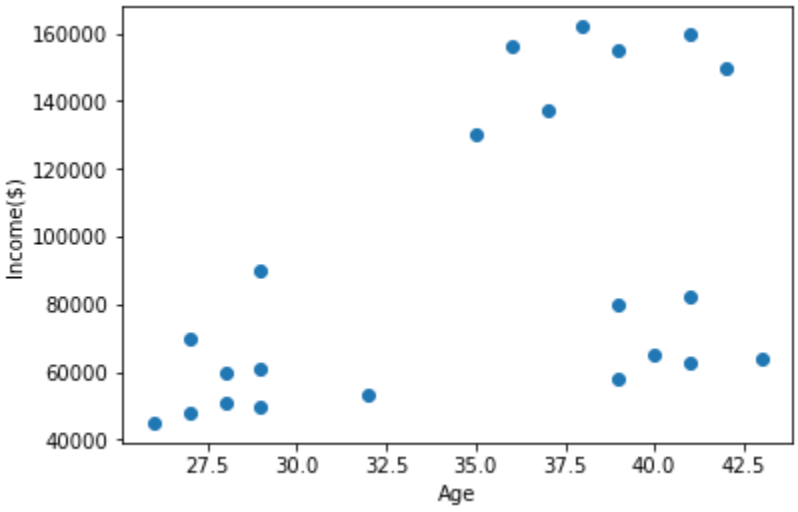
Program:

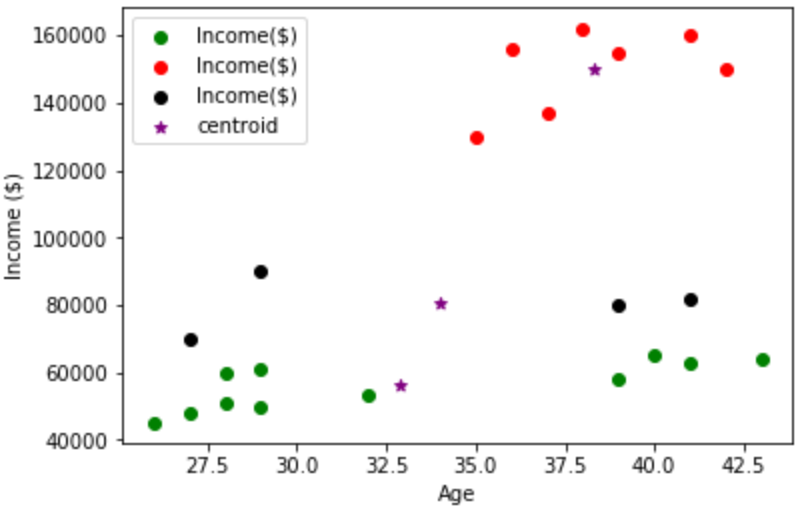
```
from sklearn.cluster import KMeans import
pandas as pd
from sklearn.preprocessing import MinMaxScaler from
matplotlib import pyplot as plt

df = pd.read_csv("income.csv")

plt.scatter(df.Age, df['Income($)'])
plt.xlabel('Age') plt.ylabel('Income($)')

km = KMeans(n_clusters=3)
y_predicted = km.fit_predict(df[['Age','Income($)']])
df['cluster'] = y_predicted

km.cluster_centers_

df1 = df[df.cluster==0] df2
= df[df.cluster==1]
```

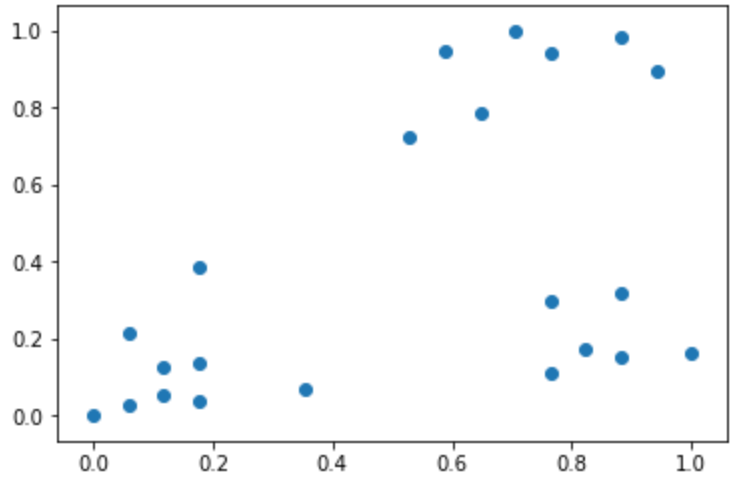
```
df3 = df[df.cluster==2]

plt.scatter(df1.Age, df1['Income($)'], color='green')
plt.scatter(df2.Age, df2['Income($)'], color='red')
plt.scatter(df3.Age, df3['Income($)'], color='black')
plt.scatter(km.cluster_centers_[:,0], km.cluster_centers_[:,1], color='purple', marker='*', label='centroid')
plt.xlabel('Age') plt.ylabel('Income
($)') plt.legend()

scaler = MinMaxScaler()

scaler.fit(df[['Income($)']])
df['Income($)'] = scaler.transform(df[['Income($)']])

scaler.fit(df[['Age']])
df['Age'] = scaler.transform(df[['Age']])

plt.scatter(df.Age, df['Income($)']) km =

KMeans(n_clusters=3)
y_predicted = km.fit_predict(df[['Age','Income($)']])
df['cluster'] = y_predicted

km.cluster_centers_

df1   =   df[df.cluster==0]
df2   =   df[df.cluster==1]
df3 = df[df.cluster==2]

plt.scatter(df1.Age, df1['Income($)'], color='green')
plt.scatter(df2.Age, df2['Income($)'], color='red')
plt.scatter(df3.Age, df3['Income($)'], color='black')
plt.scatter(km.cluster_centers_[:,0], km.cluster_centers_[:,1], color='purple', marker='*', label='centroid')
plt.legend()

sse = []
k_rng = range(1,10)  for
k in k_rng:
    km    =    KMeans(n_clusters=k)
    km.fit(df[['Age','Income($)']])
    sse.append(km.inertia_)

plt.xlabel('K')
plt.ylabel('Sum of squared error')
plt.plot(k_rng,sse)
```

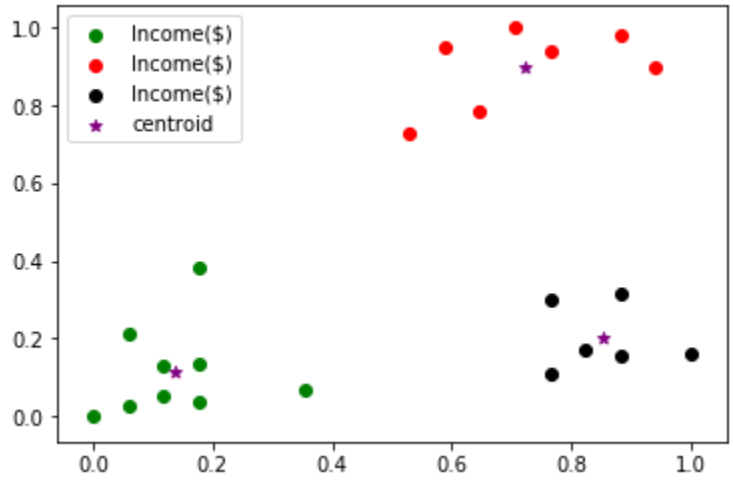
Output:

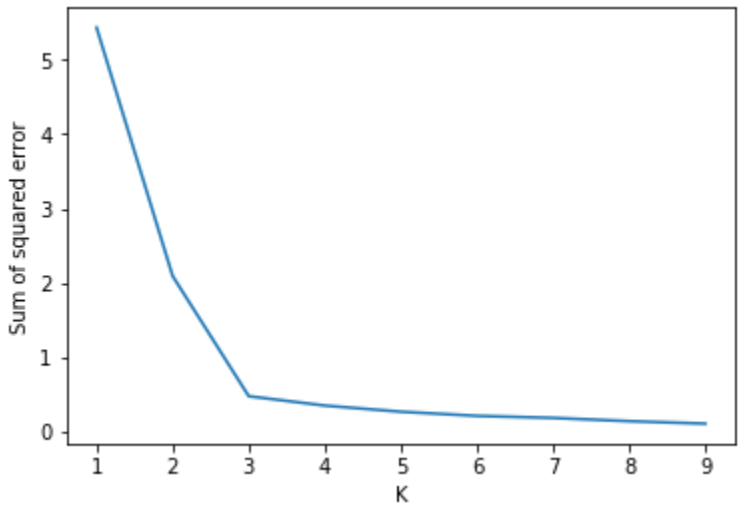
	Name	Age	Income($)
0	Rob	27	70000
1	Michael	29	90000
2	Mohan	29	61000
3	Ismail	28	60000
4	Kory	42	150000











Result:
Hence the K means clustering algorithm was implemented with the given dataset successfully.

Ex.no:. 7 Date: Reg.no: 210701062	**Implementation of Decision Tree**

Aim:
To implement a machine learning algorithm for decision tree using appropriate dataset

Dataset Description:
The data set contains the information about employees which describes which company they are in and their position and the main field is salary of the employees more than a certain amount.

Sample Dataset:

Salaries.csv

company	job	degree	salary_more_then_100k
google	sales executive	bachelors	0
google	sales executive	masters	0
google	business manager	bachelors	1
google	business manager	masters	1
google	computer programmer	bachelors	0

Program:

```
import pandas as pd
from sklearn.preprocessing import LabelEncoder from sklearn
import tree

# Read the CSV file
df = pd.read_csv("salaries.csv")

# Separate inputs and target variable
inputs = df.drop('salary_more_then_100k', axis='columns') target =
df['salary_more_then_100k']

# Label Encoding for categorical variables le_company =
LabelEncoder()
le_job = LabelEncoder() le_degree
= LabelEncoder()

inputs['company_n'] = le_company.fit_transform(inputs['company'])
inputs['job_n'] = le_job.fit_transform(inputs['job'])  inputs['degree_n'] =
le_degree.fit_transform(inputs['degree'])
```

```
inputs_n = inputs.drop(['company', 'job', 'degree'], axis='columns') # Decision

Tree Model
model = tree.DecisionTreeClassifier() model.fit(inputs_n,
target)

# Model score
model_score = model.score(inputs_n, target) print("Model
Score:", model_score)

# Predictions
prediction1 = model.predict([[2, 1, 0]])
prediction2 = model.predict([[2, 1, 1]])

print("Prediction for [2, 1, 0]:", prediction1)
print("Prediction for [2, 1, 1]:", prediction2)
```

Output:

```
Model Score: 1.0
Prediction for [2, 1, 0]:  array([0] , dtype=int64 )
Prediction for [2, 1, 1]:  array([1] , dtype=int64 )
```

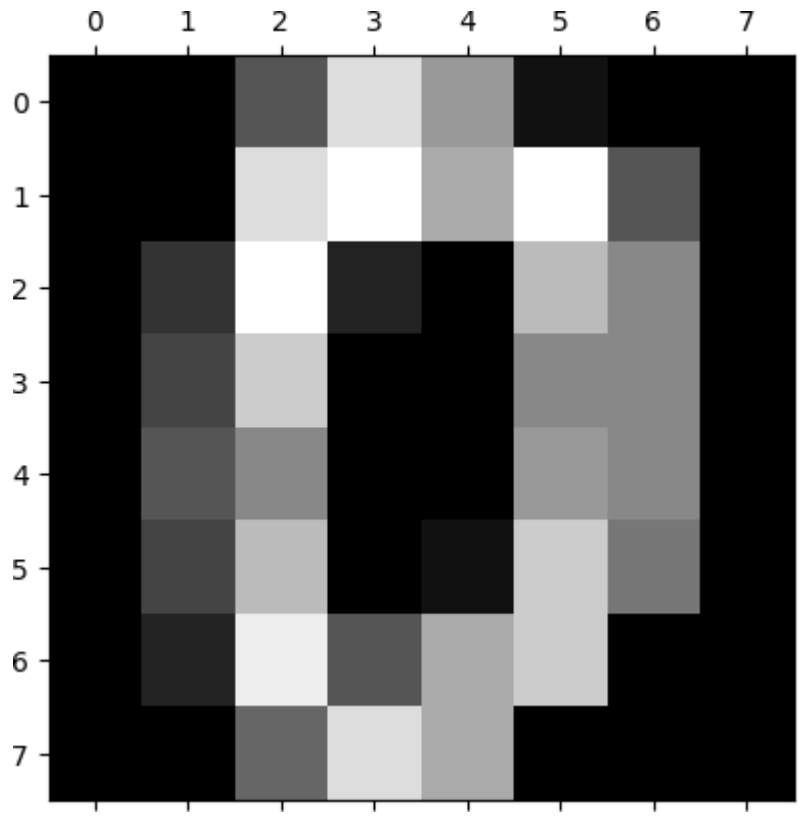
Result:
Hence the decision tree classification algorithm was implemented successfully using the given dataset.

Ex.no:. 8 Date: Reg.no: 210701062	Implementation of Random Forest

Aim:
To implement Random forest machine learning algorithm with appropriate dataset.

Dataset Description:
The dataset contains the handwritten digits which will be used for classification tasks in machine learning.

Sample Dataset:

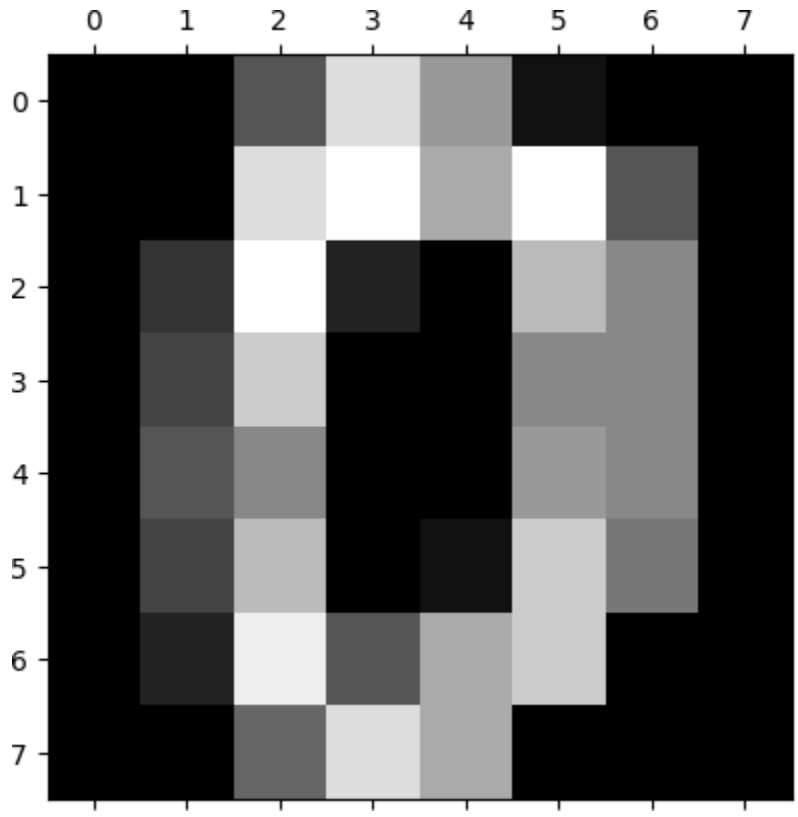


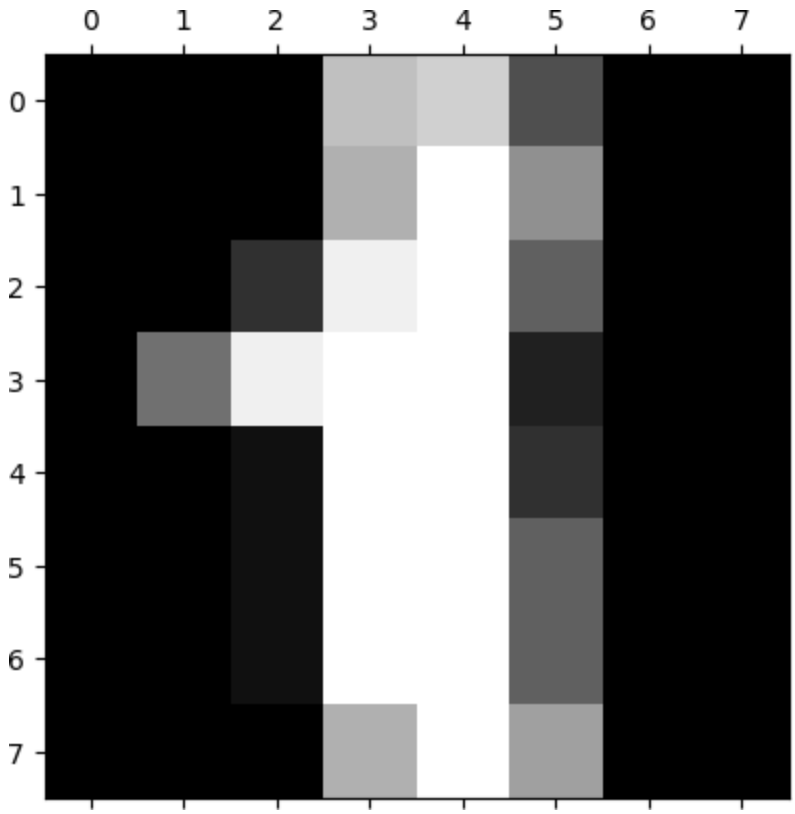
Program:

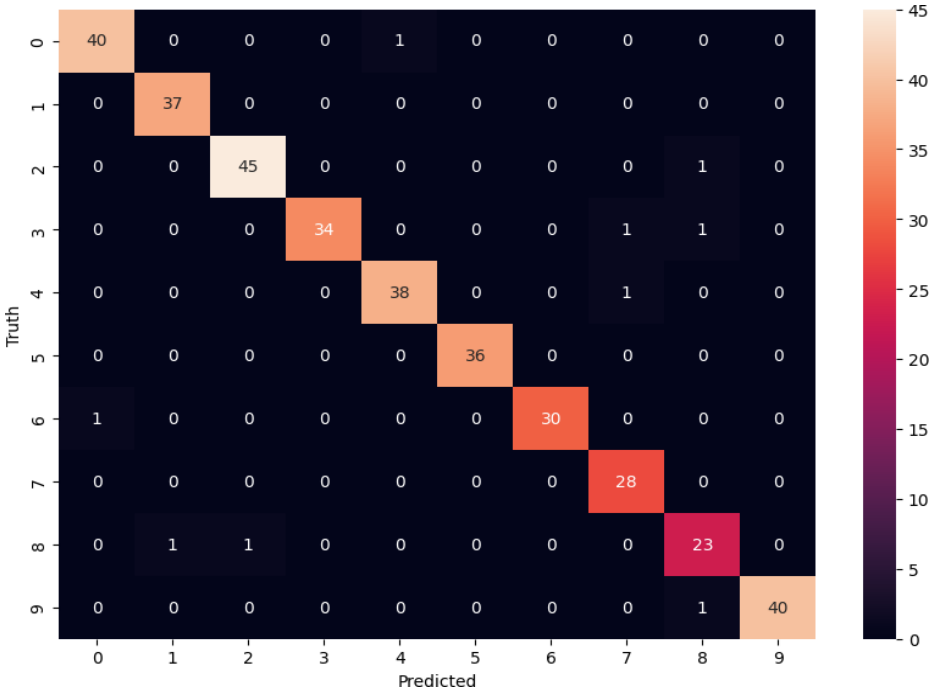
```
import pandas as pd
from sklearn.datasets import load_digits import
matplotlib.pyplot as plt
import seaborn as sn
from sklearn.ensemble import RandomForestClassifier from
sklearn.model_selection import train_test_split from
sklearn.metrics import confusion_matrix

digits = load_digits()
df = pd.DataFrame(digits.data)
df['target'] = digits.target
```

```
X = df.drop('target', axis='columns') y =
df.target

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

model = RandomForestClassifier(n_estimators=20)
model.fit(X_train, y_train)
model.score(X_test, y_test)

y_predicted = model.predict(X_test)
cm = confusion_matrix(y_test, y_predicted)

plt.figure(figsize=(10,7))
sn.heatmap(cm, annot=True)
plt.xlabel('Predicted')
plt.ylabel('Truth')
```

Output:







Result:

Hence the random forest machine learning algorithm was implemented successfully with the given dataset.

Ex.no:. 9 Date: Reg.no: 210701062	**Implementation of Naive Bayes**

Aim:
To implement Naive Bayes machine learning algorithm using appropriate dataset.

Dataset Description:
The dataset contains the category in which the email should fall and the type of messages the email contains.

Sample Dataset:
Spam.csv

Category	**Message**
ham	Go until jurong point, crazy.. Available only in bugis n great world la e buffet... Cine there got amore wat...
ham	Ok lar... Joking wif u oni...
spam	Free entry in 2 a wkly comp to win FA Cup final tkts 21st May 2005. Text FA to 87121 to receive entry question(std txt rate)T&C's apply 08452810075over18's
ham	U dun say so early hor... U c already then say...
ham	Nah I don't think he goes to usf, he lives around here though

Program:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer from
sklearn.naive_bayes import MultinomialNB
from sklearn.pipeline import Pipeline

# Read the CSV file
df = pd.read_csv("spam.csv")

# Convert 'Category' column to binary labels
df['spam'] = df['Category'].apply(lambda x: 1 if x == 'spam' else 0)

# Splitting data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(df.Message, df.spam)

# Creating a pipeline clf =
Pipeline([
    ('vectorizer', CountVectorizer()),  # Convert text to vectors ('nb',
    MultinomialNB())  # Multinomial Naive Bayes classifier
])

# Training the model
```

```
clf.fit(X_train, y_train)

# Model evaluation
accuracy = clf.score(X_test, y_test) print("Model
accuracy:", accuracy)

# Example emails emails = [
    'Hey mohan, can we get together to watch footbal game tomorrow?',
    'Upto 20% discount on parking, exclusive offer just for you. Dont miss this reward!'
]

# Predictions
predictions = clf.predict(emails)
print("Predictions:", predictions)
```

Output:

	Category	Message
0	ham	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...
3	ham	U dun say so early hor... U c already then say...
4	ham	Nah I don't think he goes to usf, he lives aro...

Model accuracy: 0.9849246231155779
Predictions: [0 1]

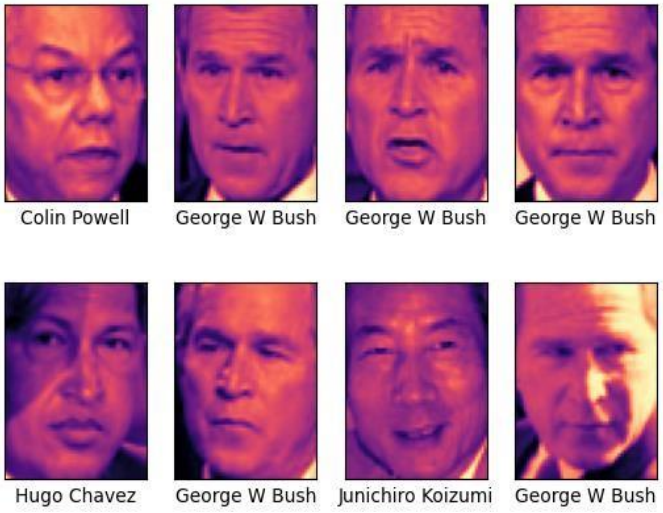
Result:
Hence the Naive Bayes algorithm was implemented using the given dataset successfully.

Ex.no:. 10 Date: Reg.no: 210701062	**Implementation of Support Vector Machine(SVM)**

Aim: To implement Support vector machine algorithm for a dataset.

Dataset Description: The dataset contains the images of a many persons.

Sample Dataset:

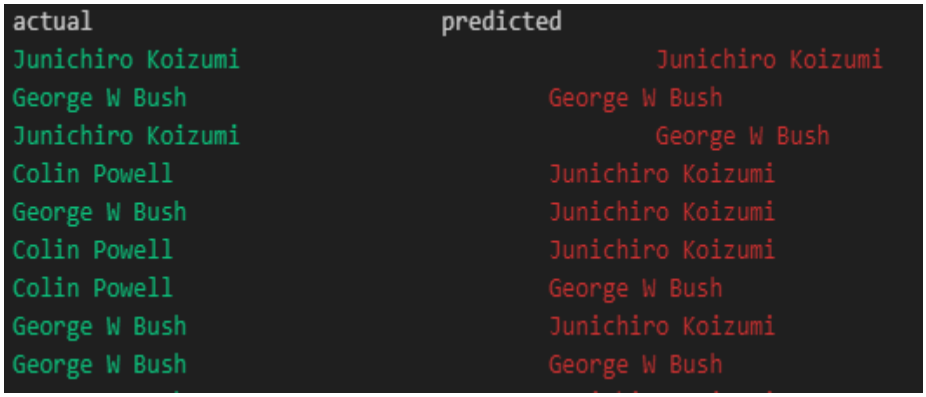


Program:

```
from sklearn.datasets import fetch_lfw_people
faces=fetch_lfw_people(min_faces_per_person=60)faces.DESCR


import matplotlib.pyplot as pltfig, splts =

plt.subplots(2, 4)

for i, ax in enumerate(splts.flat): ax.imshow(faces.images[i],
    cmap='magma')
    ax.set(xticks=[], yticks=[], xlabel=faces.target_names[faces.target[i]])
from sklearn.model_selection import train_test_splitx=faces.data
y=faces.target x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.4,random_state=42)



from sklearn.svm import SVC
from sklearn.decomposition import PCA as randomised_PCAfrom sklearn.pipeline
import make_pipeline

pca = randomised_PCA(n_components=150, whiten=True, random_state=42)svc = SVC(kernel='rbf',
class_weight='balanced')
model = make_pipeline(pca, svc)model.fit(x_train,
y_train)
```

```
from sklearn.metrics import accuracy_scorepredictions =
model.predict(x_test)
accuracy = accuracy_score(predictions, y_test)print("Accuracy:",
accuracy)
#calculating  misclassificatons from colorama import
Fore total_predictions = len(predictions)for i in
range(total_predictions):
    predictions_name = faces.target_names[predictions[i]]actual_name =
    faces.target_names[y_test[i]]
    if predictions_name != actual_name:incorrect += 1
        print("{}\t\t\t\t{}".format(Fore.GREEN + actual_name, Fore.RED +
predictions_name))
print("{} are classified as correct and {} are classified as
incorrect".format(total_predictions - incorrect, incorrect))
```

Output:



Accuracy: 0.8074074074074075

332 are classified as correct and 208 are classified as incorrect

Result: Thus SVM is implemented for the image dataset.

Ex.no:. 11 Date: Reg.no: 210701062	Implementation of Neural Networks

Aim: To implement a simple Artificial Neural Network through python.

Program:

```
from joblib.numpy_pickle_utils import xrangefrom numpy import *
class NeuralNet(object):
def___init___(self):
# Generate random numbersrandom.seed(1)
# Assign random weights to a 3 x 1 matrix, self.synaptic_weights = 2 *
random.random((3, 1)) - 1# The Sigmoid function
def___sigmoid(self, x):
return 1 / (1 + exp(-x))
# The derivative of the Sigmoid function.
# This is the gradient of the Sigmoid curve.def
_____sigmoid_derivative(self, x):
return x * (1 - x)
# Train the neural network and adjust the weights each time.def train(self, inputs,
outputs, training_iterations):
for iteration in xrange(training_iterations):
# Pass the training set through the network.output = self.learn(inputs)
# Calculate the error error = outputs -
output
# Adjust the weights by a factor
factor = dot(inputs.T, error * self._____sigmoid_derivative(output))
self.synaptic_weights += factor
# The neural network thinks.def learn(self,
inputs):
return self.____sigmoid(dot(inputs, self.synaptic_weights))if_____name__== "
____main____":
# Initialize
neural_network = NeuralNet()# The training set.
inputs = array([[0, 1, 1], [1, 0, 0], [1, 0, 1]])
outputs = array([[1, 0, 1]]).T# Train the neural
network
neural_network.train(inputs, outputs, 10000) # Test the neural
network with a test example.
print(“Prediction: ”+neural_network.learn(array([1, 0, 1])))
```

Output:

Prediction: [0.9897704]

Result:

Hence the Neural Network was successfully implemented in Python