EXPT NO: 01 **DATA PREPROCESSING**

210701063

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

Coding:

```
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 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0
1 1 0 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 1 0 1 1 1]
```

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

EXPT NO: 02 **LINEAR REGRESSION**

210701063

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:

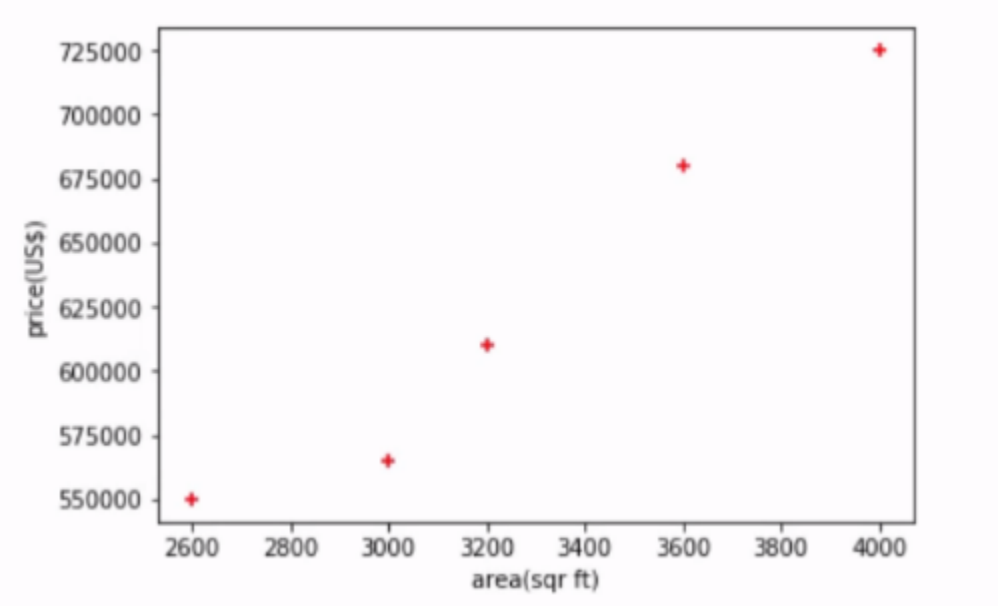
area.csv

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

Coding:

```
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)
```



Predicted price for 3300 square feet area: 628715.75342

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

EXPT NO: 03 MULTIVARIATE REGRESSION
210701063

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

Coding:

```
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 dataset for prediction.

EXPT NO: 04 POLYNOMIAL REGRESSION
210701063

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:

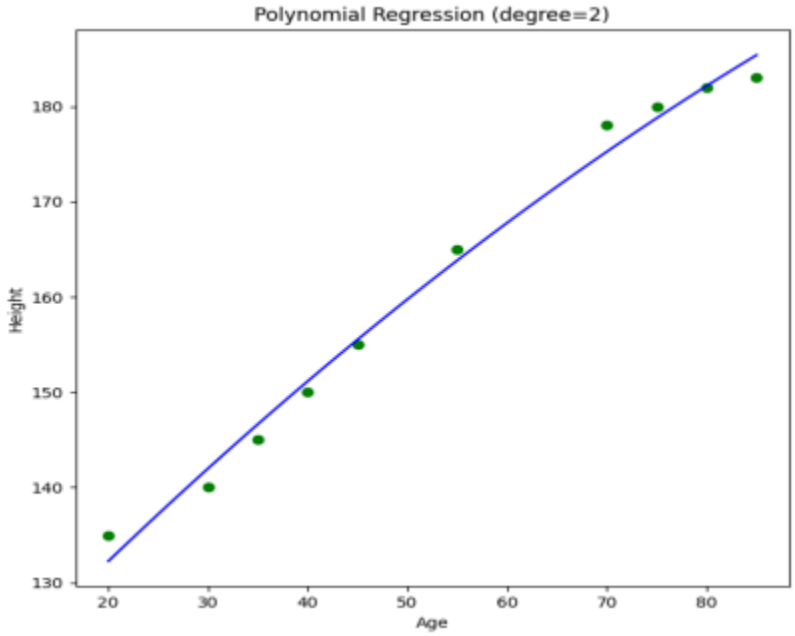
Age	Height
45	155.43
31	147.56
51	156.27
27	155.11
38	163.54

Coding:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
dataset = pd.read_csv('RBC.csv')
X = dataset.iloc[:, 0:-1].values
y = dataset.iloc[:, -1].values
from sklearn.linear_model import LinearRegression
linear_reg = LinearRegression()
linear_reg.fit(X, y)
from sklearn.preprocessing import PolynomialFeatures
polynomial_reg = PolynomialFeatures(degree = 6)
X_poly = polynomial_reg.fit_transform(X)
linear_reg_2 = LinearRegression()
linear_reg_2.fit(X_poly, y)
plt.scatter(X, y, color = "red")
plt.plot(X, linear_reg.predict(X), color='blue')
plt.title("RBC - Linear Regression")
plt.xlabel('Age')
plt.ylabel('RBC')
plt.show()
```

```
plt.scatter(X, y, color = "red")
plt.plot(X, linear_reg_2.predict(X_poly), color='blue')
plt.title("RBC - Polynomial Regression")
plt.xlabel('Age')
plt.ylabel('RBC')
plt.show()
linear_reg.predict([[6.5]])
```

Output:



Result: Thus Polynomial regression was successfully implemented.

EXPT NO: 05 LOGISTIC REGRESSION
210701063

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:

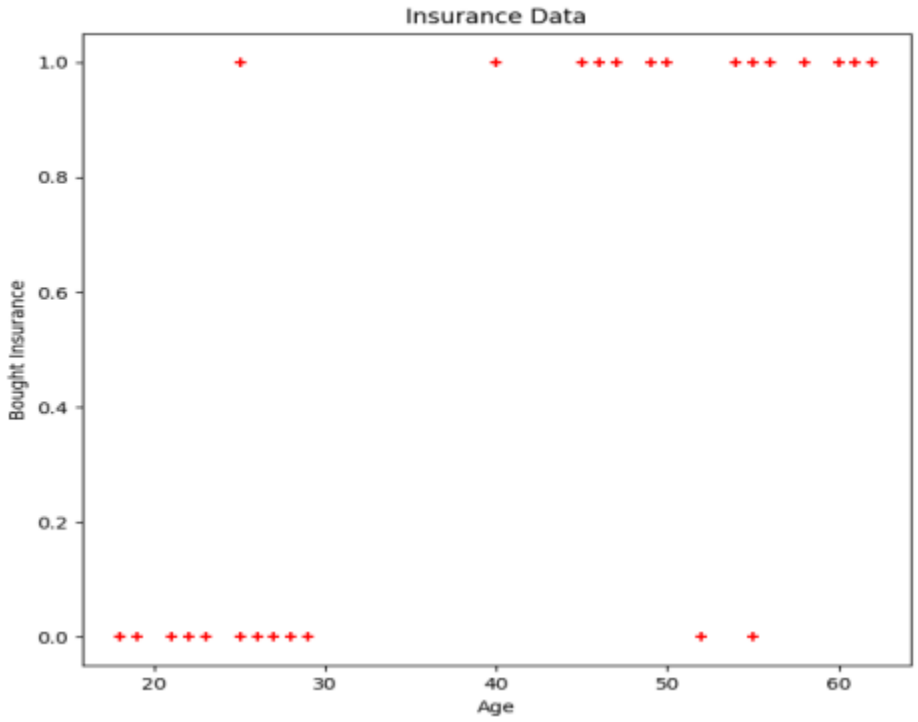
age	bought_insurance
22	0
25	0
47	1
52	0
46	1

Coding:

```
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:



Result: Thus Logistic regression was successfully implemented.

EXPT NO: 06 K-MEANS CLUSTERING
210701063

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:

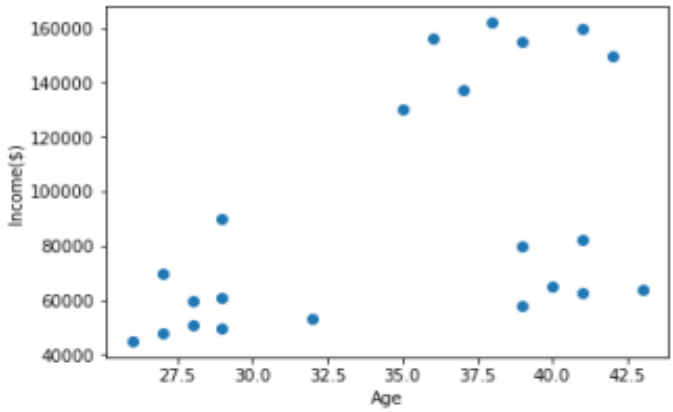
Name	Age	Income($)
Rob	27	70000
Michael	29	90000
Mohan	29	61000
Ismail	28	60000
Kory	42	150000

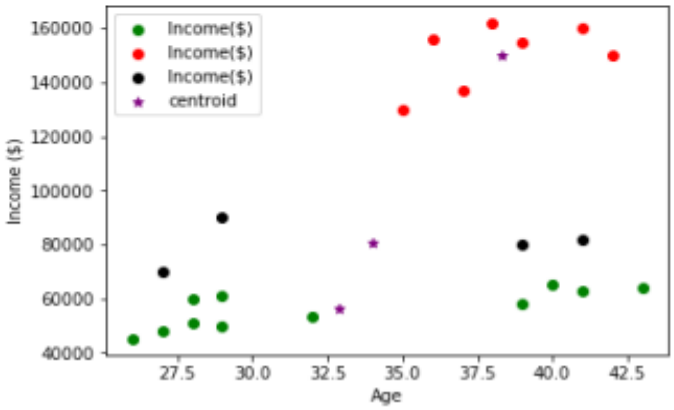
Coding:

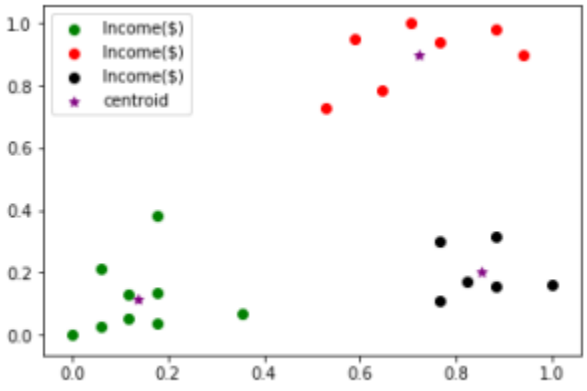
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from sklearn import tree
from sklearn.preprocessing import MinMaxScaler
from sklearn.cluster import KMeans
%matplotlib inline
df=pd.read_csv('income_kmeans.csv')
df.head()
plt.scatter(df['Age'],df['Income'])
km=KMeans(n_clusters=3)
km
y_pred=km.fit_predict(df[['Age','Income']])
y_pred
df['cluster']=y_pred
df.head()
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='blue')
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()#to properly scale the  x and y axis-->Scaling and predict again
scaler.fit(df[['Income']])
df['Income']=scaler.transform(df[['Income']])
scaler1=MinMaxScaler()
scaler1.fit(df[['Age']])
df['Age']=scaler1.transform(df[['Age']])
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:







Result: Thus the K-Means program is implemented successfully.

EXPT NO: 07 DECISION TREE
210701063

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:

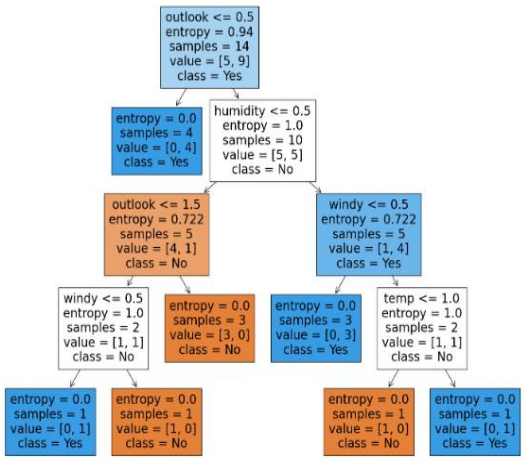
company	job	degree	salary_more_then_100k
google	sales executive	bachelors	0
google	sales executive	masters	0
google	business manager	bachelors	1
google	business manager	masters	1
google	computer programmer	bachelors	0

Coding:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn import linear_model

df=pd.read_csv('company_salary_decision.csv')
df.head()

input=df.drop(['salary more than 100k'],axis='columns')
target=df['salary more than 100k']
target


from sklearn.preprocessing import LabelEncoder

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.head()


inputs_n =inputs.drop(['Company','job','degree','salary more than
100k'],axis='columns')
inputs_n


from sklearn import tree
model=tree.DecisionTreeClassifier()
model.fit(inputs_n,target)
model.score(inputs_n, target)
model.predict([[2, 2, 1]])
```

Output:



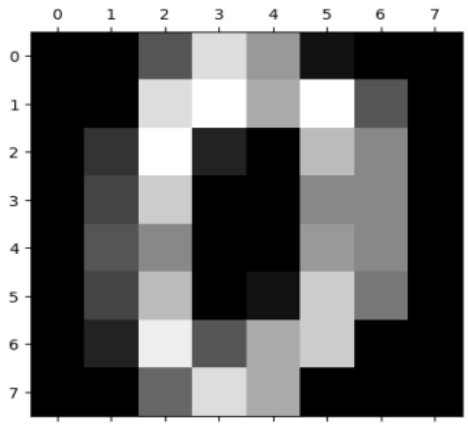
Result: Thus the decision tree algorithm is implemented.

EXPT NO: 08 **RANDOM FOREST**
210701063

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:



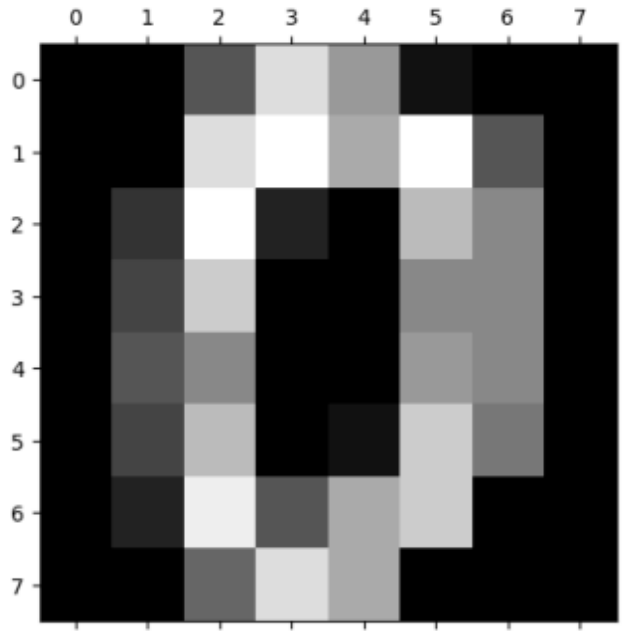
Coding:

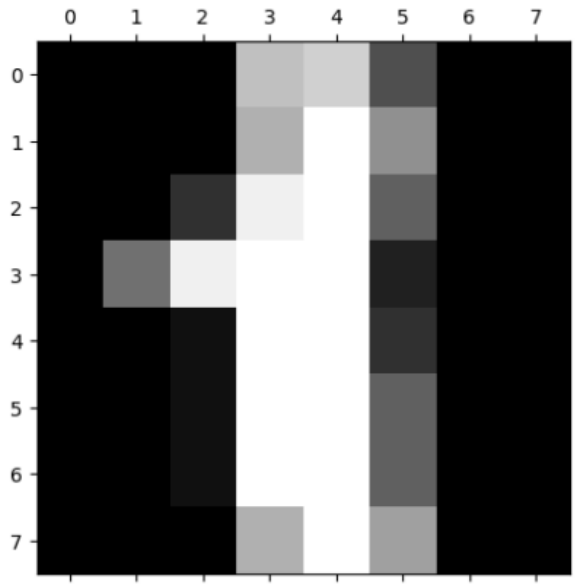
```
# Importing the required libraries
import pandas as pd, numpy as np
import matplotlib.pyplot as plt, seaborn as sns
%matplotlib inline
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
df = pd.read_csv('Social_network_ads.csv')
print(df.head())
# Putting feature variable to X
X = df.drop('Purchased',axis=1)
# Putting response variable to y
y = df['Purchased']
# now lets split the data into train and test
from sklearn.model_selection import train_test_split
```

```
# Splitting the data into train and test
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.7,
random_state=42)
X_train.shape, X_test.shape
import pandas as pd

# Assuming 'sex' is a categorical variable
X_encoded = pd.get_dummies(X)

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_encoded, y, test_size=0.2,
random_state=42)

# Initialize and train the RandomForestClassifier
classifier_rf = RandomForestClassifier(random_state=42, n_jobs=-1,
max_depth=5, n_estimators=100, oob_score=True)
classifier_rf.fit(X_train, y_train)
rf = RandomForestClassifier(random_state=42, n_jobs=-1)

params = {
   'max_depth': [2,3,5,10,20],
   'min_samples_leaf': [5,10,20,50,100,200],
   'n_estimators': [10,25,30,50,100,200]
}

from sklearn.model_selection import GridSearchCV

# Instantiate the grid search model
grid_search = GridSearchCV(estimator=rf,
                           param_grid=params,
                           cv = 3,
                           n_jobs=-1, verbose=1, scoring="accuracy")

grid_search.fit(X_train, y_train)
grid_search.best_score_rf_best = grid_search.best_estimator_
rf_best
from sklearn.tree import plot_tree
import matplotlib.pyplot as plt
```

```
# Convert the Index object to a list
feature_names_list = X.columns.tolist()

plt.figure(figsize=(10,10))
plot_tree(rf_best.estimators_[5], feature_names=feature_names_list,
class_names=['Yes', 'No'], filled=True);
from sklearn.tree import plot_tree
import matplotlib.pyplot as plt

# Convert the Index object to a list
feature_names_list = X.columns.tolist()

plt.figure(figsize=(10,10))
plot_tree(rf_best.estimators_[7], feature_names=feature_names_list,
class_names=['Yes', 'No'], filled=True);
```

OUTPUT:





Result: Thus random forest algorithm is implemented for the given dataset.

EXPT NO: 09 **NAIVE BAYES**
210701063

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:

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

Output:

```
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)
```

Ouput:

	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...

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

EXPT NO:10 SUPPORT VECTOR MACHINE
210701063

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

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

Sample Dataset:

Coding:

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


import matplotlib.pyplot as plt

fig, 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_split
x=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_PCA
from 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_score
predictions = 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:

```
actual                          predicted
Junichiro Koizumi                       Junichiro Koizumi
George W Bush                   George W Bush
Junichiro Koizumi                       George W Bush
Colin Powell                    Junichiro Koizumi
George W Bush                   Junichiro Koizumi
Colin Powell                    Junichiro Koizumi
Colin Powell                    George W Bush
George W Bush                   Junichiro Koizumi
George W Bush                   George W Bush
```

Accuracy: 0.8074074074074075

332 are classified as correct and 208 are classified as incorrect

Result: Thus SVM is implemented for the image dataset.

EXPT NO 11 NEURAL NETWORKS
210701063

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

Code:

```
from joblib.numpy_pickle_utils import xrange
from numpy import *
class NeuralNet(object):
def __init__(self):
# Generate random numbers
random.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