Sr.No	Topic	Date	Sign
1	Implementing K-means Classification Technique	11-02-2023	
2	Implementing Linear Regression using following raw data. a. Homeprices b. Weightwaist c. Canada percapita income	09-03-2023	
3	Implementing Logistic Regression	15-03-2023	
4	Implement an application that stores big data in MongoDB and manipulate it using python. a.insert_one, update_one, delete_one b.insert_many, update_many, delete_many	23-03-2023	
5	Implement SVM Classification Technique.	17-04-2023	
6	Implement Decision Tree Classification Technique.	21-04-2023	
7	Text Analysis Implementation	25-04-2023	
8	Sentiment Analysis	04-05-2023	
9	Install, configure and run Hadoop and HDFS	11-05-2023	
10	Basic Commands of HDFS Mkdir, ls, cat, get, put, copyToLocal, copyFromLocal, mv, tail, touchz, cp, rm, rmr, chmod.	16-05-2023	

Practical 1

Aim: - Implementing K-means Classification Technique

Description: - K-means classification is a clustering algorithm that can be used for classification. It partitions a dataset into k clusters by iteratively assigning data points to the nearest centroid and updating the centroids. Once convergence is reached, class labels are assigned to the clusters based on the majority class of the data points within each cluster. K-means classification is simple, scalable, and fast, but it assumes spherical clusters of similar sizes and is sensitive to initial centroid positions. While it provides a straightforward approach to classification, its effectiveness depends on the data and underlying assumptions.

Method: -

1. `pd.DataFrame()`: This method from the pandas library is used to create a DataFrame object. The DataFrame holds the data points for clustering, with 'x' and 'y' representing the coordinates of each data point.
2. `np.random.seed()`: This method from the numpy library is used to set the random seed. By setting a seed, it ensures that the random numbers generated for initializing the centroids are the same each time the code is executed. This allows for reproducibility of results.
3. `plt.scatter()`: This method from the matplotlib.pyplot library is used to create scatter plots. It is used to plot the data points and centroids on the figure.
4. `plt.xlim()` and `plt.ylim()`: These methods set the x-axis and y-axis limits for the plot, respectively.
5. `assignment()`: This function calculates the Euclidean distance between each data point and the centroids. It assigns each data point to the closest centroid and updates the 'closest' column in the DataFrame. Additionally, it assigns a color to each data point based on the closest centroid.
6. `print(df.head())`: This line prints the first few rows of the DataFrame to display the assigned centroids and colors for each data point.
7. `plt.show()`: This method displays the figure with the plotted data points and centroids.
8. `update()`: This function updates the positions of the centroids based on the mean of the data points assigned to each centroid.
9. `copy.deepcopy()`: This method from the copy module is used to create a deep copy of the 'centroids' dictionary. It is used to store the previous centroid positions for visualization purposes.
10. `ax.arrow()`: This method is used to plot arrows representing the movement of the centroids from their old positions to the updated positions.

Code:

Initial stage

prac1_kmeans.py - E:\Sem 2\Big Data\practical\prac1_kmeans.py (3.10.10)

File Edit Format Run Options Window Help

```
plt.scatter(df['x'],df['y'],color=df['color'],alpha=0.5,edgecolor='k')
for i in centroids.keys():
    plt.scatter(*centroids[i],color=colmp[i])
plt.xlim(0,80)
plt.ylim(0,80)
plt.show()


#update stage
import copy
old_centroids=copy.deepcopy(centroids)

def update(k):
    for i in centroids.keys():
        centroids[i][0] =np.mean(df[df['closest'] == i]['x'])
        centroids[i][1] =np.mean(df[df['closest'] == i]['y'])
    return k

centroids=update(centroids)

fig=plt.figure(figsize=(5,5))
ax=plt.axes()
plt.scatter(df['x'],df['y'],color=df['color'],alpha=0.5,edgecolor='k')
for i in centroids.keys():
    plt.scatter(*centroids[i],color=colmp[i])
plt.xlim(0,80)
plt.ylim(0,80)
for i in old_centroids.keys():
    old_x=old_centroids[i][0]
    old_y=old_centroids[i][1]
    dx= (centroids[i][0]-old_centroids[i][0])*0.75
    dy= (centroids[i][1]-old_centroids[i][1])*0.75
    ax.arrow(old_x,old_y,dx,dy,head_width=2,head_length=3,fc=colmp[i],ec=colmp[i])
plt.show()
```

Assignment stage

```
#assignment stage
def assignment(df,centroids):
    for i in centroids.keys():
        df['distance_from_{}'.format(i)]=(
            np.sqrt(
                (df['x']-centroids[i][0]) ** 2
                +(df['y']-centroids[i][1]) ** 2
            )
        )
    centroid_distance_cols=['distance_from_{}'.format(i) for i in centroids.keys()]
    df['closest']=df.loc[:,centroid_distance_cols].idxmin(axis=1)
    df['closest']=df['closest'].map(lambda x:int(x.lstrip('distance_from_')))
    df['color']=df['closest'].map(lambda x:colmp[x])
    return df

df=assignment(df,centroids)
print(df.head())

fig=plt.figure(figsize=(5,5))
plt.scatter(df['x'],df['y'],color=df['color'],alpha=0.5,edgecolor='k')
for i in centroids.keys():
    plt.scatter(*centroids[i],color=colmp[i])
plt.xlim(0,80)
plt.ylim(0,80)
plt.show()
```

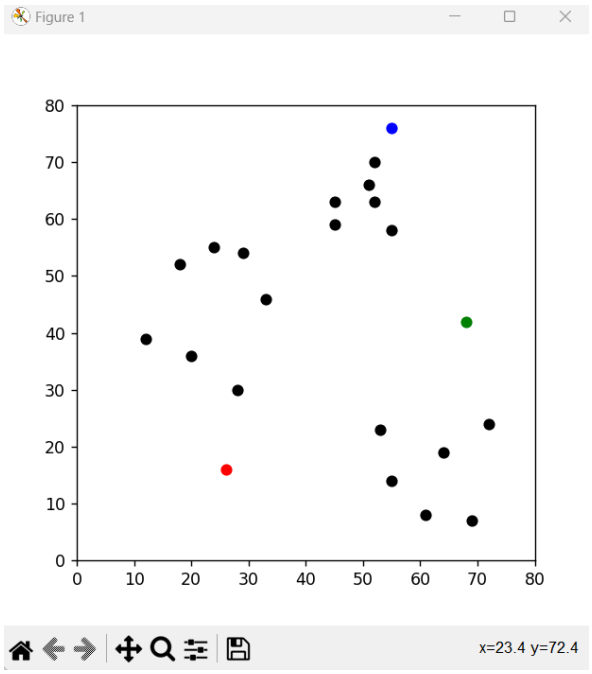
Update stage

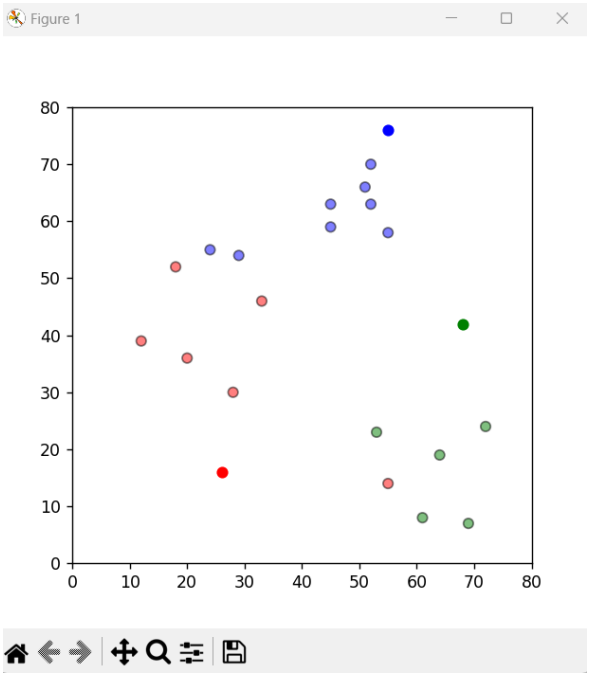
```
#update stage
import copy
old_centroids=copy.deepcopy(centroids)

def update(k):
    for i in centroids.keys():
        centroids[i][0] =np.mean(df[df['closest'] == i]['x'])
        centroids[i][1] =np.mean(df[df['closest'] == i]['y'])
    return k

centroids=update(centroids)

fig=plt.figure(figsize=(5,5))
ax=plt.axes()
plt.scatter(df['x'],df['y'],color=df['color'],alpha=0.5,edgecolor='k')
for i in centroids.keys():
    plt.scatter(*centroids[i],color=colmp[i])
plt.xlim(0,80)
plt.ylim(0,80)
for i in old_centroids.keys():
    old_x=old_centroids[i][0]
    old_y=old_centroids[i][1]
    dx= (centroids[i][0]-old_centroids[i][0])*0.75
    dy= (centroids[i][1]-old_centroids[i][1])*0.75
    ax.arrow(old_x,old_y,dx,dy,head_width=2,head_length=3,fc=colmp[i],ec=colmp[i])
plt.show()
```

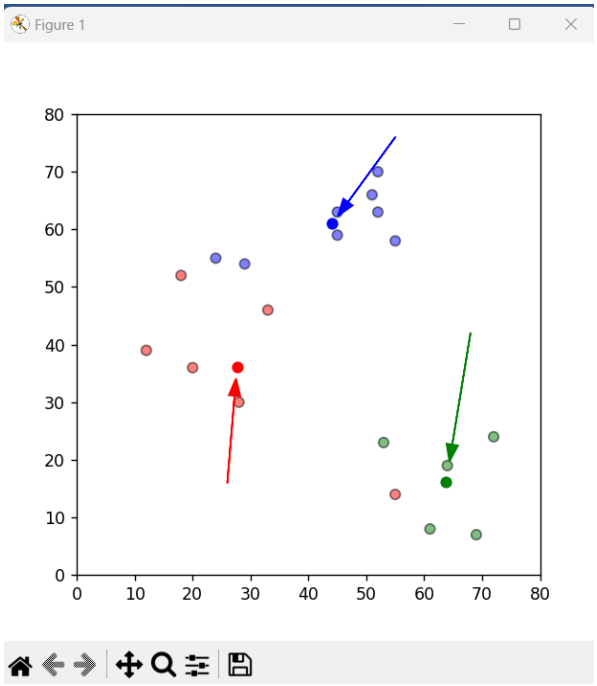
Update stage and Continue until all assigned Categories don’t change anymore

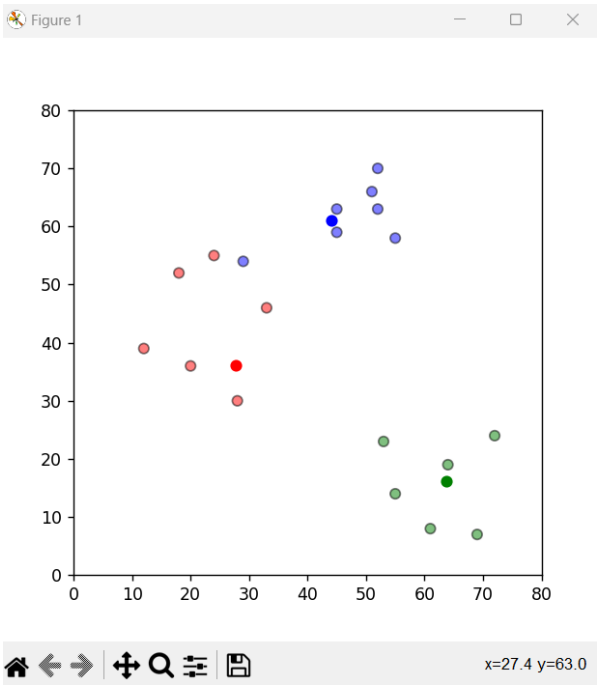
```
#repeatassignmentstage
df=assignment(df,centroids)

fig=plt.figure(figsize=(5,5))
plt.scatter(df['x'],df['y'],color=df['color'],alpha=0.5,edgecolor='k')
for i in centroids.keys():
    plt.scatter(*centroids[i],color=colmp[i])
plt.xlim(0,80)
plt.ylim(0,80)
plt.show()


#continue until all assigned categories don't change anymore:
while True:
    closest_centroids=df['closest'].copy(deep=True)
    centroids =update(centroids)
    df= assignment(df,centroids)
    if closest_centroids.equals(df['closest']):
        break

fig=plt.figure(figsize=(5,5))
plt.scatter(df['x'],df['y'],color=df['color'],alpha=0.5,edgecolor='k')
for i in centroids.keys():
    plt.scatter(*centroids[i],color=colmp[i])
plt.xlim(0,80)
plt.ylim(0,80)
plt.show()
print("vighnesh kargutkar MLDC 03")
```

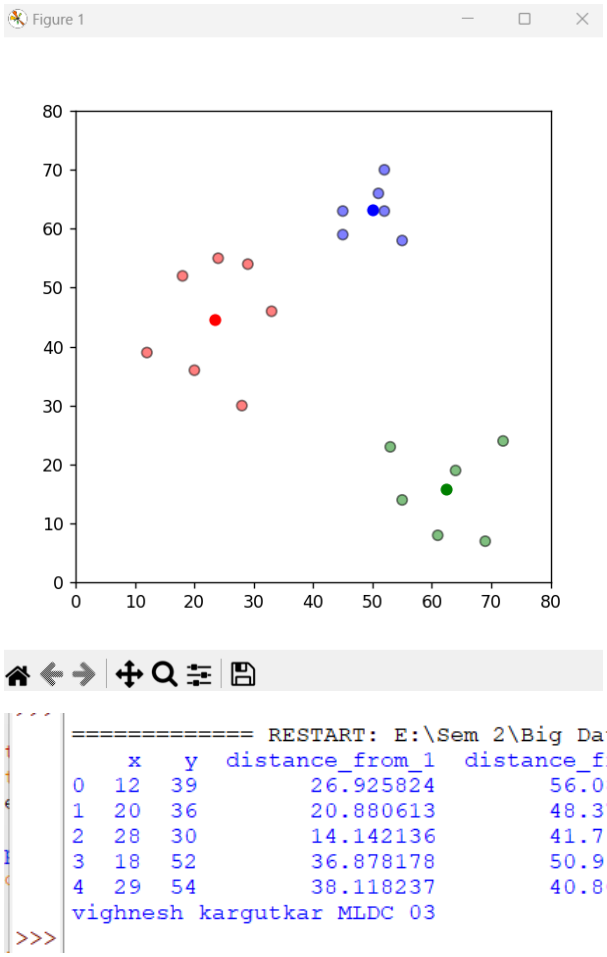
Output












```
============== RESTART: E:\Sem 2\Big Data\practical\prac1_kmeans.py ==============
    x   y  distance_from_1  distance_from_2  distance_from_3  closest color
0  12  39        26.925824        56.080300        56.727418        1     r
1  20  36        20.880613        48.373546        53.150729        1     r
2  28  30        14.142136        41.761226        53.338541        1     r
3  18  52        36.878178        50.990195        44.102154        1     r
4  29  54        38.118237        40.804412        34.058773        3     b
vighnesh kargutkar MLDC 03
>>>
```

Practical 2

Aim: - Implementing Linear Regression using following raw data.

Description**:** - Linear regression is a statistical modeling technique used to establish a relationship between a dependent variable and one or more independent variables. It assumes a linear relationship between the variables, with the goal of finding the best-fit line that minimizes the difference between predicted and actual values. It involves data preparation, model training, evaluation, and prediction. The equation for simple linear regression is y = b0 + b1 * x, where y is the dependent variable, x is the independent variable, b0 is the y-intercept, and b1 is the coefficient (slope). Linear regression is used for tasks such as prediction, analyzing variable impact, and trend estimation.

Methods**:** -

1. `pd.read_csv('file_path')`: This method is used to read a CSV file and load it into a pandas DataFrame. The DataFrame is assigned to the variable `df`, containing the data from the 'homeprices.csv' file.
2. `plt.xlabel()`** and **`plt.ylabel()`: These methods are used to label the x and y axes of the scatter plot.
3. `plt.scatter()`: This method is used to create a scatter plot of the 'area' against the 'price' from the DataFrame `df`.
4. `linear_model.LinearRegression()`: This creates an instance of the `LinearRegression` class from the scikit-learn library, which represents the linear regression model.
5. `model.fit()`: This method fits the linear regression model to the training data.
6. `model.coef_` and `model.intercept_`: These attributes of the model provide the coefficient (slope) and the intercept of the fitted regression line, respectively.

A] Homeprices

Code**:**

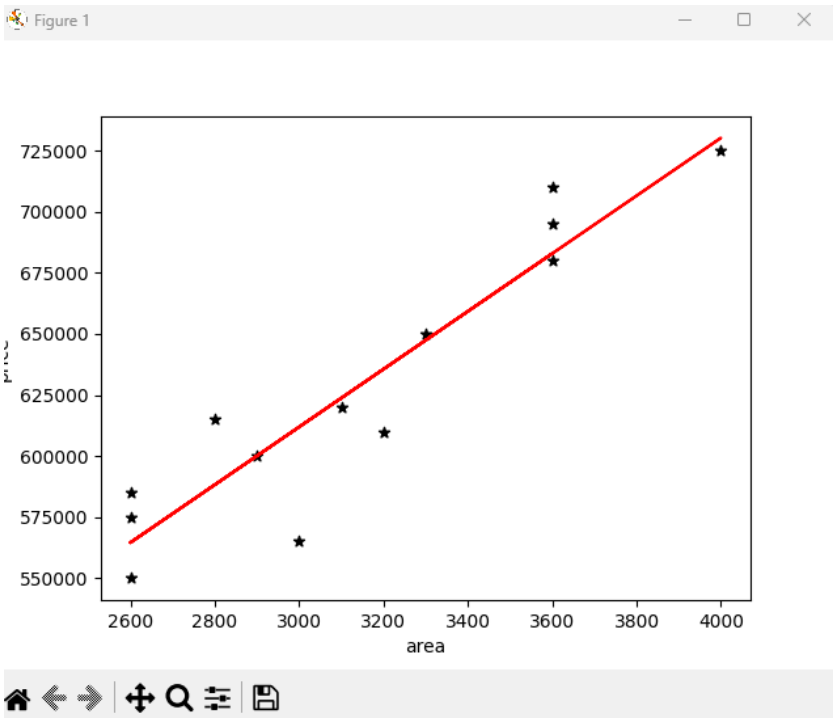
prac2a_homeprice.py - E:\Sem 2\Big Data\practical\prac2a_homeprice.py (3.10.10)

File Edit Format Run Options Window Help

```
import pandas as pd
import numpy as np
from sklearn import linear_model
import matplotlib.pyplot as plt
df=pd.read_csv('E:\Sem 2\Big Data\practical\homeprices.csv')
print(df)
plt.xlabel('area')
plt.ylabel('price')
plt.scatter(df.area,df.price,color='black',marker='*')
#dropping price column bcoz when we fit the linear model it expects a 2D array
new_df=df.drop('price',axis=1)
print('***********')
#making an instance of LinearRegression class
model=linear_model.LinearRegression()
#to train the model use fit method with area and price
model.fit(new_df.values,df.price)
#we want to predict price of area 1500, predict function expects 2D array
print('predicted value',model.predict([[1500]]))
print('coefficient value',model.coef_)
print('intercept value',model.intercept_)
area_df=pd.read_csv('E:\Sem 2\Big Data\practical\\area.csv')
predicted=model.predict(area_df.values)
print('predicted',predicted)
area_df['prices']=predicted
area_df.to_csv('E:\Sem 2\Big Data\practical\prediction.csv')
dff=pd.read_csv('E:\Sem 2\Big Data\practical\prediction.csv')
print(dff)
plt.plot(df.area,model.predict(df.area.values.reshape(-1,1)),color='red')
plt.show()
print("vighnesh kargutkar MLDC 03")
```

Output:

```
========== RESTART: E:\Sem 2\Big Data\practical\prac2a_homeprice.py =======
    area   price
0   2600  550000
1   3000  565000
2   3200  610000
3   3600  680000
4   4000  725000
5   2600  585000
6   2800  615000
7   3300  650000
8   3600  710000
9   2600  575000
10  2900  600000
11  3100  620000
12  3600  695000
***********
predicted value [434499.0665837]
coefficient value [118.29495955]
intercept value 257056.627255756
predicted [375351.58680772 399010.57871811 422669.5706285  469987.55444928
 493646.54635968 505476.04231487 517305.53827007 540964.53018046
 351692.59489732 374168.6372122  339863.09894213 328033.60298693
 335131.30056005]
    Unnamed: 0  area         prices
0            0  1000  375351.586808
1            1  1200  399010.578718
2            2  1400  422669.570629
3            3  1800  469987.554449
4            4  2000  493646.546360
5            5  2100  505476.042315
6            6  2200  517305.538270
7            7  2400  540964.530180
8            8   800  351692.594897
9            9   990  374168.637212
10          10   700  339863.098942
11          11   600  328033.602987
12          12   660  335131.300560
vighnesh kargutkar MLDC 03
>>>
```

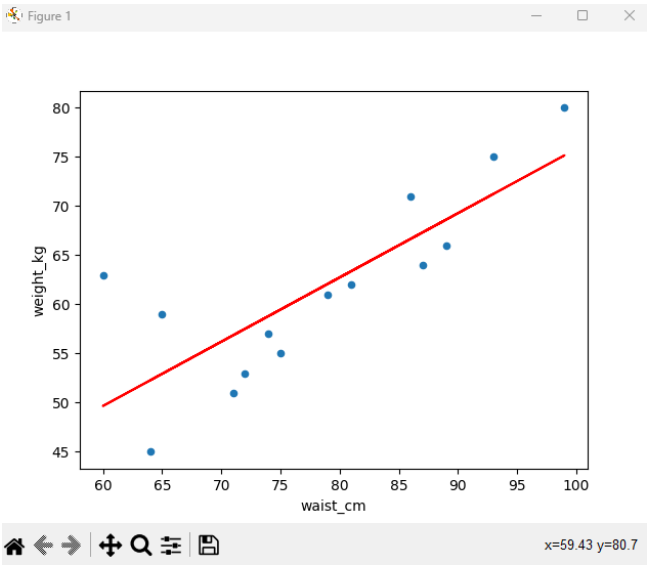


B] weightwaist

Code**:**

```
prac2b_weightwaist.py - E:\Sem 2\Big Data\practical\prac2b_weightwaist.py (3.10.10)
File  Edit  Format  Run  Options  Window  Help
import pandas as pd
import numpy as np
from sklearn import linear_model
import matplotlib.pyplot as plt
data=pd.read_csv('E:\Sem 2\Big Data\practical\weightwaist.csv')
print(data)
data.plot(kind='scatter',x='waist_cm',y='weight_kg',color='green',marker='*')
new_data=data.drop('weight_kg',axis=1)
model=linear_model.LinearRegression()
model.fit(new_data,data.weight_kg)
print(model.coef_)
print(model.intercept_)
print(model.score(new_data,data.weight_kg))
model=linear_model.LinearRegression()
model.fit(new_data.values,data.weight_kg.values)
predictions=model.predict([[98]])
print('Prediction for weight 98 is:',predictions)
plt.plot(data.waist_cm,model.predict(data.waist_cm.values.reshape(-1,1)),color='red')
plt.show()
print("vighnesh kargutkar MLDC 03")
```

Output**:**




```
========== RESTART: E:\Sem 2\Big Data\practical\prac2b_weightwaist.py ==========
    waist_cm  weight_kg
0         71         51
1         89         66
2         64         45
3         74         57
4         87         64
5         93         75
6         79         61
7         81         62
8         75         55
9         72         53
10        65         59
11        60         63
12        99         80
13        86         71
[0.65405294]
10.415144674738357
0.6377256319321334
Prediction for weight 98 is: [74.51233326]
```

C] Canada percapita income

Code**:**

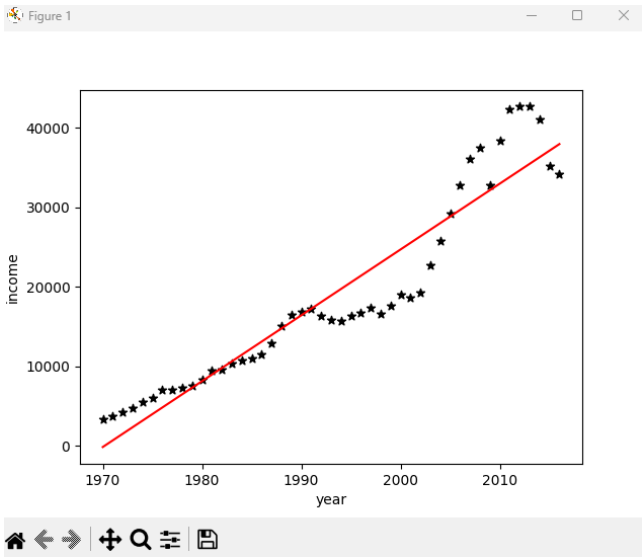
prac2c_canada_per_capita_income.py - E:\Sem 2\Big Data\practical\prac2c_canada_per_capita_income.py (3.10.10)

File Edit Format Run Options Window Help

```
print('Vighnesh kargutkar MLDC 3')
import pandas as pd
import numpy as np
from sklearn import linear_model
import matplotlib.pyplot as plt
dataset=pd.read_csv('E:\Sem 2\Big Data\practical\canada_per_capita_income.csv')
print(dataset.head(5))
dataset.rename(columns={'per capita income (US$)': 'income'}, inplace=True)
plt.xlabel('year')
plt.ylabel('income')
plt.scatter(dataset.year,dataset.income,color='black',marker='*')
print('************')
new_data=dataset.drop('income',axis=1)
print(new_data.head(5))
print('predicted values')

model=linear_model.LinearRegression()
model.fit(new_data.values,dataset.income.values)
predictions=model.predict([[2050]])
print('Prediction of Year 2050 is:',predictions)
plt.plot(dataset.year,model.predict(dataset.year.values.reshape(-1,1)),color='red')
plt.show()
print("vighnesh kargutkar MLDC 03")
```

Output:

```
==== RESTART: E:\Sem 2\Big Data\practical\prac2c_canada_per_capita_income.py ===
Vighnesh kargutkar MLDC 3
   year  per capita income (US$)
0  1970              3399.299037
1  1971              3768.297935
2  1972              4251.175484
3  1973              4804.463248
4  1974              5576.514583
************
   year
0  1970
1  1971
2  1972
3  1973
4  1974
predicted values
Prediction of Year 2050 is: [66142.6463511]
vighnesh kargutkar MLDC 03
```



Practical 3

Aim: - Implementing Logistic Regression

Description: -

Logistic regression is a statistical modeling technique used for binary classification problems, where the goal is to predict one of two possible outcomes. It calculates the probability of an event occurring based on input features and uses a logistic function, also known as the sigmoid function, to map the input to a value between 0 and 1. The logistic regression model assumes a linear relationship between the input features and the log-odds of the event occurring. It estimates the coefficients for each feature using a method called maximum likelihood estimation. These coefficients represent the impact of each feature on the predicted outcome.

Method: -

1. `make_classification`: This method generates a random binary classification dataset with specified characteristics. It is used to create a synthetic dataset for demonstration purposes.
2. `plt.scatter`: This method is used to create a scatter plot of the generated dataset. It visualizes the relationship between the input features (`x`) and the corresponding class labels (`y`).
3. `train_test_split`: This method is used to split the dataset into training and testing sets. It randomly divides the data into two portions based on the specified test size or train size.
4. `LogisticRegression`: This is the logistic regression model class from scikit-learn. It represents the logistic regression algorithm and provides methods to fit the model to the training data and make predictions.
5. `model.fit`: This method fits the logistic regression model to the training data. It learns the coefficients of the model based on the input features (`x_train`) and the corresponding class labels (`y_train`).
6. `model.coef_`: This attribute of the logistic regression model returns the coefficients (weights) assigned to the input features. It provides insight into the impact of each feature on the predicted outcome.
7. `model.intercept_`: This attribute returns the intercept (bias) term of the logistic regression model.
8. `model.score`: This method calculates the accuracy of the model on the given data. It returns the mean accuracy of the predicted labels compared to the true labels.
9. `model.predict`: This method is used to predict the class labels for the test data (`x_test`) based on the trained logistic regression model.
10. `confusion_matrix`: This method calculates the confusion matrix, which is a table that summarizes the performance of a classification model. It compares the predicted labels (`y_pred`) with the true labels (`y_test`).

Code:

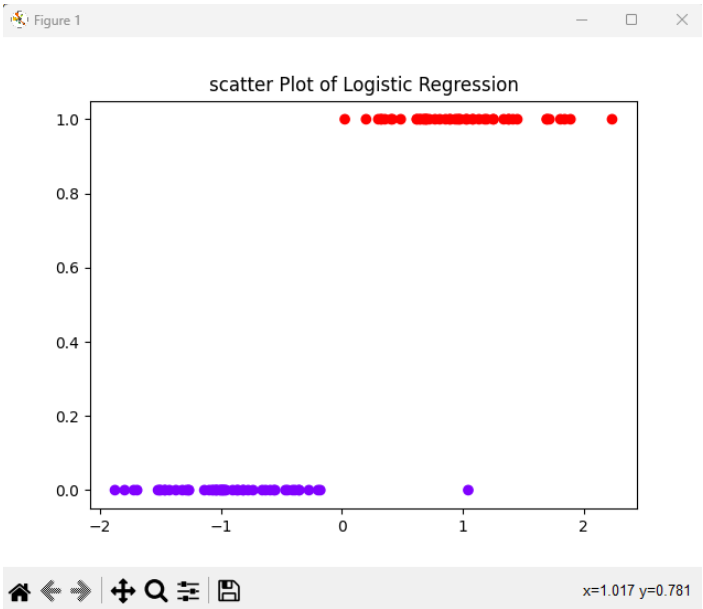
prac3_logisticregression.py - E:\Sem 2\Big Data\practical\prac3_logisticregression.py (3.10.10)

File Edit Format Run Options Window Help

```
from sklearn.datasets import make_classification
from matplotlib import pyplot as plt
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
import pandas as pd
x,y = make_classification(
    n_samples=100,
    n_features=1,
    n_classes=2,
    n_clusters_per_class=1,
    flip_y=0.03,
    n_informative=1,
    n_redundant=0,
    n_repeated=0)
plt.scatter(x,y,c=y, cmap = 'rainbow')
plt.title('scatter Plot of Logistic Regression')
plt.show()
x_train, x_test, y_train, y_test = train_test_split(x, y, random_state=1)
model = LogisticRegression()
model.fit(x_train, y_train)
print("coeficent",model.coef_)
print("intercept",model.intercept_)
print("predicted",model.score(x_train,y_train))
y_pred = model.predict(x_test)
print("Confusion matrix")
print(confusion_matrix(y_test, y_pred))
model.score(x_train, y_train)
print("vighnesh kargutkar MLDC 03")
```

Output: -

```
======= RESTART: E:\Sem 2\Big Data\practical\prac3_logisticregression.py
coeficent [[2.89123276]]
intercept [-0.43976278]
predicted 0.9866666666666667
Confusion matrix
[[ 6  0]
 [ 1 18]]
vighnesh kargutkar MLDC 03
```



Practical 4

Aim: - Implement an application that stores big data in MongoDB and manipulate it using python.

Description: -

MongoDB is a popular open-source document-oriented NoSQL database that provides a flexible and scalable solution for storing and managing data. Unlike traditional relational databases, MongoDB stores data in a format called BSON (Binary JSON), which allows for storing and querying complex, hierarchical data structures. MongoDB is widely used in a range of applications, including web and mobile applications, content management systems, real-time analytics, IoT platforms, and more. Its flexible data model, scalability, and performance make it a popular choice for developers working with modern data-driven applications.

Method: -

1. `MongoClient`: This class from the `pymongo` library is used to establish a connection to the MongoDB server. In this case, it connects to the server running on the local machine at the default port 27017.
2. `client.get_database()`: This method is used to get a reference to a specific database in MongoDB. In this code, it retrieves the "mscit" database from the connected MongoDB server.
3. `records.count_documents({})`: This method is used to count the number of documents in the "student" collection. It passes an empty filter `{}` to count all documents.
4. `records.find()`: This method returns a cursor that iterates over all the documents in the "student" collection. It is used in a loop to print each document.
5. `records.insert_one()`: This method is used to insert a single document into the "student" collection. It takes a dictionary as input, which represents the document to be inserted.
6. `records.update_one()`: This method is used to update a single document in the "student" collection. It takes a query dictionary to match the document to be updated and a new values dictionary to specify the changes to be made.
7. `records.delete_one()`: This method is used to delete a single document from the "student" collection. It takes a query dictionary to match the document to be deleted.
8. `records.insert_many()`: This method is used to insert multiple documents into the "student" collection. It takes a list of dictionaries as input, where each dictionary represents a document to be inserted.
9. `records.update_many()`: This method is used to update multiple documents in the "student" collection. It takes a query dictionary to match the documents to be updated and a new values dictionary to specify the changes to be made
10. `records.delete_many()`: This method is used to delete multiple documents from the "student" collection. It takes a query dictionary to match the documents to be deleted.

A] insert_one, update_one, delete_one

Code:

prac4a.py - E:/Sem 2/Big Data/practical/prac4a.py (3.10.10)

File Edit Format Run Options Window Help

```
from pymongo import MongoClient
client=MongoClient('localhost:27017')
db=client.get_database('mscit')
records=db.student
print("Count of records ",records.count_documents({}))
print("Records")
for v in records.find():
    print(v)

print("Inserting one record")
insertquery={"FirstName":"Rohit","RollNo":4,"Age":21,"Subject":"BDA"}
records.insert_one(insertquery)
for v in records.find():
    print(v)

print("Update one record")
query={"RollNo":2}
newvalues={"$set":{"Age":22}}
records.update_one(query,newvalues)
for v in records.find():
    print(v)

print("Delete one record")
records.delete_one({"RollNo":4})
for v in records.find():
    print(v)
print("Vighnesh Kargutkar MLDC 3")
```

Output: -

```
================ RESTART: E:/Sem 2/Big Data/practical/prac4a.py ================
Count of records  2
Records
{'_id': ObjectId('6467b1841d177599b77cd96f'), 'FirstName': 'Vighnesh', 'RollNo': 3, 'Age': 21, 'Subject': 'DS'}
{'_id': ObjectId('6467b1a31d177599b77cd970'), 'FirstName': 'Siddhi', 'RollNo': 2, 'Age': 22, 'Subject': 'DS'}
Inserting one record
{'_id': ObjectId('6467b1841d177599b77cd96f'), 'FirstName': 'Vighnesh', 'RollNo': 3, 'Age': 21, 'Subject': 'DS'}
{'_id': ObjectId('6467b1a31d177599b77cd970'), 'FirstName': 'Siddhi', 'RollNo': 2, 'Age': 22, 'Subject': 'DS'}
{'_id': ObjectId('6467c1aeebf65c303bbde922'), 'FirstName': 'Rohit', 'RollNo': 4, 'Age': 21, 'Subject': 'BDA'}
Update one record
{'_id': ObjectId('6467b1841d177599b77cd96f'), 'FirstName': 'Vighnesh', 'RollNo': 3, 'Age': 21, 'Subject': 'DS'}
{'_id': ObjectId('6467b1a31d177599b77cd970'), 'FirstName': 'Siddhi', 'RollNo': 2, 'Age': 22, 'Subject': 'DS'}
{'_id': ObjectId('6467c1aeebf65c303bbde922'), 'FirstName': 'Rohit', 'RollNo': 4, 'Age': 21, 'Subject': 'BDA'}
Delete one record
{'_id': ObjectId('6467b1841d177599b77cd96f'), 'FirstName': 'Vighnesh', 'RollNo': 3, 'Age': 21, 'Subject': 'DS'}
{'_id': ObjectId('6467b1a31d177599b77cd970'), 'FirstName': 'Siddhi', 'RollNo': 2, 'Age': 22, 'Subject': 'DS'}
Vighnesh Kargutkar MLDC 3
```

B] insert_many, update_many, delete_many

Code: -

prac4b.py - E:/Sem 2/Big Data/practical/prac4b.py (3.10.10)

File Edit Format Run Options Window Help

```
from pymongo import MongoClient
client=MongoClient('localhost:27017')
db=client.get_database('mscit')
records=db.student
print("Count of records ",records.count_documents({}))
print("Records")
for v in records.find():
    print(v)

print("Inserting manu record")
insertquery=[{"FirstName":"Joyel","RollNo":4,"Age":22,"Subject":"CC"},
             {"FirstName":"Jake","RollNo":5,"Age":22,"Subject":"CC"},
             {"FirstName":"Rachel","RollNo":6,"Age":21,"Subject":"MSA"}]
records.insert_many(insertquery)
for v in records.find():
    print(v)

print("Update many record")
query={"Subject":"CC"}
newvalues={"$set":{"Subject":"MN"}}
records.update_many(query,newvalues)
for v in records.find():
    print(v)

print("Delete many record")
records.delete_many({"Subject":"MN"})
for v in records.find():
    print(v)
print("Vighnesh Kargutkar MLDC 3")
```

Output: -

```
========== RESTART: E:/Sem 2/Big Data/practical/prac4b.py ==========
Count of records  2
Records
{'_id': ObjectId('6467b1841d177599b77cd96f'), 'FirstName': 'Vighnesh', 'RollNo': 3, 'Age': 21, 'Subject': 'DS'}
{'_id': ObjectId('6467b1a31d177599b77cd970'), 'FirstName': 'Siddhi', 'RollNo': 2, 'Age': 22, 'Subject': 'DS'}
Inserting manu record
{'_id': ObjectId('6467b1841d177599b77cd96f'), 'FirstName': 'Vighnesh', 'RollNo': 3, 'Age': 21, 'Subject': 'DS'}
{'_id': ObjectId('6467b1a31d177599b77cd970'), 'FirstName': 'Siddhi', 'RollNo': 2, 'Age': 22, 'Subject': 'DS'}
{'_id': ObjectId('6467c2862257c47037b5d8aa'), 'FirstName': 'Joyel', 'RollNo': 4, 'Age': 22, 'Subject': 'CC'}
{'_id': ObjectId('6467c2862257c47037b5d8ab'), 'FirstName': 'Jake', 'RollNo': 5, 'Age': 22, 'Subject': 'CC'}
{'_id': ObjectId('6467c2862257c47037b5d8ac'), 'FirstName': 'Rachel', 'RollNo': 6, 'Age': 21, 'Subject': 'MSA'}
Update many record
{'_id': ObjectId('6467b1841d177599b77cd96f'), 'FirstName': 'Vighnesh', 'RollNo': 3, 'Age': 21, 'Subject': 'DS'}
{'_id': ObjectId('6467b1a31d177599b77cd970'), 'FirstName': 'Siddhi', 'RollNo': 2, 'Age': 22, 'Subject': 'DS'}
{'_id': ObjectId('6467c2862257c47037b5d8aa'), 'FirstName': 'Joyel', 'RollNo': 4, 'Age': 22, 'Subject': 'MN'}
{'_id': ObjectId('6467c2862257c47037b5d8ab'), 'FirstName': 'Jake', 'RollNo': 5, 'Age': 22, 'Subject': 'MN'}
{'_id': ObjectId('6467c2862257c47037b5d8ac'), 'FirstName': 'Rachel', 'RollNo': 6, 'Age': 21, 'Subject': 'MSA'}
Delete many record
{'_id': ObjectId('6467b1841d177599b77cd96f'), 'FirstName': 'Vighnesh', 'RollNo': 3, 'Age': 21, 'Subject': 'DS'}
{'_id': ObjectId('6467b1a31d177599b77cd970'), 'FirstName': 'Siddhi', 'RollNo': 2, 'Age': 22, 'Subject': 'DS'}
{'_id': ObjectId('6467c2862257c47037b5d8ac'), 'FirstName': 'Rachel', 'RollNo': 6, 'Age': 21, 'Subject': 'MSA'}
Vighnesh Kargutkar MLDC 3
```

Practical 5

Aim: - Implement SVM Classification Technique.

Description: -

SVM (Support Vector Machine) is a powerful machine learning algorithm for classification. It finds the best decision boundary that separates different classes in the data space by maximizing the margin between classes. SVM uses a subset of training data called support vectors, which lie closest to the decision boundary. It can handle linearly separable data and non-linearly separable data through the use of kernel functions. SVM is effective in high-dimensional spaces, less prone to overfitting, and can accurately classify new, unseen data points. It is a popular choice for various classification tasks due to its versatility and performance.

Methods: -

1. `pd.read_csv('E:\Sem 2\Big Data\practical\social.csv')`: This method from the pandas library is used to read the CSV file located at the specified path and load it into a pandas DataFrame. The DataFrame represents the input data.
2. `df.iloc[:,[2,3]]` and `df.iloc[:,4]`: These lines extract the features (columns 2 and 3) and the target variable (column 4) from the DataFrame `df`.
3. `train_test_split()`: This function from the sklearn.model_selection module is used to split the dataset into training and testing sets. It takes the features (`x`) and the target variable (`y`) as input and splits them based on the specified test size (25% in this case) and a random seed (random_state).
4. `StandardScaler()`: This class from the sklearn.preprocessing module is used to perform feature scaling on the training and testing data. It standardizes the features by removing the mean and scaling to unit variance.
5. `classifier = SVC(kernel='linear', random_state=0)`: This line creates an instance of the Support Vector Classifier (SVC) from the sklearn.svm module. It uses a linear kernel for classification and sets the random state to 0 for reproducibility.
6. `classifier.fit(x_train, y_train)`: This method is used to train the SVM classifier on the training data. It learns the relationship between the features and the target variable.
7. `classifier.predict(x_test)`: This method predicts the target variable values for the test data using the trained classifier.
8. `metrics.accuracy_score(y_test, y_pred)`: This function from the sklearn.metrics module is used to calculate the accuracy score by comparing the predicted target variable values (`y_pred`) with the actual values (`y_test`).

Code:

prac5_SVM.py - E:/Sem 2/Big Data/practical/prac5_SVM.py (3.10.10)

File Edit Format Run Options Window Help

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
df=pd.read_csv('E:\Sem 2\Big Data\practical\social.csv')
print('Input Data Values ==================================')
print(df)
x=df.iloc[:,[2,3]]
y=df.iloc[:,4]
#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=0.25,random_state=0)
print("Training data:",x_train)
print('********************')
print("Testing data:",x_test)
#Feature scaling
from sklearn.preprocessing import StandardScaler
sc_x=StandardScaler()
x_train=sc_x.fit_transform(x_train)
x_test=sc_x.transform(x_test)
from sklearn.svm import SVC
classifier=SVC(kernel='linear', random_state=0)
classifier.fit(x_train,y_train)
#predicting the test set results
y_pred=classifier.predict(x_test)
print(y_pred)
from sklearn import metrics
print('accuracy score with linear kernel')
print(metrics.accuracy_score(y_test,y_pred))
print("vighnesh kargutkar MLDC 3")
```

Output: -

```
=============== RESTART: E:/Sem 2/Big Data/practical/prac5_SVM.py ==============
Input Data Values ==================================
    userid  gender  age  estimatedsalary  purchased
0      155    male   19            16000          0
1      156    male   22            23000          0
2      157  female   56            44000          1
3      158    male   33            22000          0
4      159   femle   23            22000          0
5      160   femle   54            22000          0
6      161   femle   21            22000          0
7      162   femle   51            22000          0
8      163   femle   22            22000          0
9      164   femle   33            22000          0
10     165   femle   19            44000          1
11     166   femle   22            44000          1
12     167   femle   32            44000          1
13     168   femle   19            44000          1
14     169   femle   43            44000          1
15     170   femle   45            44000          1
16     171   femle   33            44000          1
17     172    male   55            34000          0
18     173    male   33            34000          0
19     174  female   23            34000          0
20     175    male   44            34000          0
21     176    male   34            34000          0
22     177    male   23            22000          0
23     178    male   23            22000          0
24     179  female   44            22000          0
25     180    male   65            22000          0
```

```
Training data:      age  estimatedsalary
13    19             44000
18    33             34000
19    23             34000
16    33             44000
1     22             23000
10    19             44000
25    65             22000
24    44             22000
8     22             22000
6     21             22000
4     23             22000
9     33             22000
7     51             22000
23    23             22000
3     33             22000
0     19             16000
21    34             34000
15    45             44000
12    32             44000
********************
Testing data:      age  estimatedsalary
2     56             44000
20    44             34000
14    43             44000
17    55             34000
5     54             22000
11    22             44000
22    23             22000
[1 0 1 0 0 1 0]
accuracy score with linear kernel
1.0
vighnesh kargutkar MLDC 3
```

Practical 6

Aim: - Implement Decision Tree Classification Technique.

Description: -

Decision Tree Classification: Decision Tree is a popular supervised machine learning algorithm for classification tasks. It creates a tree-like model by recursively splitting the data based on features, aiming to maximize class separation. Each node represents a feature, branches represent feature values, and leaves hold class labels. Decision Trees are interpretable and handle both categorical and numerical data. They can capture non-linear relationships and are often used in ensemble methods like Random Forest. However, Decision Trees are prone to overfitting, so techniques like pruning are used to control complexity. To predict a class label, a sample traverses the tree based on feature values until it reaches a leaf node.

Methods: -

1. `pd.read_csv()`: This method from the pandas library reads the CSV file located at the specified path and loads it into a pandas DataFrame named `df`.
2. `MinMaxScaler()`: This class from the sklearn.preprocessing module is used to scale the feature values between zero and one using the Min-Max scaling technique. It is necessary because the decision tree algorithm can benefit from having scaled features.
3. `DecisionTreeClassifier()`: This class from the sklearn.tree module is used to create an instance of the Decision Tree Classifier model.
4. `train_test_split()`: This function from the sklearn.model_selection module is used to split the dataset into training and testing sets. It takes the features (`X`) and the target variable (`Y`) as input and splits them based on the specified test size (25% in this case).
5. `fit()`: This method is used to train the Decision Tree Classifier model on the scaled training data. It learns the relationship between the features and the target variable.
6. `predict()`: This method is used to predict the target variable values for the scaled test data using the trained model.
7. `plt.scatter()`: This method from the matplotlib.pyplot module is used to create a scatter plot. It visualizes the test data points with their corresponding class labels, where red dots represent class 0 (not purchased) and blue dots represent class 1 (purchased).
8. `score()`: This method is used to calculate the accuracy score of the model by comparing the predicted target variable values (`Y_predict`) with the actual values (`Y_test`).

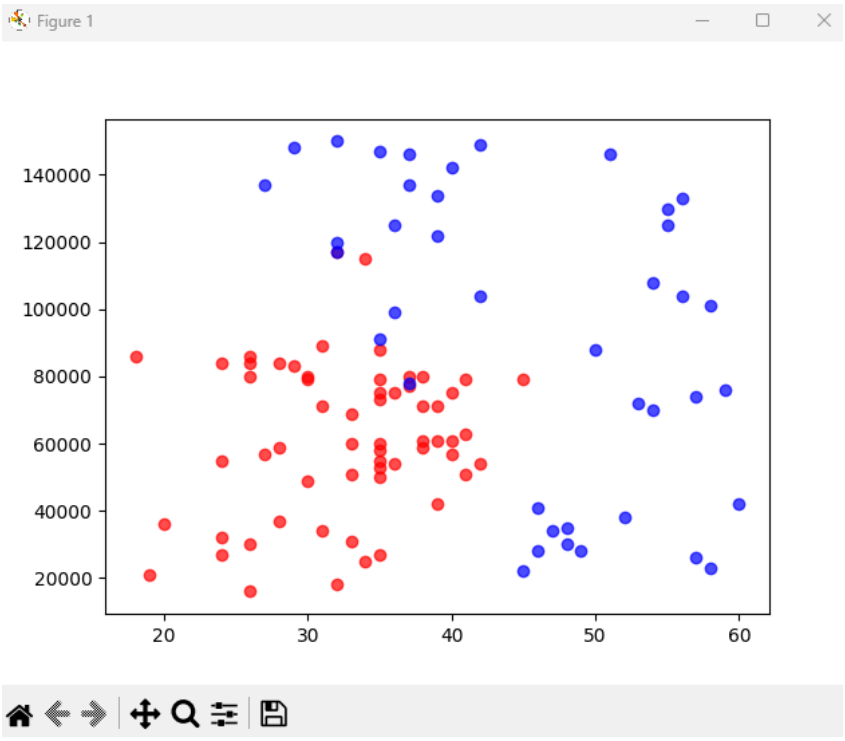
Code:

prac6_DT.py - E:/Sem 2/Big Data/practical/prac6_DT.py (3.10.10)

File Edit Format Run Options Window Help

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

#to range the values between zero and one MinMaxScalaer is needed
from sklearn.preprocessing import MinMaxScaler
from sklearn.tree import DecisionTreeClassifier
#READ DATASET
df=pd.read_csv('E:\Sem 2\Big Data\practical\Social_Network_Ads.csv')
print(df)
X=df[['Age','EstimatedSalary']]
print(X)
Y=df['Purchased']
print(Y)
#split dataset into X_train X_test,y_train and y_test
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)
print(X_train.shape, Y_train.shape, X_test.shape, Y_test.shape)

#feature scaling
SS=MinMaxScaler()
SS.fit(X_train)
X_train_scaled=SS.transform(X_train)
SS.fit(X_test)
X_test_scaled=SS.transform(X_test)


#implement decision tree
Model_DT=DecisionTreeClassifier()
#fit is for training the model
Model_DT.fit(X_train_scaled, Y_train)
Y_predict=Model_DT.predict(X_test_scaled)
plt.scatter(X_test[Y_test==0]['Age'],X_test[Y_test==0]['EstimatedSalary'],c='red',alpha=0.7)

plt.scatter(X_test[Y_test==1]['Age'],X_test[Y_test==1]['EstimatedSalary'],c='blue',alpha=0.7)
plt.show()
#accuracy level of the model
print(Model_DT.score(X_test_scaled, Y_test))
print("vighnesh kargutkar MLDC 3")
```

Output: -



```
=============== RESTART: E:/Sem 2/Big Data/practical/prac6_DT.py ===============
      User ID  Gender  Age  EstimatedSalary  Purchased
0    15624510    Male   19            19000          0
1    15810944    Male   35            20000          0
2    15668575  Female   26            43000          0
3    15603246  Female   27            57000          0
4    15804002    Male   19            76000          0
..        ...     ...  ...              ...        ...
395  15691863  Female   46            41000          1
396  15706071    Male   51            23000          1
397  15654296  Female   50            20000          1
398  15755018    Male   36            33000          0
399  15594041  Female   49            36000          1

[400 rows x 5 columns]
     Age  EstimatedSalary
0     19            19000
1     35            20000
2     26            43000
3     27            57000
4     19            76000
..   ...              ...
395   46            41000
396   51            23000
397   50            20000
398   36            33000
399   49            36000

[400 rows x 2 columns]
0      0
1      0
2      0
3      0
4      0
      ..
395    1
396    1
397    1
398    0
399    1
Name: Purchased, Length: 400, dtype: int64
(300, 2) (300,) (100, 2) (100,)
0.85
vighnesh kargutkar MLDC 3
```

Practical 7

Aim: - Text Analysis Implementation

Description: -

Text analysis, also known as text mining or natural language processing (NLP), is the process of extracting meaningful insights from textual data. It involves several key steps, including data preprocessing, text representation, feature extraction, and applying machine learning algorithms. Data is cleaned and transformed into numerical representations suitable for analysis. Additional features like sentiment analysis and named entity recognition can be extracted. Machine learning algorithms, such as Naive Bayes or Support Vector Machines (SVM), are utilized for tasks like text classification. Text analysis enables the understanding of text-based data, supporting applications in sentiment analysis, topic modeling, document categorization, and information retrieval.

Methods: -

1. sent_tokenize(text): This method from the NLTK library is used for sentence tokenization. It takes a text input and splits it into individual sentences.
2. word_tokenize(text): This method is used for word tokenization. It takes a text input and splits it into individual words or tokens.
3. FreqDist(tokenized_word): This method is used to calculate the frequency distribution of words in the tokenized text. It returns a frequency distribution object that can be used to get the count of each word.
4. most_common(n): This method is used to get the n most common words and their respective counts from the frequency distribution. In the provided code, `fdist.most_common(3)` returns the three most common words and their counts.
5. fdist.plot(): This method is used to plot the frequency distribution as a histogram. It shows the distribution of word frequencies in the text.
6. set(stopwords.words("english")): This method is used to get a set of stopwords from the NLTK library for the English language. Stopwords are common words (e.g., "the", "is", "and") that are often removed in text analysis as they don't carry significant meaning.

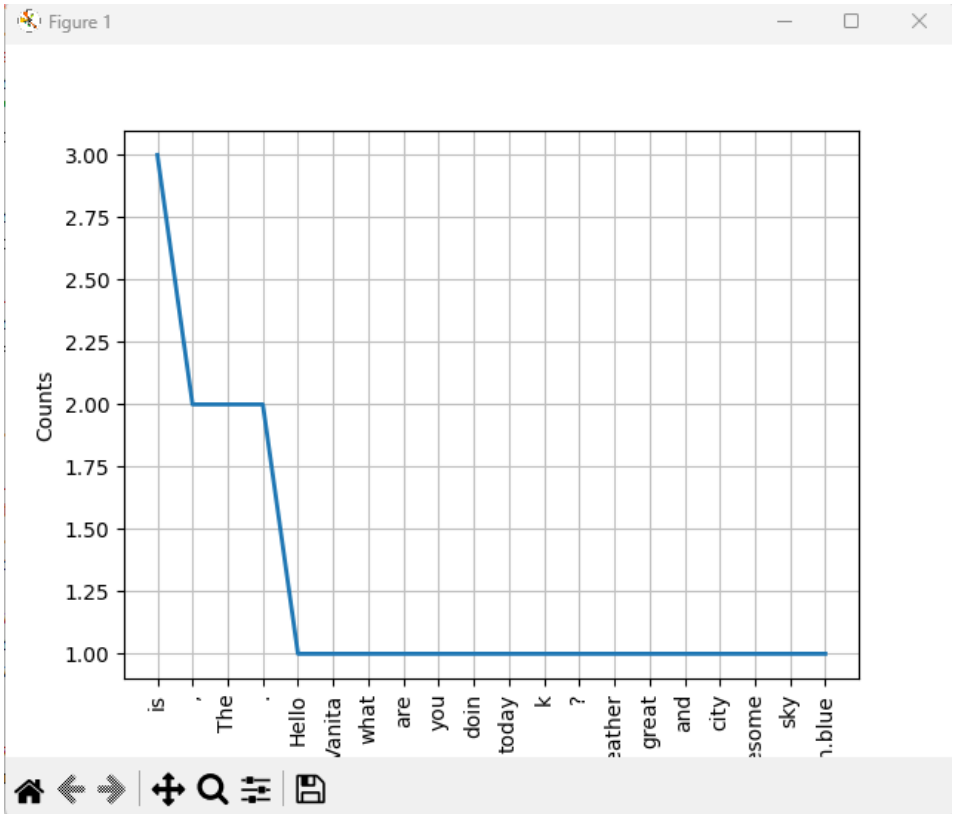
Code: -

prac7_TXT.py - E:/Sem 2/Big Data/practical/prac7_TXT.py (3.10.10)

File Edit Format Run Options Window Help

```
import nltk
#nltk.download('all')
#sentence tokenization
from nltk.tokenize import sent_tokenize
text="Hello Miss.Vanita, what are you doin today k? The weather is great, and city is awesome. The sky is pinkish.blue."
tokenized_sent=sent_tokenize(text)
print(tokenized_sent)
#word tokenization
from nltk.tokenize import word_tokenize
tokenized_word=word_tokenize(text)
print(tokenized_word)
#Frequency Distribution
from nltk.probability import FreqDist
fdist=FreqDist(tokenized_word)
print(fdist)

fdist.most_common(3)

#frequency distribution plot
import matplotlib.pyplot as plt
fdist.plot()
plt.show()

#stopwords
from nltk.corpus import stopwords
stop_words=set(stopwords.words("english"))
print(stop_words)

#removing stopwords
filtered_sent=[]
for w in tokenized_word:
    if w not in stop_words:
        filtered_sent.append(w)
print("#######")
print(w)
print("Tokenized sentence:",tokenized_word)
print("Filtered sentence:",filtered_sent)
print("Vighnesh Kargutkar MLDC 3")
```

Output: -



```
=============== RESTART: E:/Sem 2/Big Data/practical/prac7_TXT.py ==============
['Hello Miss.Vanita, what are you doin today k?', 'The weather is great, and cit
y is awesome.', 'The sky is pinkish.blue.']
['Hello', 'Miss.Vanita', ',', 'what', 'are', 'you', 'doin', 'today', 'k', '?', '
The', 'weather', 'is', 'great', ',', 'and', 'city', 'is', 'awesome', '.', 'The',
 'sky', 'is', 'pinkish.blue', '.']
<FreqDist with 20 samples and 25 outcomes>
{'no', 'am', 'and', 'too', 'are', "hadn't", 'his', "isn't", 'once', 'here', 'tho
se', "you've", 'being', 'wouldn', 'other', "wouldn't", 'because', 'most', 'had',
 'do', 'each', 'below', 'yourself', 'to', 'where', 'by', 'further', 'd', 'aren',
 'the', 'been', 'into', 's', 'only', 'them', 'himself', 'weren', 'hasn', 'above'
, 'up', 'that', 'has', 'again', 'own', 'm', 'whom', "won't", 'as', 'off', 'which
', 'about', 'him', 'isn', 'when', 'now', "haven't", 'until', 'he', 'its', 'hadn'
, 'same', 'wasn', 'me', 'a', 'doesn', 'couldn', 'of', 'any', 'you', 'have', 'not
', 'few', 'on', 'don', 'with', 'herself', 'we', 'i', 'or', 'hers', 'before', 'mu
stn', 'these', 'all', 'at', 'having', 'her', 'such', 'll', "it's", 've', "wasn't
", 'how', 'y', 'itself', "weren't", 'down', "shouldn't", 'this', 'ma', 'was', 'a
gainst', 'yourselves', 'nor', 'out', "didn't", 'won', 'doing', 'haven', 'your',
'their', "needn't", 'were', 'then', 'over', 'so', 'through', 'who', 'ain', 'why'
, 'ours', "you'd", 'from', 'in', 'an', 'both', 'didn', 'there', 'ourselves', "mi
ghtn't", "you'll", 'can', 'is', 'between', "that'll", 'during', 'more', 'shouldn
', 'some', 'o', "doesn't", 'under', 'our', 'does', 'yours', 'very', 'if', "could
n't", 'be', 't', 're', "hasn't", 'they', 'will', 'for', "don't", "you're", "must
n't", 'needn', 'myself', 'should', "she's", 'she', 'what', 'but', 'mightn', 'aft
er', 'it', 'theirs', "should've", "shan't", 'did', 'than', 'themselves', 'while'
, 'shan', "aren't", 'my', 'just'}
#######
.
Tokenized sentence: ['Hello', 'Miss.Vanita', ',', 'what', 'are', 'you', 'doin',
'today', 'k', '?', 'The', 'weather', 'is', 'great', ',', 'and', 'city', 'is', 'a
wesome', '.', 'The', 'sky', 'is', 'pinkish.blue', '.']
Filtered sentence: ['Hello', 'Miss.Vanita', ',', 'doin', 'today', 'k', '?', 'The
', 'weather', 'great', ',', 'city', 'awesome', '.', 'The', 'sky', 'pinkish.blue'
, '.']
Vighnesh Kargutkar MLDC 3
```

Practical 8

Aim: - Sentiment Analysis

Description: -

Sentiment Analysis, also known as opinion mining, is a text analysis technique used to determine the sentiment or emotion expressed in a piece of text. It involves analyzing the subjective information present in the text to classify it as positive, negative, or neutral. Sentiment analysis can be performed on various types of text data, such as customer reviews, social media posts, and feedback comments. It has applications in various fields, including market research, brand monitoring, customer feedback analysis, and social media sentiment tracking. Machine learning algorithms, such as Naive Bayes, Support Vector Machines (SVM), or deep learning models, are commonly used in sentiment analysis to automatically classify the sentiment of text data and provide valuable insights for decision-making.

Methods: -

1. read_csv(file_path): This method from the pandas library is used to read a CSV file and create a DataFrame. It takes the file path as input and returns a DataFrame containing the data from the CSV file.
2. head(): This method is used to display the first few rows of the DataFrame. In the provided code, it is used to display the first few rows of the "at" DataFrame.
3. plt.rcParams['figure.figsize']: This statement is used to get the current figure size parameters of the matplotlib.pyplot module.
4. plt.rcParams['figure.figsize'] = plot_size: This statement is used to set the figure size parameters of the matplotlib.pyplot module to the specified values in the "plot_size" list.
5. at.airline.value_counts().plot(kind='pie', autopct='1.0'): This line of code is used to create a pie chart representing the count of each unique value in the "airline" column of the DataFrame "at". The "kind" parameter is set to 'pie', and the "autopct" parameter is set to '1.0' to display the percentage values as whole numbers.
6. plt.show(): This method is used to display the plot generated by matplotlib.
   airline_sentiment.plot(kind='bar'): This line of code is used to create a bar chart from the "airline_sentiment" DataFrame. The "kind" parameter is set to 'bar' to create a vertical bar chart.
7. sns.barplot(x='airline_sentiment', y='airline_sentiment_confidence', data=at)**: This line of code uses the seaborn library to create a bar plot. It takes the "airline_sentiment" column as the x-axis and the "airline_sentiment_confidence" column as the y-axis from the "at" DataFrame.

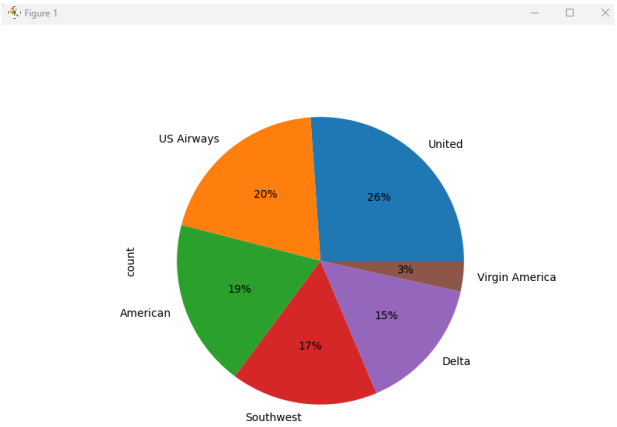
Code: -

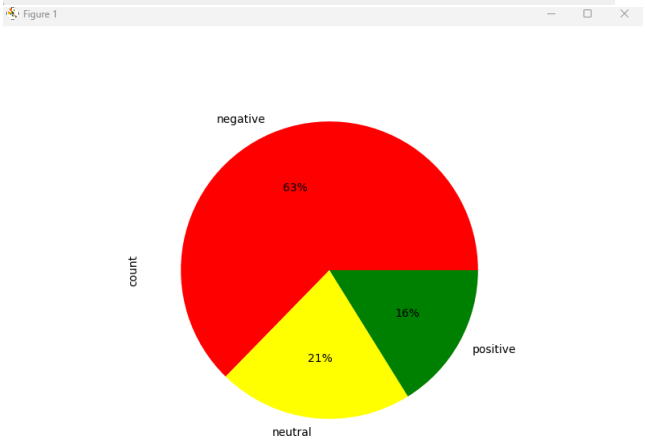
prac8_sentiment.py - E:/Sem 2/Big Data/practical/prac8_sentiment.py (3.10.10)

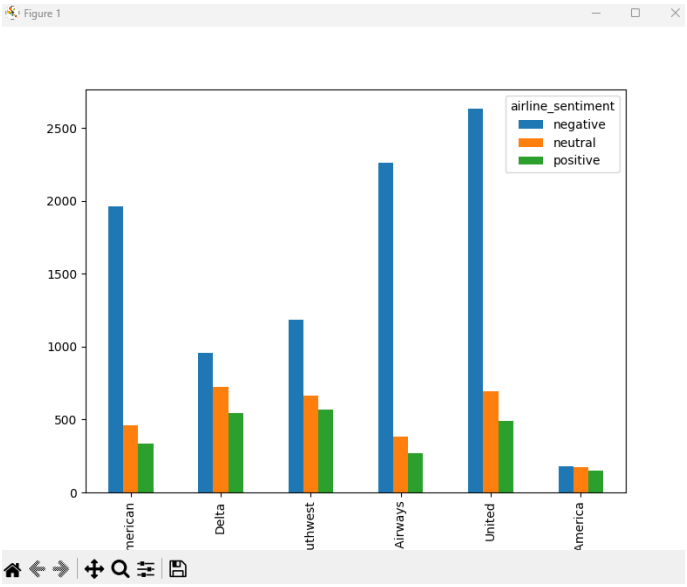
File Edit Format Run Options Window Help

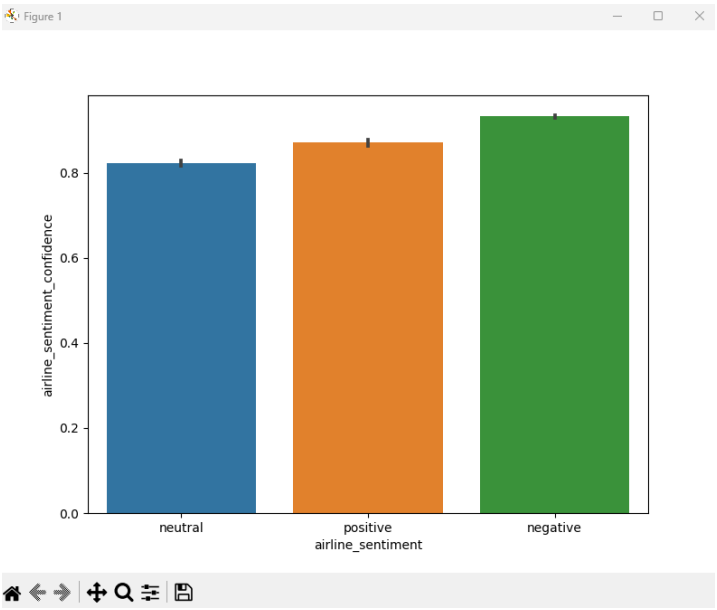
```
import numpy as np
import pandas as pd
import re
import nltk
import matplotlib.pyplot as plt
import seaborn as sns

at=pd.read_csv(r'E:\Sem 2\Big Data\practical\Tweets.csv')
at.head()
plot_size=plt.rcParams['figure.figsize']
print(plot_size[0])
print(plot_size[1])
plot_size[0]=8
plot_size[1]=6
plt.rcParams['figure.figsize']=plot_size
at.airline.value_counts().plot(kind='pie',autopct='%1.0f%%')
plt.show()
at.airline_sentiment.value_counts().plot(kind='pie',autopct='%1.0f%%',colors=['red','yellow','green'])
plt.show()
airline_sentiment=at.groupby(['airline','airline_sentiment']).airline_sentiment.count().unstack()
airline_sentiment.plot(kind='bar')
plt.show()
sns.barplot(x='airline_sentiment',y='airline_sentiment_confidence',data=at)
plt.show()
print("Vighnesh Kargutkar MLDC 3")
```

Output: -










```
============ RESTART: E:/Sem 2/Big Data/practical/prac8_sentiment.py ==
6.4
4.8
Vighnesh Kargutkar MLDC 3
```

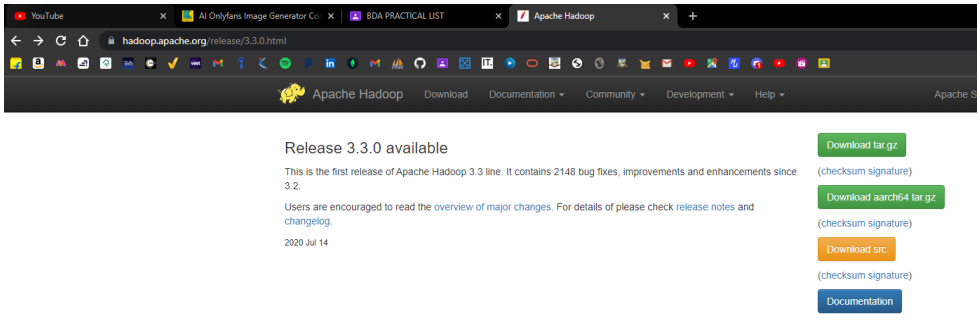
Practical 9

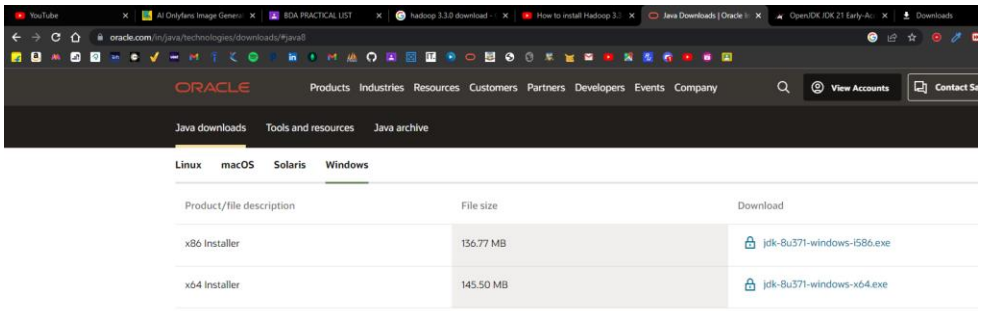
Aim: - Install, configure and run Hadoop and HDFS

Step 1: Download Hadoop 3.3.0 and JDK from the link

https://hadoop.apache.org/release/3.3.0.html

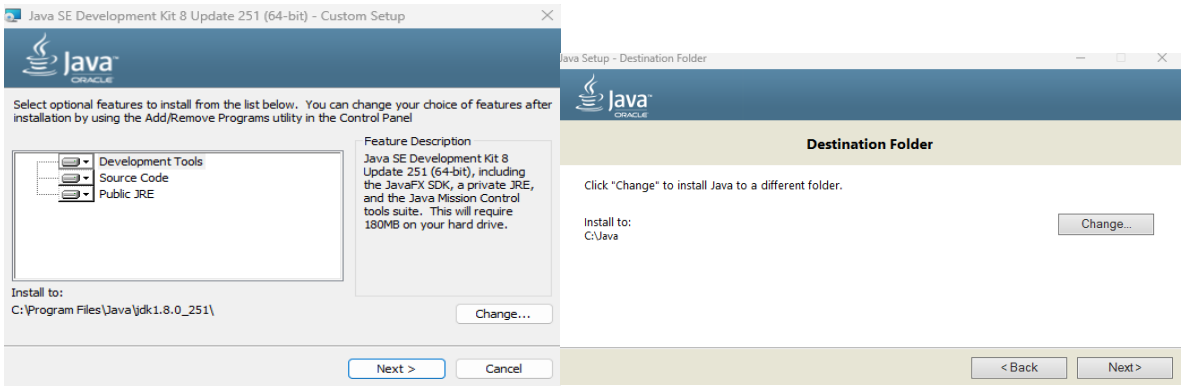
https://www.oracle.com/in/java/technologies/downloads/#java8



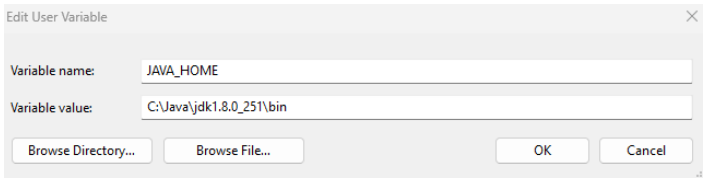


Setp2: -

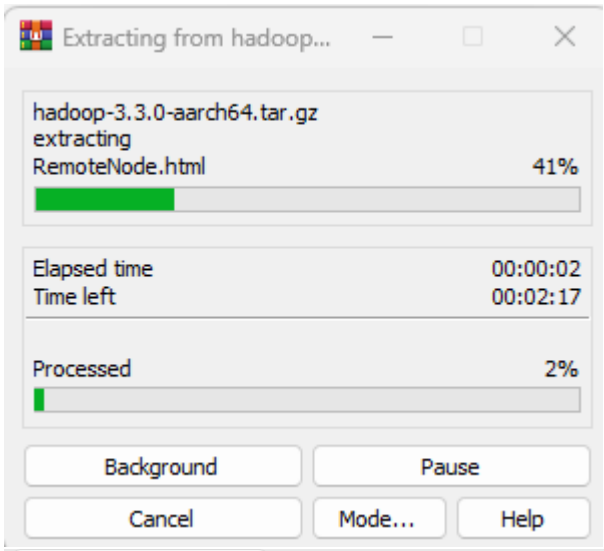
Open exe file and click on next then change path to Java

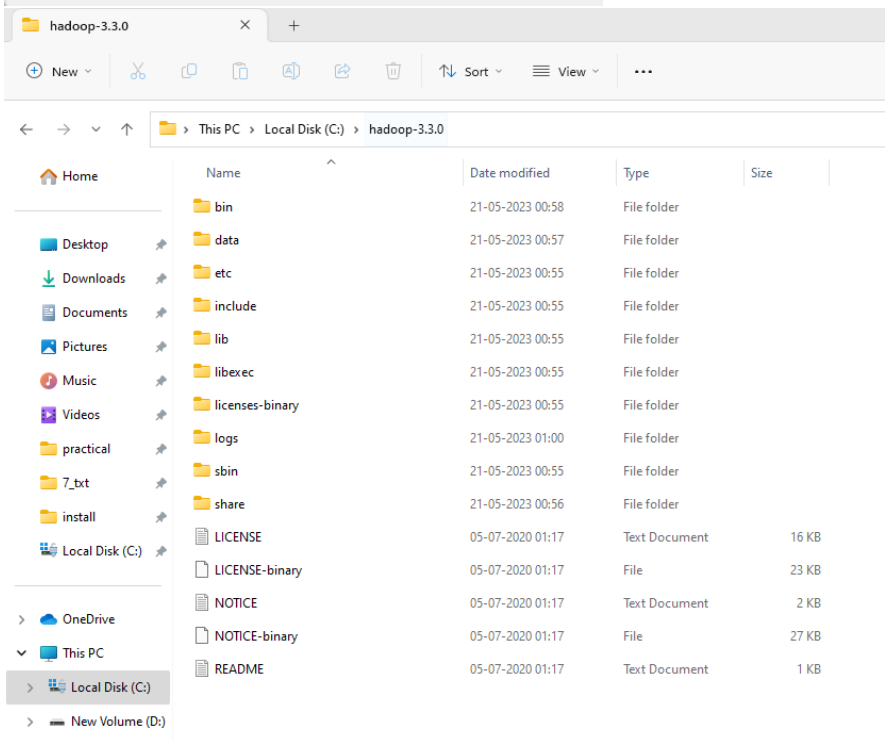


Set environment variable path



Step3: -Extract downloaded Hadoop folder and paste in C:/ Drive





Step4: -

Make changes in following file core-site.xml,hdfs-site.xml,mapred.xml,yarn-site.xml and Hadoop-env.cmd

Before making changes add data folder in Hadoop file then inside Hadoop folder add two new folder datanode and namenode

1] core-site.xml

Add following code in configuration

```
<configuration>
 <property>
```

```
 <name>fs.default.name</name>
 <value>hdfs://localhost:9000</value>
 </property>
</configuration>
```

2] hdfs-site.xml

Add following code in configuration

```
<configuration>
 <property>
<name>dfs.replication</name>
 <value>1</value>
 </property>
 <property>
 <name>dfs.namenode.name.dir</name>
 <value>file:///C:/hadoop-3.3.0/data/namenode</value>
 </property>
 <property>
 <name>dfs.datanode.data.dir</name>
 <value>file:///C:/hadoop-3.3.0/data/datanode</value>
 </property>
</configuration>
```

3] mapred-site.xml

Add following code in configuration

```
<configuration>
 <property>
 <name>mapreduce.framework.name</name>
 <value>yarn</value>
 </property>
</configuration>
```

4] yarn-site.xml

Add following code in configuration

```
<configuration>
 <property>
 <name>yarn.nodemanager.aux-services</name>
 <value>mapreduce_shuffle</value>
 </property>
 <property>
 <name>yarn.nodemanager.auxservices.mapreduce.shuffle.class</name>
<value>org.apache.hadoop.mapred.ShuffleHandler</value>
 </property>
```

```
</configuration>
```

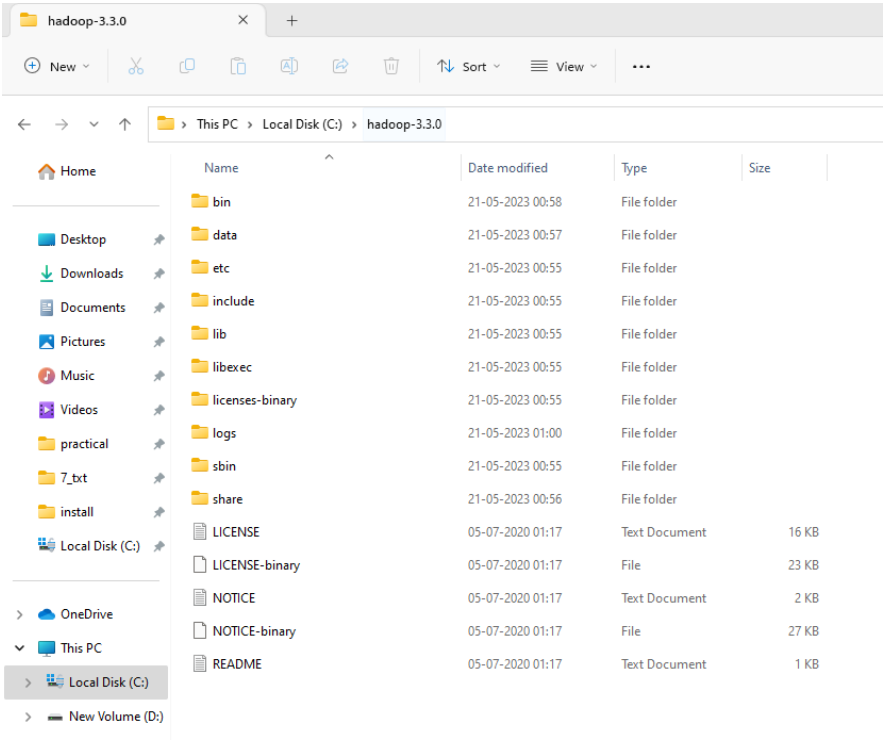
5] Hadoop-env.cmd

Change path

```
set JAVA_HOME=C:\Java\jdk1.8.0_251
```

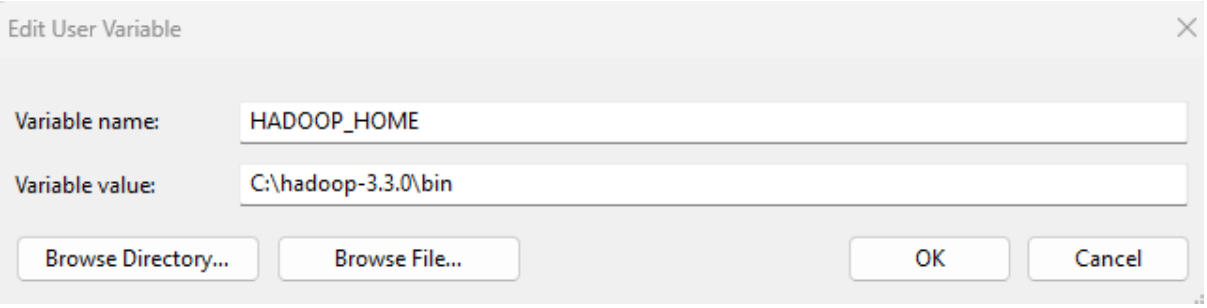
step5: -download the configuration file

extract file - > delete the existing bin folder ->paste extracted bin folder in hadoop file

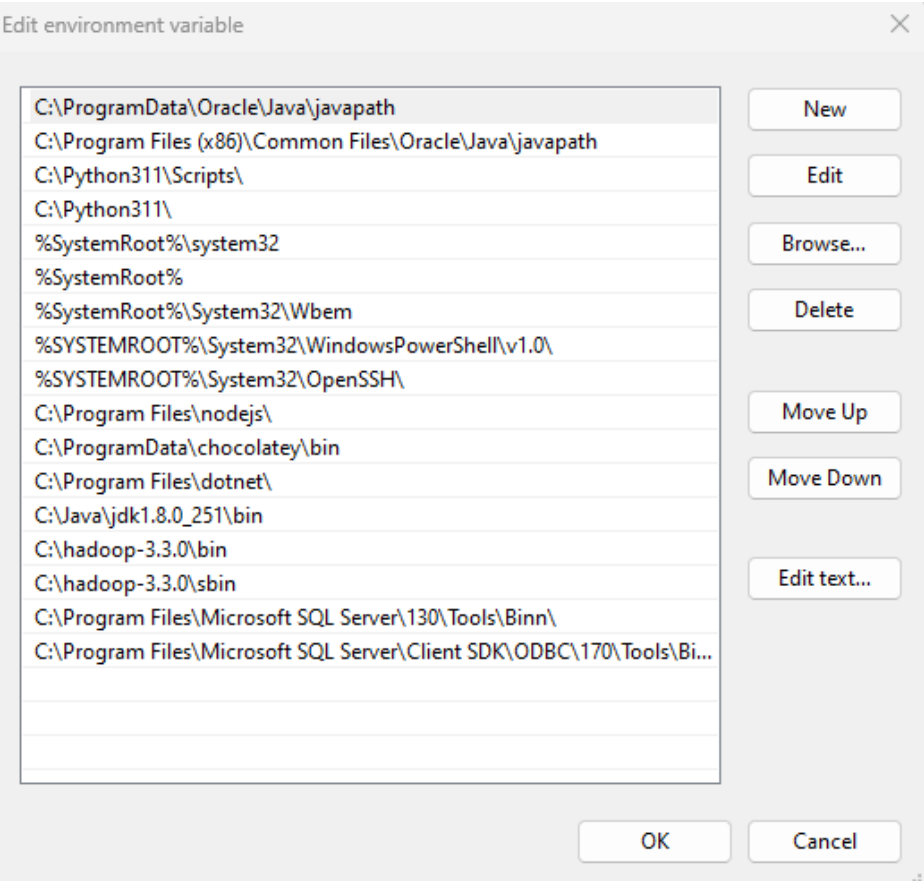


step6: -

add Hadoop home path



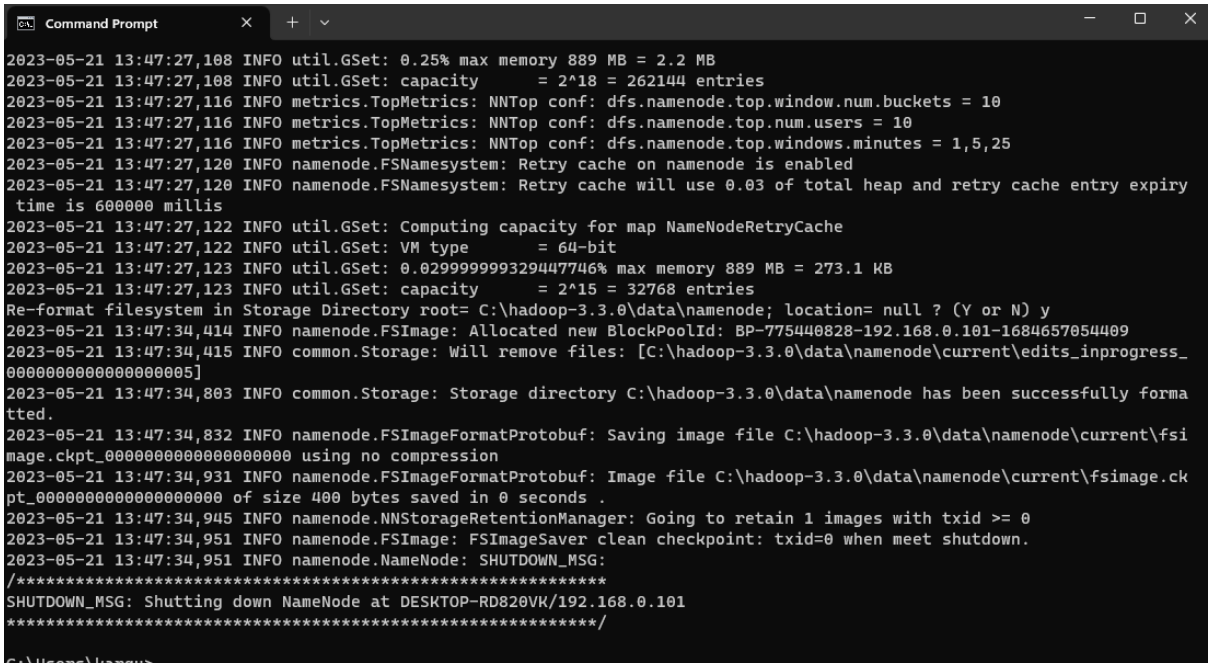
Add Hadoop bin and sbin path



Step7: -

Open cmd and type

hdfs namenode -format



Hadoop have successfully

Practical 10

Aim: - Basic Commands of HDFS

Commands: -

1] start-all: -

Start all Hadoop daemons, the namenode, datanode, the jobtracker and the tasktracker.

2] jps: -

This command is used to check all Hadoop daemons are properly running.This is basic check to see if all the Hadoop services are running or not before going forward.

3] mkdir: -

This command creates directory in HDFS if it does not already exist.

4] ls: -

This command shows the list of file/content in a directory.

5] touchz: -

This command creates a file in HDFS with file size equals to 0 byte

6] copyfromlocal: -

Hadoop copyFromLocal command is used to copy the file from your local file system to the HDFS.

7] cat: -

This command reads the file in HDFS and displays the content of the file.

8] put: -

This command is used to copy the file from the local file system to the Hadoop HDFS file system.

9] copytolocal: -

This command is used to copy the data from HDFS to the local filesystem.

10] get: -

This command copies files from HDFS file system to local file system.

11] mv: -

This command moves the files or directory from the source to a destination within HDFS.

12] rm:-

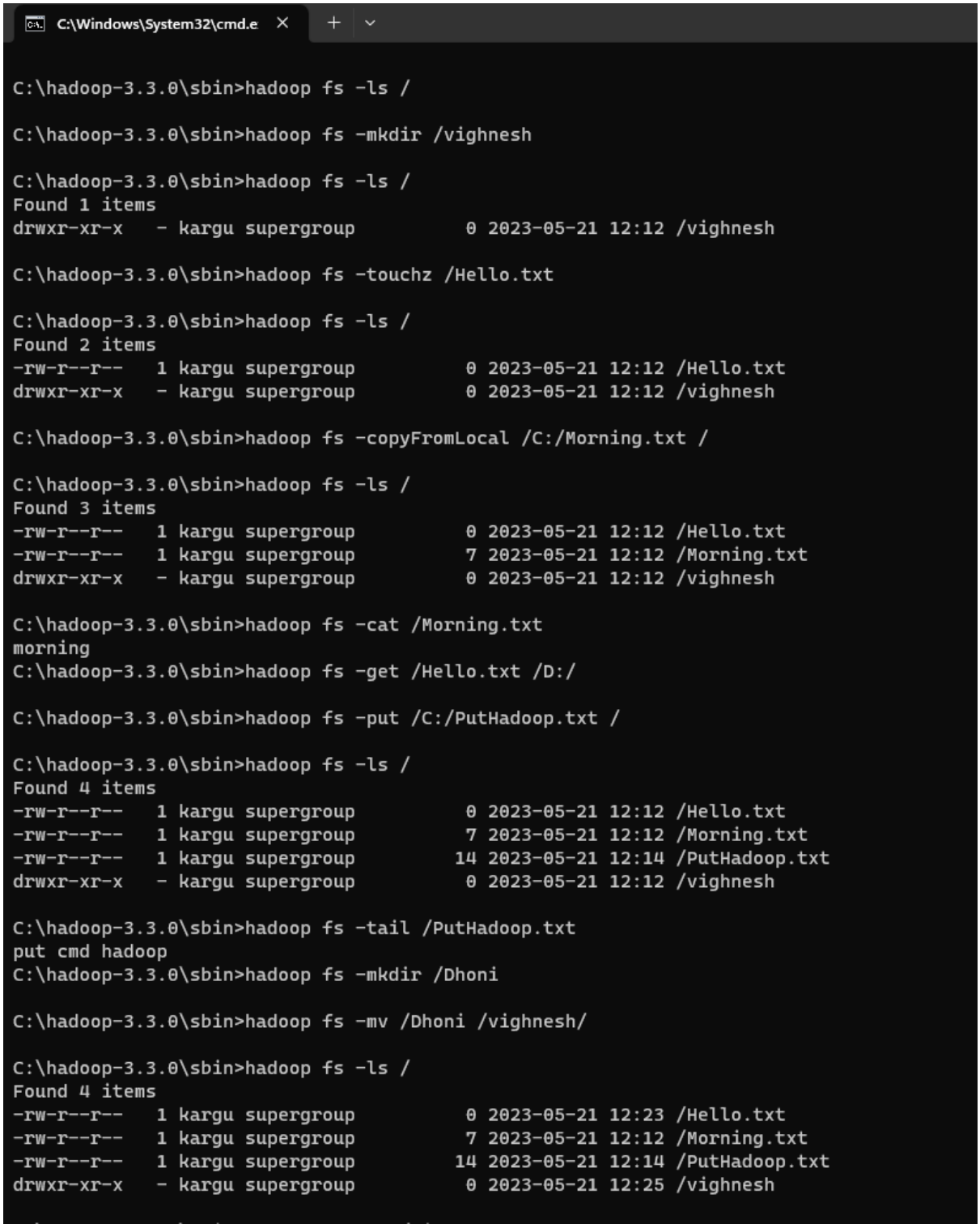
This command removes a file from HDFS.

13] rmr: -

This command removes a file from HDFS and can be used to delete

14] tail: -

This command shows the last 1KB of the file on the console



```
C:\hadoop-3.3.0\sbin>hadoop fs -ls /vighnesh
Found 1 items
drwxr-xr-x   - kargu supergroup          0 2023-05-21 12:24 /vighnesh/Dhoni

C:\hadoop-3.3.0\sbin>hadoop fs -cp -p /Hello.txt /vighnesh/Dhoni/

C:\hadoop-3.3.0\sbin>hadoop fs -ls /vighnesh/Dhoni/
Found 1 items
-rw-r--r--   1 kargu supergroup          0 2023-05-21 12:23 /vighnesh/Dhoni/Hello.txt

C:\hadoop-3.3.0\sbin>hadoop fs -rmr /vighnesh
rmr: DEPRECATED: Please use '-rm -r' instead.
Deleted /vighnesh

C:\hadoop-3.3.0\sbin>hadoop fs -ls /
Found 3 items
-rw-r--r--   1 kargu supergroup          0 2023-05-21 12:23 /Hello.txt
-rw-r--r--   1 kargu supergroup          7 2023-05-21 12:12 /Morning.txt
-rw-r--r--   1 kargu supergroup         14 2023-05-21 12:14 /PutHadoop.txt

C:\hadoop-3.3.0\sbin>hadoop fs -rm /Hello.txt
Deleted /Hello.txt

C:\hadoop-3.3.0\sbin>hadoop fs -ls /
Found 2 items
-rw-r--r--   1 kargu supergroup          7 2023-05-21 12:12 /Morning.txt
-rw-r--r--   1 kargu supergroup         14 2023-05-21 12:14 /PutHadoop.txt

C:\hadoop-3.3.0\sbin>
```