INDEX

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	Implementing Text Analysis Technique	25-04-2023	
8	Implementing Sentimental Analysis Technique	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 NO. :01

Aim: Implementing k-means classification Technique.

Description :-

- K-means is an unsupervised classification algorithm, also called clusterization, that groups objects into k groups based on their characteristics.
- The grouping is done minimizing the sum of the distances between each object and the group or cluster centroid.
- The algorithm will categorize the items into k groups of similarity.
- To calculate that similarity, we will use the Euclidean distance as measurement.
- The algorithm works as follows: First, we initialize k points, called means, randomly. We categorize each item to its closest mean and we update the mean's coordinates, which are the averages of the items categorized in that mean so far. We repeat the process for a given number of iterations and at the end, we have our clusters.

Methods :-

1. numpy.random.randint(low, high=None, size=None) : Return random integers from low (inclusive) to high (exclusive).
2. matplotlib.pyplot.figure(figsize=(x,y)) :Create a new figure, or activate an existing figure.
3. matplotlib.pyplot.scatter(x, y, color ='k') : With Pyplot, you can use the scatter() function to draw a scatter plot.
4. matplotlib.pyplot.xlim(*args, **kwargs) : The xlim() function in pyplot module of matplotlib library is used to get or set the xlimits of the current axes.
5. matplotlib.pyplot.ylim(*args, **kwargs) : The ylim() function in pyplot module of matplotlib library is used to get or set the ylimits of the current axes.
6. matplotlib.pyplot.show() : This method is used to display the graph.
7. df.head() : This method is used to obtain size of the dataset.

Here we are taking 3 number of clusters i.e., Red, Green, Yellow.

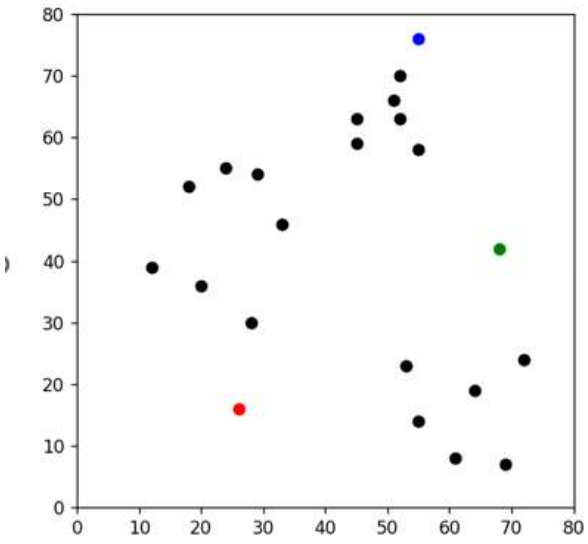
Program Code :

1. Initialize Stage

kmeans algorithm.py - C:\sem2\bda\prac\kmeans algorithm.py (3.10.0)

File Edit Format Run Options Window Help

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

df=pd.DataFrame({'x': [12,20,28,18,29,33,24,45,45,52,51,52,55,53,55,61,64,69,72],
                 'y': [39,36,30,52,54,46,55,59,63,70,66,63,58,23,14,8,19,7,24]

})

np.random.seed(200)
k=3
centroids={
    i+1: [np.random.randint(0,80),np.random.randint(0,80)]
    for i in range(k)
}

fig=plt.figure(figsize=(5,5))
plt.scatter(df['x'],df['y'],color='k')
colmp={1:'r',2:'g',3:'b'}
for i in centroids.keys():
    plt.scatter(*centroids[i],color=colmp[i])
plt.xlim(0,80)
plt.ylim(0,80)
plt.show()
```

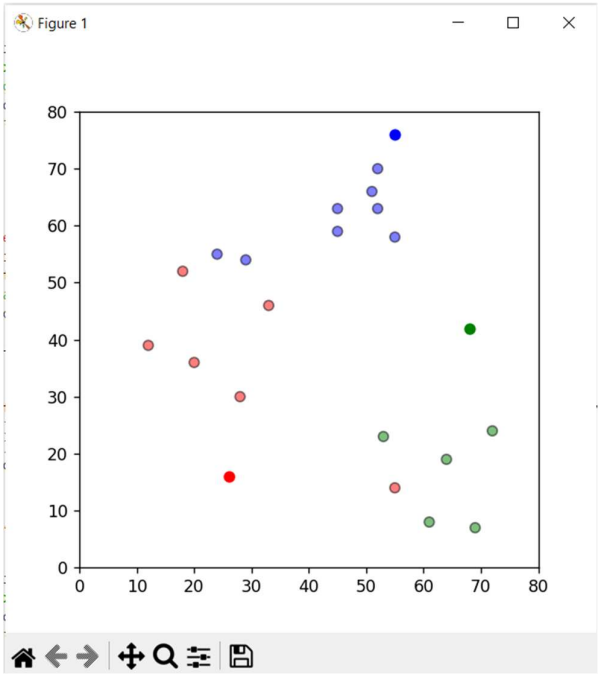


Stage 2 : Assignment stage.

Code:

```
#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()
```

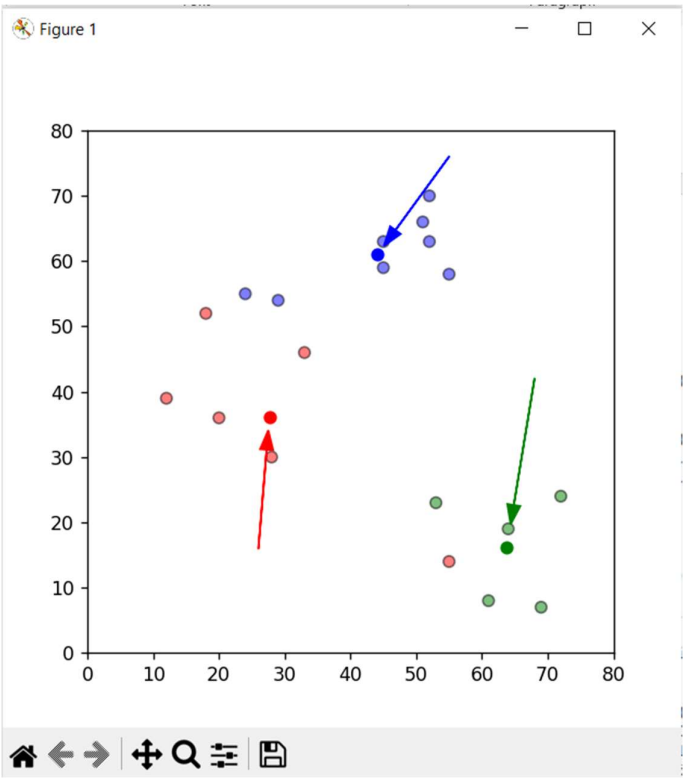
```
================== RESTART: C:\sem2\bda\prac\kmeans algorithm.py ================
    x   y  distance_from_1  distance_from_2  distance_from_3  closest color
0  12  39        26.925824        56.080300        56.727418        1     r
1  20  36        20.880613        48.373546        53.150729        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
```



Stage 3: Update Stage.

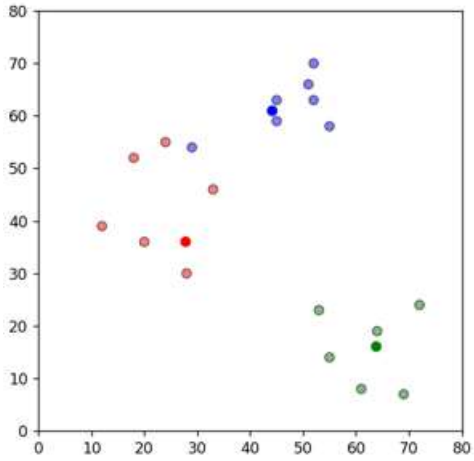
Code:

```
#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()
```

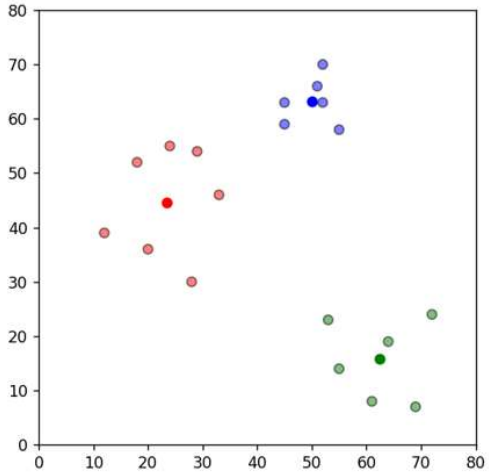


Stage 4: Repeat Assignment stage

```
#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()
```



PRACTICAL NO.2

Aim: Implementing Linear Regression

Description: -

- Linear regression is one of the easiest and most popular Machine Learning algorithms. It is a statistical method that is used for predictive analysis.
- Linear regression makes predictions for continuous/real or numeric variables such as sales, salary, age, product price, etc.
- Linear regression algorithm shows a linear relationship between a dependent (y) and one or more independent (y) variables, hence called as linear regression.
- Since linear regression shows the linear relationship, which means it finds how the value of the dependent variable is changing according to the value of the independent variable.

Methods :-

1. **pd.read_csv(inputfilename) :** This method is used to read the csv files.
2. **dataframe.iloc[:,[colno_1,colon_3]] :** This method is used to fetch specific row of specific columns.
3. **train_test_split(x,y,test_size=0.25,random_state=0) :** This method is used to split dataframe into training and testing dataset.
4. **StandardScaler() :**This method is used for feature scaling.
5. **SVC(kernel='linear', random_state=0) :** This method is used for linear support vector classifier.
6. **metrics.accuracy_score(y_test,y_pred) :** This method is used to check the accuracy score.
7. **model.coef_** : model.coef_ is used to obtain coefficient value.
8. **model.intercept_** : model.intercept_ is used to obtain intercept value.
9. **model.score(waist,weight)** : This method is used to check the accuracy of the model.
10. **model.predict(waist_new)** :This method is used to predict the value based on trained dataset.
11. **data.corr()** : This method is used to obtain correlation.
12. **lm.fit(waist, weight)** : fit() is used to train model.

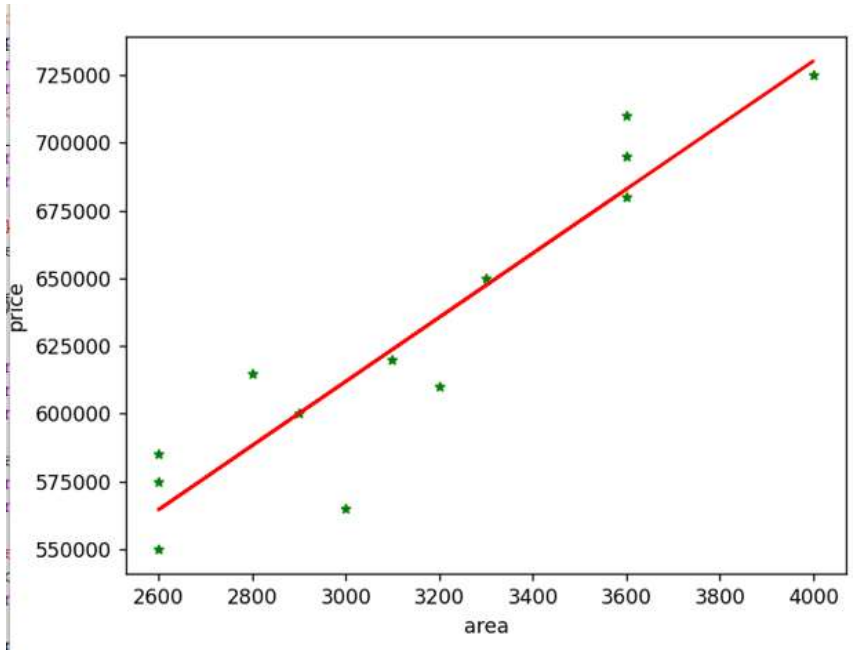
Code :

Applying Linear Regression to predict Area Vs Price based on given data set.

```
import pandas as pd
import numpy as np
from sklearn import linear_model
import matplotlib.pyplot as plt
df=pd.read_csv(r'C:\DS\DATA SCIENCE\DATA\homeprices.csv')
print("Fill Data :\n",df.head(3))
print(df.shape)
#dropping price column bcuz when we fit the linear model it expects a 2D array
new_df=df.drop('price',axis=1)
print('\n After dropping price column :\n', new_df.head(3))
print(new_df.shape)

#making as instance of linearRegression class
model=linear_model.LinearRegression()
#to train the model  use fit method with area and price
model.fit(new_df,df.price)

#we want to predict price of area 1500,predict function expects 2D array
print("price predicted  value of area 1500:",model.predict([[1500]]))
print("value of Coeficent Value :",model.coef_)
print("value of the intercept:",model.intercept_)

area_df=pd.read_csv(r'C:\DS\DATA SCIENCE\DATA\area.csv')
print('\n Area which is suppose to predict:\n',area_df.head(3))
print(area_df.shape)

#predicting area
predicted_value=model.predict(area_df)
print("\n Price predicted for all area: \n",predicted_value)

df.plot(kind='scatter',x='area',y='price',color='green',marker='*')
plt.plot(df.area,model.predict(df.area.values.reshape(-1,1)),color='red')
plt.show()
```

OUTPUT:-

```
Fill Data :
    area   price
0  2600  550000
1  3000  565000
2  3200  610000
(13, 2)

 After dropping price column :
    area
0  2600
1  3000
2  3200
(13, 1)
price predicted  value of area 1500: [434499.0665837]
value of Coeficent Value : [118.29495955]
value of the intercept: 257056.627255756

 Area which is suppose to predict:
    area
0  1000
1  1200
2  1400
(13, 1)

 Price predicted for all area:
 [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]
```



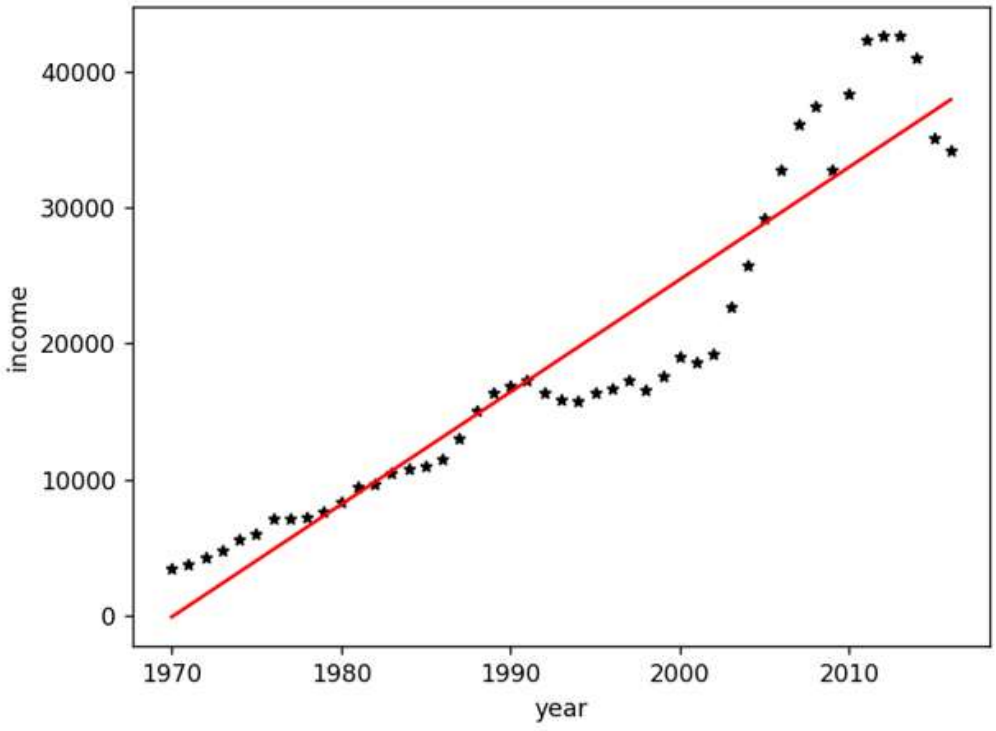
CODE: -

Applying Linear Regression to predict income based on given data set.

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

df=pd.read_csv('C:\DS\DATA SCIENCE\DATA\canada_per_capita_income.csv')
print('Given data set :\n',df.head(5))
print(df.shape)

#copying year in sepreated data frame
year=pd.DataFrame(df['year'].values.reshape(-1,1))
print('\n from given data set, year is seprated :\n',year.head())

#Building model
model=linear_model.LinearRegression()
model.fit(year,df.income)

#Data which is suppose to predict
x=[2018,2019,2020,2030]
print('\n Income is supposed to predict of this year\n',x)
b=pd.DataFrame(x)
y=model.predict(b)
d=pd.DataFrame(y)
data=pd.concat([b,d],axis=1,keys=['Year','predicted income'])
print('\n Result :\n',data)

#plotting graph
df.plot(kind='scatter',x='year',y='income',color='black',marker='*')
#write after data ploting
plt.plot(df.year,model.predict(df.year.values.reshape(-1,1)),color='red')
plt.show()
```

OUTPUT: -

```
Given data set :
    year       income
0   1970  3399.299037
1   1971  3768.297935
2   1972  4251.175484
3   1973  4804.463248
4   1974  5576.514583
(47, 2)

 from given data set, year is seprated :
       0
0   1970
1   1971
2   1972
3   1973
4   1974

 Income is supposed to predict of this year
 [2018, 2019, 2020, 2030]

 Result :
    Year predicted income
      0                0
0   2018     39631.763944
1   2019     40460.229019
2   2020     41288.694094
3   2030     49573.344847
```



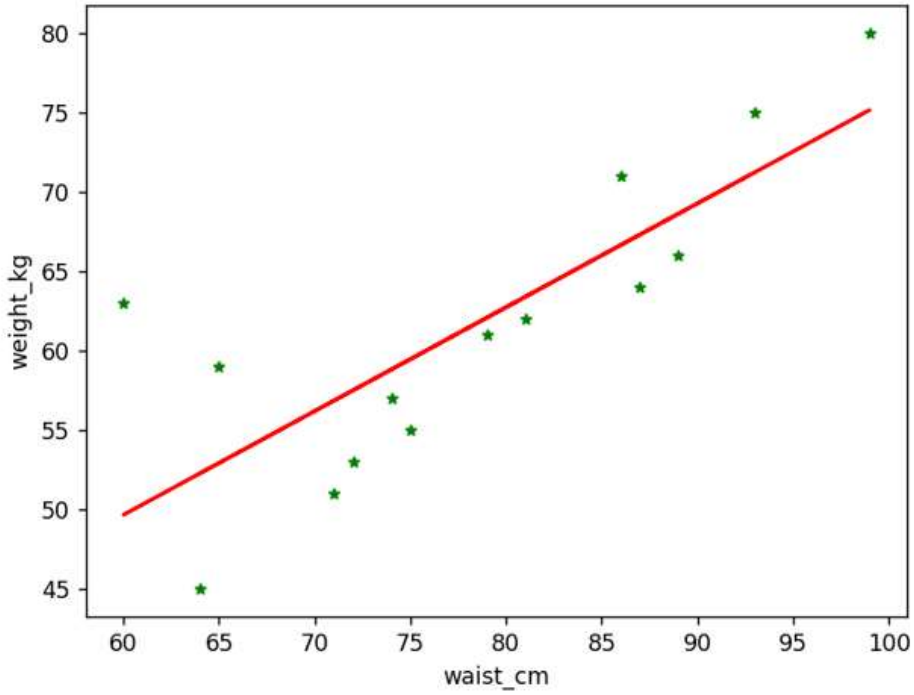
CODE :-

Applying Linear Regression to predict weight based on the waist.

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

df=pd.read_csv('C:\DS\DATA SCIENCE\DATA\weightwaist.csv')
print('Given data set : \n',df.head(3))
print('Data shape:',df.shape)

#copying waist from dataset into dataframe or drop column
waist=pd.DataFrame(df['waist_cm'])
print('\n Copying waist in seperate DataFrame :\n',waist.head(3),waist.shape)

#Building model
model=linear_model.LinearRegression()
model.fit(waist,df.weight_kg)

#predicting coeficent and intercept y=mx+C
print('\n value of model coeffficent :',model.coef_)
print('Value of model intercept:',model.intercept_)
print('model score:',model.score(waist,df.weight_kg))

#Taking new data to predict here both variable should be dataframe
x=[67,78,94]
c=pd.DataFrame(x)
d=model.predict(c)
e=pd.DataFrame(d)
predicted=pd.concat([c,e],axis=1,keys=['Waist_cm','  predicted_weight_kg'])
print('\n Result :',predicted)

#predicting single value weight
print('\n predicted weight od single waist:',model.predict([[97]]))
df.plot(kind='scatter',x='waist_cm',y='weight_kg',color='green',marker='*')
plt.plot(waist,model.predict(df.waist_cm.values.reshape(-1,1)),color='red')
plt.show()
```

OUTPUT:-

```
Given data set :
    waist_cm  weight_kg
0         71         51
1         89         66
2         64         45
Data shape: (14, 2)

 Copying waist in seperate DataFrame :
    waist_cm
0         71
1         89
2         64 (14, 1)

 value of model coeffficent : [0.65405294]
Value of model intercept: 10.415144674738357
model score: 0.6377256319321334

 Result :   Waist_cm   predicted_weight_kg
            0                           0
0        67              54.236692
1        78              61.431274
2        94              71.896121

 predicted weight od single waist: [73.85828032]
```



PRACTICAL NO.: 03

Aim : Implementing Logistic Regression.

Description :-

- Logistic regression is one of the most popular Machine Learning algorithms, which comes under the Supervised Learning technique.
- It is used for predicting the categorical dependent variable using a given set of independent variables.
- Logistic regression predicts the output of a categorical dependent variable. Therefore the outcome must be a categorical or discrete value.
- It can be either Yes or No, 0 or 1, true or False, etc. but instead of giving the exact value as 0 and 1, it gives the probabilistic values which lie between 0 and 1.
- Logistic Regression is much similar to the Linear Regression except that how they are used. Linear Regression is used for solving Regression problems, whereas Logistic regression is used for solving the classification problems.
- In Logistic regression, instead of fitting a regression line, we fit an "S" shaped logistic function, which predicts two maximum values (0 or 1).
- The curve from the logistic function indicates the likelihood of something such as whether the cells are cancerous or not, a mouse is obese or not based on its weight, etc.
- Logistic Regression is a significant machine learning algorithm because it has the ability to provide probabilities and classify new data using continuous and discrete datasets.

Method:

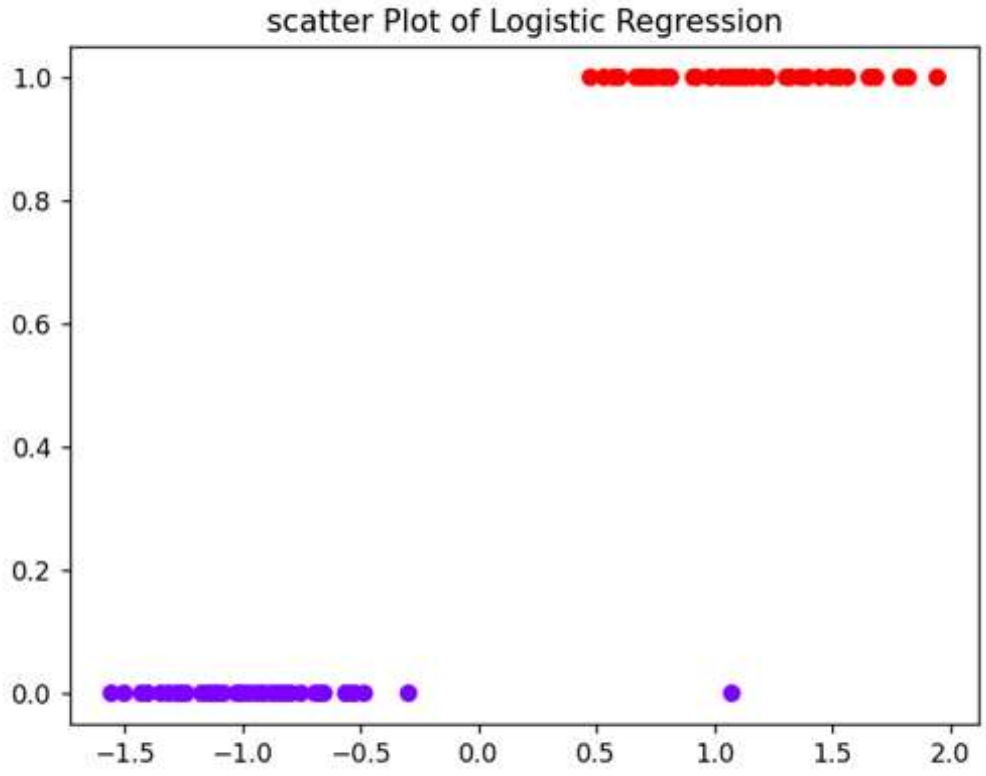
1. LogisticRegression() : This method is used to implement logistic regression.
2. train_test_split(x,y,test_size=0.25,random_state=0) : This method is used to split dataframe into training and testing dataset.
3. Model.fit(X, y[, sample_weight]): Fit the model according to the given training data.
4. model.predict(x_test) :This method is used to predict the value based on trained dataset.
5. model.coef_ : model.coef_ is used to obtain coefficient value.
6. model.intercept_ : model.intercept_ is used to obtain intercept value.

CODE: -

```
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

## Generating a dataset for logistic regression
x,y=make_classification(
    n_samples=100, # The number of samples
    n_features=1,# by default=20,
    n_classes=2,#Number of classes of classification function
    n_clusters_per_class=1, # by default=2 the number of cluster per class
    #flip_y=0.3, # by default=0.1
    n_informative=1,# useful Feautures
    n_redundant=0,#height-cm and Height_feet are same.
    n_repeated=0)#informative,redundant,repeated < n_features
print("\nShape of Data Generated (x) (y): ",x.shape,y.shape)

#split the dataset into training and testing dataset
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=1)
print("Shape of (x_train) (y_train) (x_test) (y_test) :",
      x_train.shape,y_train.shape,x_test.shape,y_test.shape)

#Building Model
from sklearn.linear_model import LogisticRegression
model=LogisticRegression()
model.fit(x_train,y_train)

#Finding the value of the coefficient and intercept
print("\nValue of Coefficient : ",model.coef_)
print("Value of Intercpet : ",model.intercept_)

#Predicting the value of the x_test Dataset
y_pred=model.predict(x_test)
print("predicted value :",y_pred)

#Show the confusion Matrix
from sklearn.metrics import confusion_matrix
print("\nConfusion Matrix : \n",confusion_matrix(y_test,y_pred))

#Create a scatter plot
plt.scatter(x,y,c=y,cmap='rainbow'),plt.title
("Scatter plot of logistic regression"),plt.show()
```

OUTPUT: -

```
Shape of Data Generated (x) (y):  (100, 1) (100,)
Shape of (x_train) (y_train) (x_test) (y_test) : (75, 1) (75,) (25, 1) (25,)

Value of Coefficient :  [[3.14868435]]
Value of Intercpet :  [0.04059558]
predicted value : [1 0 0 1 0 0 0 1 1 1 1 0 1 0 0 1 0 1 0 0 1 1 0 0 0]

Confusion Matrix :
 [[14  0]
 [ 0 11]]
```



PRACTICAL NO.4

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

Description :

- MongoDB, the most popular NoSQL database, is an open-source document-oriented database.
- The term 'NoSQL' means 'non-relational'. It means that MongoDB isn't based on the table-like relational database structure but provides an altogether different mechanism for storage and retrieval of data.
- SQL databases store data in tabular format.
- This data is stored in a predefined data model which is not very much flexible for today's real-world highly growing applications.
- Modern applications are more networked, social and interactive than ever.
- Applications are storing more and more data are accessing it at higher rates.
- Relational Database Management System (RDBMS) is not the correct choice when it comes to handling big data by the virtue of their design since they are not horizontally scalable.
- If the database runs on a single server, then it will reach a scaling limit.
- NOSQL databases are more scalable and provide superior performance.
- MongoDB is such a NoSQL database that scales by adding more and more servers and increases productivity with its flexible document model.

Methods :

1. **MongoClient('localhost:27017') :** This method is used to get at which port monodb is running.
2. **client.get_database('database_name') :** This method is used to access the database.
3. **db.records_name :** This method is used to access the collection of database.
4. **records.count_documents({}) :** This method is used to count the number of records in the collection.
5. **list(records.find())** : This method is used to print all the records in collection.
6. **records.update_one({"$set":{"key","value"}}) :** This method is used to update one record in collection.
7. **records.insert_one({"eno":6,"name":"Raj","location":"India"}) :** This method is used to insert one record in collection.
8. **records.delete_one({"name":"Raj"})** : This method is used to delete one record from collection.

Steps : For mongoDB operation

Here We need Python -version 3.11.2 and MongoDB Version 5.0.15 2008R2Plus.

```
> use msc
switched to db msc
```

```
> show dbs
admin   0.000GB
config  0.000GB
local   0.000GB
msc     0.000GB
```

```
> show collections
collection
person
student
```

```
> db.student.insertOne({"First Name":"Rajesh","Last Name":"Prajapati","age":23})
```

```
> db.collection.insertMany([{"First Name":"Ramesh","Last name":"Ram","age":29},{"First Name":"Suresh","Last Name":"babu","age":25}])
```

```
> db.collection.find().pretty()
```

```
> db.collection.update({"First Name":"Rakesh"},{$set:{"age":23}})
```

```
> db.collection.update( {"First Name":"Ramesh"}, {$set:{"age":40,"Last name":"Chaube"}})
```

Code : For the Retrieve data

```
from pymongo import MongoClient
client = MongoClient('localhost:27017')
db = client.msc
collection = db.student
print("Whole Record : \n")
for x in collection.find({},{ "_id": 0, "name": 1,"Div":1,'address':1}):
 print(x)
print("************************************************************\n")
##Query the data by query operator $in
print("printing those value containing Div B and A both :\n")
for record in collection.find({'Div':{"$in":['B','A']}},{ "_id": 0, "name": 1,"Div":1,'address':1}):
  print(record)
print("************************************************************\n")
print("printing those value containing Div A  :\n")
for record in collection.find({'Div':{"$in":['A']}},{ "_id": 0, "name": 1,"Div":1,'address':1}):
  print(record)
print("************************************************************\n")
##And and Query Operator
print(" AND Operator, those value containing only and only bandra,Sandy :\n")
for records in collection.find({"name":"Sandy","address":{"$lt":"bandra"}},{ "_id": 0, "name": 1,"Div":1,'address':1}):
  print("Record : ",records)
print("************************************************************\n")
print("OR operator,those value containing Suraj or Delhi  :\n")
for records in collection.find({"$or":[{"name":"Suraj"},{"address":"Delhi"}]},{ "_id": 0, "name": 1,"Div":1,'address':1}):
  print("Record : ",records)
```

Output :

```
Whole Record :

{}
{}
{'name': 'Suraj', 'address': 'bandra', 'Div': 'A'}
{'name': 'Sandy', 'address': 'Delhi', 'Div': 'B'}
{'name': 'Sandy', 'address': 'Ocean blvd 2', 'Div': 'B'}
************************************************************

printing those value containing Div B and A both :

{'name': 'Suraj', 'address': 'bandra', 'Div': 'A'}
{'name': 'Sandy', 'address': 'Delhi', 'Div': 'B'}
{'name': 'Sandy', 'address': 'Ocean blvd 2', 'Div': 'B'}
************************************************************

printing those value containing Div A  :

{'name': 'Suraj', 'address': 'bandra', 'Div': 'A'}
************************************************************

 AND Operator, those value containing only and only bandra,Sandy :

Record :  {'name': 'Sandy', 'address': 'Delhi', 'Div': 'B'}
Record :  {'name': 'Sandy', 'address': 'Ocean blvd 2', 'Div': 'B'}
************************************************************

OR operator,those value containing Suraj or Delhi  :

Record :  {'name': 'Suraj', 'address': 'bandra', 'Div': 'A'}
Record :  {'name': 'Sandy', 'address': 'Delhi', 'Div': 'B'}
```

Code: for deleting and updating records

```
from pymongo import MongoClient
client = MongoClient('localhost:27017')
db = client.msc
collection = db.student
print("Whole Data:")
for record in collection.find({},{ "_id": 0, "name": 1,"Div":1,'address':1}):
  print(record)
print("*****************************************************\n")
#updating records update({},{$set:{}})
print("Updating Suraj's Address and Div : \n")
collection.update_many({"name":"Suraj"},{"$set":{"address":"Santacruz","Div":"D"}})
for record in collection.find({"name":"Suraj"},{ "_id": 0, "name": 1,"Div":1,'address':1}):
  print(record)
print("*****************************************************\n")

# delete Record
print("Deleting where Address is ocean blvd 2 : \n")
collection.delete_many({"address":"Ocean blvd 2"})
for record in collection.find({},{ "_id": 0, "name": 1,"Div":1,'address':1}):
  print(record)
```

Output :

```
Whole Data:
{}
{}
{'name': 'Suraj', 'address': 'Santacruz', 'Div': 'D'}
{'name': 'Sandy', 'address': 'Delhi', 'Div': 'B'}
{'name': 'Sandy', 'address': 'Ocean blvd 2', 'Div': 'B'}
*****************************************************


Updating Suraj's Address and Div :

{'name': 'Suraj', 'address': 'Santacruz', 'Div': 'D'}
*****************************************************

Deleting where Address is ocean blvd 2 :

{}
{}
{'name': 'Suraj', 'address': 'Santacruz', 'Div': 'D'}
{'name': 'Sandy', 'address': 'Delhi', 'Div': 'B'}
```

Code : For inserting single/Multiple Records also Deleting Multiple Records

```
from pymongo import MongoClient
client = MongoClient('localhost:27017')
db = client.msc
collection = db.student
print("Whole Data :\n")
for record in collection.find({},{ "_id": 0, "name": 1,"Div":1,'address':1}):
  print(record)
print("**************************************************\n")

print("inserting Single Rocord : \n")
collection.insert_one({"name":"Rajesh","address":"Vile Parle","Div":"A"},)
for record in collection.find({},{ "_id": 0, "name": 1,"Div":1,'address':1}):
  print(record)
print("**************************************************\n")
print("Inserting Many Records : \n")
collection.insert_many([{"name":"Rakesh","address":"Bandra","Div":"A"},{"name":"Rahul","address":"Andheri","Div":"B"}])
for record in collection.find({},{ "_id": 0, "name": 1,"Div":1,'address':1}):
  print(record)
print("**************************************************\n")

print("deleting many records : \n")
collection.delete_many({"$or":[{"name":"Rajesh"},{"address":"Bandra"},{"name":"Rahul"}]})
for record in collection.find({},{ "_id": 0, "name": 1,"Div":1,'address':1}):
  print(record)
```

OUTPUT: -

```
Whole Data :

{}
{}
{'name': 'Suraj', 'address': 'Santacruz', 'Div': 'D'}
{'name': 'Sandy', 'address': 'Delhi', 'Div': 'B'}
{'name': 'Rajesh', 'address': 'Vile Parle', 'Div': 'A'}
**************************************************

inserting Single Rocord :

{}
{}
{'name': 'Suraj', 'address': 'Santacruz', 'Div': 'D'}
{'name': 'Sandy', 'address': 'Delhi', 'Div': 'B'}
{'name': 'Rajesh', 'address': 'Vile Parle', 'Div': 'A'}
{'name': 'Rajesh', 'address': 'Vile Parle', 'Div': 'A'}
**************************************************

Inserting Many Records :

{}
{}
{'name': 'Suraj', 'address': 'Santacruz', 'Div': 'D'}
{'name': 'Sandy', 'address': 'Delhi', 'Div': 'B'}
{'name': 'Rajesh', 'address': 'Vile Parle', 'Div': 'A'}
{'name': 'Rajesh', 'address': 'Vile Parle', 'Div': 'A'}
{'name': 'Rakesh', 'address': 'Bandra', 'Div': 'A'}
{'name': 'Rahul', 'address': 'Andheri', 'Div': 'B'}
**************************************************

deleting many records :

{}
{}
{'name': 'Suraj', 'address': 'Santacruz', 'Div': 'D'}
{'name': 'Sandy', 'address': 'Delhi', 'Div': 'B'}
```

PRACTICAL NO.5

Aim: Implement SVM classification Technique.

Description :-

- SVM is a famous supervised machine learning algorithm used for classification as well as regression algorithms.
- However, mostly it is preferred for classification algorithms.
- It basically separates different target classes in a hyperplane in n-dimensional or multidimensional space.
- The main motive of the SVM is to create the best decision boundary that can separate two or more classes(with maximum margin) so that we can correctly put new data points in the correct class.Because it chooses extreme vectors or support vectors to create the hyperplane, that's why it is named so.

Methods :-

1. StandardScaler() : It is used for feature scaling. Use to transform the not scale data to scale data. X is in less range y is in high range then might be x get less importance by model itself so avoid this, where It calculates the mean and standard deviation for the independent and dependent variable. Each datapoint – mean of columns and divide by the standard deviation. By doing so mean of the respected columns become 0 and deviation become 1. When you plot your data you will see that all your data points centered towards the mean value. So all the features start getting the equal importance by the model

2. SVC(kernel='linear', random_state=0) : This method is used for implementing SVM.

3. metrics.accuracy_score(y_test,y_pred) : This method is used to check the accuracy score of the model.

4. df.iloc() : The iloc function in Python returns a view of the selected rows and columns from a Pandas DataFrame.

   Fit.transform() : where fit with parameter calculates the mean and standard deviation and transform() does subtract form each data point – mean divide by deviation

5. : Dependent variable is not in the form of binary then ---df[" purchased "] = df["purchased"] . map ({"any name":0 , "anyanme" : 1 })

6. If data points in purchased are more than 2 types then

   df=df [df ["purchased"] ! = " 3rd type "]

CODE: -

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
df=pd.read_csv("C:\MSCITPractical\BDA\SVM implementationn\social.csv")
print('Input Data Values ==================================')
print(df.head(10),df.shape)

# Either you can drop the columns and store into x,y or you can do this
x=df.iloc[:,[2,3]]# before " : " means entire rows after that columns selecting
y=df.iloc[:,4]
print("\nOnly Cols 2,3 taking in variable x : \n",x.head(5),x.shape)
print("\nOnly cols 4 Taking in variable Y \n",y.head(5),y.shape)

#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("\nTraining data:\n",x_train.head(),x_train.shape)
print('********************')
print("\nTesting data:\n",x_test.head(),x_test.shape)

#Feature scaling performed to scaling down the feature between 0 to 1
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
x_train_scaled=scaler.fit_transform(x_train)
x_test_scaled=scaler.fit_transform(x_test)

from sklearn.svm import SVC
classifier=SVC(kernel='linear', random_state=0)
classifier.fit(x_train_scaled,y_train)

#predicting the test set results
y_pred=classifier.predict(x_test_scaled)
print("\nTest set Result Prediction : ",y_pred)

from sklearn import metrics
print('\naccuracy score with linear kernel : ')
print(metrics.accuracy_score(y_test,y_pred))
```

OUTPUT: -

```
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 (26, 5)

Only Cols 2,3 taking in variable x :
    age  estimatedsalary
0   19            16000
1   22            23000
2   56            44000
3   33            22000
4   23            22000 (26, 2)

Only cols 4 Taking in variable Y
 0    0
1    0
2    1
3    0
4    0
Name: purchased, dtype: int64 (26,)

Training data:
     age  estimatedsalary
13   19            44000
18   33            34000
19   23            34000
16   33            44000
1    22            23000 (19, 2)
********************

Testing data:
     age  estimatedsalary
2    56            44000
20   44            34000
14   43            44000
17   55            34000
5    54            22000 (7, 2)

Test set Result Prediction :  [1 0 1 0 0 1 0]

accuracy score with linear kernel :
1.0
```

PRACTICAL NO. : 06

Aim: Implement Decision Tree classification Technique.

Description :-

- Decision Tree is a supervised learning method used in data mining for classification and regression methods.
- It is a tree that helps us in decision-making purposes.
- The decision tree creates classification or regression models as a tree structure.
- It separates a data set into smaller subsets, and at the same time, the decision tree is steadily developed.
- The final tree is a tree with the decision nodes and leaf nodes.
- A decision node has at least two branches.
- The leaf nodes show a classification or decision.
- We can't accomplish more split on leaf nodes.
- The uppermost decision node in a tree that relates to the best predictor called the root node. Decision trees can deal with both categorical and numerical data.

Methods :

1. MinMaxScaler() : This method is used for feature scaling.

2. DecisionTreeClassifier() : This method is used to implement decision tree

CODE: -

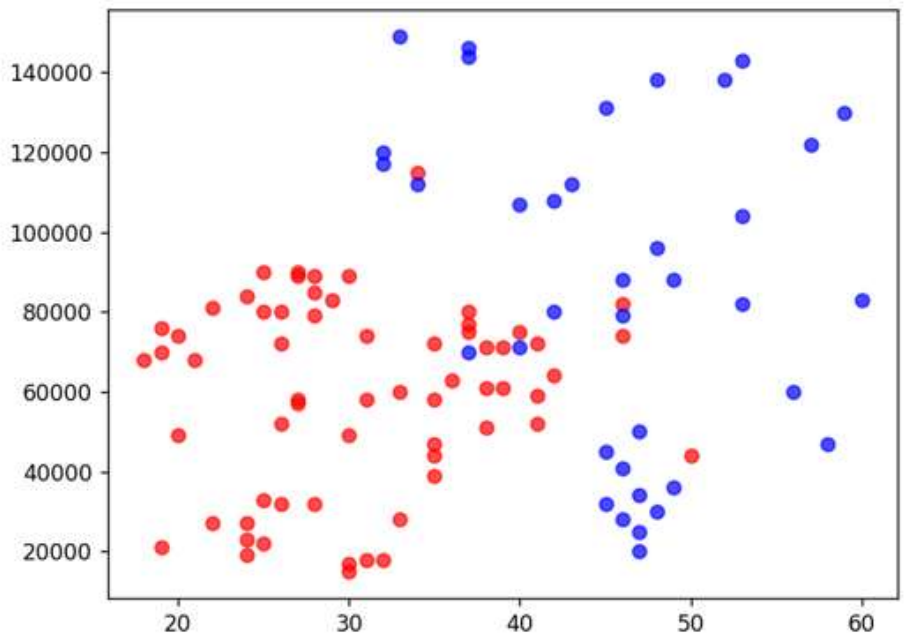
DT.py - C:\sem2\bda\prac\DT.py (3.10.0)

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("C:/sem2/bda/prac/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))
```

OUTPUT: -

```
======================== RESTART: C:\sem2\bda\prac\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,)
```



Practical No : 7

Aim: - Implementing Text Analysis Technique

Description :

- Text Analysis refers to the representation, processing, and modelling of textual data to derive useful insights.
- An important component of text analysis is text mining, the process of discovering relationships and interesting patterns in large text collections.
- Text analysis suffers from the curse of high dimensionality.
- Text analysis of high dimensionality.
- Text analysis often deals with textual data that is far more complex.
- A corpus is a large collection of texts used for various purposes in Natural Language Processing. Another major challenge with Text analysis is most of the time the text is not structured.

Code :

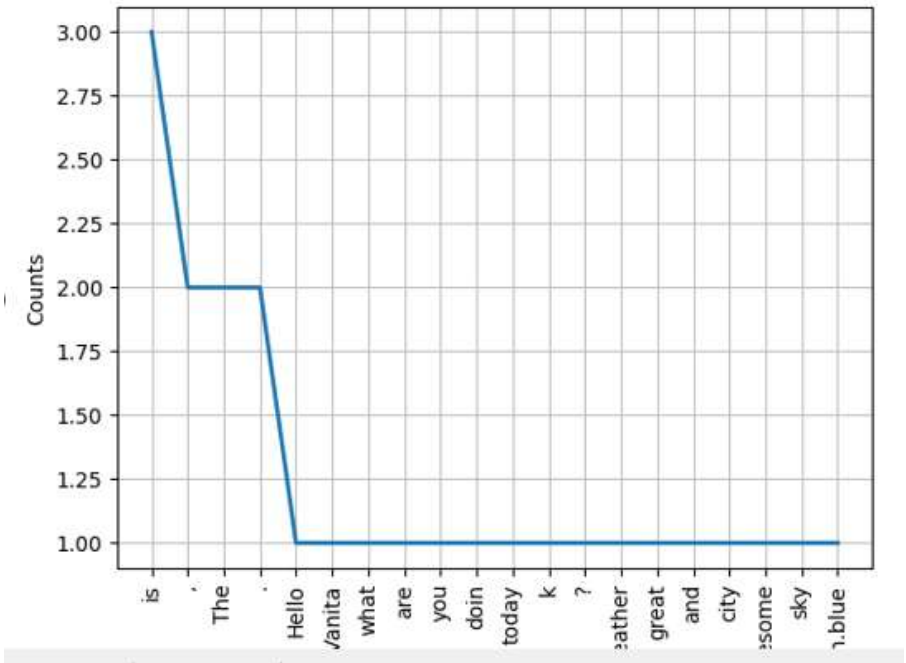
text anaylsis.py - C:\sem2\bda\prac\text anaylsis.py (3.10.0)

File Edit Format Run Options Window Help

```
import nltk
#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)
```

OUTPUT: -

```
['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>
{'have', 'haven', 'it', 'your', 'because', "shan't", 'where', 'a', 'won', 'herse
lf', "hadn't", "mightn't", "doesn't", 'only', "didn't", 'couldn', "shouldn't", '
not', 'whom', 'once', 'needn', "weren't", 'when', 'off', 'they', 'is', 'can', 'd
o', 'both', 'am', 'ours', 'itself', 'having', "she's", 'very', 'down', "you've",
 'above', 'through', 'her', 'were', 'how', 'd', 'been', 'few', 'doesn', 'yoursel
ves', 'his', 'there', "you're", 'm', "isn't", 'too', 'my', 'no', 'has', 's', 'wi
th', 'will', 'from', 'nor', "wouldn't", 'by', 'should', 'you', 'for', 'just', 'a
ren', 'hadn', 'after', 'himself', 'about', 'their', 'and', 'these', 'our', 'whic
h', 'don', 'are', 'own', "you'd", 'as', 'weren', 'yours', 'he', 'until', "you'll
", 'an', 'or', 'before', 'here', 'this', "couldn't", 'we', 'between', 've', 'doe
s', 'if', 'ain', 'theirs', 'myself', 'other', "that'll", 'them', 'had', 'below',
 'those', 'll', 'why', 'but', 'shouldn', 'mustn', 'that', 't', "needn't", "shoul
d've", 'against', 'did', 'isn', 'its', 'the', 'being', 'y', 'him', 'again', 'doi
ng', 'at', 'under', 'further', "won't", 'ma', 'any', 'same', 'hers', 'o', 'durin
g', 'ourselves', 'most', 'of', 'while', 'over', 'more', 'to', 'was', 'than', "ar
en't", 'all', 'mightn', 'wouldn', 'she', 'so', 'into', 'out', 'themselves', 'som
e', 'what', "hasn't", 'shan', 'i', 'then', 'me', 'be', 'on', 'each', "mustn't",
"don't", 'didn', 'who', "wasn't", 'such', 're', 'yourself', 'hasn', "haven't", '
in', 'wasn', "it's", 'now', 'up'}
#######
.
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'
, '.']
```



Practical No. : 08

Aim: Implementing Sentimental Analysis Technique

Description:

- Sentiment analysis is a natural language processing technique that identifies the polarity of a given text.
- It refers to analyse an opinion or feelings about something using data like text or images, regarding almost anything.
- Sentiment analysis helps companies in their decision-making process.
- For example, if public sentiments towards a product is not so good, a company may try to modify the product or stop the production altogether in order to avoid the losses. There are different flavours of sentiment analysis, but one of the most widely used techniques labels data into positive, negative and neutral.

Methods:

- unstack () is used to reshape the given Pandas DataFrame by transposing specified row level to column level. By default, it transposes the innermost row level into a column level. This is one of the techniques for reshaping the DataFrame. When we want to analyze or reshape the data, Pandas provides in-built functions.
- The count () method counts the number of not empty values for each row, or column if you specify the axis parameter as axis='columns', and returns a Series object with the result for each row (or column).
- Groupby(): name of the column on the basis of you can create the group like school , sector means creating a group inside your data

CODE: -

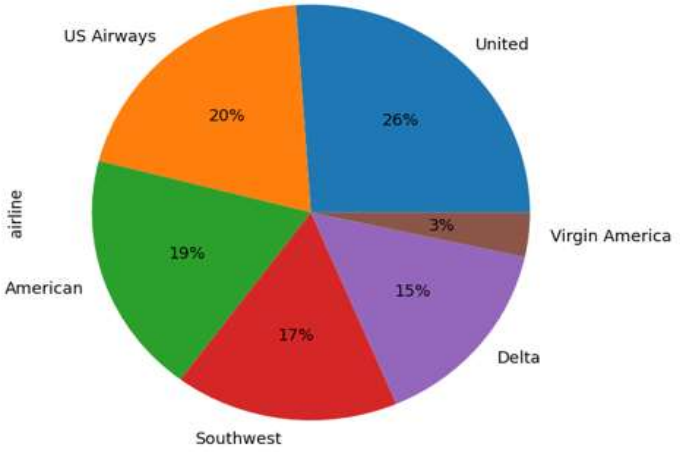
Sentiment analysis.py - C:\sem2\bda\prac\Sentiment analysis.py (3.10.0)

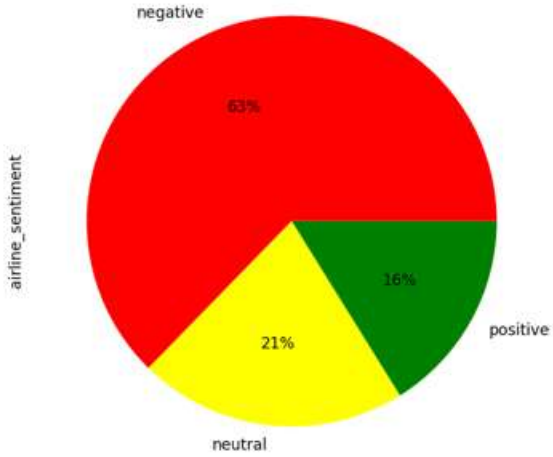
File Edit Format Run Options Window Help

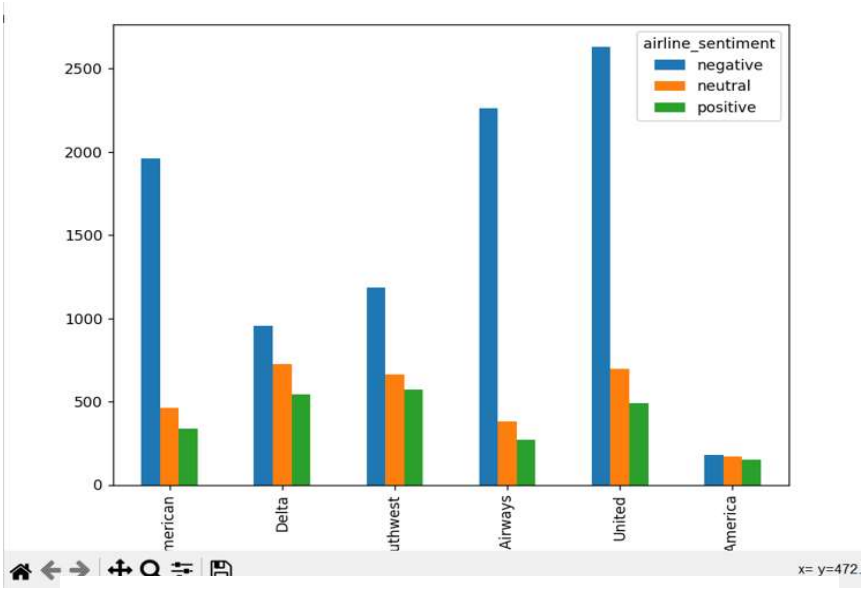
```
import numpy as np
import pandas as pd
import re
import nltk
import matplotlib.pyplot as plt
import seaborn as sns

at=pd.read_csv(r'C:\sem2\bda\prac\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()
```

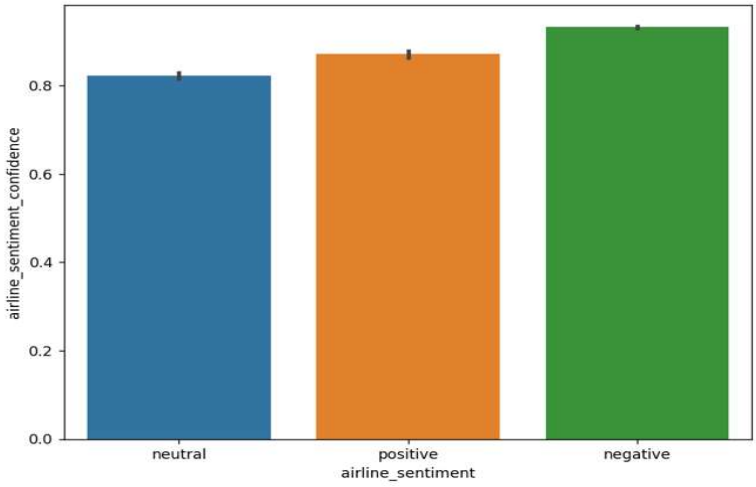
OUTPUT: -

```
================ RESTART: C:\sem2\bda\prac\Sentiment analysis.py ================
6.4
4.8
```









Practical No. 9

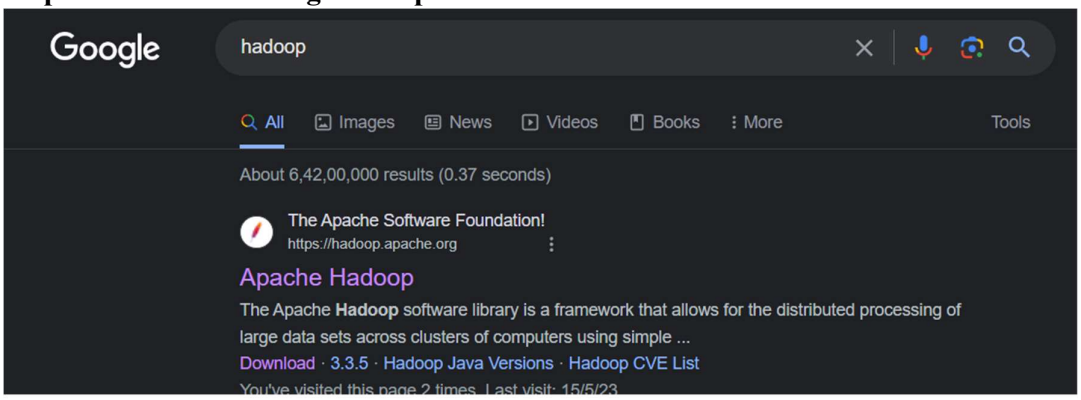
Aim: Implementing Hadoop Installations, configure and run Hadoop and HDFS

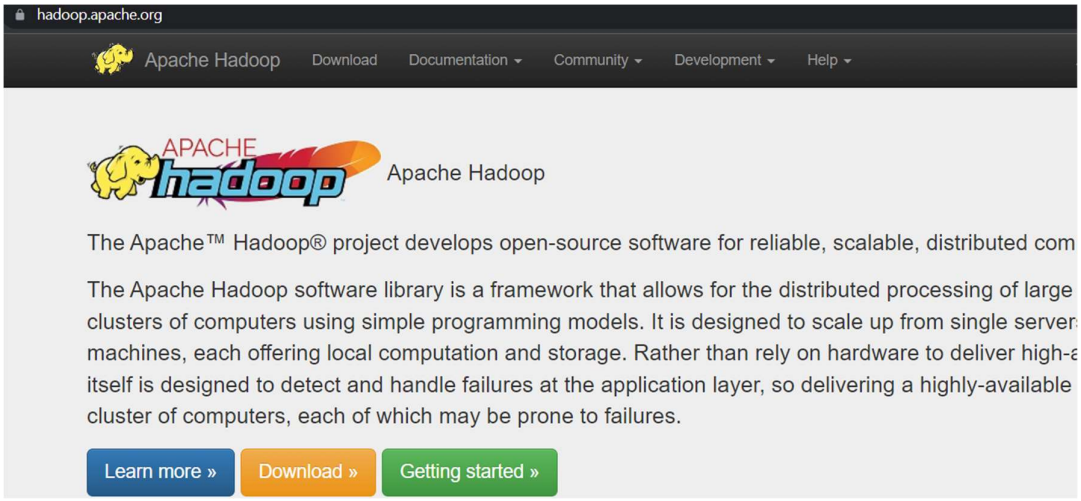
Description:

- Hadoop is an open source distributed processing framework that manages data processing and storage for big data applications in scalable clusters of computer servers.
- Hadoop systems can handle various forms of structured and unstructured data, giving users more flexibility for collecting, processing, analyzing and managing data than relational databases and data warehouses provide.
- The core components of Hadoop are HDFS and YARN. In hadoop clusters, YARN sits between HDFS and the processing engines deployed by the users.

A] Hadoop installation and configuration.

Steps : For Downloading Hadoop.





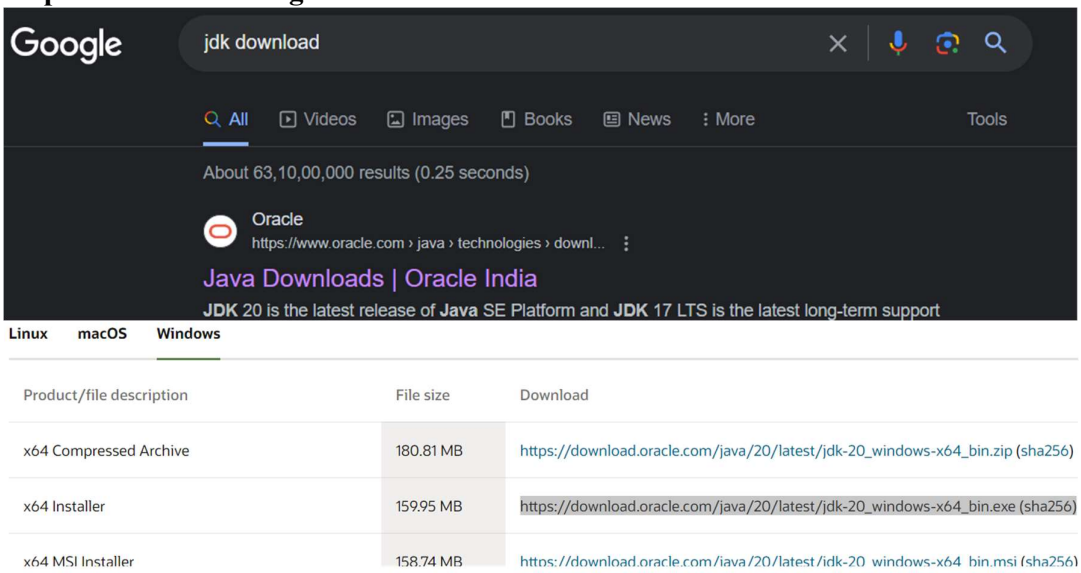
Download

Hadoop is released as source code tarballs with corresponding binary tarballs for convenience. The downloads are distributed via using GPG or SHA-512.

Version	Release date	Source download	Binary download
3.3.5	2023 Mar 22	source (checksum signature)	binary (checksum signature) binary-aarch64 (checksum signatu
3.2.4	2022 Jul 22	source (checksum signature)	binary (checksum signature)
2.10.2	2022 May 31	source (checksum signature)	binary (checksum signature)

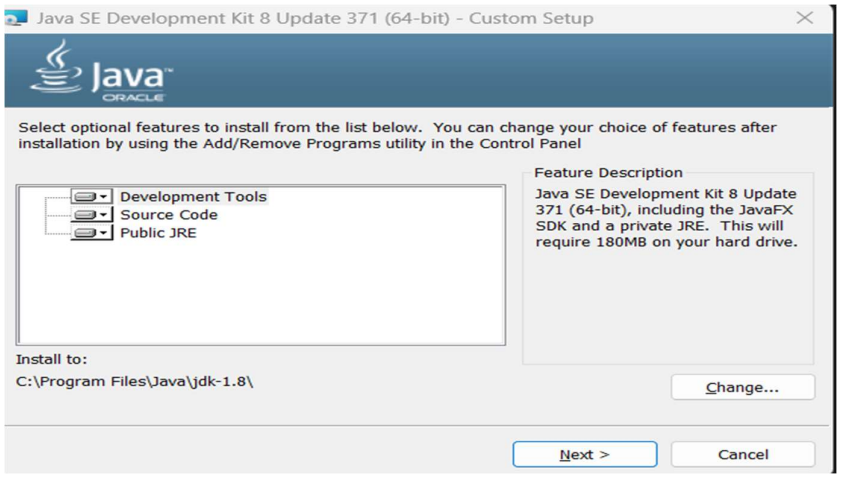
Extract with WinRAR.

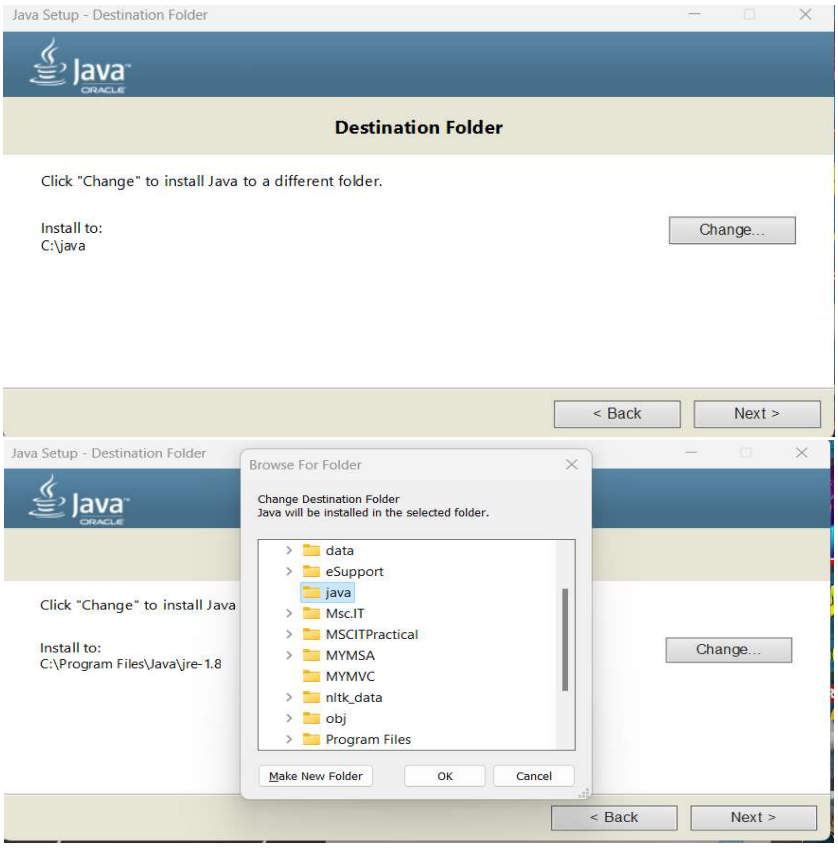
Step 2:for downloading JDK.



Then Sign-in into Oracle account by using your Email Password.

Lets Start installing JDK.





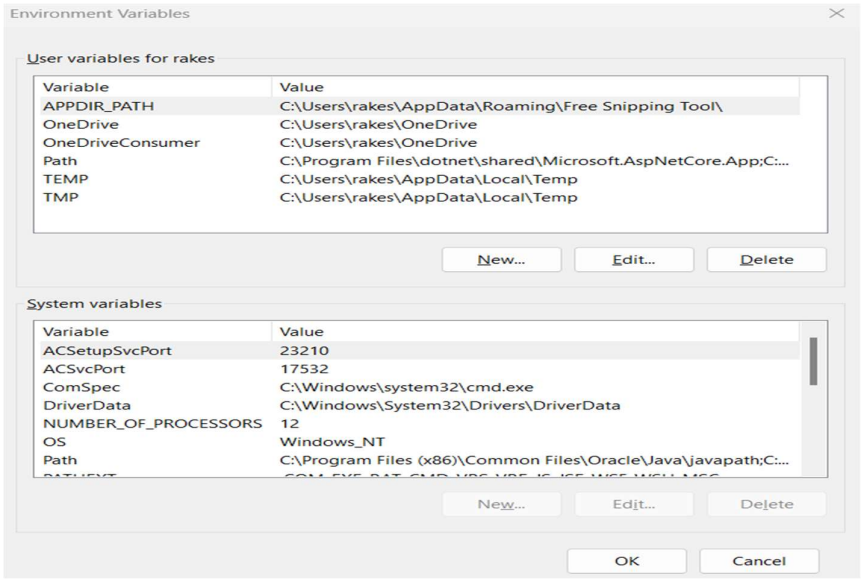
Step 3: We have to move jdk file from C:\Program Files\Java to newly created java folder C:\java.

Delete “java folder from C:\Program Files\Java” in C drive.

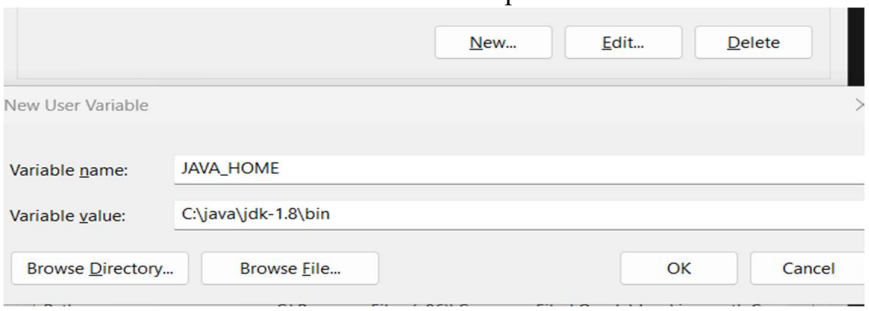
C:\java because sometimes It throws error while setting environment variable.

Step 4: Now we have to set environment variable and path for JDK

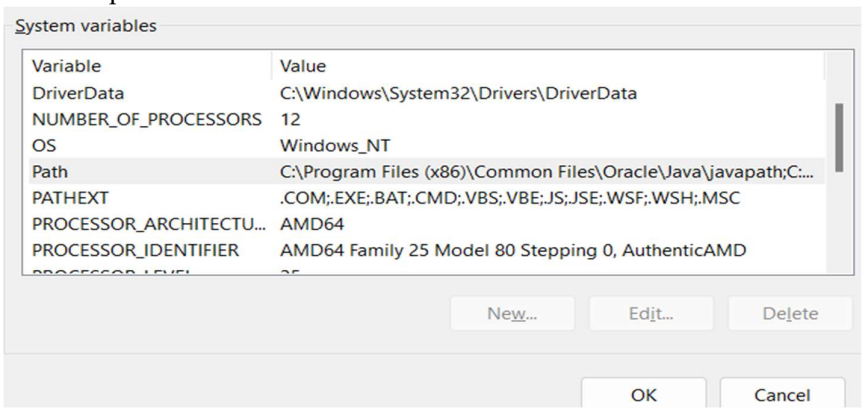
Click window and search for



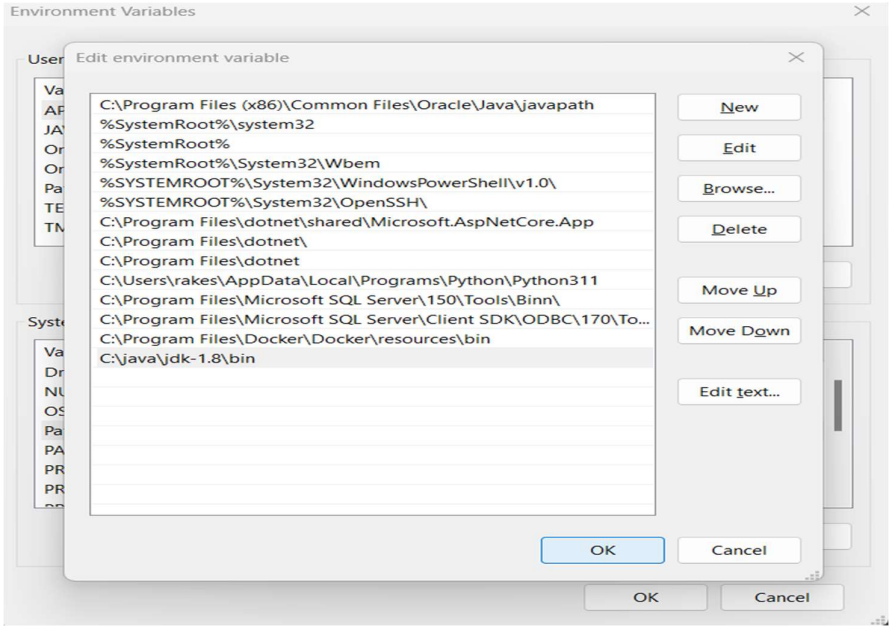
Click on New and Give Variable name and path.



Click on path



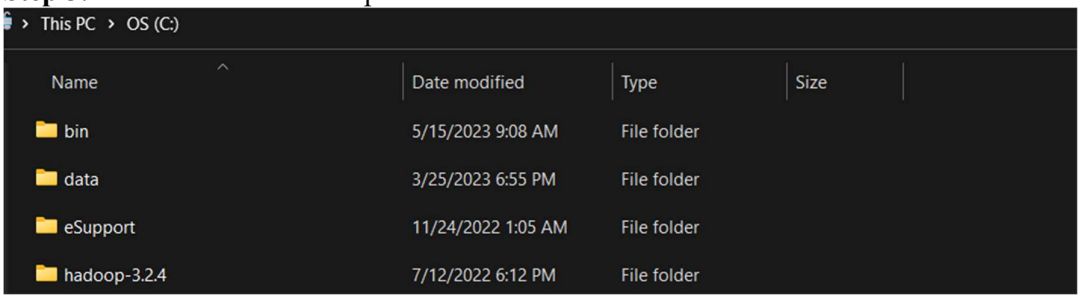
Click on edit and then click on new and paste the jdk/bin path



Now you can check installation of jdk in cmd.
Open CMD run as administrator.

```
C:\Windows\System32>java -version
java version "1.8.0_371"
Java(TM) SE Runtime Environment (build 1.8.0_371-b11)
Java HotSpot(TM) 64-Bit Server VM (build 25.371-b11, mixed mode)
```

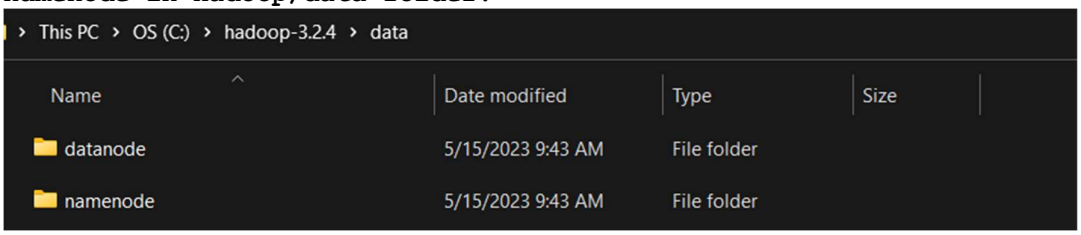
Step 5: Now extract the Hadoop tar file with WinRar and move the file in C drive.



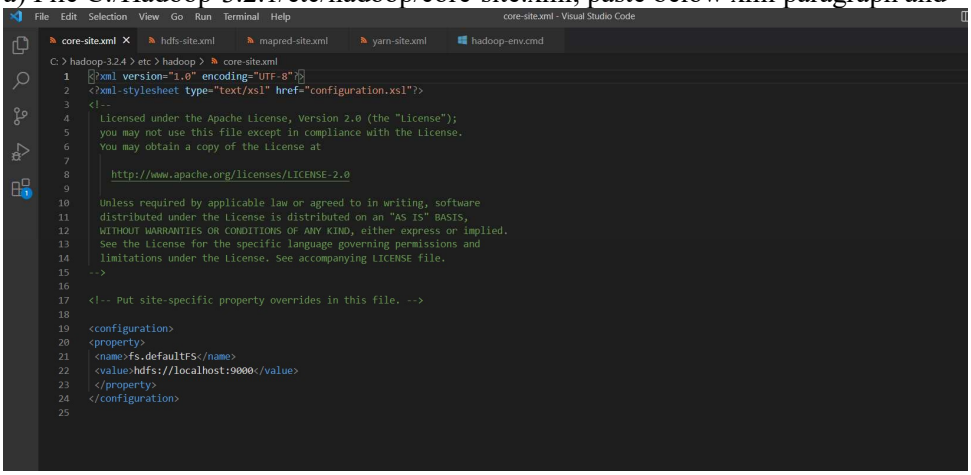
Now we need to perform some configuration in ETC folder files.
To edit this, click the desire file and right click open all simultaneously.

1. core-site
2. hdfs-site
3. mapred-site
4. yarn-site
5. hadoop-env here set JAVA_HOME=C:\java\jdk-1.8

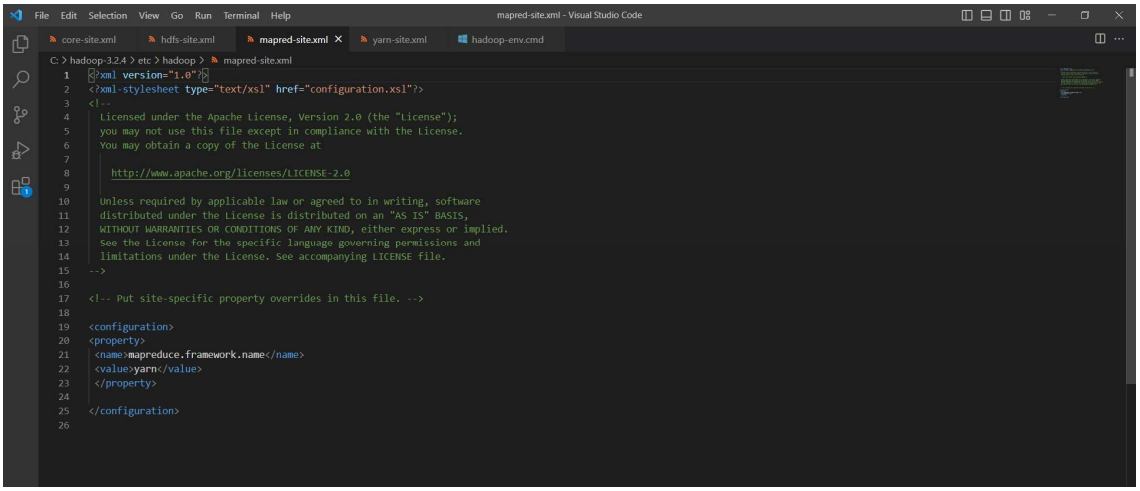
Lets first add the folder data and two more folder datanode and namenode in hadoop/data folder.



a) File C:/Hadoop-3.2.1/etc/hadoop/core-site.xml, paste below xml paragraph and



b) C:/Hadoop-3.2.1/etc/hadoop/mapred-site.xml, paste below xml paragraph and save this file.

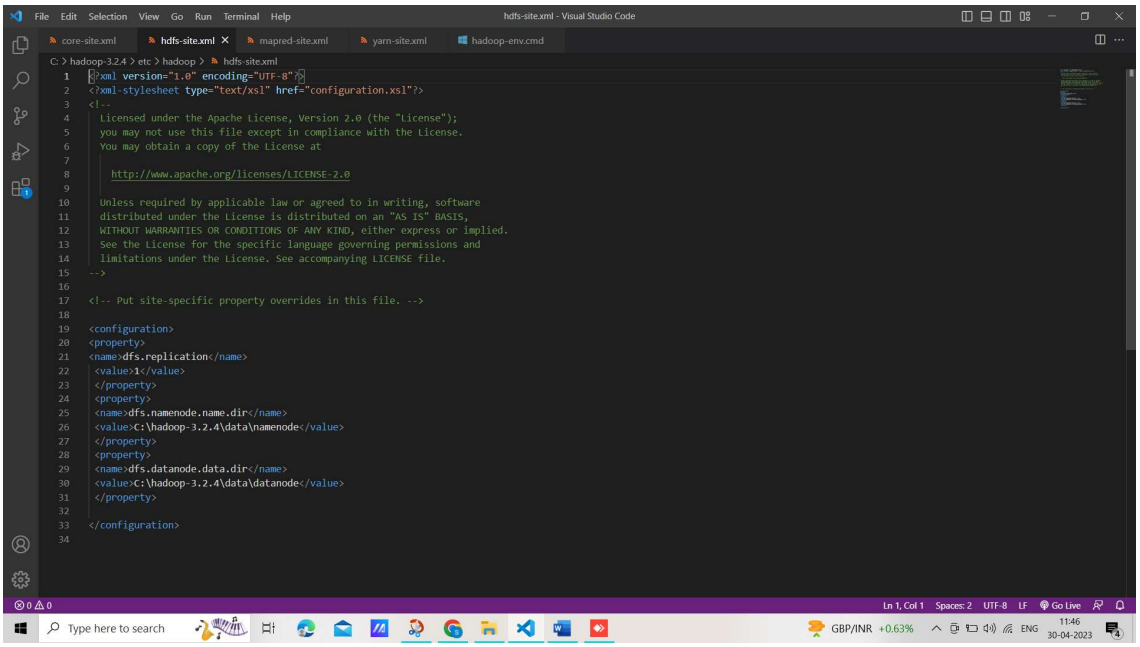


c) Create folder "data" under "C:\Hadoop-3.2.1"

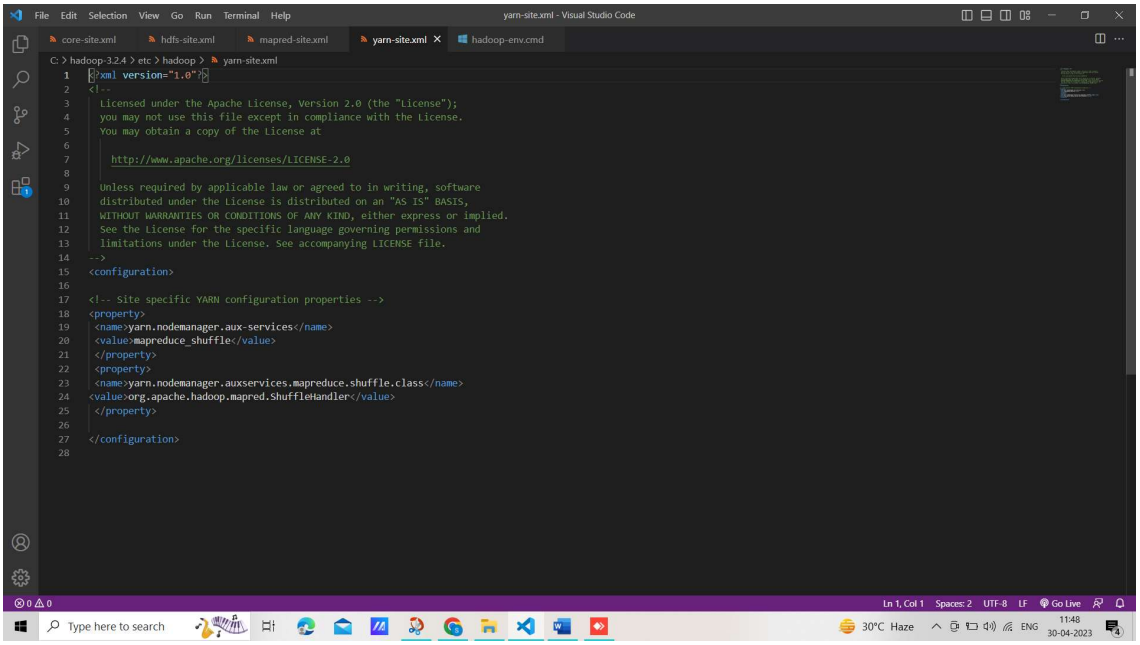
1) Create folder "datanode" under "C:\Hadoop-3.2.1\data"

2) Create folder "namenode" under "C:\Hadoop-3.2.1\data" data

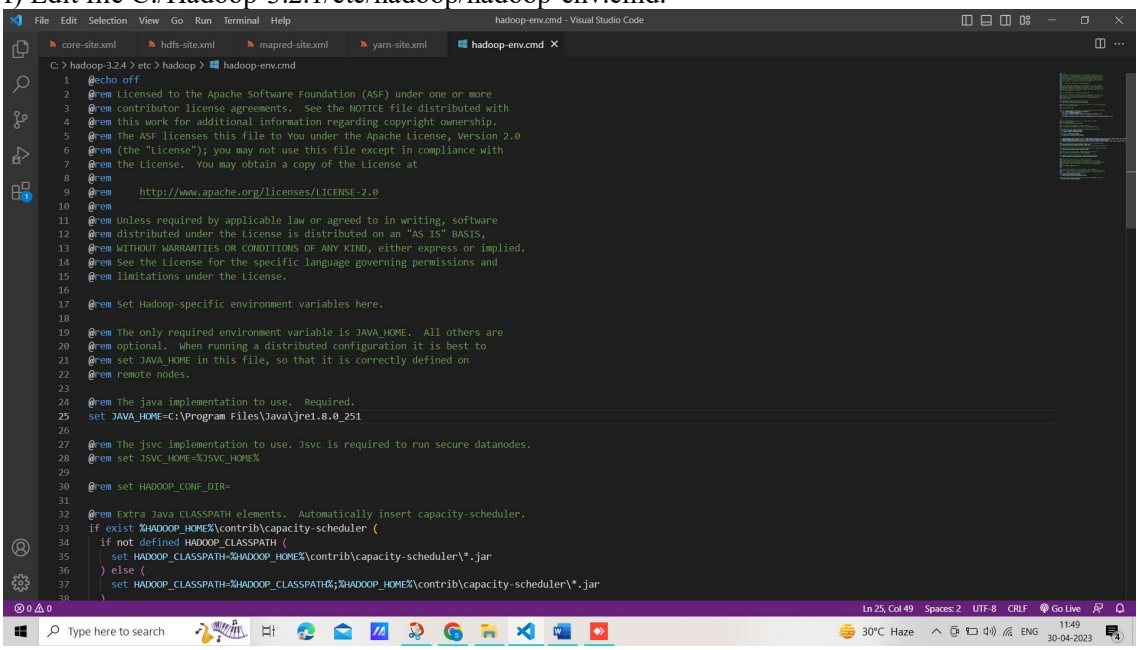
d) Edit file C:\Hadoop-3.2.1/etc/hadoop/hdfs-site.xml, paste below xml paragraph and save this file.



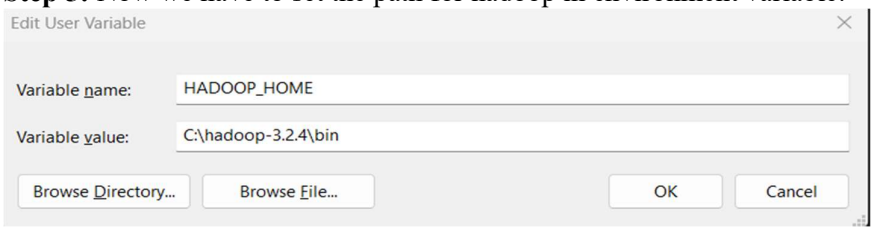
e) Edit file C:/Hadoop-3.2.1/etc/hadoop/yarn-site.xml, paste below xml paragraph and save this file.



f) Edit file C:/Hadoop-3.2.1/etc/hadoop/hadoop-env.cmd.



Step 5: Now we have to set the path for hadoop in environment variable.



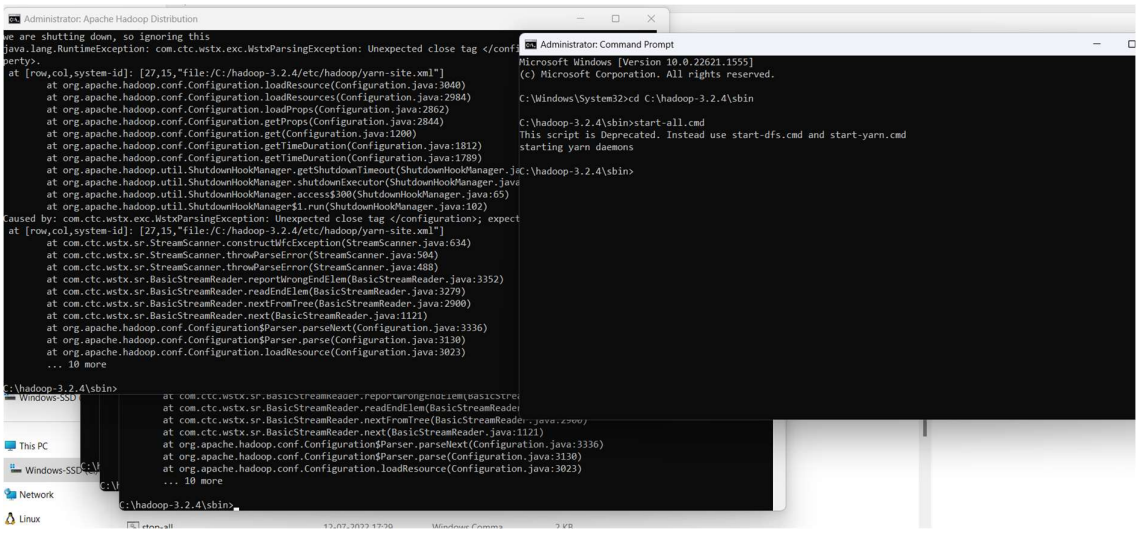
Path

Step 6:

If you see hadoop/bin folder, you'll see some important files are missing.
Download the bin configuration file and extract it.
Now delete the bin folder of hadoop and paste new bin folder.

Step 7:

Start the Hadoop Environment in CMD.



Practical No. : 10

Aim : Basic Commands of HDFS : Mkdir, ls, cat, get, put, copyToLocal, copyFromLocal, mv, tail, touchz, cp, rm, rmr, chmod.

>start-all: Start all Hadoop daemons, the namenode, datanode, the jobtracker and the tasktracker.

>**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.

>**mkdir:**This command creates directory in HDFS if it does not already exist.

>**ls:**This command shows the list of file/content in a directory.

```
C:\hadoop-3.2.4\sbin>start-all
This script is Deprecated. Instead use start-dfs.cmd and start-yarn.cmd
starting yarn daemons

C:\hadoop-3.2.4\sbin>jps
14160 Jps
14292 NameNode
6996 ResourceManager
19816 DataNode
21032 NodeManager

C:\hadoop-3.2.4\sbin>hadoop fs -mkdir /Aniket

C:\hadoop-3.2.4\sbin>hadoop fs -mkdir /XYZ

C:\hadoop-3.2.4\sbin>hadoop fs -ls /
Found 3 items
drwxr-xr-x   - Siddhi supergroup          0 2023-05-21 11:05 /Aniket
drwxr-xr-x   - Siddhi supergroup          0 2023-05-21 11:05 /XYZ
-rw-r--r--   1 Siddhi supergroup          0 2023-05-21 10:47 /hello1
```

>touchz: This command creates a file in HDFS with file size equals to 0 byte.

```
C:\hadoop-3.2.4\sbin>hadoop fs -touchz /hello1.txt
```

>**copyfromlocal:** Hadoop copyFromLocal command is used to copy the file from your local file system to the HDFS.
>**cat:** This command reads the file in HDFS and displays the content of the file.
>**put:**This command is used to copy the file from the local file system to the Hadoop HDFS file system.

```
C:\hadoop-3.2.4\sbin>hadoop fs -copyFromLocal C:/Aniket/hello1.txt /Aniket

C:\hadoop-3.2.4\sbin>hadoop fs -cat /Aniket/hello1.txt
hello Aniket
How are You?
C:\hadoop-3.2.4\sbin>hadoop fs -put C:/Aniket/hadoop.txt /Aniket

C:\hadoop-3.2.4\sbin>hadoop fs -cat /Aniket/hello1.txt
hello Aniket
```

>**copytolocal:**This command is used to copy the data from HDFS to the local filesystem.
>**get:**This command copies files from HDFS file system to local file system.

```
C:\hadoop-3.2.4\sbin>hadoop fs -copyToLocal /Aniket C:/Aniket/hello1.txt
copyToLocal: `/Aniket/hello1.txt': File exists

C:\hadoop-3.2.4\sbin>hadoop fs -get /Aniket C:/Aniket/hadoop.txt
get: `/Aniket/hadoop.txt': File exists
```

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

```
C:\hadoop-3.2.4\sbin>hadoop fs -mv /Aniket /XYZ
```

>rm:This command removes a file from HDFS.

```
C:\hadoop-3.2.4\sbin>hadoop fs -rm /hello1.txt
Deleted /hello1.txt
```

>rmr:This command removes a file from HDFS and can be used to delete files recursively.

```
C:\hadoop-3.2.4\sbin>hadoop fs -rmr /XYZ
rmr: DEPRECATED: Please use '-rm -r' instead.
Deleted /XYZ
```

>tail: This command shows the last 1KB of the file on the console.

```
C:\hadoop-3.2.4\sbin>hadoop fs -tail /siddhi/hellohadoop.txt
hi   siddhi
how are uh?
```