Ex.No 1: Procedure to set up single node cluster using hadoop

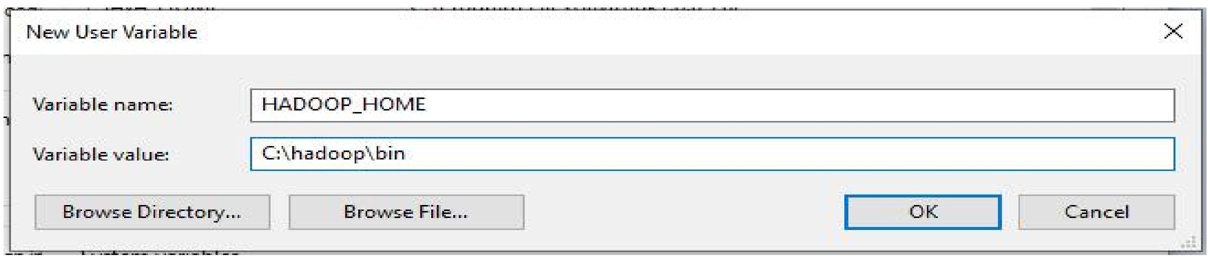
Step 1: Verify the Java installed

javac -version

Step 2: Extract Hadoop at C:\Hadoop

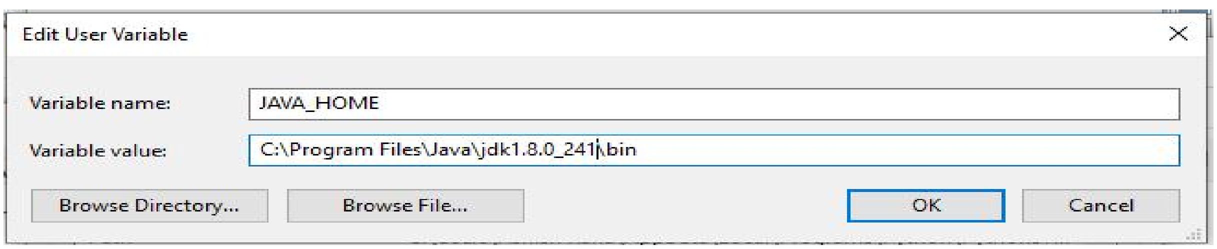
Step 3: Setting up the HADOOP_HOME variable

Use windows environment variable setting for Hadoop Path setting.

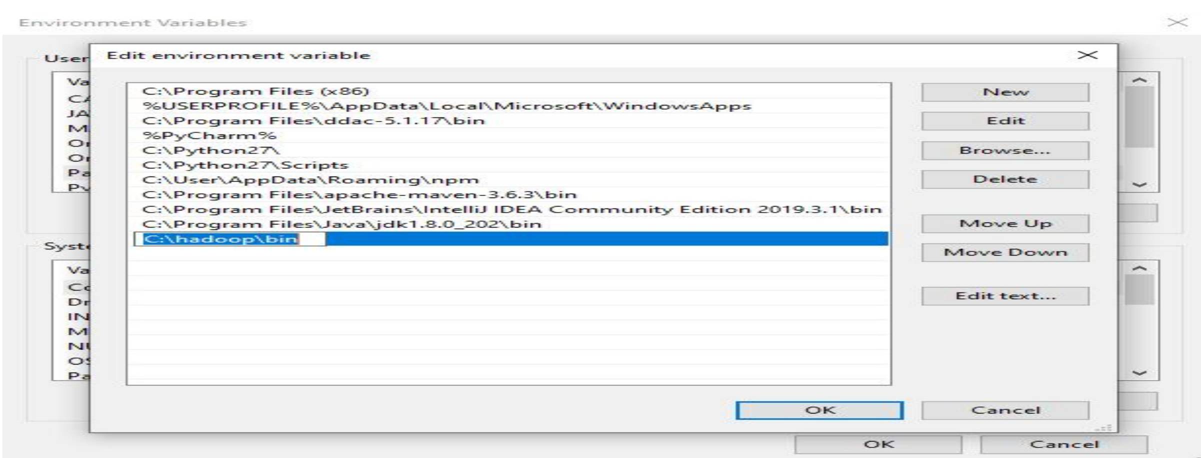


Step 4: Set JAVA_HOME variable

Use windows environment variable setting for Hadoop Path setting.



Step 5: Set Hadoop and Java bin directory path



Step 6: Hadoop Configuration :

For Hadoop Configuration we need to modify Six files that are listed below-

1. Core-site.xml

2. Mapred-site.xml

3. Hdfs-site.xml

4. Yarn-site.xml

5. Hadoop-env.cmd

6. Create two folders datanode and namenode

Step 6.1: Core-site.xml configuration

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

Step 6.2: Mapred-site.xml configuration

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

Step 6.3: Hdfs-site.xml configuration

```
<configuration>
  <property>
     <name>dfs.replication</name>
     <value>1</value>
  </property>
  <property>
     <name>dfs.namenode.name.dir</name>
     <value>C:\hadoop-2.8.0\data\namenode</value>
  </property>
  <property>
```

<name>dfs.datanode.data.dir</name>

<value>C:\hadoop-2.8.0\data\datanode</value>

</property>

</configuration>

Step 6.4: Yarn-site.xml configuration

<configuration>

<property>

<name>yarn.nodemanager.aux-services</name>

<value>mapreduce_shuffle</value>

</property>

<property>

<name>yarn.nodemanager.auxservices.mapreduce.shuffle.class</name>

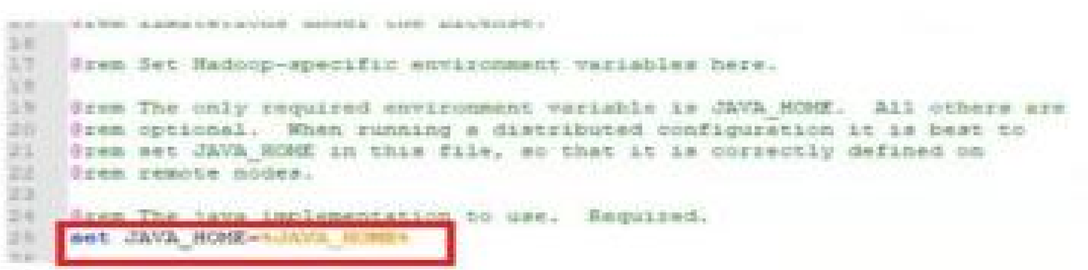
<value>org.apache.hadoop.mapred.ShuffleHandler</value>

</property>

</configuration>

Step 6.5: Hadoop-env.cmd configuration

Set "JAVA_HOME=C:\Java" (On C:\java this is path to file jdk.18.0)



Step 6.6: Create datanode and namenode folders

1. Create folder "data" under "C:\Hadoop-2.8.0"

2. Create folder "datanode" under "C:\Hadoop-2.8.0\data"

3. Create folder "namenode" under "C:\Hadoop-2.8.0\data"

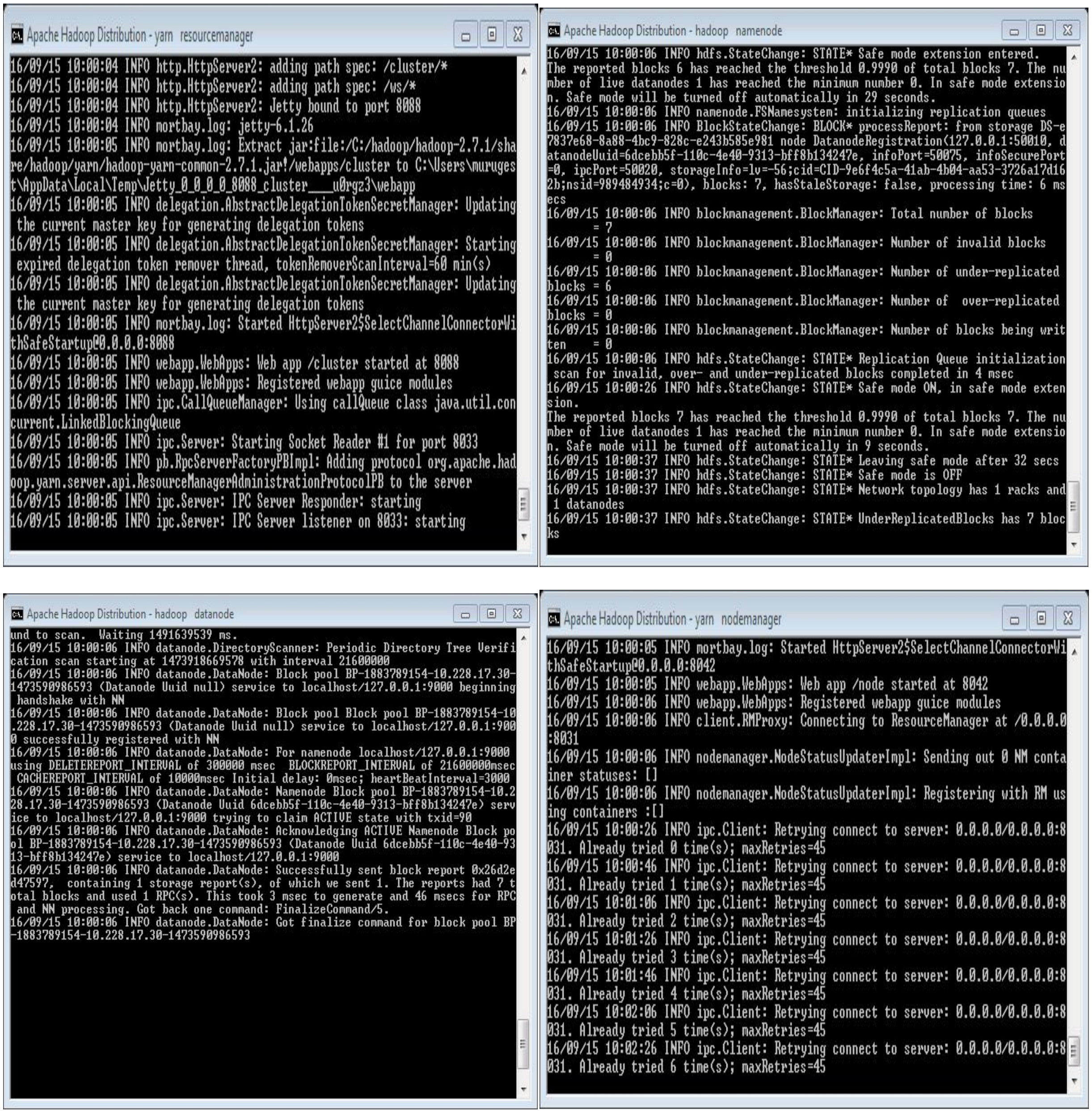
Step 7: Format the namenode folder

Open command window (cmd) and typing command “hdfs namenode –format”

Step 8: Testing the setup

Open command window (cmd) and typing command “start-all.cmd”

C:\hadoop\hadoop-2.7.1\sbin>start-all.cmd



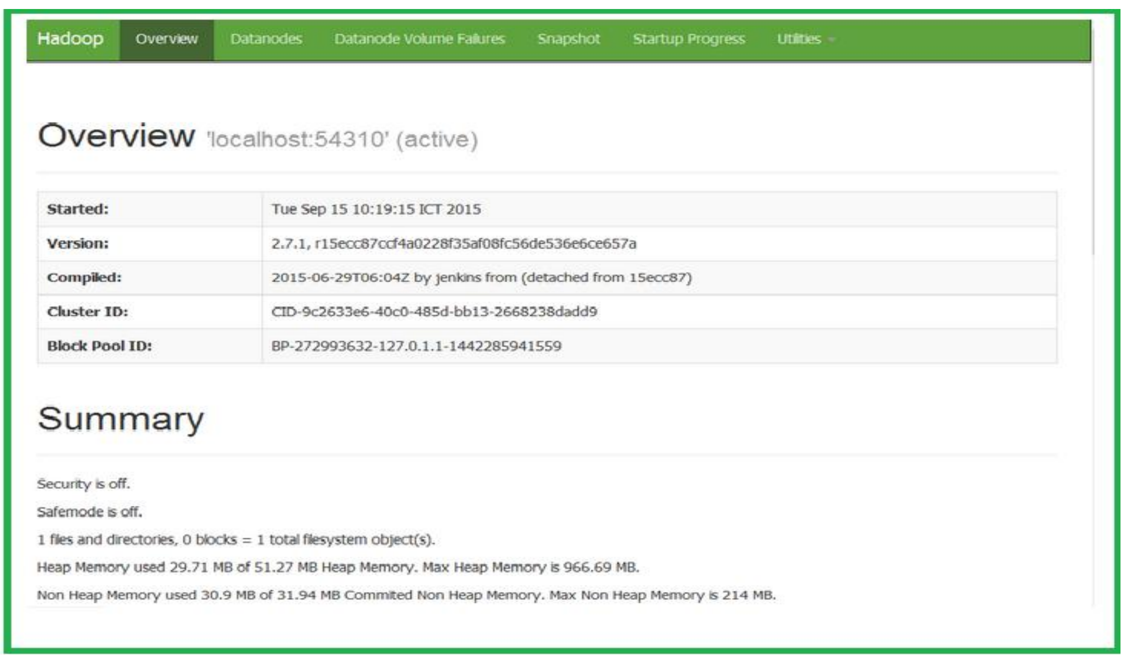
C:\hadoop\hadoop-2.7.1\bin>hadoop fs -mkdir /cluster1

C:\hadoop\hadoop-2.7.1\bin>

Ensure that namenode, datanode, and Resource manager are running

Step 9: Open: http://localhost:8088

Step 10: Open: http://localhost:50070



Ex.No 2: Word Count Program to use of Map and Reduce Tasks

```
import java.io.IOException;
import java.util.StringTokenizer;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
public class WordCount {
  public static class TokenizerMapper
       extends Mapper<Object, Text, Text, IntWritable>{
    private final static IntWritable one = new IntWritable(1);
    private Text word = new Text();
    public void map(Object key, Text value, Context context
                    ) throws IOException, InterruptedException {
      StringTokenizer itr = new StringTokenizer(value.toString());
      while (itr.hasMoreTokens()) {
        word.set(itr.nextToken());
        context.write(word, one);
      }
    }
  }

  public static class IntSumReducer
       extends Reducer<Text,IntWritable,Text,IntWritable> {
    private IntWritable result = new IntWritable();
    public void reduce(Text key, Iterable<IntWritable> values,
                       Context context
                       ) throws IOException, InterruptedException {
      int sum = 0;
      for (IntWritable val : values) {
        sum += val.get();
      }
      result.set(sum);
      context.write(key, result);
    }
  }

  public static void main(String[] args) throws Exception {
```

```
    Configuration conf = new Configuration();
    Job job = Job.getInstance(conf, "word count");
    job.setJarByClass(WordCount.class);
    job.setMapperClass(TokenizerMapper.class);
    job.setCombinerClass(IntSumReducer.class);
    job.setReducerClass(IntSumReducer.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);
    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));
    System.exit(job.waitForCompletion(true) ? 0 : 1);
  }
}
```

Output:

C:\hadoop\hadoop-2.7.1\bin>set classpath=c:\hadoop\hadoop-core-1.2.1.jar

C:\hadoop\hadoop-2.7.1\bin>set classpath=c:\program files\java\jdk1.7.0\bin

C:\hadoop\hadoop-2.7.1\bin>javac WordCount.java

C:\hadoop\hadoop-2.7.1\bin>jar -cvf wc.jar WordCount*.class

C:\hadoop\hadoop-2.7.1\bin>hadoop fs -mkdir /f1

C:\hadoop\hadoop-2.7.1\bin>hadoop fs -cat file1.txt

Hello world Bye world

C:\hadoop\hadoop-2.7.1\bin>hadoop fs -put file1.txt /f1

C:\hadoop\hadoop-2.7.1\bin>hadoop jar wc.jar WordCount /f1 /f2

File System Counters

FILE: Number of bytes read=6604

FILE: Number of bytes written=566732

FILE: Number of read operations=0

FILE: Number of large read operations=0

FILE: Number of write operations=0

HDFS: Number of bytes read=42

HDFS: Number of bytes written=22

HDFS: Number of read operations=13

HDFS: Number of large read operations=0

HDFS: Number of write operations=4

Map-Reduce Framework

Map input records=1

Map output records=4

Map output bytes=38

Map output materialized bytes=40

Input split bytes=99

Combine input records=4

```
        Combine output records=3
        Reduce input groups=3
        Reduce shuffle bytes=40
        Reduce input records=3
        Reduce output records=3
        Spilled Records=6
        Shuffled Maps =1
        Failed Shuffles=0
        Merged Map outputs=1
        GC time elapsed (ms)=0
        Total committed heap usage (bytes)=641728512
    Shuffle Errors
        BAD_ID=0
        CONNECTION=0
        IO_ERROR=0
        WRONG_LENGTH=0
        WRONG_MAP=0
        WRONG_REDUCE=0
    File Input Format Counters
        Bytes Read=21
    File Output Format Counters
        Bytes Written=22
C:\hadoop\hadoop-2.7.1\sbin>hadoop fs -cat /f3/part-r-00000
Bye     1
Hello   1
world   2
```

Ex.No 3: Python basic for pandas

1. import pandas as pd

2. # Reading data from google drive in to a data frame
   data=pd.read_csv("/content/drive/My Drive/ Code/data/cafe.csv")

3. # Viewing data head (default: first 5 rows)
   data.head()

4. # Sum of a column
   data["AFF-Mon"].sum()

5. # Minimum value in a column
   data["AFF-Mon"].min()

6. # Numbr of values in a column
   data["AFF-Mon"].count()

7. # Mean value of a column
   data["AFF-Mon"].mean()

8. # Basic statistical values of a column
   data["AFF-Mon"].describe()

9. # STatistics of the whole data frame
   data.describe()

10. # STatistics of the whole data frame including all columns (numeric and non-numeric)
    data.describe(include="all")

11. # View last 5 rows of a data frame
    data.tail()

12. # View column names
    data.columns

13. # Unique values in a column
    data["State"].unique()

14. # Number of unique values in a column
    data["State"].nunique()

15. # Number of rows and columns in a data frame
    data.shape

16. # Add column to a data frame
    # First create a list of numbers for 85 rows
    x=[i for i in range(85)]
    # Assign the list to new column
    data["New Index"]=x

17. # View column names
    data.columns

18. data.shape

19. data.head()

20. # Dropping (removing) a column
    data.drop("New Index",axis=1,inplace=True) # axis=1 for columns and axis=0 for rows.

21. # Conditional statements (or filtering data using conditions)
22. # Example, all store data from the state Tamil Nadu (TN)
    data[data["State"]=="TN"]

23. # Detecting missing values
    data.isna()

24. # Detecting missing values in a single column
    data['Store ID'].isna()

25. # Number of missing values per column
    data.isna().sum()

26. # Number of missing valyes in the data frame
    data.isna().sum().sum()

27. # Reading "train.csv" data from google drive in to a data frame
    df=pd.read_csv("/content/drive/My Drive/Colab Notebooks/ATAL FDP GGV Python Code/data/train.csv")

28. # View first 5 rows
    df.head()

29. # Detecting missing values
    df.isna().sum()

30. # Filling missing values with mean data
31. df["Age"]=df["Age"].fillna(df["Age"].mean())

32. df.isna().sum()

Ex.No 4: Python Code for Matplotlib

1. import matplotlib.pyplot as plt

2. plt.plot([1,2,3,4,5],[1,4,6,8,9],"ro")
   plt.show()

3. plt.plot([1,2,3,4,5],[1,4,6,8,9])
   plt.title("Graph")
   plt.xlabel("X-axis")
   plt.ylabel("Y-axis")
   plt.show()

4. import numpy as np

5. t=np.arange(0.,5.,0.2)
   t

6. array([0. , 0.2, 0.4, 0.6, 0.8, 1. , 1.2, 1.4, 1.6, 1.8, 2. , 2.2, 2.4,
   2.6, 2.8, 3. , 3.2, 3.4, 3.6, 3.8, 4. , 4.2, 4.4, 4.6, 4.8])
   plt.plot(t,t,"r--",t,t**2,'bs',t,t**3,'g^')
   plt.title("Linear, Quadratic and Cubic graphs")
   plt.xlabel("X-axis")
   plt.ylabel("Y-axis")
   plt.show()

7. x1=[5,8,10]
   y1=[12,6,16]
   x2=[6,9,11]
   y2=[6,15,17]
   plt.plot(x1,y1,"green",label="Age",linewidth=5)
   plt.plot(x2,y2,"blue",label="Income",linewidth=5)
   plt.legend()
   plt.grid(True,color='lightgrey')
   plt.show()

8. # Bar graph
   x1=[5,8,10]
   y1=[12,6,16]
   plt.bar(x1,y1,color="green")
   plt.show()

9. # Scatter plot
   x=[6,9,11,14,20,24,30]
   y=[6,15,17,22,30,37,44]
   plt.scatter(x,y)

plt.show()

10. # Plotting categorial graph
    x=[100,200,300,400]
    y=[12,20,30,40]
    plt.figure(figsize=(9,3))
    plt.subplot(131)
    plt.bar(x,y)
    plt.subplot(132)
    plt.scatter(x,y)
    plt.subplot(133)
    plt.plot(x,y)

11. import pandas as pd
    data=pd.read_csv("/content/drive/My Drive/cafe.csv")
    data.head()

12. addr=data["Address"].head()
    addr

13. mon=data["AFF-Mon"].head()
    plt.bar(addr,mon)
    plt.show()

14. plt.bar(addr,mon)
    plt.xticks(addr,('a','b','c','d','e'))
    plt.shw()

15. tue=data["AFF-Tue"].head()
    wed=data["AFF-Wed"].head()
    # Saving graph to a file
    plt.plot(addr,mon,'r--',addr,tue,'bs',addr,wed,'g^')
    plt.xticks(rotation=90)
    plt.savefig("/content/drive/My Drive/Colab Notebooks/MyGraph.png",format='png')
    plt.show()

16. plt.scatter(addr,mon)
17. plt.xticks(rotation=90)
18. plt.show()

19. import folium as f
    f.Map(location=[18.932308,72.834091])

Ex.No 5: Python Code for Seaborn

1.
   ```
   import pandas as pd
   import numpy as np
   import matplotlib.pyplot as plt
   import seaborn as sns
   data=pd.read_csv("/content/drive/My Drive/train.csv")
   data.head()
   ```

2.
   ```
   # Default Seaborn datasets
   sns.get_dataset_names()
   ```

3.
   ```
   # Loading Seaborn predefined dataset
   df=sns.load_dataset('tips')
   df.head()
   ```

4.
   ```
   # Range and outliers
   plt.scatter(data["Sex"],data["Age"])
   plt.show()
   ```

5.
   ```
   plt.style.use("seaborn")
   fig,ax=plt.subplots(figsize=(15,9))
   ax.boxplot([data["Fare"]])
   plt.show()
   ```

6.
   ```
   data["Fare"].max()
   512.3292
   ```

7.
   ```
   # Identifying outliers using z-score
   z_score=(data["Age"]-data["Age"].mean())/data["Age"].std()
   print(z_score)
   ```

8.
   ```
   # Example: use only data where z-score is less than 3
   z_score<3
   ```

9.
   ```
   # Adding z_score column to data
   data['z_score']=z_score
   data.head()
   ```

10.
    ```
    # Filtering data where z_score < 3
    data[data['z_score']<3]
    ```

11.
    ```
    data.skew()
    ```

12. # Histogram (Frequency distribution)
```
sns.displot(data["Age"],kde=True)
plt.show()
```

13. `data.groupby(["Sex","Ticket"]).describe()`

14.
```
data.groupby(["Embarked","Sex","Pclass"]).size()
data.groupby(["Embarked","Sex","Pclass"]).size().unstack()
```

15. # Arranging data into bins
```
group_data=pd.DataFrame(pd.cut(data["Age"],bins=[0,15,30,45,60,75,90]))
group_data.head()
```

16. `group_data.groupby(["Age"]).size()`

17. # Bar graph
```
group_data.groupby(["Age"]).size().plot(kind="bar")
plt.show()
```

18. # Pie plot
```
group_data.groupby(["Age"]).size().plot(kind="pie")
plt.show()
```

```
# 3D plots
```
19.
```
from mpl_toolkits import mplot3d
fig=plt.figure()
ax=plt.axes(projection="3d")
plt.show()
```

20.
```
ax=plt.axes(projection="3d")
z=np.linspace(0,15,1000)
x=np.sin(z)
y=np.cos(z)
ax.plot3D(x,y,z,"grey")

zdata=15*np.random.random(100)
xdata=np.sin(zdata)+0.1*np.random.random(100)
ydata=np.cos(zdata)+0.1*np.random.random(100)

plt.figure(figsize=(10,7))
sns.pairplot(data)
plt.show()
```

Ex.No 6: Regression Algorithm

1. import pandas as pd

2. Stock_Market = {'Year':
[2017,2017,2017,2017,2017,2017,2017,2017,2017,2017,2017,2017,2016,2016,2016,2016,2016,2016,2016,2016,2016,2016,2016,2016],
'Month': [12, 11,10,9,8,7,6,5,4,3,2,1,12,11,10,9,8,7,6,5,4,3,2,1],
'Interest_Rate':
[2.75,2.5,2.5,2.5,2.5,2.5,2.5,2.25,2.25,2.25,2,2,2,1.75,1.75,1.75,1.75,1.75,1.75,1.75,1.75,1.75,1.75,1.75],
'Unemployment_Rate':
[5.3,5.3,5.3,5.3,5.4,5.6,5.5,5.5,5.5,5.6,5.7,5.9,6,5.9,5.8,6.1,6.2,6.1,6.1,6.1,5.9,6.2,6.2,6.1],
'Stock_Index_Price':
[1464,1394,1357,1293,1256,1254,1234,1195,1159,1167,1130,1075,1047,965,943,958,971,949,884,866,876,822,704,719]
}

3. df = pd.DataFrame Stock_Market,columns=
['Year','Month','Interest_Rate','Unemployment_Rate', 'Stock_Index_Price'])
print (df)

4. import os
os.chdir('/content/drive/My Drive/')
os.getcwd()

5. df.to_csv('data/stock.csv', index=False)
df=pd.read_csv('data/stock.csv')
df.head()

6. # Check for Linearity
import matplotlib.pyplot as plt
plt.scatter(df['Interest_Rate'], df['Stock_Index_Price'], color='red')
plt.title('Stock Index Price Vs Interest Rate', fontsize=14)
plt.xlabel('Interest Rate', fontsize=14)
plt.ylabel('Stock Index Price', fontsize=14)
plt.grid(True)
plt.show()

7. plt.scatter(df['Unemployment_Rate'], df['Stock_Index_Price'], color='blue')
plt.title('Stock Index Price Vs Unemployment Rate', fontsize=14)
plt.xlabel('Unemployment Rate', fontsize=14)
plt.ylabel('Stock Index Price', fontsize=14)
plt.grid(True)
plt.show()

8. from sklearn import linear_model
   import statsmodels.api as sm

9. # Perform Multi-Linear Regression
   # here we have 2 variables for multiple regression. If you just want to use one
   # variable for simple linear regression, then use X = df['Interest_Rate'] for
   # example. Alternatively, you may add additional variables within the brackets
   X = df[['Interest_Rate','Unemployment_Rate']]
   Y = df['Stock_Index_Price']
   # with sklearn
   regr = linear_model.LinearRegression()
   regr.fit(X, Y)
   print('Intercept: \n', regr.intercept_)
   print('Coefficients: \n', regr.coef_)

   # prediction with sklearn
   New_Interest_Rate = 2.75
   New_Unemployment_Rate = 5.3
   print ('Predicted Stock Index Price: \n',
   regr.predict([[New_Interest_Rate ,New_Unemployment_Rate]]))

   # With StatModels
   X = sm.add_constant(X) # adding a constant
   model = sm.OLS(Y, X).fit() # Ordinary Least Squares
   predictions = model.predict(X)
   print_model = model.summary()
   print(print_model)

Ex.No 7: Model Training

1. from google.colab import drive
   drive.mount('/content/drive')

2. import os
   os.chdir('/content/drive/My Drive/ ')
   os.getcwd()

3. from keras import layers
   from keras import models

4. model = models.Sequential()

5. model.add(layers.Conv2D(32, (3, 3), activation='relu',
   input_shape=(150, 150, 3)))

6. model.add(layers.MaxPooling2D((2, 2)))
   model.add(layers.Conv2D(64, (3, 3), activation='relu'))
   model.add(layers.MaxPooling2D((2, 2)))
   model.add(layers.Conv2D(128, (3, 3), activation='relu'))
   model.add(layers.MaxPooling2D((2, 2)))
   model.add(layers.Conv2D(128, (3, 3), activation='relu'))
   model.add(layers.MaxPooling2D((2, 2)))
   model.add(layers.Flatten())
   model.add(layers.Dropout(0.5))
   model.add(layers.Dense(512, activation='relu'))
   model.add(layers.Dense(1, activation='sigmoid'))

7. ***Let's look at how the dimensions of the feature maps change with every successive layer:***
   model.summary()

8. # **Configuring the model for training**
   For the compilation step, you'll go with the RMSprop optimizer. Because you ended the network with a single sigmoid unit, you'll use binary crossentropy as the loss
   from keras import optimizers
   model.compile(loss='binary_crossentropy',
   optimizer=optimizers.RMSprop(lr=1e-4),
   metrics=['accuracy'])

9. # **Data preprocessing**

Currently, the data sits on a drive as JPEG files, so the steps for getting it into the network are roughly as follows:

1. Read the picture files.
2. Decode the JPEG content to RGB grids of pixels.
3. Convert these into floating-point tensors.
4. Rescale the pixel values (between 0 and 255) to the [0, 1] interval (as you know, neural networks prefer to deal with small input values).

Using ImageDataGenerator to read images from directories

```
from keras.preprocessing.image import ImageDataGenerator
train_datagen = ImageDataGenerator(rescale=1./255)
#test_datagen = ImageDataGenerator(rescale=1./255)
train_generator = train_datagen.flow_from_directory('data/images',target_size=(150, 150),
batch_size=20, class_mode='binary')
```

10. The output of one of these generators: it yields batches of 150 × 150 RGB images (shape (20, 150, 150, 3)) and binary labels (shape (20,)).There are 20 samples in each batch (the batch size).

```
for data_batch, labels_batch in train_generator:
print('data batch shape:', data_batch.shape)
print('labels batch shape:', labels_batch.shape)
break
```

11. # **Fitting the model using a batch generator**

```
history = model.fit_generator( train_generator,steps_per_epoch=10,epochs=5)
```

12. # **Saving the model**

```
model.save('output/mymodel.h5')
```

13. ## **Displaying curves of loss and accuracy during training**

```
import matplotlib.pyplot as plt
acc = history.history['accuracy']
loss = history.history['loss']
epochs = range(1, len(acc) + 1)
plt.plot(epochs, acc, 'b', label='Training acc')
plt.title('Training accuracy')
plt.legend()
plt.figure()
plt.plot(epochs, loss, 'b', label='Training loss')
plt.title('Training loss')
plt.legend()
plt.show()
```

Ex.No 8: Neural_Networks

1.
```
import keras
from keras.layers import Dense
from keras.models import Sequential
```

2.
```
import os
os.chdir('/content/drive/My Drive/')
os.getcwd()
```

3.
```
import pandas as pd
import numpy as np
```

4.
```
ds=pd.read_csv("Iris.csv")
ds.head()
```

5.
```
arr=ds.values
x=arr[:,0:4]
y=arr[:,4]
```

6.
```
model=Sequential()
model.add(Dense(units=64,input_dim=4,activation="relu"))
model.add(Dense(units=32,activation="relu"))
model.add(Dense(units=3,activation="softmax"))
```

7.
```
model.compile(loss='categorical_crossentropy', optimizer='adam',
              metrics=['accuracy'])
```

8.
```
from sklearn.preprocessing import LabelEncoder
from keras.utils import np_utils
encoder = LabelEncoder()
encoder.fit(y)
encoded_Y = encoder.transform(y)
dummy_y = np_utils.to_categorical(encoded_Y)
```

9. h=model.fit(x.astype('float'),dummy_y,epochs=10)

```
Epoch 1/10
5/5 [==============================] - 1s 5ms/step - loss: 3.8956 - accuracy: 0.5867
Epoch 2/10
5/5 [==============================] - 0s 3ms/step - loss: 1.5611 - accuracy: 0.4000
Epoch 3/10
5/5 [==============================] - 0s 4ms/step - loss: 0.9363 - accuracy: 0.4800
Epoch 4/10
5/5 [==============================] - 0s 3ms/step - loss: 0.9004 - accuracy: 0.5667
Epoch 5/10
5/5 [==============================] - 0s 4ms/step - loss: 0.7987 - accuracy: 0.6133
Epoch 6/10
5/5 [==============================] - 0s 3ms/step - loss: 0.6825 - accuracy: 0.6600
Epoch 7/10
5/5 [==============================] - 0s 3ms/step - loss: 0.6173 - accuracy: 0.7933
Epoch 8/10
5/5 [==============================] - 0s 5ms/step - loss: 0.6038 - accuracy: 0.7667
Epoch 9/10
5/5 [==============================] - 0s 3ms/step - loss: 0.5508 - accuracy: 0.7733
Epoch 10/10
5/5 [==============================] - 0s 3ms/step - loss: 0.5352 - accuracy: 0.7867
```

10.
```
from matplotlib import pyplot as plt
plt.plot(h.history['accuracy'])
plt.plot(h.history['loss'])
plt.ylabel('Accuracy/Loss')
plt.xlabel('Epoch')
plt.legend(['Accuracy', 'Loss'], loc='upper left')
plt.show()
```