UNIVERSITY SCHOOL OF AUTOMATION & ROBOTICS (USAR)

Guru Gobind Singh Indraprastha University, East Delhi Campus, Surajmal Vihar, Delhi 110092

Big Data Analytics Lab

(ARD-353)

Submitted to:
Dr. Sanjay Kumar Singh
Asst. Professor, USAR

Submitted By:
Name: Pratham Kumar
Batch: AI-DS (B1-A)
Roll No: 00119011921

Index

S.No	Program	Signature
1	1.1) Installation of Ubuntu using WSL/Virtual box/Dual Boot etc. 1.2) Installation of hadoop latest version in Ubuntu stand-alone mode.	
2	Setting up a Single Node Cluster. Hadoop Installation: Psuedo Distributed Mode(Locally and YARN)	
3	File Management tasks in Hadoop. Creation of folder, deletion of folder, put a file from local drive to hdfs, show the content of hdfs file, download the hdfs file to local drive, delete hdfs file.	
4	Word count Map Reduce program using Java.	
5	Word count Map Reduce program using python.	
6	6.1) Installing apache spark/pyspark. 6.2) WAP to create pyspark configuration, session and count number of words present in a given textfile.	
7	WAP to implement various regression models and evaluate their performance using pyspark.	
8	WAP to implement various classification models and evaluate their performance using pyspark.	
9	WAP to implement various K-means clustering and find the best value of K using the elbow method using pyspark.	
10	WAP to implement a recommender system using pyspark.	

Practical 1.1: Installation of Ubuntu using WSL.

Aim: To install Ubuntu using Windows Subsystem for Linux (WSL).

Objectives:

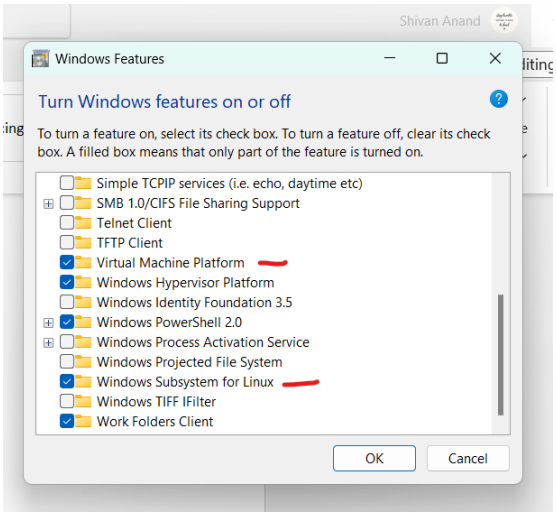
1. To understand the process of setting up WSL.
2. To install Ubuntu distribution within WSL.

Requirements:

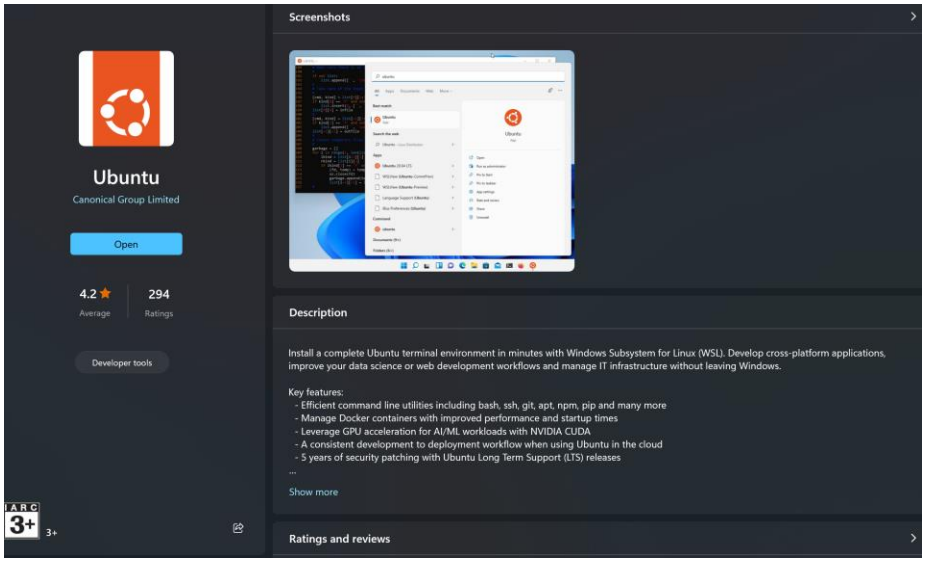
1. Windows 11 operating system or later.
2. Stable internet connection.
3. Access to the Microsoft Store.

Procedure:

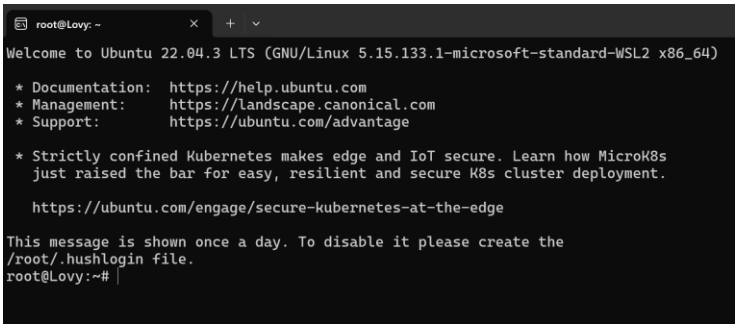
1. Open “Turn Windows features on or off” and check the checkbox for “Virtual MachinePlatform” and “Windows Subsystem for Linux”.



2. Install Ubuntu 22.04.2 LTS from Microsoft Office.



3. Launch Ubuntu 22.04.2 LTS. Enter a username. This will create a local user account and youwill be automatically logged in to Ubuntu 18.04 as this user. Enter a password for the user and enter a second time to confirm.



Observations

1. The download and installation process may take some time depending on the speed of theinternet connection.
2. During the installation, the progress will be displayed in the Microsoft Store.

Results

1. Ubuntu has been successfully installed using WSL.
2. The Ubuntu terminal is accessible from the Start menu and can be used for various Linux operations.

Conclusion

The successful installation of Ubuntu using WSL allows users to leverage the advantages of both Windows and Ubuntu operating systems, enabling a more versatile and comprehensive computing environment.

Practical 1.2: Installation of Hadoop in Ubuntu Stand-Alone Mode. Aim: To install the latest version of Hadoop in stand-alone mode on the Ubuntu operating system.

Objectives:

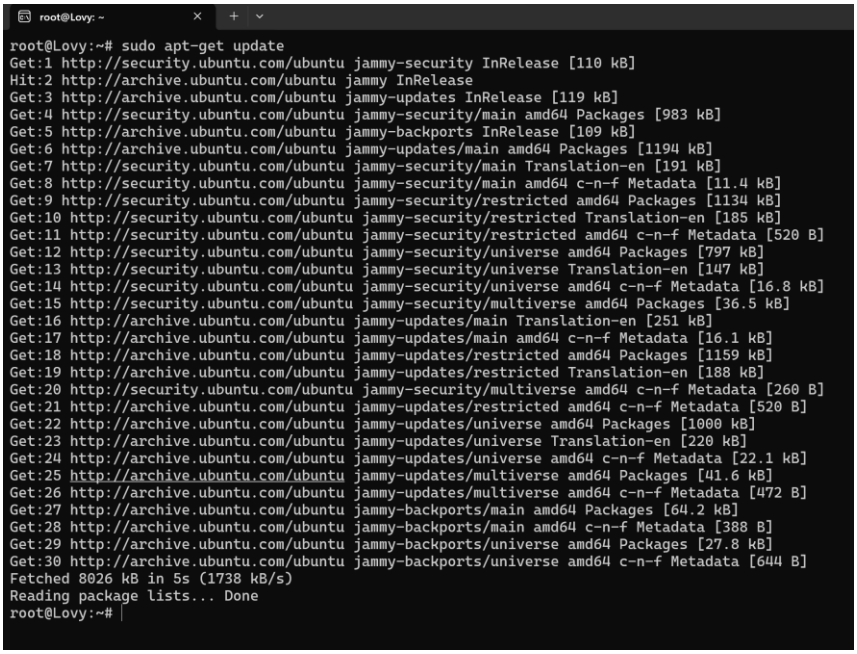
1. Setting up the environment for Hadoop installation.
2. Downloading the latest version of Hadoop.
3. Configuring the Hadoop environment variables.
4. Verifying the successful installation of Hadoop.

Requirements:

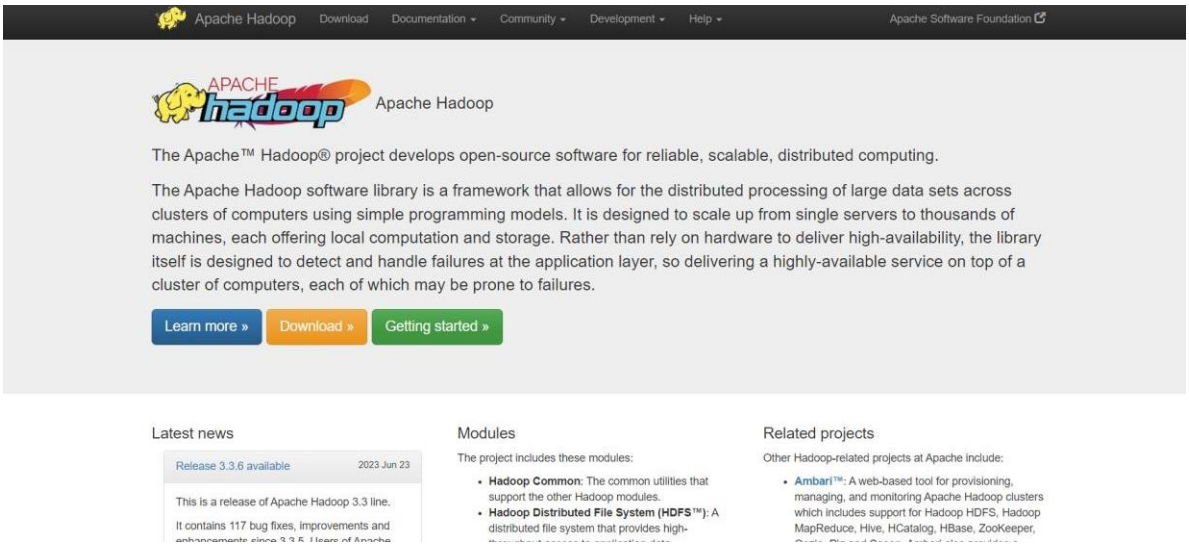
1. Ubuntu operating system installed using WSL or any other method.
2. Stable internet connection.
3. Basic understanding of the Ubuntu command line.

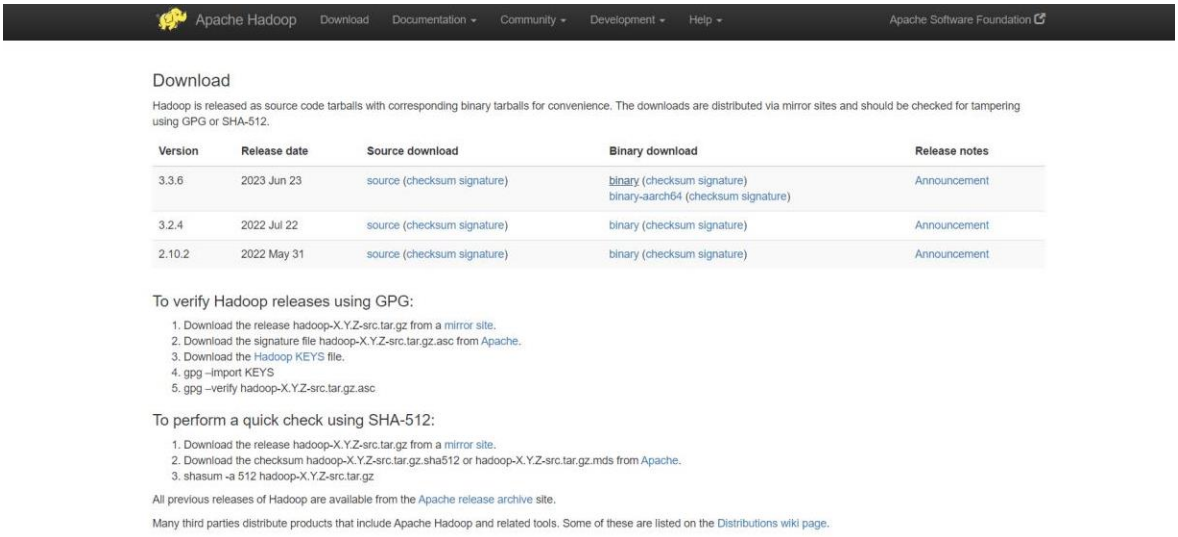
Procedure:

1. Open the Ubuntu terminal.
2. Update the system packages using the command: `*sudo apt-get update*`.



3. Go to https://hadoop.apache.org/ , Click on “Getting Started”, then download the latest Binary download, in our case the latest Hadoop version is 3.3.6.

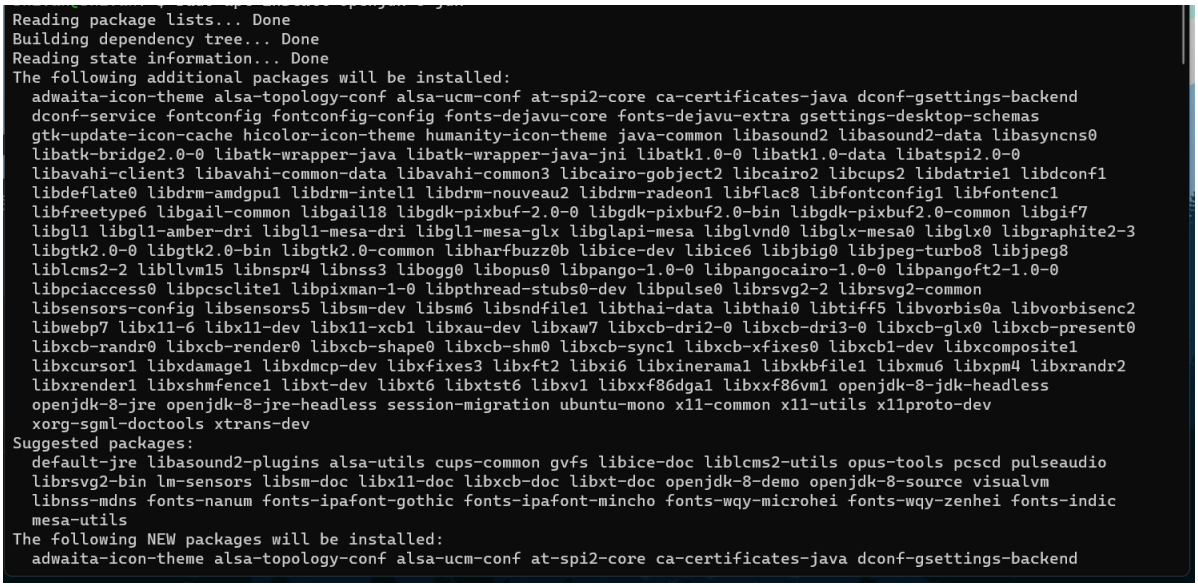




4. Extract the downloaded Hadoop package using the command: `*tar -xzvf /mnt/c/Users/prath/Downloads/hadoop-3.3.6.tar.gz -C /home/prath/* `.

```
hadoop-3.3.6/share/hadoop/hdfs/jdiff/Apache_Hadoop_HDFS_3.0.0.xml
hadoop-3.3.6/share/hadoop/hdfs/jdiff/Apache_Hadoop_HDFS_3.3.1.xml
hadoop-3.3.6/share/hadoop/hdfs/jdiff/Apache_Hadoop_HDFS_3.2.2.xml
hadoop-3.3.6/share/hadoop/hdfs/jdiff/Apache_Hadoop_HDFS_3.2.3.xml
hadoop-3.3.6/share/hadoop/hdfs/jdiff/Apache_Hadoop_HDFS_2.8.2.xml
hadoop-3.3.6/share/hadoop/hdfs/jdiff/Apache_Hadoop_HDFS_3.3.3.xml
hadoop-3.3.6/share/hadoop/hdfs/jdiff/Null.java
hadoop-3.3.6/share/hadoop/hdfs/jdiff/Apache_Hadoop_HDFS_2.8.3.xml
hadoop-3.3.6/share/hadoop/hdfs/jdiff/Apache_Hadoop_HDFS_3.3.5.xml
hadoop-3.3.6/share/hadoop/hdfs/jdiff/Apache_Hadoop_HDFS_2.8.0.xml
hadoop-3.3.6/share/hadoop/hdfs/jdiff/Apache_Hadoop_HDFS_3.0.3.xml
hadoop-3.3.6/share/hadoop/hdfs/jdiff/hadoop-hdfs_0.22.0.xml
hadoop-3.3.6/share/hadoop/hdfs/jdiff/Apache_Hadoop_HDFS_2.9.1.xml
hadoop-3.3.6/share/hadoop/hdfs/jdiff/Apache_Hadoop_HDFS_3.1.1.xml
hadoop-3.3.6/share/hadoop/hdfs/jdiff/hadoop-hdfs_0.20.0.xml
hadoop-3.3.6/share/hadoop/hdfs/jdiff/Apache_Hadoop_HDFS_3.0.0-alpha4.xml
hadoop-3.3.6/share/hadoop/hdfs/jdiff/Apache_Hadoop_HDFS_3.2.0.xml
hadoop-3.3.6/share/hadoop/hdfs/jdiff/Apache_Hadoop_HDFS_2.9.2.xml
hadoop-3.3.6/share/hadoop/hdfs/jdiff/Apache_Hadoop_HDFS_3.0.0-alpha2.xml
hadoop-3.3.6/share/hadoop/hdfs/jdiff/Apache_Hadoop_HDFS_3.0.2.xml
hadoop-3.3.6/share/hadoop/hdfs/jdiff/Apache_Hadoop_HDFS_2.10.0.xml
hadoop-3.3.6/share/hadoop/hdfs/jdiff/Apache_Hadoop_HDFS_3.1.0.xml
hadoop-3.3.6/share/hadoop/hdfs/jdiff/Apache_Hadoop_HDFS_3.0.1.xml
hadoop-3.3.6/share/hadoop/hdfs/jdiff/Apache_Hadoop_HDFS_3.2.1.xml
hadoop-3.3.6/share/hadoop/hdfs/jdiff/Apache_Hadoop_HDFS_3.2.4.xml
hadoop-3.3.6/share/hadoop/hdfs/jdiff/hadoop-hdfs_0.21.0.xml
hadoop-3.3.6/share/hadoop/hdfs/jdiff/Apache_Hadoop_HDFS_3.1.3.xml
hadoop-3.3.6/share/hadoop/hdfs/hadoop-hdfs-client-3.3.6-tests.jar
hadoop-3.3.6/share/hadoop/hdfs/hadoop-hdfs-httpfs-3.3.6.jar
```

5. Install Java JDK using the command: `*sudo apt install openjdk-8 -jdk*`.



6. Open .bashrc file using the command "*sudo nano .bashrc*"

```
root@Lovy:~# sudo nano .bashrc
```

and paste these commands:

export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64

export PATH=$PATH:/usr/lib/jvm/java-8-openjdk-amd64/bin

export HADOOP_HOME=~/hadoop-3.3.6/

export PATH=$PATH:$HADOOP_HOME/bin

export

PATH=$PATH:$HADOOP_HOME/sbin

export HADOOP_MAPRED_HOME=$HADOOP_HOME

export YARN_HOME=$HADOOP_HOME

export HADOOP_CONF_DIR=$HADOOP_HOME/etc/hadoop

export

HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native

export HADOOP_OPTS="-

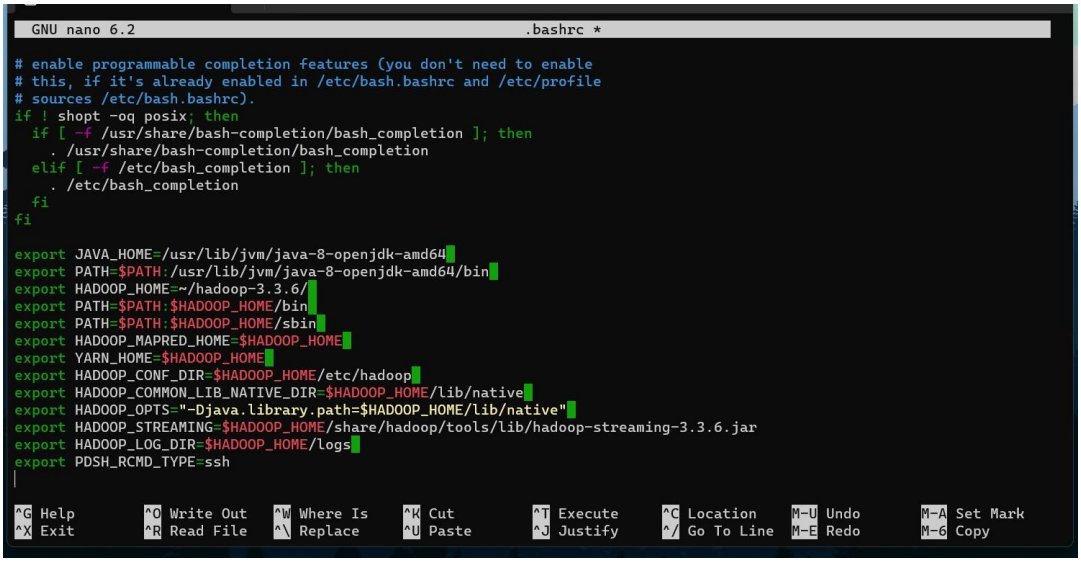
Djava.library.path=$HADOOP_HOME/lib/native"

export
HADOOP_STREAMING=$HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-3.3.6.jar

export

HADOOP_LOG_DIR=$HADOOP_HOME/logs

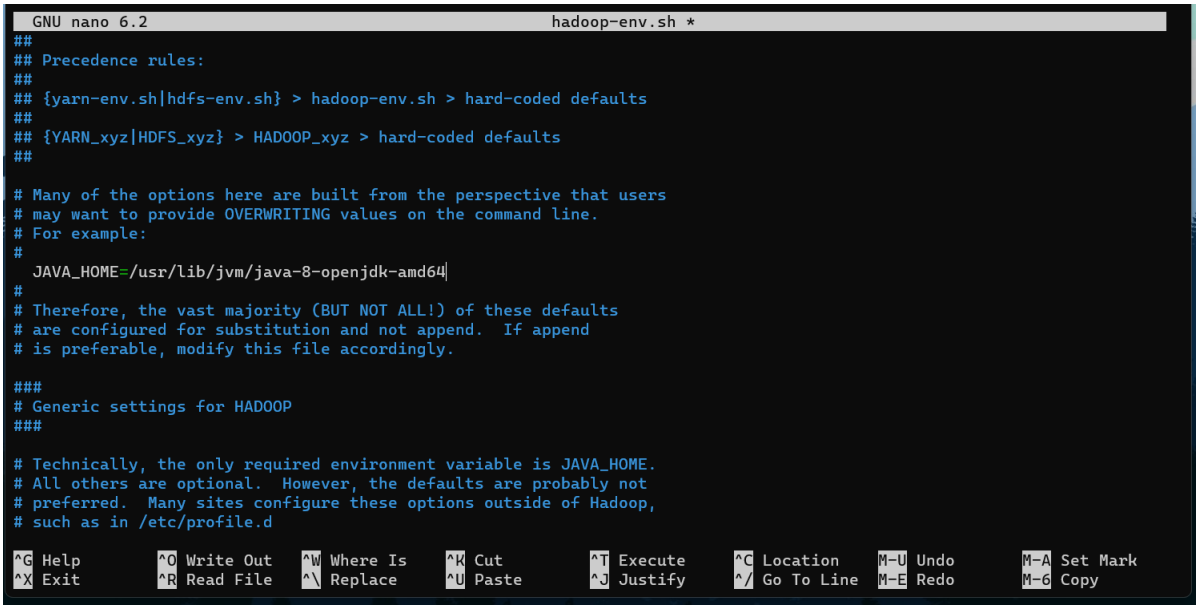
export PDSH_RCMD_TYPE=ssh



Press ctrl + O to write out and then ctrl + x to save the file

7. Go to hadoop-3.3.6/etc/hadoop

8. Set the JAVA_HOME on Hadoop Environment by opening the file hadoop-env.sh using thecommand "*sudo nano hadoop-env.sh*" add the path of JAVA_HOME and save the file as follows



Observations:

1. The installation and configuration processes may take some time depending on the system specifications and internet speed.
2. Any errors or warnings during the installation process should be carefully noted for troubleshooting.

Results:

Hadoop has been successfully installed in stand-alone mode on the Ubuntu operating system.

Conclusion:

The successful installation of the latest version of Hadoop in stand-alone mode on Ubuntu provides users with a platform to explore the core functionalities of Hadoop and gain insights into distributed computing and data processing.

Practical 2: Setting up a Single Node Cluster and understanding itsfunctionalities.

Aim: Setting up a Single Node Cluster. Hadoop Installation: Psuedo Distributed Mode(Locally and YARN)

Objectives:

1. Configuring Hadoop in Pseudo-Distributed Mode.
2. Understanding the setup of a Single Node Cluster.
3. Exploring the functionalities of the Hadoop ecosystem in a pseudo-distributed environment.

Requirements:

1. Ubuntu operating system with Hadoop installed.
2. Basic understanding of Hadoop configuration files and settings.

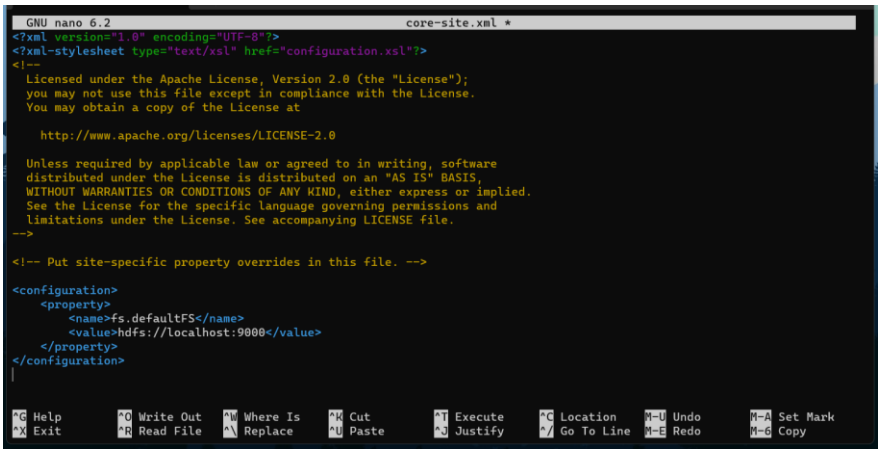
Procedure:

1. Go to hadoop-3.3.6/etc/hadoop

2. Open the file core-site.xml using the command "*sudo nano core-site.xml*"

3. Edit the `core-site.xml` file to define the Hadoop core parameters, which are:

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



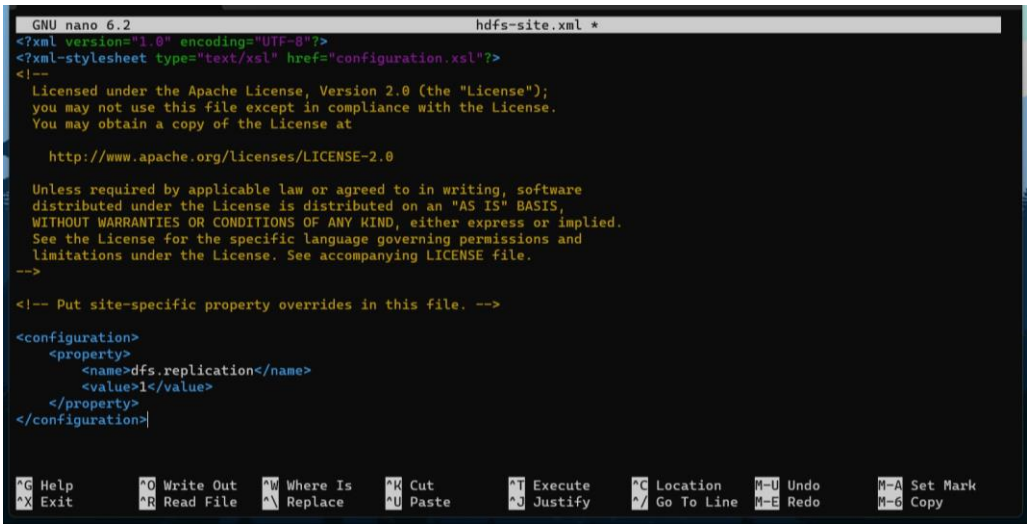
Save the file.

4. Open the file hdfs-site.xml using the command "*sudo nano hdfs-site.xml*"
5. Modify the `hdfs-site.xml` file to configure HDFS settings, which are:

```
<configuration>
  <property>
    <name>dfs.replication</name>
    <value>1</value>
```

</property>

</configuration>



Save the file.

6. Open the file mapred-site.xml using the command "*sudo nano mapred-site.xml*"

7. Configure the parameters of mapred-site.xml as follows:

<configuration>

<property>

<name>mapreduce.framework.name</name>

<value>yarn</value>

</property>

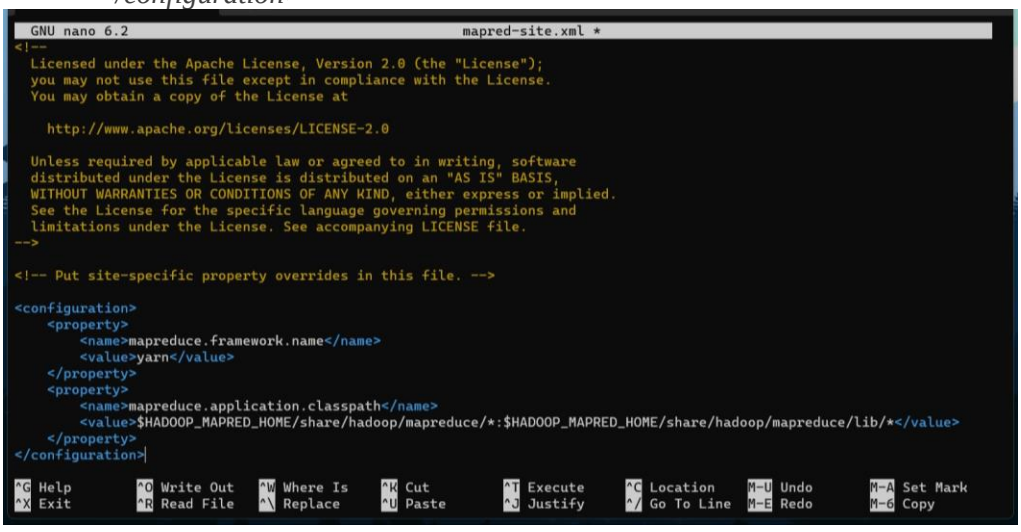
<property>

<name>mapreduce.application.classpath</name>

<value>$HADOOP_MAPRED_HOME/share/hadoop/mapreduce/*:$HADOOP_MAPRED_HOME/share/hadoop/mapreduce/lib/</value>*

</property>

</configuration>

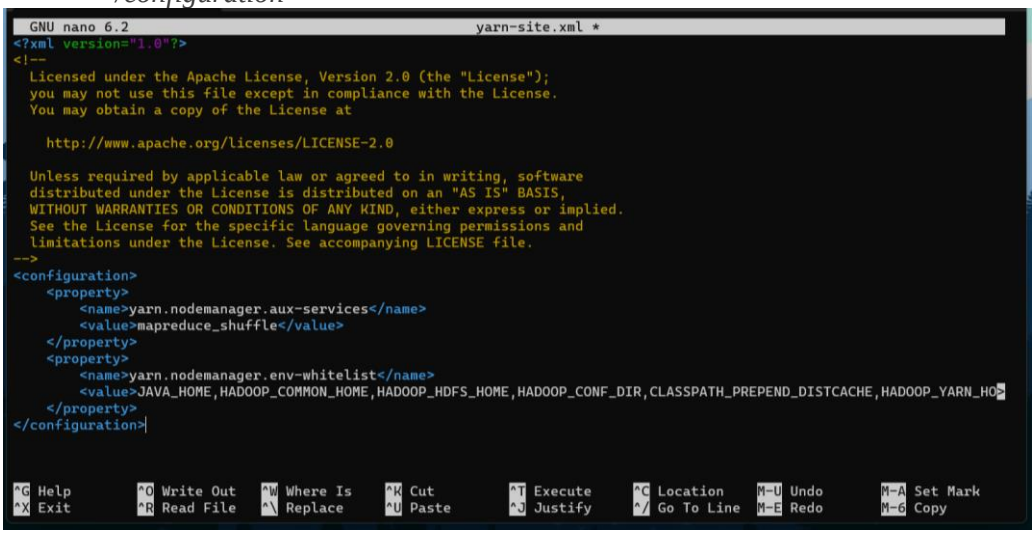


Save the file.

8. Open the file yarn-site.xml using the command "*sudo nano yarn-site.xml*"

9. Configure the `yarn-site.xml` file to set up the YARN settings. Configurations are:

```
<configuration>
  <property>
    <name>yarn.nodemanager.aux-services</name>
    <value>mapreduce_shuffle</value>
  </property>
  <property>
    <name>yarn.nodemanager.env-whitelist</name>

<value>JAVA_HOME,HADOOP_COMMON_HOME,HADOOP_HDFS_HOME,HA
DOOP_
CONF_DIR,CLASSPATH_PREPEND_DISTCACHE,HADOOP_YARN_HOME,HA
DOOP_HO ME,PATH,LANG,TZ,HADOOP_MAPRED_HOME</value>
  </property>
</configuration>
```



Save the file.

10. Now check that you can ssh to the localhost without a passphrase, using the command "*ssh localhost*"

11. Execute the following commands:

```
ssh-keygen -t rsa -P '' -f ~/.ssh/id_rsa
cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
chmod 0600 ~/.ssh/authorized_keys
```

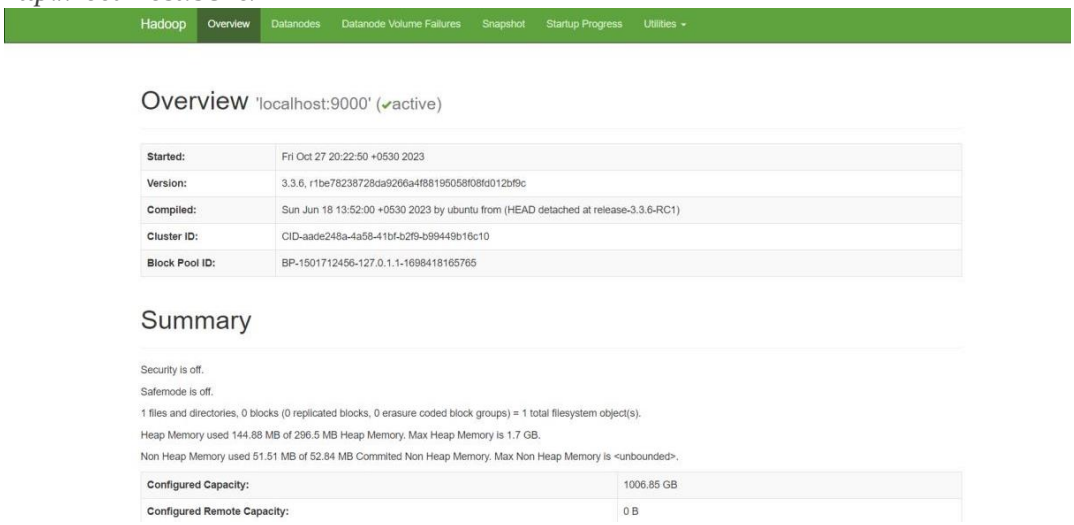
```
root@Lovy:~# ssh-keygen -t rsa -P '' -f ~/.ssh/id_rsa
Generating public/private rsa key pair.
Created directory '/root/.ssh'.
Your identification has been saved in /root/.ssh/id_rsa
Your public key has been saved in /root/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:gCliV+xGC9wDFYQzCHtWCf4i6XGWE/460ly4KG/hC1o root@Lovy
The key's randomart image is:
+---[RSA 3072]----+
|o +oO*.          |
| + BoB           |
|o.=oO +          |
|.=ooo+ .         |
|o..*+   S        |
|..=oo.           |
|.oE.o.           |
|o=o=.            |
|oo+o.            |
+----[SHA256]-----+
root@Lovy:~#
```

12. Format the filesystem using the command "*hdfs namenode -format*"

13. Start NameNode daemon and DataNode daemon using the command "*start-dfs.sh*"

```
hadoop@Lovy:~$ start-dfs.sh
Starting namenodes on [0.0.0.0]
Starting datanodes
Starting secondary namenodes [Lovy]
```

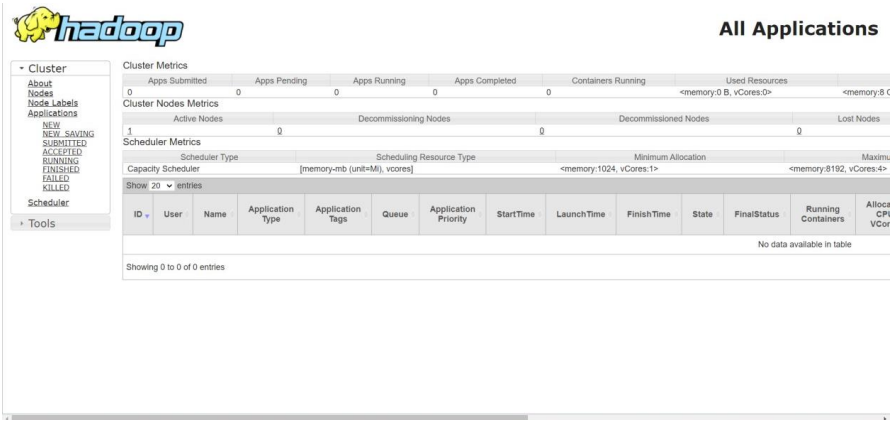
14. Browse the web interface for the NameNode; by default it is available at: NameNode - *http://localhost:9870/*



15. Start ResourceManager daemon and NodeManager daemon using the command "*start-yarn.sh*"

```
hadoop@Lovy:~$ start-yarn.sh
Starting resourcemanager
Starting nodemanagers
```

16. Browse the web interface for the ResourceManager; by default it is available at: *http://localhost:8088/*



Observations:

Pay attention to the log messages and error reports during the configuration and startup process.

Results:

The Single Node Cluster has been successfully set up in Pseudo-Distributed Mode.

Conclusion:

The successful setup of the Single Node Cluster in Pseudo-Distributed Mode facilitates the understanding of Hadoop's distributed architecture and operations, serving as a foundation for further exploration and learning in the field of Big Data processing and analytics.

Practical 3: Implementing file management tasks in Hadoop.

Aim: File Management tasks in Hadoop. Creation of folder, deletion of folder, put afile from local drive to hdfs, show the content of hdfs file, download the hdfs file tolocal drive, delete hdfs file.

Objectives:

1. Creating folders in HDFS.
2. Deleting folders from HDFS.
3. Uploading files from the local drive to HDFS.
4. Displaying the contents of an HDFS file.
5. Downloading an HDFS file to the local drive.
6. Deleting an HDFS file.

Requirements:

1. Hadoop installed and configured in Pseudo-Distributed or Standalone mode.
2. A sample file for uploading to HDFS.

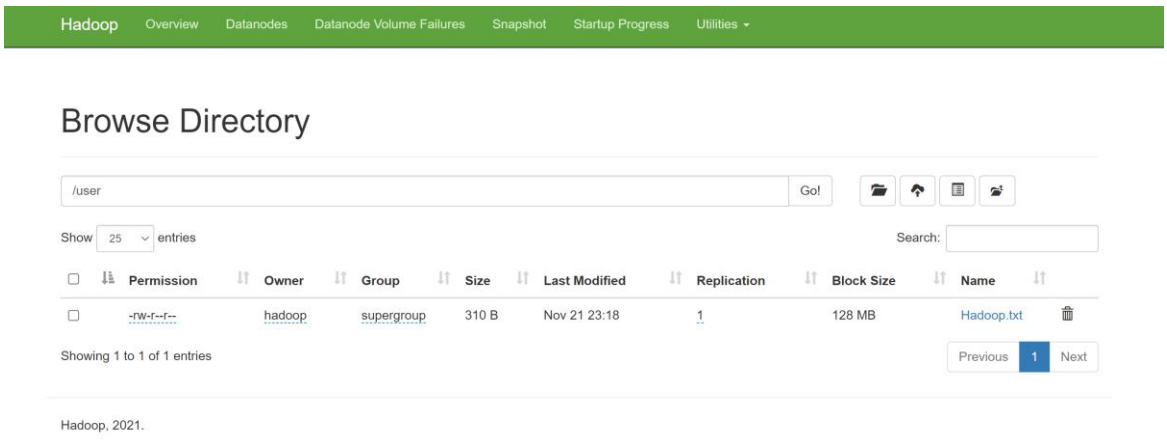
Procedure:

1. Use the "*hadoop fs -mkdir*" command to create a folder in HDFS.

```
hadoop@Lovy:~$ hadoop fs -mkdir /user
hadoop@Lovy:~$ hdfs dfs -ls /
Found 1 items
drwxr-xr-x   - hadoop supergroup          0 2023-11-21 22:59 /user
```

2. Upload a file from the local drive to HDFS using the "*hdfs dfs -put*" command.

```
hadoop@Lovy:~$ cd /mnt/c/Users/prath/BigData
hadoop@Lovy:/mnt/c/Users/prath/BigData$ hdfs dfs -put Hadoop.txt /user/Hadoop.txt
hadoop@Lovy:/mnt/c/Users/prath/BigData$ hdfs dfs -ls /user
Found 1 items
-rw-r--r--   1 hadoop supergroup        310 2023-11-21 23:18 /user/Hadoop.txt
```



3. Display the contents of an HDFS file using the "*hdfs dfs -cat*" command.

```
hadoop@Lovy:~$ hdfs dfs -cat /user/Hadoop.txt
Hadoop, an open-source framework, facilitates distributed storage and processing of vast data sets across clusters of co
mputers. Its scalability and fault tolerance make it pivotal for big data analytics, utilizing MapReduce for computation
 and HDFS for storage, revolutionizing data management and analysis.
```

4. Download an HDFS file to the local drive using the "*hdfs dfs -get*" command.

```
hadoop@Lovy:~$ hdfs dfs -get /user/Hadoop.txt Hadoop1.txt
```

5. Delete an HDFS file using the "*hdfs dfs -rm*" command.

```
hadoop@Lovy:~$ hdfs dfs -rm /user/Hadoop.txt
Deleted /user/Hadoop.txt
```

6. Use the "*hdfs dfs -rm -r*" command to delete a folder from HDFS.

```
hadoop@Lovy:~$ hdfs dfs -ls /
Found 1 items
drwxr-xr-x   - hadoop supergroup          0 2023-11-22 00:23 /user
hadoop@Lovy:~$ hdfs dfs -rm -r /user
Deleted /user
```

Observations:

1. Note any errors or warnings during the file management tasks.
2. Monitor the file system changes in the Hadoop user interface.

Results:

1. Folders have been created and deleted successfully in HDFS.
2. Files have been uploaded, downloaded, and deleted from HDFS.

Conclusion:

The successful execution of file management tasks in Hadoop demonstrates the capabilities of HDFS in handling data storage, retrieval, and deletion, showcasing its role in managing Big Data effectively.

Practical 4: Writing a Word count MapReduce program in Java.

Aim: To implement a Word Count MapReduce program using Java.

Objectives:

1. Understanding the basics of MapReduce programming paradigm.
2. Writing Mapper and Reducer functions for word counting.
3. Compiling and executing the Java program on the Hadoop cluster.

Requirements:

1. Hadoop installed and configured in Pseudo-Distributed or Standalone mode.
2. Basic understanding of Java programming.
3. Text input files for testing the Word Count program.

Procedure:

1. Write a Java file named WordCount.java containing the provided source code.

```
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);

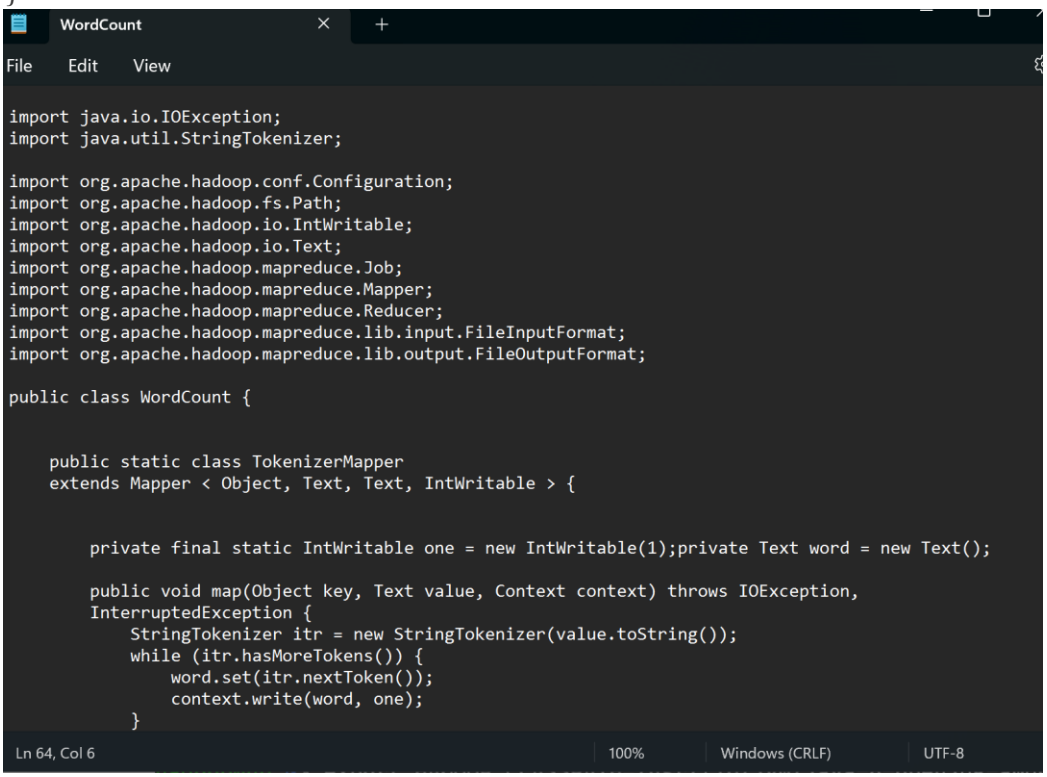
FileInputFormat.addInputPath(job, new Path(args[0]));

FileOutputFormat.setOutputPath(job, new

Path(args[1]));System.exit(job.waitForCompletion(true)

? 0 : 1);

}



2. Set the JAVA_HOME variable, using the command "*export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64*"

```
hadoop@Lovy:~$ export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
```

3. Set the HADOOP_CLASSPATH variable, using the command "*export HADOOP_CLASSPATH=/usr/lib/jvm/java-8-openjdk-amd64/lib/tools.jar*"

```
hadoop@Lovy:~$ export HADOOP_CLASSPATH=/usr/lib/jvm/java-8-openjdk-amd64/lib/tools.jar
```

4. Compile the WordCount.java file and create a JAR file using the provided commands

hadoop com.sun.tools.javac.Main WordCount.java

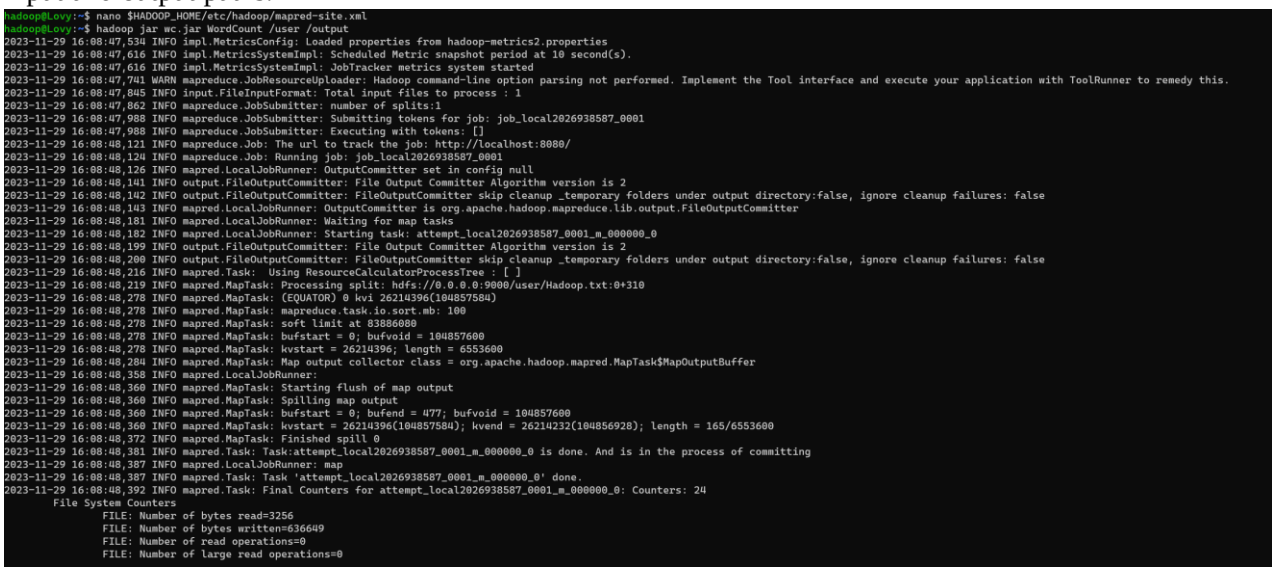
jar cf wc.jar WordCount.class*

```
hadoop@Lovy:~$ touch WordCount.java
hadoop@Lovy:~$ nano WordCount.java
hadoop@Lovy:~$ hadoop com.sun.tools.javac.Main WordCount.java
```

5. Prepare sample text files as input and store them in the Hadoop file system *(Refer Practical 3)*.

```
hadoop@Lovy:~$ ls
 Hadoop.txt   'WordCount$IntSumReducer.class'     WordCount.class   hdfs      id_rsa.pub
 Hadoop1.txt  'WordCount$TokenizerMapper.class'   WordCount.java    id_rsa
hadoop@Lovy:~$ jar cf wc.jar WordCount*.class
hadoop@Lovy:~$ ls
 Hadoop.txt   'WordCount$IntSumReducer.class'     WordCount.class   hdfs      id_rsa.pub
 Hadoop1.txt  'WordCount$TokenizerMapper.class'   WordCount.java    id_rsa    wc.jar
```

6. Execute the MapReduce application using the "*hadoop jar*" command with appropriate input and output paths.



```
hadoop@Lovy:~$ nano $HADOOP_HOME/etc/hadoop/mapred-site.xml
hadoop@Lovy:~$ hadoop jar wc.jar WordCount /user /output
2023-11-29 16:08:47,534 INFO impl.MetricsConfig: Loaded properties from hadoop-metrics2.properties
2023-11-29 16:08:47,616 INFO impl.MetricsSystemImpl: Scheduled Metric snapshot period at 10 second(s).
2023-11-29 16:08:47,616 INFO impl.MetricsSystemImpl: JobTracker metrics system started
2023-11-29 16:08:47,741 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application with ToolRunner to remedy this.
2023-11-29 16:08:47,845 INFO input.FileInputFormat: Total input files to process : 1
2023-11-29 16:08:47,862 INFO mapreduce.JobSubmitter: number of splits:1
2023-11-29 16:08:47,988 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_local2026938587_0001
2023-11-29 16:08:47,988 INFO mapreduce.JobSubmitter: Executing with tokens: []
2023-11-29 16:08:48,121 INFO mapreduce.Job: The url to track the job: http://localhost:8080/
2023-11-29 16:08:48,124 INFO mapreduce.Job: Running job: job_local2026938587_0001
2023-11-29 16:08:48,126 INFO mapred.LocalJobRunner: OutputCommitter set in config null
2023-11-29 16:08:48,141 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 2
2023-11-29 16:08:48,142 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders under output directory:false, ignore cleanup failures: false
2023-11-29 16:08:48,143 INFO mapred.LocalJobRunner: OutputCommitter is org.apache.hadoop.mapreduce.lib.output.FileOutputCommitter
2023-11-29 16:08:48,181 INFO mapred.LocalJobRunner: Waiting for map tasks
2023-11-29 16:08:48,182 INFO mapred.LocalJobRunner: Starting task: attempt_local2026938587_0001_m_000000_0
2023-11-29 16:08:48,199 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 2
2023-11-29 16:08:48,200 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders under output directory:false, ignore cleanup failures: false
2023-11-29 16:08:48,216 INFO mapred.Task:  Using ResourceCalculatorProcessTree : [ ]
2023-11-29 16:08:48,219 INFO mapred.MapTask: Processing split: hdfs://0.0.0.0:9000/user/Hadoop.txt:0+310
2023-11-29 16:08:48,278 INFO mapred.MapTask: (EQUATOR) 0 kvi 26214396(104857584)
2023-11-29 16:08:48,278 INFO mapred.MapTask: mapreduce.task.io.sort.mb: 100
2023-11-29 16:08:48,278 INFO mapred.MapTask: soft limit at 83886080
2023-11-29 16:08:48,278 INFO mapred.MapTask: bufstart = 0; bufvoid = 104857600
2023-11-29 16:08:48,278 INFO mapred.MapTask: kvstart = 26214396; length = 6553600
2023-11-29 16:08:48,284 INFO mapred.MapTask: Map output collector class = org.apache.hadoop.mapred.MapTask$MapOutputBuffer
2023-11-29 16:08:48,358 INFO mapred.LocalJobRunner:
2023-11-29 16:08:48,360 INFO mapred.MapTask: Starting flush of map output
2023-11-29 16:08:48,360 INFO mapred.MapTask: Spilling map output
2023-11-29 16:08:48,360 INFO mapred.MapTask: bufstart = 0; bufend = 477; bufvoid = 104857600
2023-11-29 16:08:48,360 INFO mapred.MapTask: kvstart = 26214396(104857584); kvend = 26214232(104856928); length = 165/6553600
2023-11-29 16:08:48,372 INFO mapred.MapTask: Finished spill 0
2023-11-29 16:08:48,381 INFO mapred.Task: Task:attempt_local2026938587_0001_m_000000_0 is done. And is in the process of committing
2023-11-29 16:08:48,387 INFO mapred.LocalJobRunner: map
2023-11-29 16:08:48,387 INFO mapred.Task: Task 'attempt_local2026938587_0001_m_000000_0' done.
2023-11-29 16:08:48,392 INFO mapred.Task: Final Counters for attempt_local2026938587_0001_m_000000_0: Counters: 24
        File System Counters
                FILE: Number of bytes read=3256
                FILE: Number of bytes written=636649
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
```

7. Check the output using the "*hadoop fs -cat*" command to verify the word count results.

```
hadoop@Lovy:~$ cd java-output/
hadoop@Lovy:~/java-output$ ls
_SUCCESS  part-r-00000
hadoop@Lovy:~/java-output$ cat part-r-00000
HDFS    1
Hadoop, 1
Its     1
MapReduce       1
across  1
an      1
analysis.       1
analytics,      1
and     4
big     1
clusters        1
computation     1
computers.      1
data    3
distributed     1
facilitates     1
fault   1
for     3
framework,      1
it      1
make    1
management      1
of      2
open-source     1
pivotal 1
processing      1
revolutionizing 1
scalability     1
sets    1
storage 1
storage,        1
tolerance       1
utilizing       1
vast    1
```

Observations:

1. Observe the execution logs and any error messages for debugging purposes.
2. Compare the output with the expected results for accuracy verification.

Results:

1. The Word Count program has successfully counted the occurrences of each word in the provided input text files.
2. The output displays the word counts for each unique word in the text.

Conclusion:

The successful implementation of the Word Count MapReduce program in Java underscores the potential of Hadoop's distributed processing capabilities in handling data-intensive tasks, laying the groundwork for exploring more complex data processing operations in the Hadoop ecosystem.

Practical 5: Writing a Word count MapReduce program in Python.

Aim: To implement a Word Count MapReduce program using Python.

Objectives:

1. Implementing the concept of Hadoop Steaming
2. Understanding the fundamentals of the MapReduce programming model.
3. Implementing the Mapper and Reducer functions in Python for counting words.
4. Executing the Python program on the Hadoop cluster.

Requirements:

1. Hadoop installed and configured in Pseudo-Distributed or Standalone mode.
2. Basic understanding of Python programming.

Procedure:

1. Create a file with the name word_count_data.txt and add some data to it.

```
hadoop@Lovy:/mnt/c/Users/prath/Downloads$ touch word_count_data.txt
hadoop@Lovy:/mnt/c/Users/prath/Downloads$ nano word_count_data.txt
hadoop@Lovy:/mnt/c/Users/prath/Downloads$ cat word_count_data.tx
cat: word_count_data.tx: No such file or directory
hadoop@Lovy:/mnt/c/Users/prath/Downloads$ cat word_count_data.txt
Hadoop is an open-source framework designed for distributed storage and processing of large
-scale data across clusters of computers using simple programming models. It comprises two
main components: the Hadoop Distributed File System (HDFS) for storage and the MapReduce pr
ogramming model for processing. HDFS breaks data into chunks and distributes them across no
des in a cluster, ensuring reliability and fault tolerance. MapReduce processes data by div
iding tasks into smaller sub-tasks, assigning them to nodes, and consolidating the results.
 Hadoop's scalability allows for handling massive datasets and enables parallel processing,
 speeding up computations significantly. Beyond MapReduce, Hadoop's ecosystem includes vari
ous tools like Hive for data warehousing, Pig for data analysis, and Spark for in-memory pr
ocessing, providing a robust ecosystem for diverse data operations. Its flexibility and abi
lity to work with different data types make it a cornerstone in big data analytics and stor
age solutions.
```

2. Create a mapper.py file that implements the mapper logic. It will read the data from STDIN and will split the lines into words, and will generate an output of each word with its individual count. Copy the below code to the mapper.py file

```
#!/usr/bin/env python


# import sys because we need to read and write data to STDIN and

STDOUTimport sys


# reading entire line from STDIN (standard

input)for line in sys.stdin:

    # to remove leading and trailing whitespace

    line = line.strip()

    # split the line into words

    words = line.split()


    # we are looping over the words array and printing the word

    # with the count of 1 to the STDOUT

    for word in words:

        # write the results to STDOUT (standard
```

output);# what we output here will be the input

for the

 # Reduce step, i.e. the input for reducer.py

print ('%s\t%s' % (word, 1))

```
hadoop@Lovy:/mnt/c/Users/prath/Downloads$ touch mapper.py
hadoop@Lovy:/mnt/c/Users/prath/Downloads$ nano mapper.py
hadoop@Lovy:/mnt/c/Users/prath/Downloads$ cat mapper.py
#!/usr/bin/env python

# import sys because we need to read and write data to STDIN and STDOUT
import sys

# reading entire line from STDIN (standard input)
for line in sys.stdin:
        # to remove leading and trailing whitespace
        line = line.strip()
        # split the line into words
        words = line.split()

        # we are looping over the words array and printing the word
        # with the count of 1 to the STDOUT
        for word in words:
                # write the results to STDOUT (standard output);
                # what we output here will be the input for the
                # Reduce step, i.e. the input for reducer.py
                print '%s\t%s' % (word, 1)
```

3. Create a reducer.py file that implements the reducer logic. It will read the output of mapper.py from STDIN(standard input) and will aggregate the occurrence of each word and will write the final output to STDOUT. Copy the below code to the reducer.py file

#!/usr/bin/env python

from operator import itemgetter

import sys

current_word =

Nonecurrent_count

= 0 word = None

 # input comes from

STDINfor line in

sys.stdin:

line = line.strip()

```
    word, count = line.split('\t', 1)

    try:
        count =
    int(count)except
    ValueError:
        continue

    if current_word ==
        word: current_count
        += count
    else:
        if current_word:
            # write result to STDOUT
            print('%s\t%s' % (current_word,
        current_count))current_count = count
        current_word = word

# do not forget to output the last word if needed!
if current_word == word:
    print('%s\t%s' % (current_word, current_count))
```

```
hadoop@Lovy:/mnt/c/Users/prath/Downloads$ touch reducer.py
hadoop@Lovy:/mnt/c/Users/prath/Downloads$ nano reducer.py
hadoop@Lovy:/mnt/c/Users/prath/Downloads$ cat reducer.py
#!/usr/bin/env python

from operator import itemgetter
import sys

current_word = None
current_count = 0
word = None

# read the entire line from STDIN
for line in sys.stdin:
        # remove leading and trailing whitespace
        line = line.strip()
        # splitting the data on the basis of tab we have provided in mapper.py
        word, count = line.split('\t', 1)
        # convert count (currently a string) to int
        try:
                count = int(count)
        except ValueError:
                # count was not a number, so silently
                # ignore/discard this line
                continue

        # this IF-switch only works because Hadoop sorts map output
        # by key (here: word) before it is passed to the reducer
        if current_word == word:
                current_count += count
        else:
                if current_word:
                        # write result to STDOUT
                        print '%s\t%s' % (current_word, current_count)
                current_count = count
                current_word = word

# do not forget to output the last word if needed!
if current_word == word:
        print '%s\t%s' % (current_word, current_count)
```

4. Now make a directory word_count_in_python in our HDFS in the root directory that will store our word_count_data.txt file with the command "*hadoop fs -mkdir /word_count_in_python*"

```
hadoop@Lovy:/mnt/c/Users/prath/Downloads$ hadoop fs -mkdir /word_count_in_python
```

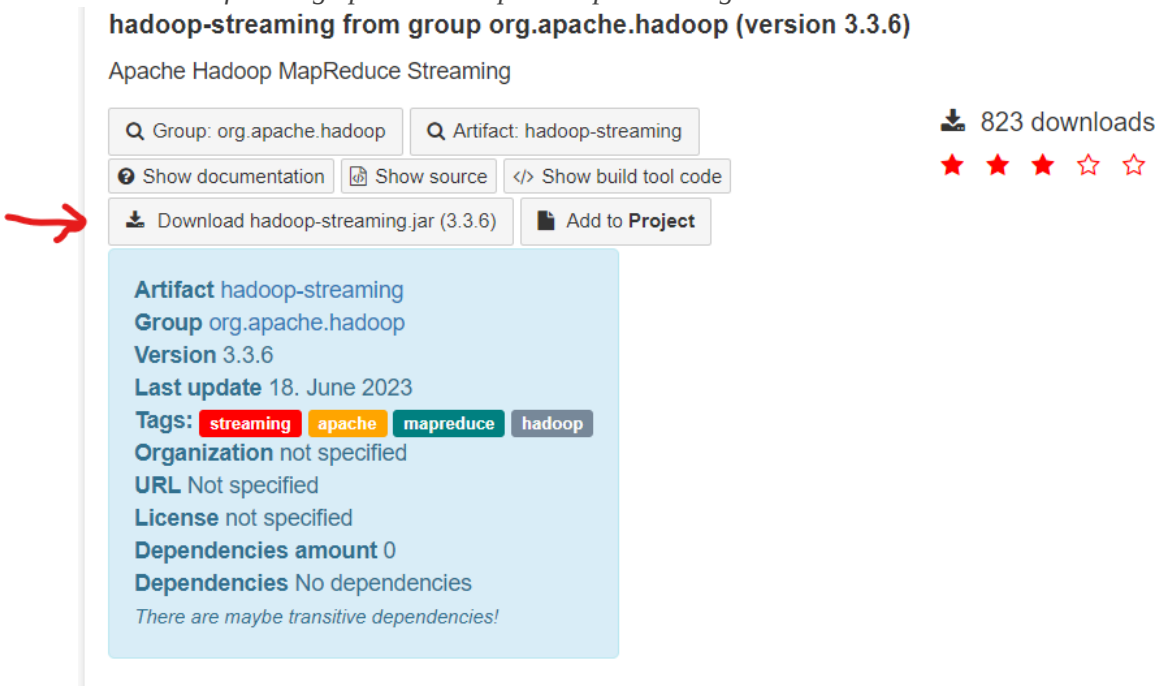
5. Copy word_count_data.txt to this folder in our HDFS

```
hadoop@Lovy:/mnt/c/Users/prath/Downloads$ hadoop dfs -put word_count_data.txt /word_co
unt_in_python/word_count_data.txt
```

6. Give executable permission to the mapper.py and reducer.py with the help of the command "*chmod 777 mapper.py reducer.py*".

```
hadoop@Lovy:/mnt/c/Users/prath/Downloads$ sudo chmod 777 mapper.py reducer.py
[sudo] password for hadoop:
```

7. Now download the latest hadoop-streaming jar file from "*https://jar-download.com/artifacts/org.apache.hadoop/hadoop-streaming/3.3.6*".




```
hadoop@Lovy:~$ sudo mv mapper.py /home/hadoop/mapreduce_python/
[sudo] password for hadoop:
mv: cannot stat 'mapper.py': No such file or directory
hadoop@Lovy:~$ sudo mv /mnt/f/mapper.py /home/hadoop/mapreduce_python/
hadoop@Lovy:~$ sudo mv /mnt/f/reducer.py /home/hadoop/mapreduce_python/
hadoop@Lovy:~$ sudo mv /mnt/f/hadoop-streaming-3.3.6.jar.py /home/hadoop/mapreduce_python/
mv: cannot stat '/mnt/f/hadoop-streaming-3.3.6.jar.py': No such file or directory
hadoop@Lovy:~$ sudo mv /mnt/f/hadoop-streaming-3.3.6.jar /home/hadoop/mapreduce_python/
hadoop@Lovy:~$ ls
 Hadoop.txt   'WordCount$IntSumReducer.class'     WordCount.class   hdfs     id_rsa.pub    mapreduce_python   word_count_d
ata.txt
 Hadoop1.txt  'WordCount$TokenizerMapper.class'   WordCount.java    id_rsa   java-output   wc.jar
hadoop@Lovy:~$ cd mapreduce_python/
hadoop@Lovy:~/mapreduce_python$ ls
hadoop-streaming-3.3.6.jar  mapper.py  reducer.py
hadoop@Lovy:~/mapreduce_python$ nano mapper.py
hadoop@Lovy:~/mapreduce_python$ nano reducer.py
hadoop@Lovy:~/mapreduce_python$ nano data.txt
```

8. Run the python files with the help of the Hadoop streaming utility as shown below.

*hadoop jar hadoop-streaming-3.3.6.jar *

*-files mapper.py,reducer.py *

*-mapper mapper.py *

*-reducer reducer.py *

*-input /user/shivan/word_count_data.txt *

-output /user/shivan/word_count_output

```
hadoop@Lovy:~/mapreduce_python$ hadoop jar hadoop-streaming-3.3.6.jar -input /wordpython/input/data.txt -output /wordpython/output -mapper mapper.py -reducer reducer.py
2023-12-06 12:50:56,976 INFO impl.MetricsConfig: Loaded properties from hadoop-metrics2.properties
2023-12-06 12:50:57,062 INFO impl.MetricsSystemImpl: Scheduled Metric snapshot period at 10 second(s).
2023-12-06 12:50:57,062 INFO impl.MetricsSystemImpl: JobTracker metrics system started
2023-12-06 12:50:57,073 WARN impl.MetricsSystemImpl: JobTracker metrics system already initialized!
2023-12-06 12:50:57,258 INFO mapred.FileInputFormat: Total input files to process : 1
2023-12-06 12:50:57,301 INFO mapreduce.JobSubmitter: number of splits:1
2023-12-06 12:50:57,421 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_local430809991_0001
2023-12-06 12:50:57,421 INFO mapreduce.JobSubmitter: Executing with tokens: []
2023-12-06 12:50:57,522 INFO mapreduce.Job: The url to track the job: http://localhost:8080/
2023-12-06 12:50:57,524 INFO mapreduce.Job: Running job: job_local430809991_0001
2023-12-06 12:50:57,524 INFO mapred.LocalJobRunner: OutputCommitter set in config null
2023-12-06 12:50:57,530 INFO mapred.LocalJobRunner: OutputCommitter is org.apache.hadoop.mapred.FileOutputCommitter
2023-12-06 12:50:57,536 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 2
2023-12-06 12:50:57,536 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders under output directory:false, ignore cleanup failures: false
2023-12-06 12:50:57,577 INFO mapred.LocalJobRunner: Waiting for map tasks
2023-12-06 12:50:57,580 INFO mapred.LocalJobRunner: Starting task: attempt_local430809991_0001_m_000000_0
2023-12-06 12:50:57,598 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 2
2023-12-06 12:50:57,599 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders under output directory:false, ignore cleanup failures: false
2023-12-06 12:50:57,616 INFO mapred.Task:  Using ResourceCalculatorProcessTree : [ ]
2023-12-06 12:50:57,623 INFO mapred.MapTask: Processing split: hdfs://0.0.0.0:9000/wordpython/input/data.txt:0+492
2023-12-06 12:50:57,642 INFO mapred.MapTask: numReduceTasks: 1
2023-12-06 12:50:57,675 INFO mapred.MapTask: (EQUATOR) 0 kvi 26214396(104857584)
2023-12-06 12:50:57,675 INFO mapred.MapTask: mapreduce.task.io.sort.mb: 100
2023-12-06 12:50:57,675 INFO mapred.MapTask: soft limit at 83886080
2023-12-06 12:50:57,675 INFO mapred.MapTask: bufstart = 0; bufvoid = 104857600
2023-12-06 12:50:57,675 INFO mapred.MapTask: kvstart = 26214396; length = 6553600
2023-12-06 12:50:57,683 INFO mapred.MapTask: Map output collector class = org.apache.hadoop.mapred.MapTask$MapOutputBuffer
2023-12-06 12:50:57,692 INFO streaming.PipeMapRed: PipeMapRed exec [/home/hadoop/mapreduce_python/./mapper.py]
2023-12-06 12:50:57,695 INFO Configuration.deprecation: mapred.work.output.dir is deprecated. Instead, use mapreduce.task.output.dir
2023-12-06 12:50:57,696 INFO Configuration.deprecation: mapred.local.dir is deprecated. Instead, use mapreduce.cluster.local.dir
2023-12-06 12:50:57,697 INFO Configuration.deprecation: map.input.file is deprecated. Instead, use mapreduce.map.input.file
2023-12-06 12:50:57,697 INFO Configuration.deprecation: map.input.length is deprecated. Instead, use mapreduce.map.input.length
2023-12-06 12:50:57,697 INFO Configuration.deprecation: mapred.job.id is deprecated. Instead, use mapreduce.job.id
2023-12-06 12:50:57,697 INFO Configuration.deprecation: mapred.task.partition is deprecated. Instead, use mapreduce.task.partition
2023-12-06 12:50:57,698 INFO Configuration.deprecation: map.input.start is deprecated. Instead, use mapreduce.map.input.start
2023-12-06 12:50:57,698 INFO Configuration.deprecation: mapred.task.is.map is deprecated. Instead, use mapreduce.task.ismap
2023-12-06 12:50:57,698 INFO Configuration.deprecation: mapred.task.id is deprecated. Instead, use mapreduce.task.attempt.id
2023-12-06 12:50:57,698 INFO Configuration.deprecation: mapred.tip.id is deprecated. Instead, use mapreduce.task.id
2023-12-06 12:50:57,699 INFO Configuration.deprecation: mapred.skip.on is deprecated. Instead, use mapreduce.job.skiprecords
2023-12-06 12:50:57,699 INFO Configuration.deprecation: user.name is deprecated. Instead, use mapreduce.job.user.name
2023-12-06 12:50:57,769 INFO streaming.PipeMapRed: R/W/S=1/0/0 in:NA [rec/s] out:NA [rec/s]
2023-12-06 12:50:57,770 INFO streaming.PipeMapRed: Records R/W=1/1
2023-12-06 12:50:57,772 INFO streaming.PipeMapRed: MRErrorThread done
2023-12-06 12:50:57,773 INFO streaming.PipeMapRed: mapRedFinished
2023-12-06 12:50:57,776 INFO mapred.LocalJobRunner:
2023-12-06 12:50:57,776 INFO mapred.MapTask: Starting flush of map output
2023-12-06 12:50:57,776 INFO mapred.MapTask: Spilling map output
2023-12-06 12:50:57,776 INFO mapred.MapTask: bufstart = 0; bufend = 664; bufvoid = 104857600
```

9. Check the output using the "*hadoop fs -cat*" command to verify the word count results.

```
hadoop@Lovy:~/mapreduce_python$ hadoop fs -cat /wordpython/output/part-00000
'Content        1
'lorem  1
English.        1
Ipsum   2
It      1
Lorem   2
Many    1
The     1
a       5
and     2
as      2
at      1
be      1
by      1
content 2
default 1
desktop 1
distracted      1
distribution    1
editors 1
established     1
fact    1
for     1
has     1
here',  1
here,   1
in      1
infancy.        1
ipsum'  1
is      2
it      2
its     1
layout. 1
letters,        1
like    1
long    1
look    1
looking 1
making  1
many    1
model   1
more-or-less    1
normal  1
now     1
of      3
opposed 1
packages        1
page    2
point   1
publishing      1
readable        2
reader  1
search  1
sites   1
```

Observations:

1. Monitor the execution logs and error messages to facilitate debugging, if necessary.
2. Compare the output generated by the Python program with the expected results.

Results:

1. The Word Count MapReduce program in Python successfully counts the occurrences of each word in the provided input text files.
2. The output demonstrates the word counts for each unique word in the text.

Conclusion:

The successful implementation of the Word Count MapReduce program in Python highlights the versatility of Hadoop's distributed computing capabilities, paving the way for exploring more intricate data processing tasks within the Hadoop ecosystem.

Practical 6: WAP to create pyspark configuration, session and count number of words present in a given textfile.

In [2]:
```
from pyspark import SparkConf,SparkContext
conf=SparkConf().setAppName('abc').setMaster('local')
sc=SparkContext(conf=conf)
sc.setLogLevel('ERROR')

from pyspark.sql import SparkSession
spark=SparkSession.builder.appName('abc').config('','').getOrCreate()

import numpy as np
v1=np.array([1,2,3,4,5])
print(v1)
v2=[1,2,3,4,5,6]
print(v2)
from pyspark.mllib.linalg import Vectors
v3=Vectors.dense([3,4,5,6])
print(v3)
v4 = Vectors.sparse(3, [0, 2], [1.0, 3.0])
print(v4)
```

```
[1 2 3 4 5]
[1, 2, 3, 4, 5, 6]
[3.0,4.0,5.0,6.0]
(3,[0,2],[1.0,3.0])
```

In [3]:
```
df = spark.read.csv('data.csv',header=True,inferSchema=True)
df.show(5,0)
```

```
+-------------+------+
|Time_to_Study|Grades|
+-------------+------+
|1            |1.5   |
|5            |2.7   |
|7            |3.1   |
|3            |2.1   |
|2            |1.8   |
+-------------+------+
only showing top 5 rows
```

In [4]:
```
df.count()
```

Out[4]: 50

In [5]:
```
df.select('Grades').distinct().count()
```

Out[5]: 14

In [6]:
```
df.printSchema()
```

```
root
 |-- Time_to_Study: integer (nullable = true)
 |-- Grades: double (nullable = true)
```

In [7]:
```
df.show()
```

```
+-------------+------+
|Time_to_Study|Grades|
+-------------+------+
|            1|   1.5|
|            5|   2.7|
|            7|   3.1|
|            3|   2.1|
|            2|   1.8|
|            9|   3.9|
|            6|   2.9|
|           12|   4.5|
|           11|   4.3|
|            2|   1.8|
|            4|   2.4|
|            8|   3.5|
|           13|   4.8|
|            9|   3.9|
|           14|   5.0|
|           10|   4.1|
|            6|   2.9|
|           12|   4.5|
|            1|   1.5|
|            4|   2.4|
+-------------+------+
only showing top 20 rows
```

In [8]:
```
df.describe().show()
```

```
+-------+-----------------+------------------+
|summary|    Time_to_Study|            Grades|
+-------+-----------------+------------------+
|  count|               50|                50|
|   mean|             7.12|3.2220000000000004|
| stddev|4.048884956102742|1.1047744252164082|
|    min|                1|               1.5|
|    max|               14|               5.0|
+-------+-----------------+------------------+
```

In [9]:
```
feature_cols = df.columns[:-1]

from pyspark.ml.feature import VectorAssembler
vect_assembler = VectorAssembler(inputCols = feature_cols, outputCol="featu

data_w_features = vect_assembler.transform(df)
```

In [10]:
```
finalized_data = data_w_features.select("features","Grades")
finalized_data.show()
```

```
+--------+------+
|features|Grades|
+--------+------+
|   [1.0]|   1.5|
|   [5.0]|   2.7|
|   [7.0]|   3.1|
|   [3.0]|   2.1|
|   [2.0]|   1.8|
|   [9.0]|   3.9|
|   [6.0]|   2.9|
|  [12.0]|   4.5|
|  [11.0]|   4.3|
|   [2.0]|   1.8|
|   [4.0]|   2.4|
|   [8.0]|   3.5|
|  [13.0]|   4.8|
|   [9.0]|   3.9|
|  [14.0]|   5.0|
|  [10.0]|   4.1|
|   [6.0]|   2.9|
|  [12.0]|   4.5|
|   [1.0]|   1.5|
|   [4.0]|   2.4|
+--------+------+
only showing top 20 rows
```

In [11]:
```
train_dataset, test_dataset = finalized_data.randomSplit([0.7, 0.3])
print(df.count())
print(train_dataset.count())
print(test_dataset.count())
```

```
50
36
14
```

In [12]:
```
from pyspark.ml.regression import LinearRegression
LinReg = LinearRegression(featuresCol="features", labelCol="Grades")
```

In [13]:
```
model = LinReg.fit(train_dataset)
```

In [14]:
```
pred = model.evaluate(test_dataset)
```

In [15]:
```
pred.predictions.show()
```

```
+--------+------+-----------------+
|features|Grades|       prediction|
+--------+------+-----------------+
|   [1.0]|   1.5|1.5487462730338715|
|   [2.0]|   1.8|1.8209980937484715|
|   [4.0]|   2.4|2.3655017351776717|
|   [4.0]|   2.4|2.3655017351776717|
|   [5.0]|   2.7| 2.637753555892272|
|   [6.0]|   2.9|2.9100053766068714|
|   [8.0]|   3.5|3.4545090180360716|
|   [9.0]|   3.9|3.7267608387506717|
|   [9.0]|   3.9|3.7267608387506717|
|  [11.0]|   4.3| 4.271264480179871|
|  [11.0]|   4.3| 4.271264480179871|
|  [12.0]|   4.5|4.5435163008944714|
|  [13.0]|   4.8|4.8157681216090715|
|  [14.0]|   5.0| 5.088019942323672|
+--------+------+-----------------+
```

In [16]:
```
coefficient = model.coefficients
print ("The coefficient of the model is : %a" %coefficient)

intercept = model.intercept
print ("The Intercept of the model is : %f" %intercept)
```

```
The coefficient of the model is : DenseVector([0.2723])
The Intercept of the model is : 1.276494
```

In [17]:
```
from pyspark.ml.evaluation import RegressionEvaluator
evaluation = RegressionEvaluator(labelCol="Grades", predictionCol="predicti

rmse = evaluation.evaluate(pred.predictions, {evaluation.metricName: "rmse"
print("RMSE: %.3f" % rmse)

mse = evaluation.evaluate(pred.predictions, {evaluation.metricName: "mse"})
print("MSE: %.3f" % mse)

mae = evaluation.evaluate(pred.predictions, {evaluation.metricName: "mae"})
print("MAE: %.3f" % mae)

r2 = evaluation.evaluate(pred.predictions, {evaluation.metricName: "r2"})
print("r2: %.3f" %r2)
```

```
RMSE: 0.077
MSE: 0.006
MAE: 0.058
r2: 0.995
```

Practical 7: WAP to implement various regression models and evaluate their performance using pyspark

In [2]:
```
from pyspark import SparkConf,SparkContext
conf=SparkConf().setAppName('abc').setMaster('local') #
sc=SparkContext(conf=conf)
sc.setLogLevel('ERROR')
from pyspark.sql import SparkSession
spark=SparkSession.builder.appName('abc').config('','').getOrCreate()
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [3]:
```
df = spark.read.csv('iris.csv', header=None, inferSchema=True)
df.show(5,0)
```

```
+---+---+---+---+-----------+
|_c0|_c1|_c2|_c3|_c4        |
+---+---+---+---+-----------+
|5.1|3.5|1.4|0.2|Iris-setosa|
|4.9|3.0|1.4|0.2|Iris-setosa|
|4.7|3.2|1.3|0.2|Iris-setosa|
|4.6|3.1|1.5|0.2|Iris-setosa|
|5.0|3.6|1.4|0.2|Iris-setosa|
+---+---+---+---+-----------+
only showing top 5 rows
```

In [4]:
```
df.columns
```

Out[4]:
```
['_c0', '_c1', '_c2', '_c3', '_c4']
```

In [5]:
```
df.count()
```

Out[5]:
```
150
```

In [6]:
```
df.printSchema()
```

```
root
 |-- _c0: double (nullable = true)
 |-- _c1: double (nullable = true)
 |-- _c2: double (nullable = true)
 |-- _c3: double (nullable = true)
 |-- _c4: string (nullable = true)
```

In [7]:
```
from pyspark.ml.feature import VectorAssembler, StandardScaler

assembler = VectorAssembler(inputCols=['_c0','_c1','_c2','_c3'], outputCol="features")
data_df = assembler.transform(df)
data_df.show(5,0)
```

```
+---+---+---+---+-----------+-----------------+
|_c0|_c1|_c2|_c3|_c4        |features         |
+---+---+---+---+-----------+-----------------+
|5.1|3.5|1.4|0.2|Iris-setosa|[5.1,3.5,1.4,0.2]|
|4.9|3.0|1.4|0.2|Iris-setosa|[4.9,3.0,1.4,0.2]|
|4.7|3.2|1.3|0.2|Iris-setosa|[4.7,3.2,1.3,0.2]|
|4.6|3.1|1.5|0.2|Iris-setosa|[4.6,3.1,1.5,0.2]|
|5.0|3.6|1.4|0.2|Iris-setosa|[5.0,3.6,1.4,0.2]|
+---+---+---+---+-----------+-----------------+
only showing top 5 rows
```

In [8]:
```
scaler = StandardScaler(inputCol="features", outputCol="scaled_features")
scaler_model = scaler.fit(data_df)
data_df = scaler_model.transform(data_df)
data_df.show(5,0)
```

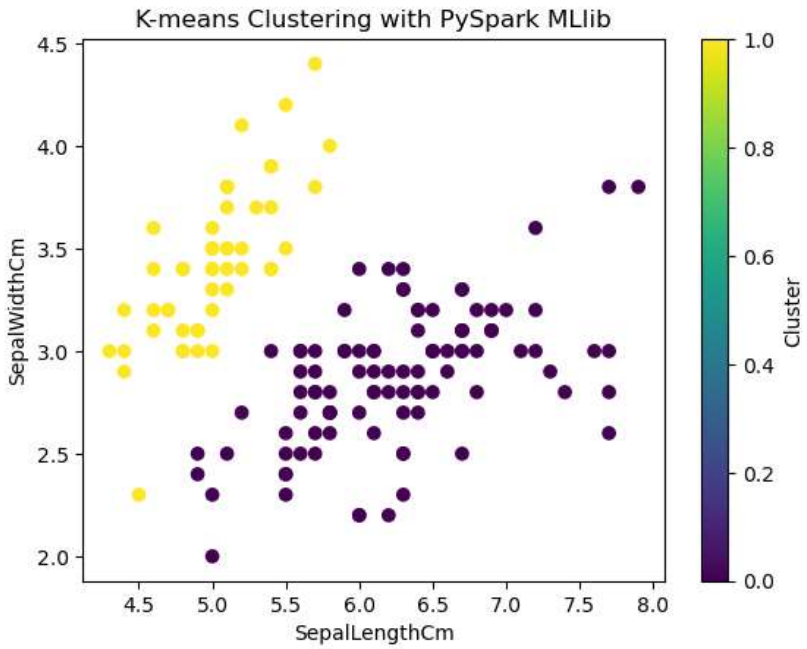
```
+---+---+---+---+-----------+-----------------+-----------------------------------------------------------------------
---------+
|_c0|_c1|_c2|_c3|_c4        |features         |scaled_features
|
+---+---+---+---+-----------+-----------------+-----------------------------------------------------------------------
---------+
|5.1|3.5|1.4|0.2|Iris-setosa|[5.1,3.5,1.4,0.2]|[6.158928408838787,8.072061621390857,0.7934616853039358,0.262067
98787142]|
|4.9|3.0|1.4|0.2|Iris-setosa|[4.9,3.0,1.4,0.2]|[5.9174018045706,6.9189099611921625,0.7934616853039358,0.2620679
8787142] |
|4.7|3.2|1.3|0.2|Iris-setosa|[4.7,3.2,1.3,0.2]|[5.675875200302412,7.38017062527164,0.7367858506393691,0.2620679
8787142] |
|4.6|3.1|1.5|0.2|Iris-setosa|[4.6,3.1,1.5,0.2]|[5.555111898168318,7.149540293231902,0.8501375199685027,0.262067
98787142]|
|5.0|3.6|1.4|0.2|Iris-setosa|[5.0,3.6,1.4,0.2]|[6.038165106704694,8.302691953430596,0.7934616853039358,0.262067
98787142]|
+---+---+---+---+-----------+-----------------+-----------------------------------------------------------------------
---------+
only showing top 5 rows
```

In [9]:
```
from pyspark.ml.clustering import KMeans
from pyspark.ml.evaluation import ClusteringEvaluator

kmeans = KMeans(k=2, featuresCol="scaled_features", predictionCol="cluster")
kmeans_model = kmeans.fit(data_df)

clustered_data = kmeans_model.transform(data_df)
```

In [10]:
```
evaluator = ClusteringEvaluator(predictionCol='cluster', featuresCol='scaled_features', metricName='silhouette',
wssse = evaluator.evaluate(clustered_data)
print(f"Within Set Sum of Squared Errors (WSSSE) = {wssse}")
```

```
Within Set Sum of Squared Errors (WSSSE) = 0.7714149126311811
```

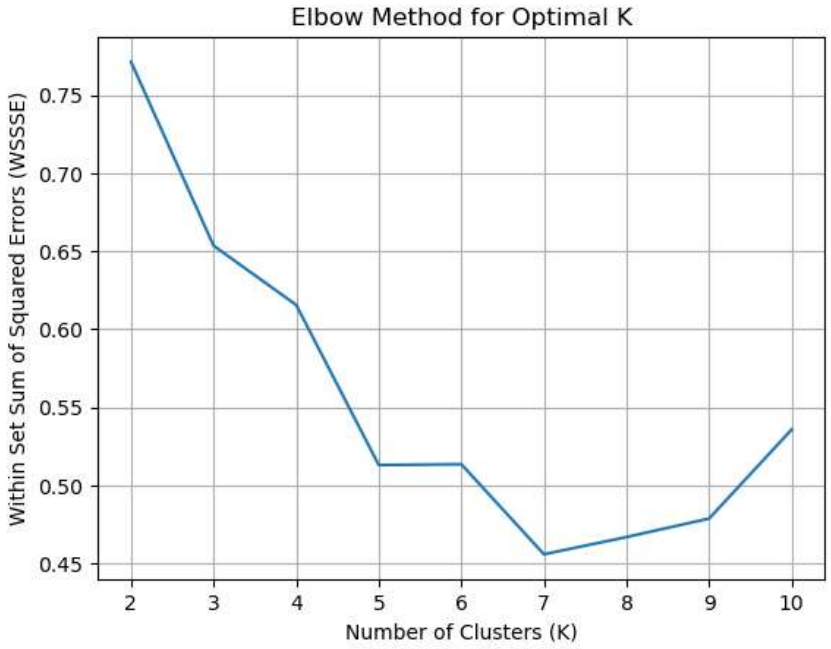
In [11]:
```
clustered_data_pd = clustered_data.toPandas()

plt.scatter(clustered_data_pd["_c0"], clustered_data_pd["_c1"], c=clustered_data_pd["cluster"], cmap='viridis')
plt.xlabel("SepalLengthCm")
plt.ylabel("SepalWidthCm")
plt.title("K-means Clustering with PySpark MLlib")
plt.colorbar().set_label("Cluster")
plt.show()
```



In [12]:
```
wssse_values =[]
evaluator = ClusteringEvaluator(predictionCol='prediction', featuresCol='scaled_features', metricName='silhouett
for i in range(2,11):
    KMeans_mod = KMeans(featuresCol='scaled_features', k=i)
    KMeans_fit = KMeans_mod.fit(data_df)
    output = KMeans_fit.transform(data_df)
    score = evaluator.evaluate(output)
    wssse_values.append(score)
    print("Silhouette Score:",score)
```

```
Silhouette Score: 0.7714149126311811
Silhouette Score: 0.6535875501205959
Silhouette Score: 0.6156585151435247
Silhouette Score: 0.5130776323604693
Silhouette Score: 0.5135784643603635
Silhouette Score: 0.4559012119042887
Silhouette Score: 0.4669358227169013
Silhouette Score: 0.47878278122125645
Silhouette Score: 0.5358603965077863
```

In [13]:
```
plt.plot( range(2,11),wssse_values)
plt.xlabel('Number of Clusters (K)')
plt.ylabel('Within Set Sum of Squared Errors (WSSSE)')
plt.title('Elbow Method for Optimal K')
plt.grid()
plt.show()
```



Practical 8: WAP to implement various classification models and evaluate their performance using pyspark.

In [2]:
```
from pyspark import SparkConf,SparkContext
conf=SparkConf().setAppName('abc').setMaster('local') #
sc=SparkContext(conf=conf)
sc.setLogLevel('ERROR')
from pyspark.sql import SparkSession
spark=SparkSession.builder.appName('abc').config('','').getOrCreate()
import numpy as np
import pandas as pd
import matplotlib as plt
```

In [3]:
```
df = spark.read.csv('bank.csv', header = True, inferSchema = True)
df.printSchema()
```

```
root
 |-- age: integer (nullable = true)
 |-- job: string (nullable = true)
 |-- marital: string (nullable = true)
 |-- education: string (nullable = true)
 |-- default: string (nullable = true)
 |-- balance: integer (nullable = true)
 |-- housing: string (nullable = true)
 |-- loan: string (nullable = true)
 |-- contact: string (nullable = true)
 |-- day: integer (nullable = true)
 |-- month: string (nullable = true)
 |-- duration: integer (nullable = true)
 |-- campaign: integer (nullable = true)
 |-- pdays: integer (nullable = true)
 |-- previous: integer (nullable = true)
 |-- poutcome: string (nullable = true)
 |-- deposit: string (nullable = true)
```

In [4]:
```
df.show(5,0)
```

```
+---+----------+-------+---------+-------+-------+-------+----+-------+---+-----+--------+--------+-----+--------+--------+
-------+
|age|job       |marital|education|default|balance|housing|loan|contact|day|month|duration|campaign|pdays|previous|poutcome|
deposit|
+---+----------+-------+---------+-------+-------+-------+----+-------+---+-----+--------+--------+-----+--------+--------+
-------+
|59 |admin.    |married|secondary|no     |2343   |yes    |no  |unknown|5  |may  |1042    |1       |-1   |0       |unknown |
yes    |
|56 |admin.    |married|secondary|no     |45     |no     |no  |unknown|5  |may  |1467    |1       |-1   |0       |unknown |
yes    |
|41 |technician|married|secondary|no     |1270   |yes    |no  |unknown|5  |may  |1389    |1       |-1   |0       |unknown |
yes    |
|55 |services  |married|secondary|no     |2476   |yes    |no  |unknown|5  |may  |579     |1       |-1   |0       |unknown |
yes    |
|54 |admin.    |married|tertiary |no     |184    |no     |no  |unknown|5  |may  |673     |2       |-1   |0       |unknown |
yes    |
+---+----------+-------+---------+-------+-------+-------+----+-------+---+-----+--------+--------+-----+--------+--------+
-------+
only showing top 5 rows
```

In [5]:
```
df.count()
```

Out[5]: 11162

In [6]:
```
numeric_features = [t[0] for t in df.dtypes if t[1] == 'int']
df.select(numeric_features).describe().toPandas().transpose()
```

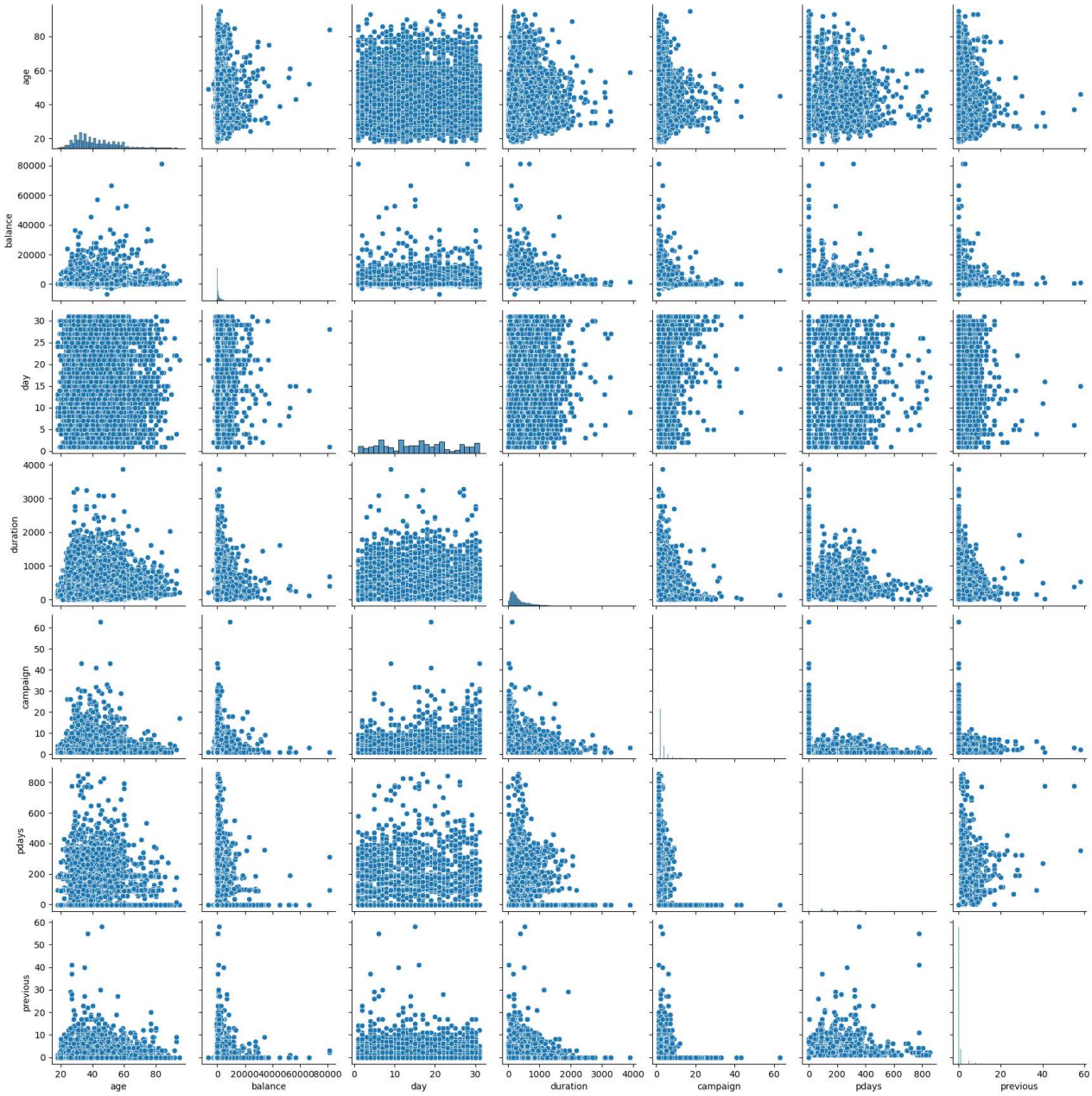
Out[6]:

	0	1	2	3	4
summary	count	mean	stddev	min	max
age	11162	41.231947679627304	11.913369192215518	18	95
balance	11162	1528.5385235620856	3225.413325946149	-6847	81204
day	11162	15.658036194230425	8.420739541006462	1	31
duration	11162	371.99381831213043	347.12838571630687	2	3881
campaign	11162	2.508421429851281	2.7220771816614824	1	63
pdays	11162	51.33040673714388	108.75828197197717	-1	854
previous	11162	0.8325568894463358	2.292007218670508	0	58

In [7]:
```
numeric_data = df.select(numeric_features).toPandas()
import seaborn as sns
sns.pairplot(numeric_data)
```

```
C:\Users\subha\anaconda3\Lib\site-packages\seaborn\axisgrid.py:118: UserWarning: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```

Out[7]: <seaborn.axisgrid.PairGrid at 0x270b09805d0>



In [8]:
```
df = df.select('age', 'job', 'marital', 'education', 'default', 'balance','housing', 'loan', 'contact', 'duration', 'campaig
'pdays', 'previous', 'poutcome', 'deposit')

cols = df.columns
df.printSchema()
```

```
root
 |-- age: integer (nullable = true)
 |-- job: string (nullable = true)
 |-- marital: string (nullable = true)
 |-- education: string (nullable = true)
 |-- default: string (nullable = true)
 |-- balance: integer (nullable = true)
 |-- housing: string (nullable = true)
 |-- loan: string (nullable = true)
 |-- contact: string (nullable = true)
 |-- duration: integer (nullable = true)
 |-- campaign: integer (nullable = true)
 |-- pdays: integer (nullable = true)
 |-- previous: integer (nullable = true)
 |-- poutcome: string (nullable = true)
 |-- deposit: string (nullable = true)
```

In [11]:
```
from pyspark.ml.feature import OneHotEncoder, StringIndexer, VectorAssembler
categoricalColumns = ['job', 'marital', 'education', 'default', 'housing','loan', 'contact', 'poutcome']
stages = []
for categoricalCol in categoricalColumns:
    stringIndexer = StringIndexer(inputCol = categoricalCol, outputCol = categoricalCol + 'Index')
    encoder = OneHotEncoder(inputCols=[stringIndexer.getOutputCol()], outputCols=[categoricalCol + "classVec"])
    stages += [stringIndexer, encoder]
label_stringIdx = StringIndexer(inputCol = 'deposit', outputCol = 'label')
stages += [label_stringIdx]
numericCols = ['age', 'balance', 'duration', 'campaign', 'pdays', 'previous']
assemblerInputs = [c + "classVec" for c in categoricalColumns] + numericCols
assembler = VectorAssembler(inputCols=assemblerInputs, outputCol="features")
stages += [assembler]
```

In [12]:
```
assemblerInputs
```

Out[12]:
```
['jobclassVec',
 'maritalclassVec',
 'educationclassVec',
 'defaultclassVec',
 'housingclassVec',
 'loanclassVec',
 'contactclassVec',
 'poutcomeclassVec',
 'age',
 'balance',
 'duration',
 'campaign',
 'pdays',
 'previous']
```

In [13]:
```
stages
```

Out[13]:
```
[StringIndexer_3555869e127d,
 OneHotEncoder_979f031d4756,
 StringIndexer_2729556f051d,
 OneHotEncoder_e1fa78fce636,
 StringIndexer_28010c65fcbe,
 OneHotEncoder_7ee26aacaeaf,
 StringIndexer_5f077fe38143,
 OneHotEncoder_4e82642b33c1,
 StringIndexer_b4fe4dfe3efc,
 OneHotEncoder_7bbafd2b416e,
 StringIndexer_5d74bc1087d5,
 OneHotEncoder_4f399195a873,
 StringIndexer_2b4a22788e6f,
 OneHotEncoder_81907173801b,
 StringIndexer_469e1fdee4a8,
 OneHotEncoder_a567a9c0422e,
 StringIndexer_94984ae01f0a,
 VectorAssembler_e7f781180497]
```

In [14]:
```
from pyspark.ml import Pipeline
pipeline = Pipeline(stages = stages)
pipelineModel = pipeline.fit(df)
df = pipelineModel.transform(df)
selectedCols = ['label', 'features'] + cols
df = df.select(selectedCols)
df.printSchema()
```

```
root
 |-- label: double (nullable = false)
 |-- features: vector (nullable = true)
 |-- age: integer (nullable = true)
 |-- job: string (nullable = true)
 |-- marital: string (nullable = true)
 |-- education: string (nullable = true)
 |-- default: string (nullable = true)
 |-- balance: integer (nullable = true)
 |-- housing: string (nullable = true)
 |-- loan: string (nullable = true)
 |-- contact: string (nullable = true)
 |-- duration: integer (nullable = true)
 |-- campaign: integer (nullable = true)
 |-- pdays: integer (nullable = true)
 |-- previous: integer (nullable = true)
 |-- poutcome: string (nullable = true)
 |-- deposit: string (nullable = true)
```

In [15]:
```
df.show(5,0)
```

```
+-----+-----------------------------------------------------------------------------------------------------------------+---+-------
---+-------+---------+-------+-------+-------+----+-------+--------+--------+-----+--------+--------+-------+
|label|features                                                                                                         |age|job
|marital|education|default|balance|housing|loan|contact|duration|campaign|pdays|previous|poutcome|deposit|
+-----+-----------------------------------------------------------------------------------------------------------------+---+-------
---+-------+---------+-------+-------+-------+----+-------+--------+--------+-----+--------+--------+-------+
|1.0  |(30,[3,11,13,16,18,20,21,24,25,26,27,28],[1.0,1.0,1.0,1.0,1.0,1.0,1.0,59.0,2343.0,1042.0,1.0,-1.0])     |59 |admin.
|married|secondary|no     |2343   |yes    |no  |unknown|1042    |1       |-1   |0       |unknown |yes    |
|1.0  |(30,[3,11,13,16,17,18,20,21,24,25,26,27,28],[1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,56.0,45.0,1467.0,1.0,-1.0])|56 |admin.
|married|secondary|no     |45     |no     |no  |unknown|1467    |1       |-1   |0       |unknown |yes    |
|1.0  |(30,[2,11,13,16,18,20,21,24,25,26,27,28],[1.0,1.0,1.0,1.0,1.0,1.0,1.0,41.0,1270.0,1389.0,1.0,-1.0])     |41 |technic
ian|married|secondary|no     |1270   |yes    |no  |unknown|1389    |1       |-1   |0       |unknown |yes    |
|1.0  |(30,[4,11,13,16,18,20,21,24,25,26,27,28],[1.0,1.0,1.0,1.0,1.0,1.0,1.0,55.0,2476.0,579.0,1.0,-1.0])      |55 |service
s  |married|secondary|no     |2476   |yes    |no  |unknown|579     |1       |-1   |0       |unknown |yes    |
|1.0  |(30,[3,11,14,16,17,18,20,21,24,25,26,27,28],[1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,54.0,184.0,673.0,2.0,-1.0])|54 |admin.
|married|tertiary |no     |184    |no     |no  |unknown|673     |2       |-1   |0       |unknown |yes    |
+-----+-----------------------------------------------------------------------------------------------------------------+---+-------
---+-------+---------+-------+-------+-------+----+-------+--------+--------+-----+--------+--------+-------+
only showing top 5 rows
```

In [16]:
```
df.select(['label','features']).show(5,0)
```

```
+-----+-----------------------------------------------------------------------------------------------------------------+
|label|features                                                                                                         |
+-----+-----------------------------------------------------------------------------------------------------------------+
|1.0  |(30,[3,11,13,16,18,20,21,24,25,26,27,28],[1.0,1.0,1.0,1.0,1.0,1.0,1.0,59.0,2343.0,1042.0,1.0,-1.0])     |
|1.0  |(30,[3,11,13,16,17,18,20,21,24,25,26,27,28],[1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,56.0,45.0,1467.0,1.0,-1.0])|
|1.0  |(30,[2,11,13,16,18,20,21,24,25,26,27,28],[1.0,1.0,1.0,1.0,1.0,1.0,1.0,41.0,1270.0,1389.0,1.0,-1.0])     |
|1.0  |(30,[4,11,13,16,18,20,21,24,25,26,27,28],[1.0,1.0,1.0,1.0,1.0,1.0,1.0,55.0,2476.0,579.0,1.0,-1.0])      |
|1.0  |(30,[3,11,14,16,17,18,20,21,24,25,26,27,28],[1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,54.0,184.0,673.0,2.0,-1.0])|
+-----+-----------------------------------------------------------------------------------------------------------------+
only showing top 5 rows
```

In [17]:
```
df2=pd.DataFrame(df.take(5),columns=df.columns).iloc[:,:2]
pd.set_option('display.max_colwidth', None)
print(df2)
```

```
   label  \
0    1.0
1    1.0
2    1.0
3    1.0
4    1.0


features
0  (0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 1.0, 0.0, 0.0, 1.0, 0.0, 1.0, 0.0, 1.0, 1.0, 0.0, 0.0,
59.0, 2343.0, 1042.0, 1.0, -1.0, 0.0)
1    (0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 1.0, 0.0, 0.0, 1.0, 1.0, 1.0, 0.0, 1.0, 1.0, 0.0, 0.
0, 56.0, 45.0, 1467.0, 1.0, -1.0, 0.0)
2  (0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 1.0, 0.0, 0.0, 1.0, 0.0, 1.0, 0.0, 1.0, 1.0, 0.0, 0.0,
41.0, 1270.0, 1389.0, 1.0, -1.0, 0.0)
3   (0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 1.0, 0.0, 0.0, 1.0, 0.0, 1.0, 0.0, 1.0, 1.0, 0.0, 0.
0, 55.0, 2476.0, 579.0, 1.0, -1.0, 0.0)
4    (0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 1.0, 0.0, 1.0, 1.0, 1.0, 0.0, 1.0, 1.0, 0.0, 0.
0, 54.0, 184.0, 673.0, 2.0, -1.0, 0.0)
```

In [18]:
```
print(df.count())
train, test = df.randomSplit([0.7, 0.3], seed = 123)
print("Training Dataset Count: " + str(train.count()))
print("Test Dataset Count: " + str(test.count()))
```

```
11162
Training Dataset Count: 7725
Test Dataset Count: 3437
```

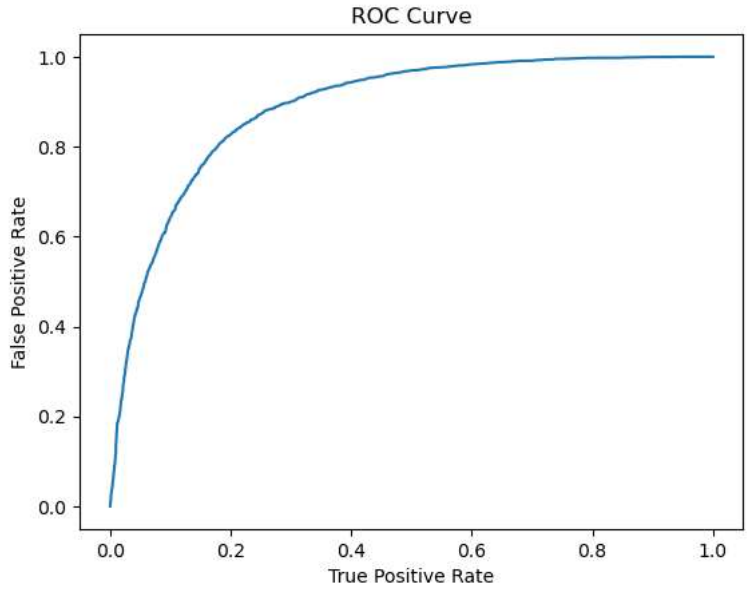
In [19]:
```
from pyspark.ml.classification import LogisticRegression
lr = LogisticRegression(featuresCol = 'features', labelCol = 'label', maxIter=10)
lrModel = lr.fit(train)
```

In [20]:
```
lrModel.coefficients
```

Out[20]:
```
DenseVector([-0.1701, -0.1772, 0.0227, 0.2651, -0.1759, 0.5859, -0.3137, 0.7606, -0.017, -0.3996, -0.155, -0.2144, 0.1096,
-0.1197, 0.2793, -0.2922, 0.3795, 0.8589, 0.5599, 0.0549, -1.3384, -0.491, -0.2352, 1.9416, 0.0002, 0.0, 0.0052, -0.1259,
0.0005, 0.0069])
```

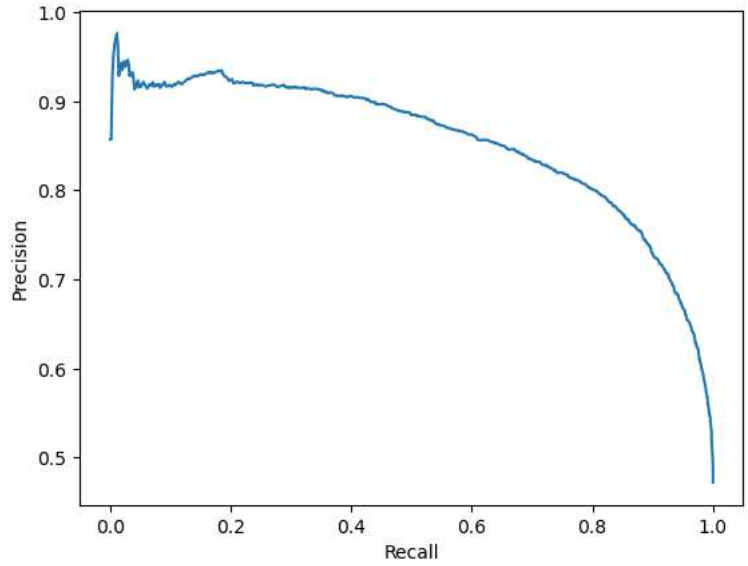
In [21]:
```
trainingSummary = lrModel.summary
```

In [22]:
```
roc = trainingSummary.roc.toPandas()
import matplotlib.pyplot as plt
plt.plot(roc['FPR'],roc['TPR'])
plt.ylabel('False Positive Rate')
plt.xlabel('True Positive Rate')
plt.title('ROC Curve')
plt.show()
print('Training set areaUnderROC: ' + str(trainingSummary.areaUnderROC))
```




```
Training set areaUnderROC: 0.8876963840801431
```

In [23]:
```
pr = trainingSummary.pr.toPandas()
plt.plot(pr['recall'],pr['precision'])
plt.ylabel('Precision')
plt.xlabel('Recall')
plt.show()
```



In [24]:
```
predictions = lrModel.transform(test)
predictions.select('age', 'job', 'label', 'rawPrediction', 'prediction', 'probability').show(10)
```

```
+---+----------+-----+--------------------+----------+--------------------+
|age|       job|label|       rawPrediction|prediction|         probability|
+---+----------+-----+--------------------+----------+--------------------+
| 35|management|  0.0|[1.99365036233196...|       0.0|[0.88012879313681...|
| 42|management|  0.0|[0.27611238609135...|       0.0|[0.56859286703856...|
| 51|management|  0.0|[3.48268389062406...|       0.0|[0.97019103738341...|
| 53|management|  0.0|[0.94274287705725...|       0.0|[0.71965337269069...|
| 54|management|  0.0|[0.20806403309255...|       0.0|[0.55182916654916...|
| 67|management|  0.0|[-0.9383151886495...|       1.0|[0.28124079221897...|
| 57|management|  0.0|[1.08993750298533...|       0.0|[0.74836995282901...|
| 46|management|  0.0|[0.43591840216268...|       0.0|[0.60728603910138...|
| 52|management|  0.0|[3.41892378893558...|       0.0|[0.96829074452287...|
| 31|management|  0.0|[1.48232796139663...|       0.0|[0.81492394805164...|
+---+----------+-----+--------------------+----------+--------------------+
only showing top 10 rows
```

In [25]:
```
from pyspark.ml.evaluation import BinaryClassificationEvaluator
evaluator = BinaryClassificationEvaluator()
print('Test Area Under ROC', evaluator.evaluate(predictions))
print("Test Area Under ROC: " + str(evaluator.evaluate(predictions, {evaluator.metricName: "areaUnderROC"})))
print("Test Area Under PR: " + str(evaluator.evaluate(predictions, {evaluator.metricName: "areaUnderPR"})))
```

```
Test Area Under ROC 0.883169179529566
Test Area Under ROC: 0.883169179529566
Test Area Under PR: 0.8498179283400261
```

In [26]:
```
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
evaluator = MulticlassClassificationEvaluator()
print('Test Accuracy', evaluator.evaluate(predictions))
print("Test Accuracy: " + str(evaluator.evaluate(predictions, {evaluator.metricName: "accuracy"})))
print("True Positive Rate of class 1: " + str(evaluator.evaluate(predictions, {evaluator.metricName: "truePositiveRateByLabel
```

```
Test Accuracy 0.8051110841633259
Test Accuracy: 0.8056444573756183
True Positive Rate of class 1: 0.7605118829981719
```

In [27]:
```
from pyspark.ml.classification import DecisionTreeClassifier
dt = DecisionTreeClassifier(featuresCol = 'features', labelCol = 'label', maxDepth = 3)
dtModel = dt.fit(train)
predictions = dtModel.transform(test)
predictions.select('age', 'job', 'label', 'rawPrediction', 'prediction', 'probability').show(10)
```

```
+---+----------+-----+--------------+----------+--------------------+
|age|       job|label| rawPrediction|prediction|         probability|
+---+----------+-----+--------------+----------+--------------------+
| 35|management|  0.0|[792.0,1238.0]|       1.0|[0.39014778325123...|
| 42|management|  0.0|[792.0,1238.0]|       1.0|[0.39014778325123...|
| 51|management|  0.0|[2456.0,475.0]|       0.0|[0.83793926987376...|
| 53|management|  0.0|[2456.0,475.0]|       0.0|[0.83793926987376...|
| 54|management|  0.0|[792.0,1238.0]|       1.0|[0.39014778325123...|
| 67|management|  0.0|[419.0,1755.0]|       1.0|[0.19273229070837...|
| 57|management|  0.0|[2456.0,475.0]|       0.0|[0.83793926987376...|
| 46|management|  0.0|[419.0,1755.0]|       1.0|[0.19273229070837...|
| 52|management|  0.0|[2456.0,475.0]|       0.0|[0.83793926987376...|
| 31|management|  0.0|[2456.0,475.0]|       0.0|[0.83793926987376...|
+---+----------+-----+--------------+----------+--------------------+
only showing top 10 rows
```

In [28]:
```
evaluator = BinaryClassificationEvaluator()
print("Test Area Under ROC: " + str(evaluator.evaluate(predictions, {evaluator.metricName: "areaUnderROC"})))
```

```
Test Area Under ROC: 0.7749096102246309
```

In [29]:
```
evaluator = BinaryClassificationEvaluator()
print("Test Area Under ROC: " + str(evaluator.evaluate(predictions, {evaluator.metricName: "areaUnderROC"})))
```

```
Test Area Under ROC: 0.7749096102246309
```

In [30]:
```
evaluator = BinaryClassificationEvaluator()
print("Test Area Under ROC: " + str(evaluator.evaluate(predictions, {evaluator.metricName: "areaUnderROC"})))
```

```
Test Area Under ROC: 0.7749096102246309
```

In [31]:
```
from pyspark.ml.classification import GBTClassifier
gbt = GBTClassifier(maxIter=10)
gbtModel = gbt.fit(train)
predictions = gbtModel.transform(test)
predictions.select('age', 'job', 'label', 'rawPrediction', 'prediction', 'probability').show(10)
```

```
+---+----------+-----+--------------------+----------+--------------------+
|age|       job|label|       rawPrediction|prediction|         probability|
+---+----------+-----+--------------------+----------+--------------------+
| 35|management|  0.0|[-0.1159357134728...|       1.0|[0.44229047152033...|
| 42|management|  0.0|[-0.2057116089783...|       1.0|[0.39857090933186...|
| 51|management|  0.0|[1.21941959425578...|       0.0|[0.91974144180373...|
| 53|management|  0.0|[0.09711126458426...|       0.0|[0.54840356967139...|
| 54|management|  0.0|[-0.1816865192908...|       1.0|[0.41014329434699...|
| 67|management|  0.0|[-0.9213839133606...|       1.0|[0.13672427531567...|
| 57|management|  0.0|[0.49413854422769...|       0.0|[0.72874748077226...|
| 46|management|  0.0|[0.37229622059617...|       0.0|[0.67799927929400...|
| 52|management|  0.0|[1.30125815243469...|       0.0|[0.93102334928564...|
| 31|management|  0.0|[1.08875556361164...|       0.0|[0.89821174634236...|
+---+----------+-----+--------------------+----------+--------------------+
only showing top 10 rows
```

In [32]:
```
evaluator = BinaryClassificationEvaluator()
print("Test Area Under ROC: " + str(evaluator.evaluate(predictions, {evaluator.metricName: "areaUnderROC"})))
```

```
Test Area Under ROC: 0.888314848217109
```

In [33]:
```
print(gbt.explainParams())
```

```
cacheNodeIds: If false, the algorithm will pass trees to executors to match instances with nodes. If true, the algorithm wi
ll cache node IDs for each instance. Caching can speed up training of deeper trees. Users can set how often should the cach
e be checkpointed or disable it by setting checkpointInterval. (default: False)
checkpointInterval: set checkpoint interval (>= 1) or disable checkpoint (-1). E.g. 10 means that the cache will get checkp
ointed every 10 iterations. Note: this setting will be ignored if the checkpoint directory is not set in the SparkContext.
(default: 10)
featureSubsetStrategy: The number of features to consider for splits at each tree node. Supported options: 'auto' (choose a
utomatically for task: If numTrees == 1, set to 'all'. If numTrees > 1 (forest), set to 'sqrt' for classification and to 'o
nethird' for regression), 'all' (use all features), 'onethird' (use 1/3 of the features), 'sqrt' (use sqrt(number of featur
es)), 'log2' (use log2(number of features)), 'n' (when n is in the range (0, 1.0], use n * number of features. When n is in
the range (1, number of features), use n features). default = 'auto' (default: all)
featuresCol: features column name. (default: features)
impurity: Criterion used for information gain calculation (case-insensitive). Supported options: variance (default: varianc
e)
labelCol: label column name. (default: label)
leafCol: Leaf indices column name. Predicted leaf index of each instance in each tree by preorder. (default: )
lossType: Loss function which GBT tries to minimize (case-insensitive). Supported options: logistic (default: logistic)
maxBins: Max number of bins for discretizing continuous features.  Must be >=2 and >= number of categories for any categori
cal feature. (default: 32)
maxDepth: Maximum depth of the tree. (>= 0) E.g., depth 0 means 1 leaf node; depth 1 means 1 internal node + 2 leaf nodes.
Must be in range [0, 30]. (default: 5)
maxIter: max number of iterations (>= 0). (default: 20, current: 10)
maxMemoryInMB: Maximum memory in MB allocated to histogram aggregation. If too small, then 1 node will be split per iterati
on, and its aggregates may exceed this size. (default: 256)
minInfoGain: Minimum information gain for a split to be considered at a tree node. (default: 0.0)
minInstancesPerNode: Minimum number of instances each child must have after split. If a split causes the left or right chil
d to have fewer than minInstancesPerNode, the split will be discarded as invalid. Should be >= 1. (default: 1)
minWeightFractionPerNode: Minimum fraction of the weighted sample count that each child must have after split. If a split c
auses the fraction of the total weight in the left or right child to be less than minWeightFractionPerNode, the split will
be discarded as invalid. Should be in interval [0.0, 0.5). (default: 0.0)
predictionCol: prediction column name. (default: prediction)
probabilityCol: Column name for predicted class conditional probabilities. Note: Not all models output well-calibrated prob
ability estimates! These probabilities should be treated as confidences, not precise probabilities. (default: probability)
rawPredictionCol: raw prediction (a.k.a. confidence) column name. (default: rawPrediction)
seed: random seed. (default: -3803952149970227115)
stepSize: Step size (a.k.a. learning rate) in interval (0, 1] for shrinking the contribution of each estimator. (default:
0.1)
subsamplingRate: Fraction of the training data used for learning each decision tree, in range (0, 1]. (default: 1.0)
thresholds: Thresholds in multi-class classification to adjust the probability of predicting each class. Array must have le
ngth equal to the number of classes, with values > 0, excepting that at most one value may be 0. The class with largest val
ue p/t is predicted, where p is the original probability of that class and t is the class's threshold. (undefined)
validationIndicatorCol: name of the column that indicates whether each row is for training or for validation. False indicat
es training; true indicates validation. (undefined)
validationTol: Threshold for stopping early when fit with validation is used. If the error rate on the validation input cha
nges by less than the validationTol, then learning will stop early (before `maxIter`). This parameter is ignored when fit w
ithout validation is used. (default: 0.01)
weightCol: weight column name. If this is not set or empty, we treat all instance weights as 1.0. (undefined)
```

In [34]:
```
from pyspark.ml.tuning import ParamGridBuilder, CrossValidator
paramGrid = (ParamGridBuilder().addGrid(gbt.maxDepth, [2, 4, 6]).addGrid(gbt.maxBins, [20, 60]).addGrid(gbt.maxIter, [10, 20

cv = CrossValidator(estimator=gbt, estimatorParamMaps=paramGrid, evaluator=evaluator, numFolds=5)

cvModel = cv.fit(train)
predictions = cvModel.transform(test)
evaluator.evaluate(predictions)
```

Out[34]:
```
0.8932867948138541
```

Practical 9: WAP to implement various K-means clustering and find the best value of K using the elbow method using pyspark

In [2]:
```
from pyspark import SparkConf,SparkContext
conf=SparkConf().setAppName('abc').setMaster('local')
sc=SparkContext(conf=conf)
sc.setLogLevel('ERROR')
from pyspark.sql import SparkSession
spark=SparkSession.builder.appName('WC').config('','').getOrCreate()
```

In [3]:
```
lines=sc.textFile('file01.txt')
print(lines)
```

```
file01.txt MapPartitionsRDD[1] at textFile at NativeMethodAccessorImpl.java:0
```

In [4]:
```
words=lines.flatMap(lambda line:line.split(' '))
print(words)
```

```
PythonRDD[2] at RDD at PythonRDD.scala:53
```

In [5]:
```
wordcounts=words.countByValue()
print(wordcounts)
```

```
defaultdict(<class 'int'>, {'Hello': 1, 'World': 2, 'Bye': 1})
```

Practical 10: Recommendation System

In [ ]:
```
!pip install pyspark
```

```
Collecting pyspark
  Downloading pyspark-3.5.0.tar.gz (316.9 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 316.9/316.9 MB 2.3 MB/s eta
0:00:00
  Preparing metadata (setup.py) ... done
Requirement already satisfied: py4j==0.10.9.7 in /usr/local/lib/python3.1
0/dist-packages (from pyspark) (0.10.9.7)
Building wheels for collected packages: pyspark
  Building wheel for pyspark (setup.py) ... done
  Created wheel for pyspark: filename=pyspark-3.5.0-py2.py3-none-any.whl s
ize=317425345 sha256=22a9242c7de9f541850d7c7d02f3c01fa3c5187fb477bba5f3bd3
ed297f13973
  Stored in directory: /root/.cache/pip/wheels/41/4e/10/c2cf2467f71c678cfc
8a6b9ac9241e5e44a01940da8fbb17fc
Successfully built pyspark
Installing collected packages: pyspark
Successfully installed pyspark-3.5.0
```

In [ ]:
```
from pyspark.sql import SparkSession
from pyspark.ml.evaluation import RegressionEvaluator
from pyspark.ml.recommendation import ALS
#Setup Spark Session
spark = SparkSession.builder.appName('Recommender').getOrCreate()
spark
```

Out[3]:

SparkSession - in-memory

SparkContext

Spark UI (http://bc2452aa52a3:4040)

Version

v3.5.0

Master

local[*]

AppName

Recommender

In [ ]:
```
data = spark.read.csv('/content/drive/MyDrive/datasets/book_ratings.csv',

inferSchema=True,header=True)
```

In [ ]:
```
data.show(5)
```

```
+-------+-------+------+
|book_id|user_id|rating|
+-------+-------+------+
|      1|    314|     5|
|      1|    439|     3|
|      1|    588|     5|
|      1|   1169|     4|
|      1|   1185|     4|
+-------+-------+------+
only showing top 5 rows
```

In [ ]:
```
data.count()
```

Out[6]: 981756

In [ ]:
```
data.describe().show()
```

```
+-------+-----------------+------------------+------------------+
|summary|          book_id|           user_id|            rating|
+-------+-----------------+------------------+------------------+
|  count|           981756|            981756|            981756|
|   mean|4943.275635697668|25616.759933221696|3.8565335989797873|
| stddev|2873.207414896114|15228.338825882167|0.9839408559620033|
|    min|                1|                 1|                 1|
|    max|            10000|             53424|                 5|
+-------+-----------------+------------------+------------------+
```

In [ ]:
```
train_data, test_data = data.randomSplit([0.8, 0.2])
```

In [ ]:
```
als = ALS(maxIter=5,
          regParam=0.01,
          userCol="user_id",
          itemCol="book_id",
          ratingCol="rating")
```

In [ ]:
```
model = als.fit(train_data)
```

In [ ]:
```
predictions = model.transform(test_data)
```

In [ ]:
```
predictions.show()
```

```
+-------+-------+------+----------+
|book_id|user_id|rating|prediction|
+-------+-------+------+----------+
|      1|  32592|     4|  4.295682|
|      1|    588|     5| 4.2512364|
|      1|  16913|     5| 3.7509816|
|      1|  32305|     5| 4.4636183|
|      1|  38475|     4|  4.085602|
|      1|  11927|     4|   5.27649|
|      1|  33065|     4|  4.986938|
|      1|  42404|     5| 4.8027167|
|      1|  21487|     4|   4.20221|
|      1|  16377|     4| 5.1734757|
|      1|  17663|     5| 4.4150023|
|      1|    439|     3| 3.7341223|
|      1|  37284|     5| 3.8447344|
|      1|   1185|     4|  3.946853|
|      1|  33872|     5| 4.4724183|
|      1|  21228|     5|  4.156422|
|      1|  44397|     5|  5.076925|
|      1|  30681|     5|  3.155237|
|      1|  23612|     4|  4.333251|
|      1|  37834|     5| 4.7758417|
+-------+-------+------+----------+
only showing top 20 rows
```

In [ ]:
```
user1 = test_data.filter(test_data['user_id']==5461).select(['book_id','use
```

In [ ]:
```
user1.show()
```

```
+-------+-------+
|book_id|user_id|
+-------+-------+
|      7|   5461|
|      8|   5461|
|     37|   5461|
|     47|   5461|
|     82|   5461|
|     86|   5461|
|    117|   5461|
|    118|   5461|
|    129|   5461|
|    130|   5461|
|    255|   5461|
|    261|   5461|
|    281|   5461|
|    304|   5461|
|    321|   5461|
|    339|   5461|
|    358|   5461|
|    396|   5461|
|    444|   5461|
|    478|   5461|
+-------+-------+
only showing top 20 rows
```

In [ ]:
```
user1.count()
```

Out[15]:
```
42
```

In [ ]:
```
recommendations = model.transform(user1)
```

In [ ]:
```
recommendations.orderBy('prediction',ascending=False).show()
```

```
+-------+-------+----------+
|book_id|user_id|prediction|
+-------+-------+----------+
|      8|   5461| 5.0335436|
|     47|   5461|  4.730926|
|    483|   5461|  4.622107|
|   1094|   5461| 4.5957804|
|    339|   5461| 4.5417233|
|    444|   5461| 4.5225058|
|    478|   5461| 4.4244857|
|   1465|   5461| 4.4110894|
|      7|   5461| 4.3999443|
|    129|   5461| 4.3938417|
|     82|   5461|   4.39313|
|    561|   5461|   4.37797|
|   1202|   5461| 4.3322597|
|   1088|   5461|  4.329598|
|    844|   5461|  4.262401|
|    130|   5461| 4.2086234|
|    885|   5461|  4.141724|
|   4877|   5461| 4.1130195|
|   1493|   5461|  4.111848|
|    304|   5461| 4.1024375|
+-------+-------+----------+
only showing top 20 rows
```

In [ ]:
```
recommendations.show()
```

```
+-------+-------+----------+
|book_id|user_id|prediction|
+-------+-------+----------+
|      7|   5461| 4.3999443|
|      8|   5461| 5.0335436|
|     37|   5461| 3.8126752|
|     47|   5461|  4.730926|
|     82|   5461|   4.39313|
|     86|   5461|  3.939124|
|    117|   5461|  3.847703|
|    118|   5461| 3.7317593|
|    129|   5461| 4.3938417|
|    130|   5461| 4.2086234|
|    255|   5461|  3.551429|
|    261|   5461| 3.1461303|
|    281|   5461| 3.1281495|
|    304|   5461| 4.1024375|
|    321|   5461|  3.316253|
|    339|   5461| 4.5417233|
|    358|   5461| 3.5786288|
|    396|   5461| 3.9388258|
|    444|   5461| 4.5225058|
|    478|   5461| 4.4244857|
+-------+-------+----------+
only showing top 20 rows
```