Practical 2

Aim: Implement word count/frequency programs using MapReduce Map

Reduce as a two-component Map and Reduce

Write a program, save as WordCount.java////////////////////////

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

) throws IOException, InterruptedException {

StringTokenizer itr = new StringTokenizer(value.toString());

while (itr.hasMoreTokens()) {//"This is the output is the"

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

{//is,3

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.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);

}

}

////////////////////////Text File:

Hello World

This is the output is theStart the server (Horton Sandbox)

Open the terminal with 192.168.119.132/4200

Enter the login: root and the password and enter

Create a folder in local directory.

Command: mkdir mscitp2

Change the directory cd mscitp2

Now create input file

Command: cat >> wordin.txt

Paste the text by right clicking on terminal

Hello World

This is the output is the

To remove the extra space type command

vi wordin.txt

After removing the extra space check the content of the file

cat wordin.txt

Create another file wordcount.java

Command: cat >> wordcount.txt

Paste the java code.

Press control d to save the file

Check both the files create with command ls

Now, to compile the java file

export HADOOP\_CLASSPATH=$(hadoop classpath)

mkdir classes (To keep the compile files)

javac -classpath ${HADOOP\_CLASSPATH} -d classes WordCount.java

Check class files are created with command ls classes

Now we have to bind all the class into single jar file with below command

jar -cvf WordCount.jar -C classes/ .

Run ls command we can see jar file is created.

wordin.txt should be present in word directory of hdfs. So we need to upload wordin.txt file.

hdfs dfs -mkdir /p2

After this its not clear

Practical 3

Aim: - Implement a MapReduce program that processes a weather dataset.

Java program:

MyMaxMin.java

///////////////////////////

// importing Libraries

import java.io.IOException;

import java.util.Iterator;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;

import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;

import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.Mapper;

import org.apache.hadoop.mapreduce.Reducer;

import org.apache.hadoop.conf.Configuration;public class MyMaxMin {

public static class MaxTemperatureMapper extends

Mapper<LongWritable, Text, Text, Text> {

public static final int MISSING = 9999;

@Override

public void map(LongWritable arg0, Text Value, Context context) throws IOException, InterruptedException {

String line = Value.toString();

// Check for the empty line

if (!(line.length() == 0)) {

// from character 6 to 14 we have

// the date in our dataset

String date = line.substring(6, 14);

// similarly we have taken the maximum

// temperature from 39 to 45 characters

float temp\_Max = Float.parseFloat(line.substring(39, 45).trim());

// similarly we have taken the minimum

// temperature from 47 to 53 characters

float temp\_Min = Float.parseFloat(line.substring(47, 53).trim()); // if maximum temperature is

// greater than 30, it is a hot day

if (temp\_Max > 30.0) {

date),

// Hot day

context.write(new Text("The Day is Hot Day :" + date),new

Text(String.valueOf(temp\_Max)));

} // if the minimum temperature is

date),

// less than 15, it is a cold day

if (temp\_Min < 15) {

// Cold day

context.write(new Text("The Day is Cold Day :" + date),new

Text(String.valueOf(temp\_Min)));

}

}

} }// Reducer

/\*MaxTemperatureReducer class is static

and extends Reducer abstract class

having four Hadoop generics type

Text, Text, Text, Text.

\*/

//The Day is Cold Day :20150101 ,-21.8

public static class MaxTemperatureReducer extends

Reducer<Text, Text, Text, Text> { /\*\*

\* @method reduce

\* This method takes the input as key and

\* list of values pair from the mapper,

\* it does aggregation based on keys and

\* produces the final context.

\*/

context)

public void reduce(Text Key, Iterator<Text> Values, Context

throws IOException, InterruptedException {

// putting all the values in

// temperature variable of type String

String temperature = Values.next().toString();

context.write(Key, new Text(temperature));

} } /\*\*

\* @method main

\* This method is used for setting

\* all the configuration properties.

\* It acts as a driver for map-reduce

public static void main(String[] args) throws Exception { // reads the default configuration of the

// cluster from the configuration XML files

Configuration conf = new Configuration();

// Initializing the job with the

// default configuration of the cluster

Job = new Job(conf, "weather example");

of mapper

// Assigning the driver class name

job.setJarByClass(MyMaxMin.class); // Key type coming out job.setMapOutputKeyClass(Text.class);

// value type coming out of mapper

job.setMapOutputValueClass(Text.class); // Defining the mapper class name

job.setMapperClass(MaxTemperatureMapper.class);

// Defining the reducer class name

job.setReducerClass(MaxTemperatureReducer.class); // Defining input Format class which is

// responsible to parse the dataset

// into a key value pair

job.setInputFormatClass(TextInputFormat.class);

// Defining output Format class which is

// responsible to parse the dataset

// into a key value pair

job.setOutputFormatClass(TextOutputFormat.class); // setting the second argument

// as a path in a path variable

Path OutputPath = new Path(args[1]); // Configuring the input path

// from the filesystem into the job

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

// Configuring the output path from

// the filesystem into the job

FileOutputFormat.setOutputPath(job, new Path(args[1])); // deleting the context path automatically

// from hdfs so that we don't have

// to delete it explicitly

OutputPath.getFileSystem(conf).delete(OutputPath);

// flag value becomes false

System.exit(job.waitForCompletion(true) ? 0 : 1); }

}

Result

Open the terminal with 192.168.119.132/4200

Enter the login: root and the password and enter

Create a folder in local directory.

Command: mkdir mscitp3

Change the directory cd mscitp3

Now create input file

Command: cat >> weatherin2.txt

Paste the weather dataset by right clicking on terminal

Ctrl d will save the file

Run command ls to see the file.

Create java file

Command: cat >>MyMaxMin.java

Paste the java code and ctrl d to save the file

export HADOOP\_CLASSPATH=$(hadoop classpath) ////compile and to create jar file

mkdir classes

javac -classpath ${HADOOP\_CLASSPATH} -d classes MyMaxMin.java

After compile need to create a jar file

jar -cvf MyMaxMin.jar -C classes/ .

Now, put weatherin.txt in hdfs

Before that create a folder

Command: hdfs dfs -mkdir /p3input123

Then run command: hdfs dfs -put weatherin2.txt /p3input123

hadoop jar MyMaxMin.jar MyMaxMin /p3inputw /output123

Check outfile is created

Command: hdfs dfs -ls /output123

hdfs dfs -cat /output123/\*

Practical 4 A

Aim: Implement the program using Pig.

Dataset:

Practical 4 A

Aim: Implement the program using Pig.

001,Rajiv,Reddy,21,9848022337,Hyderabad

002,siddarth,Battacharya,22,9848022338,Kolkata

003,Rajesh,Khanna,22,9848022339,Delhi

004,Preethi,Agarwal,21,9848022330,Pune

005,Trupthi,Mohanthy,23,9848022336,Bhuwaneshwar

006,Archana,Mishra,23,9848022335,Chennai

007,Komal,Nayak,24,9848022334,trivendram

008,Bharathi,Nambiayar,24,9848022333,Chennai

#student.txtcreate a directory and get into that directory

Command: mkdir p5mscit

Create a file

Command: cat >>student.txt

Right click and paste the text

Remove the space with vi editor

Command: vi student.txt and press i for insert mode

After editing: wq and enter

Create a program file

/script start

student = LOAD 'student.txt' USING PigStorage(',')

as (id:int, firstname:chararray, lastname:chararray, age:int, phone:chararray, city:chararray);

student\_order = ORDER student BY age DESC;student\_limit = LIMIT

student\_order 4;Dump student\_limit;

////////script end

Upload student on hdfs

Command: hdfs dfs -put student.txt /user/root/

Run the pig program

pig program.pig

Practical 5

Aim: Implement the application in Hive.

Dataset:

001,Rajiv,Reddy,21,9848022337,Hyderabad

002,siddarth,Battacharya,22,9848022338,Kolkata

003,Rajesh,Khanna,22,9848022339,Delhi

004,Preethi,Agarwal,21,9848022330,Pune

005,Trupthi,Mohanthy,23,9848022336,Bhuwaneshwar

006,Archana,Mishra,23,9848022335,Chennai

007,Komal,Nayak,24,9848022334,trivendram

008,Bharathi,Nambiayar,24,9848022333,Chennai

#student.txtcreate a directory and get into that directory

Command: mkdir p6mscit

Create a file

Command: cat >>data.txt

Right click and paste the text

Remove the space with vi editor Command: vi student.txt and press i for insert mode

After editing: wq and enter

Print the content and see the text

Now start the hive terminal

Command: hive

Copy paste below command on hive and entercreate table

CREATE TABLE IF NOT EXISTS employee ( eid int, fname String,

lname String, age int, contact String, city String)

COMMENT 'Employee details'

ROW FORMAT DELIMITED

FIELDS TERMINATED BY ','

LINES TERMINATED BY '\n'

STORED AS TEXTFILE;

Run command: LOAD DATA LOCAL INPATH 'data.txt' OVERWRITE INTO TABLE

employee;

Run the command like select \* from employee;

Practical 6

Aim: Implement an application that stores big data in HBase/ Python

What is HBase?

HBase is a distributed column-oriented database built on top of the Hadoop file system. It is an open-source project and is horizontally scalable. It is a part of the Hadoop ecosystem that provides random real-time read/write access to data in the Hadoop File System. Go to GUI page and start the HBase service.

Click on OK to start the service.

Now we must start region server.

Check zooperkeeper server is started.

Check hbase and region server are started.

Command: whichapplication-name gives directory in which application-name is installed.

Open the shell

192.168.119.132:4200

Command: hbase shell

It will start the server

Enter the command create 'test','cf' and it will create the tab

Check the table is created with command List

It will list all the tables created.

If we want to see column description of a table.

Command- describe tablename

--describe 'test'

Now, we have to put the values in table

Values:

put 'test',' row1', 'cf:a', 'value1'

put 'test',' row2', 'cf:b', 'value2'

put 'test',' row3', 'cf:c', 'value2'

copy paste the data in shell.

We to display the records of table

Command: scan 'test'

Python: storage/retrieval

Start the service with command

Hbase thrift start -p 9090 inforport 9095

Create the table the way we did it in hbase and see the records using scan

command

Create a program file

Import happybase as hb

conn = hb.connection('192.168.119.132',9090)

print(conn.table('test').row('row1')

print(conn.table('test').row('row2')

print(conn.table('test').row('row3')

print(conn.table('test').row('row4')

table = conn.table('test')

table.put(b'row5',{b'cf:r':b'value5'})

print(conn.table('test').row('row5))

Run a scan command on shell to display the values

Now, try with duplicate value at row 5 say value t

Practical 7

Implement Decision tree classification techniquesDecision Trees (DTs) are a non-parametric supervised learning method used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features. A tree can be seen as a piecewise constant approximation.Using the Iris dataset, we can construct a tree as follows:

from sklearn.datasets import load\_iris

from sklearn import tree

iris = load\_iris()

X,y = iris.data,iris.target

clf = DecisionTreeClassifier()

clf = clf.fit(X,y)

Once trained, we can plot the tree with the plot\_tree function:

tree.plot\_tree(clf)

Practical 8

Implement SVM classification techniques

Support Vector Machines

Generally, Support Vector Machines is considered to be a classification approach, it but can be employed in both types of classification and regression problems. It can easily handle multiple continuous and categorical variables. SVM constructs a hyperplane in multidimensional space to separate different classes. SVM generates optimal hyperplane in an iterative manner, which is used to minimize an error. The core idea of SVM is to find a maximum marginal hyperplane (MMH) that best divides the dataset into classes.Loading data:

from sklearn import datasets

cancer = datasets.load\_breast\_cancer()

print("Features" , cancer.feature\_names)

print("Labels: " , cancer.target\_names)

Check the shape of the dataset using shape.

cancer.data.shape

print(cancer.data[0:5])

print(cancer.target)

Splitting Data:

To understand model performance, dividing the dataset into a training set and a test set is a good strategy.

Split the dataset by using the function train\_test\_split(). you need to pass 3 parameters features, target, and test\_set size. Additionally, you can use random\_state to select records randomly.

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(cancer.data, cancer.target,test\_size=0.3,random\_state=109)

Generate Model:

Let's build support vector machine model. First, import the SVM module and create support vector classifier object by passing argument kernel as the linear kernel in SVC() function.

Then, fit your model on train set using fit() and perform prediction on the test set using predict().

from sklearn import svm

clf = svm.SVC(kernel='linear')

clf.fit(X\_train,y\_train)

y\_pred=clf.predict(X\_test)

Evaluating the Model:

Let's estimate how accurately the classifier or model can predict the breast cancer of patients. Accuracy can be computed by comparing actual test set values and predicted values.

from sklearn import metrics

print("Accuracy:,metrics.accuracy\_score(y\_test,y\_pred))

Practical No 9

Aim: To implement REGRESSION MODEL(Linear & Logistical Regression) Software: Python editor

Packages used: numpy, pandas, sklearn

Description:

Linear Regression is a statistical method used to model the relationship between a dependent variable and one or more independent variables by fitting a straight line to the data. It assumes a linear relationship and is commonly used for predicting continuous values. The model learns by minimizing the difference between predicted and actual values using metrics such as Root Mean Squared Error (RMSE) and Rsquared score (R²), which measure the accuracy and variance explained by the model. Logistic Regression is a classification algorithm used to predict categorical outcomes, typically binary (e.g., yes/no, 0/1). Instead of fitting a straight line, it uses a logistic (sigmoid) function to estimate probabilities, mapping values between 0 and 1. The model predicts class labels based on a threshold, commonly 0.5. It is evaluated using metrics such as accuracy, precision, recall, and F1-score, which assess the model's ability to distinguish between classes effectively

Code:

Linear Regression

import numpy as np

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error, r2\_score

# Generate Sample Data

np.random.seed(42)

X = 2 \* np.random.rand(100, 1)

y = 4 + 3 \* X + np.random.randn(100, 1) # Linear relation with some noise

# Convert to DataFrame

df = pd.DataFrame(np.hstack((X, y)), columns=['Feature', 'Target'])

# Split Data into Training and Testing Sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(df[['Feature']], df['Target'],

test\_size=0.2, random\_state=42)

# Initialize and Train Linear Regression Model

model = LinearRegression()

model.fit(X\_train, y\_train)

# Make Predictions

y\_pred = model.predict(X\_test)

# Evaluate the Model

rmse = np.sqrt(mean\_squared\_error(y\_test, y\_pred))

r2 = r2\_score(y\_test, y\_pred)

# Print Results

print("Coefficients:", model.coef\_)

print("Intercept:", model.intercept\_)

print("Root Mean Squared Error (RMSE):", rmse)

print("R-squared Score (R2):", r2)

Output:

Coefficients: [2.79932366]

Intercept: 4.142913319458566

Root Mean Squared Error (RMSE): 0.8085168605026132

R-squared Score (R2): 0.8072059636181392

Logistical Regression Code

import numpy as np

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LogisticRegression

# Generate Sample Data

np.random.seed(42)

X = 2 \* np.random.rand(100, 1)

y = (X > 1).astype(int).ravel() # Binary classification based on threshold

# Convert to DataFrame

df = pd.DataFrame(np.hstack((X, y.reshape(-1, 1))), columns=['Feature', 'Target']) # Split Data into Training and Testing Sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(df[['Feature']], df['Target'],

test\_size=0.2, random\_state=42)

# Initialize and Train Logistic Regression Model

model = LogisticRegression()

model.fit(X\_train, y\_train)

# Make Predictions

y\_pred = model.predict(X\_test)

# Evaluate the Model

accuracy = accuracy\_score(y\_test, y\_pred)

report = classification\_report(y\_test, y\_pred)

# Print Results

print("Coefficients:", model.coef\_)

print("Intercept:", model.intercept\_)

print("Accuracy:", accuracy)

print("Classification Report:\n", report)

Output

Coefficients: [[3.85477457]]

Intercept: [-3.75556844]

Accuracy: 1.0

Classification Report:

precision recall f1-score support

0.0 1.00 1.00 1.00 11

1.0 1.00 1.00 1.00 9

accuracy 1.00 20

macro avg 1.00 1.00 1.00 20

weighted avg 1.00 1.00 1.00 20

Practical No. 10

Aim: To implement Classification Model

Software: Python editor

Packages used: numpy, pandas, sklearn

Description:

A classification model using the Random Forest Classifier on the Iris dataset is used. It first loads the dataset and converts it into a Pandas DataFrame, where the features represent different measurements of iris flowers, and the target variable indicates the species. The data is split into training and testing sets, with 80% used for training and 20% for evaluation. A Random Forest Classifier with 100 decision trees is trained on the dataset, and predictions are made on the test set. The model's performance is assessed using accuracy, confusion matrix, and classification report, which provide insights into the model's correctness and error distribution.

A classification model is a supervised machine learning model that categorizes input data into predefined classes or labels. It learns patterns from training data and uses them to classify new observations. Classification models are widely used in realworld applications such as spam detection, medical diagnosis, and image recognition.

Code:

Classification Model

import numpy as np

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy\_score, classification\_report,

confusion\_matrix

from sklearn.datasets import load\_iris

# Load Dataset

data = load\_iris()

df = pd.DataFrame(data.data, columns=data.feature\_names)

df['Target'] = data.target

# Split Data into Training and Testing Sets

X = df.drop(columns=['Target'])

y = df['Target']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2,

random\_state=42)

# Initialize and Train Classifier

classifier = RandomForestClassifier(n\_estimators=100, random\_state=42)

classifier.fit(X\_train, y\_train)

# Make Predictions

y\_pred = classifier.predict(X\_test)

# Evaluate the Model

accuracy = accuracy\_score(y\_test, y\_pred)

conf\_matrix = confusion\_matrix(y\_test, y\_pred)

report = classification\_report(y\_test, y\_pred)

# Print Results

print("Accuracy:", accuracy)

print("Confusion Matrix:\n", conf\_matrix)

print("Classification Report:\n", report)

Output:

Accuracy: 1.0

Confusion Matrix:

[[10 0 0]

[ 0 9 0]

[ 0 0 11]]

Classification Report:

precision recall f1-score support

0 1.00 1.00 1.00 10

1 1.00 1.00 1.00 9

2 1.00 1.00 1.00 11

accuracy 1.00 30

macro avg 1.00 1.00 1.00 30

weighted avg 1.00 1.00 1.00 30

Practical No. 11

Aim: To implement Clustering Model

Software: Python editor

Packages used: numpy, pandas, sklearn

Description:

K-Means clustering is applied with Iris dataset, grouping the data into three clusters based on feature similarities. It first loads the dataset and uses K-Means, a widely used unsupervised learning algorithm, to categorize data points into three clusters. To visualize the clusters, Principal Component Analysis (PCA) is used to reduce the dataset to two dimensions. The results are plotted using Seaborn and Matplotlib, where each cluster is represented with a different color. This helps in understanding how the algorithm groups similar data points together based on patterns in the dataset.

A clustering model is an unsupervised learning technique that groups data points into clusters based on their similarities without predefined labels. It is used in applications such as customer segmentation, anomaly detection, and image segmentation. K-Means is a popular clustering algorithm that partitions data into K clusters by minimizing the distance between points and their respective cluster centroids.

Code:

Clustering Model

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.cluster import KMeans

from sklearn.datasets import load\_iris

from sklearn.decomposition import PCA

# Load Dataset

data = load\_iris()

df = pd.DataFrame(data.data, columns=data.feature\_names)

# Apply K-Means Clustering

kmeans = KMeans(n\_clusters=3, random\_state=42, n\_init=10)

kmeans.fit(df)

df['Cluster'] = kmeans.labels\_

# Reduce Dimensions for Visualization

pca = PCA(n\_components=2)

X\_pca = pca.fit\_transform(df[data.feature\_names])

df['PCA1'] = X\_pca[:, 0]

df['PCA2'] = X\_pca[:, 1]

# Plot Clusters

plt.figure(figsize=(8, 6))

sns.scatterplot(x=df['PCA1'], y=df['PCA2'], hue=df['Cluster'], palette='viridis', s=100, alpha=0.7)

plt.title("K-Means Clustering on Iris Dataset")

plt.xlabel("Principal Component 1")

plt.ylabel("Principal Component 2")

plt.legend(title="Cluster")

plt.show()