1. DATA ANALYTICS 1

import pandas as pd import numpy as np

from sklearn import metrics import matplotlib.pyplot as plt import seaborn as sns

%matplotlib inline

# Importing the Boston Housing dataset from sklearn.datasets import load\_boston boston = load\_boston()

# Initializing the dataframe

data = pd.DataFrame(boston.data)

# See head of the dataset data.head()

#Adding the feature names to the dataframe data.columns = boston.feature\_names data.head()

#Adding target variable to dataframe data['PRICE'] = boston.target

# Median value of owner-occupied homes in $1000s

#Check the shape of dataframe data.shape

data.columns data.dtypes

# Identifying the unique number of values in the dataset data.nunique()

# Check for missing values data.isnull().sum()

# See rows with missing values data[data.isnull().any(axis=1)]

# Viewing the data statistics data.describe()

# Finding out the correlation between the features corr = data.corr()

corr.shape

# Plotting the heatmap of correlation between features plt.figure(figsize=(20,20))

sns.heatmap(corr, cbar=True, square= True, fmt='.1f', annot=True, annot\_kws={'size':15}, cmap='Greens')

# Spliting target variable and independent variables X = data.drop(['PRICE'], axis = 1)

y = data['PRICE']

# Splitting to training and testing data

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

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X,y, test\_size = 0.3, random\_state = 4)

# Import library for Linear Regression

from sklearn.linear\_model import LinearRegression

# Create a Linear regressor lm = LinearRegression()

# Train the model using the training sets lm.fit(X\_train, y\_train)

# Value of y intercept lm.intercept\_

#Converting the coefficient values to a dataframe coeffcients = pd.DataFrame([X\_train.columns,lm.coef\_]).T

coeffcients = coeffcients.rename(columns={0: 'Attribute', 1: 'Coefficients'}) coeffcients

# Model prediction on train data y\_pred = lm.predict(X\_train)

# Model Evaluation print('R^2:',metrics.r2\_score(y\_train, y\_pred)) print('Adjusted R^2:',1 - (1-metrics.r2\_score(y\_train,

y\_pred))\*(len(y\_train)-1)/(len(y\_train)-X\_train.shape[1]-1)) print('MAE:',metrics.mean\_absolute\_error(y\_train, y\_pred)) print('MSE:',metrics.mean\_squared\_error(y\_train, y\_pred)) print('RMSE:',np.sqrt(metrics.mean\_squared\_error(y\_train, y\_pred)))

# Visualizing the differences between actual prices and predicted values plt.scatter(y\_train, y\_pred)

plt.xlabel("Prices") plt.ylabel("Predicted prices") plt.title("Prices vs Predicted prices") plt.show()

# Checking residuals plt.scatter(y\_pred,y\_train-y\_pred) plt.title("Predicted vs residuals") plt.xlabel("Predicted") plt.ylabel("Residuals")

plt.show()

# Checking Normality of errors sns.distplot(y\_train-y\_pred)

plt.title("Histogram of Residuals") plt.xlabel("Residuals") plt.ylabel("Frequency") plt.show()

# Predicting Test data with the model y\_test\_pred = lm.predict(X\_test)

# Model Evaluation

acc\_linreg = metrics.r2\_score(y\_test, y\_test\_pred) print('R^2:', acc\_linreg)

print('Adjusted R^2:',1 - (1-metrics.r2\_score(y\_test, y\_test\_pred))\*(len(y\_test)-1)/(len(y\_test)-X\_test.shape[1]-1))

print('MAE:',metrics.mean\_absolute\_error(y\_test, y\_test\_pred)) print('MSE:',metrics.mean\_squared\_error(y\_test, y\_test\_pred)) print('RMSE:',np.sqrt(metrics.mean\_squared\_error(y\_test, y\_test\_pred)))

models = pd.DataFrame({ 'Model': ['Linear Regression'],

'R-squared Score': [acc\_linreg\*100,]}) models.sort\_values(by='R-squared Score', ascending=False)

#### DATA ANALYTICS 2

# This Python 3 environment comes with many helpful analytics libraries installed

# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python # For example, here's several helpful packages to load

import numpy as np # linear algebra

import pandas as pd # data processing, CSV file I/O (e.g. pd.read\_csv)

import pandas as pd import numpy as np

import matplotlib.pyplot as plt

%matplotlib inline

df = pd.read\_csv('Social\_Network\_Ads.csv') df.head(2)

df['Purchased'].value\_counts()

df = df.drop('User ID', axis = 1)

df.info()

from sklearn.preprocessing import OneHotEncoder

df\_onehot = pd.get\_dummies(df, columns=['Gender'], prefix = ['Gender'])

df\_onehot.head(10)

from sklearn.preprocessing import MinMaxScaler

scaler=MinMaxScaler(feature\_range=(0,1)) df\_onehot["EstimatedSalary"]=scaler.fit\_transform(df\_onehot[["EstimatedSalary"]])

Column\_loc = ['Age','EstimatedSalary','Gender\_Female','Gender\_Male','Purchased'] df\_onehot = df\_onehot[Column\_loc]

df\_onehot.head(2)

X = df\_onehot.iloc[:,:-1].values y = df\_onehot.iloc[:,-1].values

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

train\_X, test\_X, train\_y, test\_y = train\_test\_split(X, y, test\_size= 0.25, random\_state = 4) print(train\_X.shape,train\_y.shape,test\_X.shape,test\_y.shape)

from sklearn.linear\_model import LogisticRegression lr = LogisticRegression()

lr.fit(train\_X,train\_y)

predict\_y = lr.predict(test\_X)

predict\_y

from sklearn.metrics import accuracy\_score accuracy\_score(test\_y, predict\_y)

from sklearn.metrics import confusion\_matrix, roc\_auc\_score, precision\_score, recall\_score print(roc\_auc\_score(test\_y, predict\_y))

print(precision\_score(test\_y, predict\_y)) print(recall\_score(test\_y, predict\_y))

print(confusion\_matrix(test\_y, predict\_y))

tp,fp, fn, tn = confusion\_matrix(test\_y, predict\_y).ravel() print(f'Correctly Predicted made Purchase {tp}') print(f'Falsely Predicted made Purchase {fp}')

print(f'Falsely Predicted made did NOT made Purchase {fn}') print(f'Correctly Predicted made did NOT made Purchase {tn}')

print('Area under ROC curve is',roc\_auc\_score(test\_y, lr.predict\_proba(test\_X)[:,1]))

accuracy\_score(test\_y, predict\_y)

from sklearn.metrics import plot\_roc\_curve plot\_roc\_curve(lr, test\_X, test\_y);

Result\_Test\_X = pd.DataFrame(test\_X, columns

=['Age','EstimatedSalary','Gender\_Female','Gender\_Male']) Result\_Predict\_y = pd.DataFrame(predict\_y, columns= ['Pred\_Purchase']) pd.concat([Result\_Test\_X, Result\_Predict\_y], axis=1)

Result\_test\_y = pd.DataFrame(test\_y, columns= ['Purchased'])

Result = pd.concat([Result\_Test\_X, Result\_test\_y, Result\_Predict\_y], axis=1) Result

#### DATA ANALYTICS -3 **DONE**

# This Python 3 environment comes with many helpful analytics libraries installed

# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python # For example, here's several helpful packages to load

import numpy as np # linear algebra

import pandas as pd # data processing, CSV file I/O (e.g. pd.read\_csv) import matplotlib.pyplot as plt

import seaborn as sns

iris=pd.read\_csv('Iris.csv')

iris.head()

iris['Species'].unique()

iris.describe(include='all')

iris.info()

X=iris.iloc[:,0:4].values y=iris.iloc[:,4].values

from sklearn.preprocessing import LabelEncoder le = LabelEncoder()

y = le.fit\_transform(y)

#Metrics

from sklearn.model\_selection import KFold,train\_test\_split,cross\_val\_score from sklearn.metrics import make\_scorer, accuracy\_score,precision\_score from sklearn.metrics import classification\_report

from sklearn.metrics import confusion\_matrix

from sklearn.metrics import accuracy\_score ,precision\_score,recall\_score,f1\_score

#Model

from sklearn.naive\_bayes import GaussianNB

#Train and Test split X\_train,X\_test,y\_train,y\_test=train\_test\_split(X,y,test\_size=0.3,random\_state=0)

gaussian = GaussianNB() gaussian.fit(X\_train, y\_train) Y\_pred = gaussian.predict(X\_test)

accuracy\_nb=round(accuracy\_score(y\_test,Y\_pred)\* 100, 2) acc\_gaussian = round(gaussian.score(X\_train, y\_train) \* 100, 2)

cm = confusion\_matrix(y\_test, Y\_pred) accuracy = accuracy\_score(y\_test,Y\_pred)

precision =precision\_score(y\_test, Y\_pred,average='micro') recall = recall\_score(y\_test, Y\_pred,average='micro')

f1 = f1\_score(y\_test,Y\_pred,average='micro') print('Confusion matrix for Naive Bayes\n',cm) print('accuracy\_Naive Bayes: %.3f' %accuracy) print('precision\_Naive Bayes: %.3f' %precision) print('recall\_Naive Bayes: %.3f' %recall) print('f1-score\_Naive Bayes : %.3f' %f1)

#### 8,9. DATA VISUALISATION 1 & 2 **DONE**

import numpy as np import pandas as pd

from pandas import Series,DataFrame import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.impute import SimpleImputer

import warnings warnings.filterwarnings('ignore')

%matplotlib inline

gender\_submission = pd.read\_csv("gender\_submission.csv") test = pd.read\_csv("test.csv")

train = pd.read\_csv("train.csv")

print('Test data shape: ', train.shape) train.head()

train.describe()

train.info()

train.isnull().sum() test.isnull().sum()

imputer = SimpleImputer(np.nan, "mean")

train['Age'] = imputer.fit\_transform(np.array(train['Age']).reshape(891, 1)) train.Embarked.fillna(method='ffill', inplace=True) train.drop(['PassengerId', 'Name','Ticket'], axis=1, inplace=True) train.head()

test['Age'] = imputer.fit\_transform(np.array(test['Age']).reshape(418, 1)) test.Embarked.fillna(method='ffill', inplace=True)

test.Fare.fillna(method='ffill', inplace=True) test.drop(['Name', 'Ticket'], axis=1, inplace=True) test.head()

train['Survived'].value\_counts()

plt.figure(figsize=[10,5])

sns.countplot(x = 'Sex', hue = 'Survived', data = train) plt.xticks(rotation = 20);

sns.barplot(x='Sex', y='Survived', data=train, palette=('RdPu'));

print('% of survived females:', train['Survived'][train['Sex'] == 'female'].value\_counts(normalize = True)[1]\*100)

print('% of survived males:', train['Survived'][train['Sex'] == 'male'].value\_counts(normalize = True)[1]\*100)

fig, ax = plt.subplots(figsize = (10,6))

ax = sns.countplot(x = 'Survived', hue = 'Pclass', data = train, palette = 'YlOrRd') ax.set\_xlabel('Survived')

ax.set\_title('Survival Rate for Passenger Classes', fontsize = 14, fontweight='bold');

ax = sns.catplot(x="Pclass", hue="Sex", col="Survived", data=train, kind="count",

height=4, aspect=.7, palette = 'OrRd');

sns.countplot(x = "Pclass", hue = "Survived", data = train, palette = 'RdPu');

sns.barplot(x="Pclass", y="Survived", data= train, palette = 'BuGn');

perc = train[['Pclass', 'Survived']].groupby(['Pclass'], as\_index=False).mean().sort\_values(by='Survived', ascending=False)

perc\*100

sns.factorplot(x='Pclass', y='Survived', hue = 'Sex', data = train, palette = 'PRGn');

sns.boxplot(x='Sex', y='Age', hue = 'Survived',data=train);

grid = sns.FacetGrid(train, col='Survived') grid.map(plt.hist, 'Age', bins=25, color = 'y').add\_legend(); sns.set(style="ticks", color\_codes=True);

plt.figure(figsize=(10,5)) sns.distplot(train['Age'], bins=24, color='g');

avg\_age\_train = train ["Age"].mean() std\_age\_train = train ["Age"].std() avg\_age\_test = test["Age"].mean() std\_age\_test = test ["Age"].std()

bins = [0, 1, 12, 18, 21, 60, np.inf]

labels = ['Infant', 'Child', 'Teenager',' Young Adult', 'Adult', 'Senior'] train['AgeGroup'] = pd.cut(train["Age"], bins, labels = labels) test['AgeGroup'] = pd.cut(test["Age"], bins, labels = labels)

sns.barplot(x="AgeGroup", y="Survived", data= train) plt.show;

train.head()

# convert from float to int train['Fare'] = train['Fare'].astype(int) test['Fare']= test['Fare'].astype(int)

# get fare for survived & didn't survive passengers fare\_not\_survived = train["Fare"][train["Survived"] == 0] fare\_survived = train["Fare"][train["Survived"] == 1]

# get average and std for fare of survived/not survived passengers avgerage\_fare = DataFrame([fare\_not\_survived.mean(), fare\_survived.mean()]) std\_fare = DataFrame([fare\_not\_survived.std(), fare\_survived.std()])

# plot

train['Fare'].plot(kind='hist', figsize=(15,3),bins=100, xlim=(0,50), color = 'purple')

avgerage\_fare.index.names = std\_fare.index.names = ["Survived"] avgerage\_fare.plot(yerr=std\_fare,kind='hist',legend=False, color = 'r');

#### 10. DATA VISUALISATION 3 **DONE**

import numpy as np

import pandas as pd

df = pd.read\_csv("iris-flower-dataset.csv",header=None) df.columns = ["col1","col2","col3","col4","col5"]

df.head()

column = len(list(df)) column

df.info()

df.describe()

import seaborn as sns import matplotlib

import matplotlib.pyplot as plt

%matplotlib inline

fig, axes = plt.subplots(2, 2, figsize=(16, 8))

axes[0,0].set\_title("Distribution of First Column") axes[0,0].hist(df["col1"]);

axes[0,1].set\_title("Distribution of Second Column")

axes[0,1].hist(df["col2"]);

axes[1,0].set\_title("Distribution of Third Column") axes[1,0].hist(df["col3"]);

axes[1,1].set\_title("Distribution of Fourth Column") axes[1,1].hist(df["col4"]);

data\_to\_plot = [df["col1"],df["col2"],df["col3"],df["col4"]]

sns.set\_style("whitegrid")

# Creating a figure instance

fig = plt.figure(1, figsize=(12,8))

# Creating an axes instance ax = fig.add\_subplot(111)

# Creating the boxplot

bp = ax.boxplot(data\_to\_plot);

# If we observe closely. for the box 2, interquartile distance is roughly around 0.75 hence the values lying beyond this range of (third quartile + interquartile distance)

# i.e. roughly around 4.05 will be considered as outliers. Similarly outliers with other boxplots can be found.

## 3. Descriptive Statistics- Measures of central tendency and variability

import pandas as pd import numpy as np *# Load the data*

*# Download the Iris.csv file from https://*[*www.kaggle.com/datasets/saurabh00007/iriscsv*](http://www.kaggle.com/datasets/saurabh00007/iriscsv) *# keep the .csv file in the same folder.*

df = pd.read\_csv("Iris.csv") print(df.shape) print(df.info())

*# 50th Percentile*

def q50(x):

return x.quantile(0.5)

*# 90th Percentile*

def q90(x):

return x.quantile(0.9)

*## Solution For Problem Statement 1*

print("\n Problem Solution 1: ")

print( "\nProviding summary statistics for Species(categorical variable) groupedBy Sepal-Length(quantitative variable)\n") print(df.groupby(['Species'])[['SepalLengthCm']].agg(['mean','median','min','ma x','std',q50,q90]))

print( "\nProviding summary statistics for Species(categorical variable) groupedBy Sepal-Width(quantitative variable)") print(df.groupby(['Species'])[['SepalWidthCm']].agg(['mean','median','min','max ','std',q50,q90]))

print( "\nProviding summary statistics for Species(categorical variable) groupedBy Petal-Length(quantitative variable)") print(df.groupby(['Species'])[['PetalLengthCm']].agg(['mean','median','min','ma x','std',q50,q90]))

print( "\nProviding summary statistics for Species(categorical variable) groupedBy Petal-Width(quantitative variable)") print(df.groupby(['Species'])[['PetalWidthCm']].agg(['mean','median','min','max ','std',q50,q90]))

*## Solution For Problem Statement 2*

print("\n Problem Solution 2: ")

print("\n Mean of dataset groupedby Species") print(df.groupby(['Species'])[['SepalLengthCm','SepalWidthCm','PetalLengthCm',' PetalWidthCm']].mean())

print("\n Median of dataset groupedby Species") print(df.groupby(['Species'])[['SepalLengthCm','SepalWidthCm','PetalLengthCm',' PetalWidthCm']].median())

print("\n50 Percentile result for 'Species' in Iris dataframe") print(df.groupby(['Species'])[['SepalLengthCm','SepalWidthCm','PetalLengthCm',' PetalWidthCm']].quantile(0.5))

print("\n90 Percentile result for 'Species' in Iris dataframe") print(df.groupby(['Species'])[['SepalLengthCm','SepalWidthCm','PetalLengthCm',' PetalWidthCm']].quantile(0.9))

print("\n Standard Deviation of dataset groupedby Species") print(df.groupby(['Species'])[['SepalLengthCm','SepalWidthCm','PetalLengthCm',' PetalWidthCm']].std())

## Data Wrangling I :-

import pandas as pd import numpy as np

csvfile\_df=pd.read\_csv("titanic.csv") csvfile\_df.head()

csvfile\_df.isnull() csvfile\_df.isnull().sum() print(csvfile\_df.describe()) csvfile\_df.dtypes

del csvfile\_df['Sex'] del csvfile\_df['Ticket'] del csvfile\_df['Cabin']

del csvfile\_df['Embarked']

for column in csvfile\_df.columns: csvfile\_df[column]=csvfile\_df[column]/csvfile\_df[column].abs().max() display(csvfile\_df)

## Data Wrangling II :-

import pandas as pd import numpy as np

from sklearn.datasets import load\_boston import matplotlib.pyplot as plt df=pd.read\_csv("xAPI-Edu-Data.csv") df.describe()

df.head() df.isnull().sum()

from scipy import stats z=np.abs(stats.zscore(df['VisITedResources']))

print(z)

threshold = 3 #position of outlier

print(np.where(z > 3)) df\_scaled=df.copy()

col\_names=['raisedhands','VisITedResources','AnnouncementsView','Discussion']

features = df\_scaled[col\_names]

from sklearn.preprocessing import MinMaxScaler scaler = MinMaxScaler()

df\_scaled[col\_names] = scaler.fit\_transform(features.values) df\_scaled[col\_names] = scaler.fit\_transform(features.values) df\_scaled

## 7. Text analytics :-

import nltk

sentence\_data = "The First sentence is about Python. The Second: about Django. You can learn Python,Django and Data Ananlysis here."

nltk\_tokens = nltk.sent\_tokenize(sentence\_data) print(nltk\_tokens)

nltk\_tokens1 = nltk.word\_tokenize(sentence\_data) print(nltk\_tokens1)

import nltk

from nltk.corpus import stopwords

from nltk.tokenize import word\_tokenize, sent\_tokenize stop\_words = set(stopwords.words('english'))

#Dummy text

txt = "Sukanya, Rajib and Naba are my good friends. "\ "Sukanya is getting married next year. " \

"Marriage is a big step in one’s life." \ "It is both exciting and frightening. " \

"But friendship is a sacred bond between people." \ "It is a special kind of love between us. " \

"Many of you must have tried searching for a friend "\ "but never found the right one."

# sent\_tokenize is one of instances of

# PunktSentenceTokenizer from the nltk.tokenize.punkt module

tokenized = sent\_tokenize(txt) for i in tokenized:

# Word tokenizers is used to find the words # and punctuation in a string

wordsList = nltk.word\_tokenize(i)

# removing stop words from wordList

wordsList = [w for w in wordsList if not w in stop\_words]

# Using a Tagger. Which is part-of-speech # tagger or POS-tagger.

tagged = nltk.pos\_tag(wordsList) print(tagged)

from nltk.corpus import stopwords

from nltk.tokenize import word\_tokenize

example\_sent = """This is a sample sentence,showing off the stop words filtration.""" stop\_words = set(stopwords.words('english'))

word\_tokens = word\_tokenize(example\_sent)

filtered\_sentence = [w for w in word\_tokens if not w.lower() in stop\_words] filtered\_sentence = []

for w in word\_tokens:

if w not in stop\_words: filtered\_sentence.append(w)

print(word\_tokens) print(filtered\_sentence)

# import these modules

from nltk.stem import PorterStemmer from nltk.tokenize import word\_tokenize

ps = PorterStemmer()

# choose some words to be stemmed

words = ["program", "programs", "programmer", "programming", "programmers"]

for w in words:

print(w, " : ", ps.stem(w))

# import these modules

from nltk.stem import WordNetLemmatizer lemmatizer = WordNetLemmatizer()

print("rocks :", lemmatizer.lemmatize("rocks")) print("corpora :", lemmatizer.lemmatize("corpora"))

# a denotes adjective in "pos"

print("better :", lemmatizer.lemmatize("better", pos ="a"))

# import required module

from sklearn.feature\_extraction.text import TfidfVectorizer

# assign documents d0 = 'Geeks for geeks' d1 = 'Geeks'

d2 = 'r2j'

# merge documents into a single corpus string = [d0, d1, d2]

# create object

tfidf = TfidfVectorizer()

# get tf-df values

result = tfidf.fit\_transform(string)

# get idf values print('\nidf values:')

for ele1, ele2 in zip(tfidf.get\_feature\_names(), tfidf.idf\_): print(ele1, ':', ele2)

# get indexing print('\nWord indexes:') print(tfidf.vocabulary\_)

# display tf-idf values print('\ntf-idf value:') print(result)

# in matrix form

print('\ntf-idf values in matrix form:') print(result.toarray())

## Group B

1. Java Count

# // WC\_Runner.java

package com.wc;

import java.io.IOException; import org.apache.hadoop.fs.Path; import org.apache.hadoop.io.IntWritable; import org.apache.hadoop.io.Text; import org.apache.hadoop.mapred.FileInputFormat; import org.apache.hadoop.mapred.FileOutputFormat; import org.apache.hadoop.mapred.JobClient; import org.apache.hadoop.mapred.JobConf; import org.apache.hadoop.mapred.TextInputFormat; import org.apache.hadoop.mapred.TextOutputFormat;

public class WC\_Runner { public static void main(String[] args) throws IOException { JobConf conf = new JobConf(WC\_Runner.class); conf.setJobName("WordCount"); conf.setOutputKeyClass(Text.class); conf.setOutputValueClass(IntWritable.class); conf.setMapperClass(WC\_Mapper.class); conf.setCombinerClass(WC\_Reducer.class); conf.setReducerClass(WC\_Reducer.class); conf.setInputFormat(TextInputFormat.class); conf.setOutputFormat(TextOutputFormat.class); FileInputFormat.setInputPaths(conf,new Path(args[0]));

FileOutputFormat.setOutputPath(conf,new Path(args[1])); JobClient.runJob(conf);

}

}

# // WC\_Mapper.java

package com.wc;

import java.io.IOException; import java.util.StringTokenizer; import org.apache.hadoop.io.IntWritable; import org.apache.hadoop.io.LongWritable; import org.apache.hadoop.io.Text; import org.apache.hadoop.mapred.MapReduceBase; import org.apache.hadoop.mapred.Mapper; import org.apache.hadoop.mapred.OutputCollector; import org.apache.hadoop.mapred.Reporter;

### public class WC\_Mapper extends MapReduceBase implements

Mapper<LongWritable,Text,Text,IntWritable>{ private final static IntWritable one = new IntWritable(1); private Text word = new Text();

### public void map(

LongWritable key, Text value,

OutputCollector<Text,IntWritable> output, Reporter reporter

) throws IOException {

String line = value.toString();

StringTokenizer tokenizer = new StringTokenizer(line);

while (tokenizer.hasMoreTokens()){ word.set(tokenizer.nextToken()); output.collect(word, one);

}

}

}

# // WC\_Reducer.java

package com.wc;

import java.io.IOException; import java.util.Iterator; import org.apache.hadoop.io.IntWritable; import org.apache.hadoop.io.Text; import org.apache.hadoop.mapred.MapReduceBase; import org.apache.hadoop.mapred.OutputCollector; import org.apache.hadoop.mapred.Reducer; import org.apache.hadoop.mapred.Reporter;

### public class WC\_Reducer extends MapReduceBase implements

Reducer<Text,IntWritable,Text,IntWritable> { public void reduce( Text key,

Iterator<IntWritable> values, OutputCollector<Text,IntWritable> output, Reporter reporter

) throws IOException { int sum=0; while (values.hasNext()) { sum += values.next().get();

} output.collect(key,new IntWritable(sum)); }

}

Input:

HDFS is a storage unit of Hadoop MapReduce is a processing tool for Hadoop

Output:

HDFS 1

Hadoop 2

MapReduce 1

a 2

for 1

is 2

of 1

processing 1

storage 1

tool 1

unit 1

1. Log file of System using Map-Reduce

// SalesCountryRunner.java package SalesCountry;

import org.apache.hadoop.fs.Path; import org.apache.hadoop.io.\*; import org.apache.hadoop.mapred.\*;

public class SalesCountryRunner

{ public static void main(String[] args) {

JobClient my\_client = new JobClient();

// Create a configuration object for the job

JobConf job\_conf = new JobConf(SalesCountryDriver.class);

// Set a name of the Job job\_conf.setJobName("SalePerCountry");

// Specify data type of output key and value job\_conf.setOutputKeyClass(Text.class); job\_conf.setOutputValueClass(IntWritable.class);

// Specify names of Mapper and Reducer Class job\_conf.setMapperClass(SalesCountry.SalesMapper.class); job\_conf.setReducerClass(SalesCountry.SalesCountryReducer.class);

// Specify formats of the data type of Input and output job\_conf.setInputFormat(TextInputFormat.class); job\_conf.setOutputFormat(TextOutputFormat.class);

// Set input and output directories using command line arguments,

//arg[0] = name of input directory on HDFS, and arg[1] = name of output

directory to be created to store the output file.

FileInputFormat.setInputPaths(job\_conf, new Path(args[0])); FileOutputFormat.setOutputPath(job\_conf, new Path(args[1]));

my\_client.setConf(job\_c onf); try {

// Run the job JobClient.runJob(job\_conf);

} catch (Exception e) { e.printStackTrace();

}

}

}

// SalesMapper.java package SalesCountry; import java.io.IOException;

import org.apache.hadoop.io.IntWritable

; import org.apache.hadoop.io.LongWritabl e; import org.apache.hadoop.io.Text; import org.apache.hadoop.mapred.\*;

public class SalesMapper extends MapReduceBase implements Mapper<LongWritable, Text, Text, IntWritable> { private final static IntWritable one = new IntWritable(1);

public void map(LongWritable key, Text value, OutputCollector<Text, IntWritable>

output, Reporter reporter) throws IOException {

String valueString = value.toString(); String[] SingleCountryData = valueString.split(","); output.collect(new Text(SingleCountryData[7]), one);

}

}

// SalesCountryReducer.java package SalesCountry;

import java.io.IOException; import java.util.\*;

import org.apache.hadoop.io.IntWritabl e; import org.apache.hadoop.io.Text; import org.apache.hadoop.mapred.\*;

public class SalesCountryReducer extends MapReduceBase implements Reducer<Text, IntWritable, Text, IntWritable> {

public void reduce(Text t\_key, Iterator<IntWritable> values,

OutputCollector<Text,IntWritable> output, Reporter reporter) throws IOException { Text key = t\_key; int frequencyForCountry = 0; while (values.hasNext()) {

// replace type of value with the actual type of our value IntWritable value = (IntWritable) values.next(); frequencyForCountry += value.get();

} output.collect(key, new IntWritable(frequencyForCountry));

}

}

Output:

|  |  |
| --- | --- |
| Argentina | 1 |
| Australia | 38 |
| Austria | 7 |
| Bahrain | 1 |
| Belgium | 8 |
| Bermuda  Brazil5 | 1 |
| Bulgaria | 1 |
| CO 1  Canada76 |  |
| Cayman Isls | 1 |
| China 1 |  |
| Costa Rica | 1 |
| Country | 1 |

Czech Republic 3

Denmark 15

Dominican Republic 1

Finland 2

France27

Germany 25

Greece1

Guatemala 1

Hong Kong 1

Hungary 3

Iceland 1

India 2

|  |  |
| --- | --- |
| Ireland  Israel1 | 49 |
| Italy 15 |  |
| Japan 2 Jersey1 Kuwait1 Latvia1 |  |
| Luxembourg | 1 |
| Malaysia | 1 |
| Malta 2 |  |
| Mauritius | 1 |
| Moldova Monaco2 | 1 |
| Netherlands | 22 |
| New Zealand Norway16 | 6 |
| Philippines Poland2 | 2 |
| Romania Russia1 | 1 |
| South Africa | 5 |
| South Korea | 1 |
| Spain 12 Sweden13 |  |
| Switzerland | 36 |
| Thailand | 2 |
| The Bahamas Turkey6 | 2 |

Ukraine 1

United Arab Emirates 6

United Kingdom 100

United States462

1. Weather Data

##### // MaxTemperatureDriver.java

**package MaxMinTemp;**

**import org.apache.hadoop.conf.Configured; 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.lib.input.FileInputFormat; import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat; import org.apache.hadoop.util.Tool; import org.apache.hadoop.util.ToolRunner;**

**public class MaxTemperatureDriver extends Configured implements Tool{ public int run(String[] args) throws Exception { if(args.length !=2) {**

**System.err.println("Usage: MaxTemperatureDriver <input path>**

**<outputpath>");**

**System.exit(-1)**

**; }**

**Job job = new Job(); job.setJarByClass(MaxTemperatureDriver.class); job.setJobName("Max Temperature"); FileInputFormat.addInputPath(job, new Path(args[0])); FileOutputFormat.setOutputPath(job,new Path(args[1])); job.setMapperClass(MaxTemperatureMapper.class); job.setReducerClass(MaxTemperatureReducer.class); job.setOutputKeyClass(Text.class); job.setOutputValueClass(IntWritable.class); System.exit(job.waitForCompletion(true) ? 0:1);**

**boolean success = job.waitForCompletion(true); return success ? 0 : 1;**

**}**

**public static void main(String[] args) throws Exception { MaxTemperatureDriver driver = new MaxTemperatureDriver(); int exitCode = ToolRunner.run(driver, args); System.exit(exitCode);**

**}**

**}**

##### // MaxTemperatureMapper.java

**package MaxMinTemp;**

**import java.io.IOException; import org.apache.hadoop.io.IntWritable; import org.apache.hadoop.io.LongWritable; import org.apache.hadoop.io.Text; import org.apache.hadoop.mapreduce.Mapper;**

**public class MaxTemperatureMapper extends Mapper<LongWritable, Text, Text,**

**IntWritable> { private static final int MISSING = 9999;**

**@Override**

**public void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {**

**String line = value.toString(); String year = line.substring(15, 19); int airTemperature; if (line.charAt(87) == '+') { // parseInt doesn't like leading plus signs airTemperature = Integer.parseInt(line.substring(88, 92));**

**} else {**

**airTemperature = Integer.parseInt(line.substring(87, 92));**

**}**

**String quality = line.substring(92, 93); if (airTemperature != MISSING && quality.matches("[01459]")) { context.write(new Text(year), new IntWritable(airTemperature));**

**}**

**}**

**}**

##### // MaxTemperatureReducer.java

**package MaxMinTemp;**

**import java.io.IOException; import org.apache.hadoop.io.IntWritable; import org.apache.hadoop.io.Text; import org.apache.hadoop.mapreduce.Reducer;**

**public class MaxTemperatureReducer extends Reducer<Text, IntWritable,**

**Text, IntWritable> { @Override**

**public void reduce(Text key, Iterable<IntWritable> values, Context context)**

**throws IOException, InterruptedException { int maxValue = Integer.MIN\_VALUE; for (IntWritable value : values) { maxValue = Math.max(maxValue, value.get());**

**} context.write(key, new IntWritable(maxValue)); }**

|  |  |  |
| --- | --- | --- |
| **} OUTPUT:** |  | |
|  | ***1901*** | **317** |
|  | ***1902*** | **244** |
|  | ***1903*** | **289** |
|  | ***1904*** | **256** |
|  | ***1905*** | **283** |
|  | ***1906*** | **294** |
|  | ***1907*** | **283** |
|  | ***1908*** | **289** |
|  | ***1909*** | **278** |
|  | ***1910*** | **294** |
|  | ***1911*** | **306** |
|  | ***1912*** | **322** |

|  |  |
| --- | --- |
| ***1913*** | **300** |
| ***1914*** | **333** |
| ***1915*** | **294** |
| ***1916*** | **278** |
| ***1917*** | **317** |
| ***1918*** | **322** |
| ***1919*** | **378** |
| ***1920*** | **294** |

## Group C

Case Study/Mini\_project 4.

**import** pandas **as** pd

*# Load the data*

*# Download the 'covid\_vaccine\_statewise.csv' file from https://*[*www.kaggle.com/datasets/sudalairajkumar/covid19-in-india?resource=down*](http://www.kaggle.com/datasets/sudalairajkumar/covid19-in-india?resource=down) *load*

*# keep the .csv file in the same folder.*

df = pd.read\_csv(**"covid\_vaccine\_statewise.csv"**)

*# describing data set* print(df.describe()) print(df.shape) print(df.info())

*# no of person statewise vaccinated for 1st dose in india*

print(**"\n No of person statewise vaccinated for 1st dose in india \n"**) print(df.groupby([**'State'**])[**'First Dose Administered'**].sum())

*# no of person Statewise vaccinated for 2nd dose in india*

print(**"\n No of person statewise vaccinated for 2nd dose in india \n"**) print(df.groupby([**'State'**])[**'Second Dose Administered'**].sum())

*# Total no of males vaccinated*

print(**"\n Total no of males vaccinated\n"**) print(df[**'Male(Individuals Vaccinated)'**].sum())

*# Total no of Females vaccinated*

print(**"\n Total no of females vaccinated\n"**) print(df[**'Female(Individuals Vaccinated)'**].sum())



