CS 5433: Bigdata Management
Programming Assignment 3
Task2 – PREDICTION ALGORITHM

Group 4

Task 2: Implement prediction algorithm using (a) Linear regression (b) Random Forest and Clearly identify the variable you are predicting. Predict on the same variable for both algorithms.

Description of Dataset:

In this task we are using the dataset which is the output of the Task 1 i.e., the dataset which we have obtained after performing the data correction. The dataset consists of 10 columns and 500 records without any null values and out of range values. The columns of the dataset are,

- Institute ID which is a "Double" column
- Name – Name of the university/institute of type "Double"
- City – Name of the city where university is located, which is of type "Double"
- State – Name of the State where university is located, which is of type "Double"
- PR Score – PR Score of the university which is of type "Double"
- PR Rank – PR Rank of the university which is of type "Double"
- PR Score – PR Score of the university which is of type "Double"
- Score – Score of the university which is of type "Double"
- Year –Year (contains values 2017,2018,2019,2020 & 2021) is of type "Double"
- Rank – Rank of the university which is of type "Double"

We are using 5 of these columns for creating the feature set and one column as the variable to be predicted. The **features** that we are using for our prediction are:

- Institute ID of the university/institute
- Name of the university
- State where university is located
- Score of the university
- PR Rank of the university

The variable which we are predicting is "In which year, the universities got it's score and PR Rank".

- Prediction variable column is "Year"

PART B:

In this part, we used Random Forest Regression algorithm to predict the variable.

Random Forest Regression: Random Forest Regression is a supervised learning technique that solves classification or regression issues using an ensemble learning method. Ensemble learning is a machine learning technique that integrates predictions from numerous machine learning algorithms to get a better prediction than a single algorithm. A random forest is an estimator technique that combines the results of several decision trees to produce the best possible result.

Approach:

Below are the steps we have followed to complete Task3 of this assignment,

1. At first, we have created a python file("Assign3_Group4_Task3_PartB.py") in the Hadoop cluster. Refer to "Group_4_Task_3_Part_B_code".

2. **Code Explanation:**

   a. Imported the required libraries

   - from pyspark.sql import SparkSession → This library is imported to create a sparksession. This sparksession can be used to create a dataframe.
   - from pyspark.ml.feature import StringIndexer,VectorAssembler → StringIndexer library is imported for converting the String columns into Double type and VectorAssembler is imported for merging multiple columns into a vector column.
   - from pyspark.ml.feature import MinMaxScaler → By Using column summary statistics, MinMaxScaler rescales each feature to a common range [min, max] linearly.
   - from pyspark.sql.functions import udf – It is imported for creating a user defined function (UDF).
   - from pyspark.sql.types import DoubleType – This library is imported for representing the double precision floats.
   - from pyspark.sql.types import * - It is imported for using the pyspark sql datatypes.
   - from pyspark.ml import Pipeline – Pipeline is imported to run the stages in sequence.
   - from pyspark.ml.functions import vector_to_array - It is imported for Converting a column of MLlib sparse/dense vectors into a column of dense arrays.
   - from pyspark.sql.functions import concat_ws,col – It I imported for concatenating multiple string columns into a single column with a given delimiter.
   - from pyspark.ml.regression import RandomForestRegressor – It is imported to perform Random Forest Regression on the training data.

   b. Created Spark Session

   spark = SparkSession.builder.appName("Assign3_Group4_Task3_PartB").getOrCreate()

- ➔ Here, we provided the name to our application by setting a string "Assign3_Group4_Task3_PartB" to.appName() as a parameter. Next, used .getOrCreate() to create and instantiate SparkSession into our object "spark".

c. Reading csv file into PySpark DataFrame

```
df = spark.read.csv("hdfs://hadoopnn001.cs.okstate.edu:9000 user/sdarapu/Assign3_Group4_Task1_Output_inpfor_Task2-4/part-00000-571e77d2-85ae-4579-92f8-dd4dc788ab7f-c000.csv", header = True, inferSchema = True)
```

- ➔ By using spark.read.csv() method, we first passed the given csv file location(i.e., output file of the task 1) and we used "inferSchema" attribute and set its value as True which will automatically take schema from the given file into Pyspark Dataframe.

d. Printing Schema

df.printSchema() → Prints the schema of the dataframe "df".

e. Performing Scaling on columns used for features set

```
unlist = udf(lambda x: round(float(list(x)[0]),3), DoubleType())

col_list = ["NAME","STATE","Score","PR Rank","INSTITUTE ID"]

indexx = 0

while indexx < len(col_list):
        assembler=VectorAssembler(inputCols=[col_list[indexx]],outputCol=col_list[indexx]+"_Vect")

        # MinMaxScaler Transformation

        scaler = MinMaxScaler(inputCol=col_list[indexx]+"_Vect",outputCol=col_list[indexx]+"_Scaled")

        # Pipeline of VectorAssembler and MinMaxScaler

        pipeline = Pipeline(stages=[assembler, scaler])

        # Fitting pipeline on dataframe
```

```
df =pipeline.fit(df).transform(df).withColumn(col_list[indexx]+"_Scaled", unlist(col_list[indexx]+"_Scaled")).drop(col_list[indexx]+"_Vect")

indexx = indexx +1
```

➔ When we perform scaling on the columns specified, it will convert the values of data frame based on the min-max range. This is done to get the better accuracy in task 3.

f. Use Of VectorAssembler

```
vectorAssembler = VectorAssembler(inputCols = df.columns[9:], outputCol = 'features')
vectorAssembler.setParams(handleInvalid="skip")
transform_output=vectorAssembler.transform(df)
final_df=transform_output.select('features','Year')
```

➔ We then use VectorAssembler which is a transformer that combines a given list of columns into a single vector column. We provided 5 feature scaled columns as input to the VectorAssembler which will then combine them into a single vector column called 'features'.

g. Splitting the data into train and test data

```
(trainingData, testData) = final_df.randomSplit([0.7, 0.3],80)
```

➔ Now, we have split the dataset "final_df" into training and test data 1ith 70% and 30% respectively. We have used seed value 80 because if the code is rerun then we get the same count of rows for training and test data.

h. Print number of training and test records

```
print("Number of training records are",trainingData.count())
print("Number of test records are",testData.count())
```

➔ Prints the number of records in both training and test data.

i. Shows descriptive statistics of training and test data

```
trainingData.describe().show()
testData.describe().show()
```

➔ The above statements display the descriptive statistics like mean, stddev, etc, of training and test data.

j. Model the train data using Random Forest Regression

```
rf=RandomForestRegressor(featuresCol = 'features', labelCol='Year')
```

```
rf.setMaxBins(300)
rf_model=rf.fit(trainingData)
```

➔ Now, the data (features and Year) is prepared and transformed into a format for Random Forest Reggressor and we have set MaxBins to 300 as we are having more records.

k. Transforming the test data

```
rf_predictions = rf_model.transform(testData)
```

➔ Now, we have transformed the test data and stored the result in “rf_predictions”.

l. Select the required columns from lr_predictions and display the result

```
rf=rf_predictions.select("prediction","Year","features")

rf.show(100)
```

➔ Now, we have retrieved the columns “prediction”, “Year”, “features” from rf_predictions dataframe and stored the result into “rf”. After that, displayed the first 100 rows on the console by using show() method.

m. Converting datatype of features column

```
rf = rf.withColumn('features', vector_to_array('features'))
rf = rf.withColumn("features",concat_ws(",",col("features")))
```

➔ At first, we have converted the features column data type from vector to array and then from array to String. This is done to store the result into a csv file in the later step.

n. Store result into specified hdfs folder

```
lr.coalesce(1).write.mode("overwrite").option("header","true").csv("hdfs:////user/s
darapu/Assign3_Group4_Task2_PartB_Output")
```

➔ Stored the resultant to new specified folder under hdfs where the files in the new folder are stored in .csv format.

3. <u>Steps to execute the code:</u>

i. To run the code, we have executed below command as shown below.

```
sdarapu@hadoop-nn001:~$ spark-submit /home/sdarapu/Assign3_Group4_Task2_PartB.py
```

Fig 2.B, 1: Command to execute

ii. The above command executes as follows.

```
sdarapu@hadoop-nn001:~$ spark-submit /home/sdarapu/Assign3_Group4_Task2_PartB.py
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by org.apache.spark.unsafe.Platform (file:/usr/local/spark-3.0.1-bin-hadoop3.2/jars/spark-unsafe_2.12-3.0.1.jar) to co
DirectByteBuffer(long,int)
WARNING: Please consider reporting this to the maintainers of org.apache.spark.unsafe.Platform
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations
WARNING: All illegal access operations will be denied in a future release
2022-04-29 16:36:52,251 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
2022-04-29 16:36:53,611 INFO spark.SparkContext: Running Spark version 3.0.1
2022-04-29 16:36:53,656 INFO resource.ResourceUtils: ==============================================================
2022-04-29 16:36:53,657 INFO resource.ResourceUtils: Resources for spark.driver:

2022-04-29 16:36:53,658 INFO resource.ResourceUtils: ==============================================================
2022-04-29 16:36:53,658 INFO spark.SparkContext: Submitted application: Assignment3_Group4_Task2_PartB
2022-04-29 16:36:53,713 INFO spark.SecurityManager: Changing view acls to: sdarapu
2022-04-29 16:36:53,713 INFO spark.SecurityManager: Changing modify acls to: sdarapu
2022-04-29 16:36:53,714 INFO spark.SecurityManager: Changing view acls groups to:
2022-04-29 16:36:53,714 INFO spark.SecurityManager: Changing modify acls groups to:
2022-04-29 16:36:53,714 INFO spark.SecurityManager: SecurityManager: authentication disabled; ui acls disabled; users  with view permissions: Set(sdarapu)
 permissions: Set(); users  with modify permissions: Set(sdarapu); groups with modify permissions: Set()
2022-04-29 16:36:54,003 INFO util.Utils: Successfully started service 'sparkDriver' on port 39437.
2022-04-29 16:36:54,037 INFO spark.SparkEnv: Registering MapOutputTracker
2022-04-29 16:36:54,073 INFO spark.SparkEnv: Registering BlockManagerMaster
2022-04-29 16:36:54,095 INFO storage.BlockManagerMasterEndpoint: Using org.apache.spark.storage.DefaultTopologyMapper for getting topology information
2022-04-29 16:36:54,096 INFO storage.BlockManagerMasterEndpoint: BlockManagerMasterEndpoint up
2022-04-29 16:36:54,135 INFO spark.SparkEnv: Registering BlockManagerMasterHeartbeat
2022-04-29 16:36:54,154 INFO storage.DiskBlockManager: Created local directory at /tmp/blockmgr-3acee5be-047e-486d-a496-c0d28b9be270
2022-04-29 16:36:54,179 INFO memory.MemoryStore: MemoryStore started with capacity 434.4 MiB
2022-04-29 16:36:54,222 INFO spark.SparkEnv: Registering OutputCommitCoordinator
2022-04-29 16:36:54,320 INFO util.log: Logging initialized @3635ms to org.sparkproject.jetty.util.log.Slf4jLog
2022-04-29 16:36:54,384 INFO server.Server: jetty-9.4.z-SNAPSHOT; built: 2019-04-29T20:42:08.989Z; git: e1bc35120a6617ee3df052294e433f3a25ce7097; jvm 11.
ntu0.20.04.1
```

Fig 2.B, 2: Execution Results

```
2022-04-29 16:36:54,499 INFO handler.ContextHandler: Started o.s.j.s.ServletContextHandler@2fb41ab9{/storage,null,AVAILABLE,@Spark}
2022-04-29 16:36:54,501 INFO handler.ContextHandler: Started o.s.j.s.ServletContextHandler@9edea51{/storage/json,null,AVAILABLE,@Spark}
2022-04-29 16:36:54,502 INFO handler.ContextHandler: Started o.s.j.s.ServletContextHandler@38ddc05a{/storage/rdd,null,AVAILABLE,@Spark}
2022-04-29 16:36:54,504 INFO handler.ContextHandler: Started o.s.j.s.ServletContextHandler@29e67af9{/storage/rdd/json,null,AVAILABLE,@Spark}
2022-04-29 16:36:54,506 INFO handler.ContextHandler: Started o.s.j.s.ServletContextHandler@535ef0af{/environment,null,AVAILABLE,@Spark}
2022-04-29 16:36:54,507 INFO handler.ContextHandler: Started o.s.j.s.ServletContextHandler@7b096d40{/environment/json,null,AVAILABLE,@Spark}
2022-04-29 16:36:54,509 INFO handler.ContextHandler: Started o.s.j.s.ServletContextHandler@63e7c3ed{/executors,null,AVAILABLE,@Spark}
2022-04-29 16:36:54,510 INFO handler.ContextHandler: Started o.s.j.s.ServletContextHandler@2571ea2d{/executors/json,null,AVAILABLE,@Spark}
2022-04-29 16:36:54,512 INFO handler.ContextHandler: Started o.s.j.s.ServletContextHandler@7b0e20e9{/executors/threadDump,null,AVAILABLE,@Spark}
2022-04-29 16:36:54,514 INFO handler.ContextHandler: Started o.s.j.s.ServletContextHandler@53915366{/executors/threadDump/json,null,AVAILABLE,@Spark}
2022-04-29 16:36:54,526 INFO handler.ContextHandler: Started o.s.j.s.ServletContextHandler@24ff8a17{/static,null,AVAILABLE,@Spark}
2022-04-29 16:36:54,527 INFO handler.ContextHandler: Started o.s.j.s.ServletContextHandler@191abbae{/,null,AVAILABLE,@Spark}
2022-04-29 16:36:54,529 INFO handler.ContextHandler: Started o.s.j.s.ServletContextHandler@4247227e{/api,null,AVAILABLE,@Spark}
2022-04-29 16:36:54,530 INFO handler.ContextHandler: Started o.s.j.s.ServletContextHandler@27193784{/jobs/job/kill,null,AVAILABLE,@Spark}
2022-04-29 16:36:54,532 INFO handler.ContextHandler: Started o.s.j.s.ServletContextHandler@42fccb5c{/stages/stage/kill,null,AVAILABLE,@Spark}
2022-04-29 16:36:54,534 INFO ui.SparkUI: Bound SparkUI to 0.0.0.0, and started at http://hadoop-nn001:4041
2022-04-29 16:36:54,967 INFO client.RMProxy: Connecting to ResourceManager at hadoop-nn001.cs.okstate.edu/192.168.122.2:8032
2022-04-29 16:36:55,265 INFO yarn.Client: Requesting a new application from cluster with 12 NodeManagers
2022-04-29 16:36:55,700 INFO conf.Configuration: resource-types.xml not found
2022-04-29 16:36:55,701 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
2022-04-29 16:36:55,717 INFO yarn.Client: Verifying our application has not requested more than the maximum memory capability of the cluster (8192 MB per container)
2022-04-29 16:36:55,717 INFO yarn.Client: Will allocate AM container, with 896 MB memory including 384 MB overhead
2022-04-29 16:36:55,718 INFO yarn.Client: Setting up container launch context for our AM
2022-04-29 16:36:55,720 INFO yarn.Client: Setting up the launch environment for our AM container
2022-04-29 16:36:55,726 INFO yarn.Client: Preparing resources for our AM container
2022-04-29 16:36:55,778 WARN yarn.Client: Neither spark.yarn.jars nor spark.yarn.archive is set, falling back to uploading libraries under SPARK_HOME.
2022-04-29 16:36:58,679 INFO yarn.Client: Uploading resource file:/tmp/spark-61a52bd2-b3dd-4433-9297-ec1e9f173877/__spark_libs__15529102479982301526.zip -> hdfs://h
001.cs.okstate.edu:9000/user/sdarapu/.sparkStaging/application_1647031195237_1387/__spark_libs__15529102479982301526.zip
2022-04-29 16:37:01,190 INFO yarn.Client: Uploading resource file:/usr/local/spark/python/lib/pyspark.zip -> hdfs://hadoop-nn001.cs.okstate.edu:9000/user/sdarapu/.s
ing/application_1647031195237_1387/pyspark.zip
2022-04-29 16:37:01,258 INFO yarn.Client: Uploading resource file:/usr/local/spark/python/lib/py4j-0.10.9-src.zip -> hdfs://hadoop-nn001.cs.okstate.edu:9000/user/sd
parkStaging/application_1647031195237_1387/py4j-0.10.9-src.zip
2022-04-29 16:37:01,598 INFO yarn.Client: Uploading resource file:/tmp/spark-61a52bd2-b3dd-4433-9297-ec1e9f173877/__spark_conf__14976735293428764453.zip -> hdfs://h
001.cs.okstate.edu:9000/user/sdarapu/.sparkStaging/application_1647031195237_1387/__spark_conf__.zip
```

Fig 2.B, 3: Execution Results

```
        start time: 1651268221702
        final status: UNDEFINED
        tracking URL: http://hadoop-nn001.cs.okstate.edu:8088/proxy/application_1647031195237_1387/
        user: sdarapu
2022-04-29 16:37:03,735 INFO yarn.Client: Application report for application_1647031195237_1387 (state: ACCEPTED)
2022-04-29 16:37:04,737 INFO yarn.Client: Application report for application_1647031195237_1387 (state: ACCEPTED)
2022-04-29 16:37:05,740 INFO yarn.Client: Application report for application_1647031195237_1387 (state: ACCEPTED)
2022-04-29 16:37:06,742 INFO yarn.Client: Application report for application_1647031195237_1387 (state: RUNNING)
2022-04-29 16:37:06,743 INFO yarn.Client:
        client token: N/A
        diagnostics: N/A
        ApplicationMaster host: 10.221.247.4
        ApplicationMaster RPC port: -1
        queue: default
        start time: 1651268221702
        final status: UNDEFINED
        tracking URL: http://hadoop-nn001.cs.okstate.edu:8088/proxy/application_1647031195237_1387/
        user: sdarapu
2022-04-29 16:37:06,745 INFO cluster.YarnClientSchedulerBackend: Application application_1647031195237_1387 has started running.
2022-04-29 16:37:06,763 INFO util.Utils: Successfully started service 'org.apache.spark.network.netty.NettyBlockTransferService' on port 4
2022-04-29 16:37:06,764 INFO netty.NettyBlockTransferService: Server created on hadoop-nn001:45259
2022-04-29 16:37:06,768 INFO storage.BlockManager: Using org.apache.spark.storage.RandomBlockReplicationPolicy for block replication polic
2022-04-29 16:37:06,781 INFO storage.BlockManagerMaster: Registering BlockManager BlockManagerId(driver, hadoop-nn001, 45259, None)
2022-04-29 16:37:06,787 INFO storage.BlockManagerMasterEndpoint: Registering block manager hadoop-nn001:45259 with 434.4 MiB RAM, BlockMan
, None)
2022-04-29 16:37:06,793 INFO storage.BlockManagerMaster: Registered BlockManager BlockManagerId(driver, hadoop-nn001, 45259, None)
2022-04-29 16:37:06,794 INFO storage.BlockManager: Initialized BlockManager: BlockManagerId(driver, hadoop-nn001, 45259, None)
```

Fig 2.B, 4: Execution Results

Output displayed on the Console:

```
root
 |-- INSTITUTE ID: double (nullable = true)
 |-- NAME: double (nullable = true)
 |-- CITY: double (nullable = true)
 |-- STATE: double (nullable = true)
 |-- PR Score: double (nullable = true)
 |-- PR Rank: double (nullable = true)
 |-- Score: double (nullable = true)
 |-- Year: double (nullable = true)
 |-- Rank: double (nullable = true)
```

Fig 2.B, 5: Schema of the data frame

```
Number of train records are 349
```

Fig 2.A, 6: Displays number of train records

```
Number of test records are 152
```

Fig 2.A, 7: Displays number of test records

summary	Year
count	349
mean	2019.0601719197707
stddev	1.4180044964555496
min	2017.0
max	2021.0

Fig 2.B, 8: Statistics summary of train data

summary	Year
count	152
mean	2018.8486842105262
stddev	1.4083942687234334
min	2017.0
max	2021.0

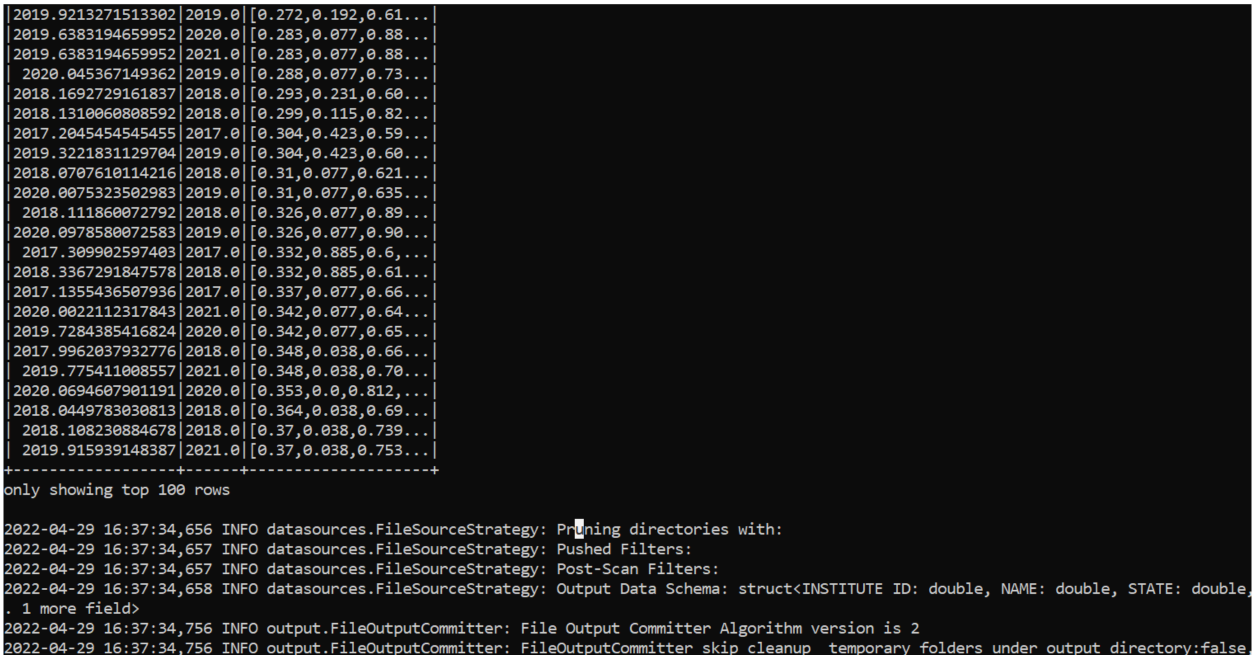
Fig 2.B, 9: Statistics summary of test data

prediction	Year	features
2020.1276026413568	2019.0	[0.011,0.0,0.919,...
2020.203992290115	2020.0	[0.011,0.0,0.965,...
2017.2932738073287	2017.0	[0.016,0.0,0.877,...
2018.0033849272481	2018.0	[0.022,0.115,0.98...
2018.058098731962	2018.0	[0.038,0.0,0.856,...
2020.0168073610869	2021.0	[0.038,0.0,0.876,...
2020.0168073610869	2019.0	[0.038,0.0,0.888,...
2017.8982539682543	2017.0	[0.049,0.038,0.60...
2019.9220865825464	2021.0	[0.049,0.038,0.65...
2020.0016276403887	2019.0	[0.054,0.231,0.94...
2017.2745454545457	2017.0	[0.06,0.269,0.602...
2019.9931863827592	2019.0	[0.06,0.269,0.659...
2019.9517907367294	2021.0	[0.06,0.269,0.688...
2017.307568216942	2017.0	[0.076,0.346,0.69...
2018.0588073629103	2018.0	[0.076,0.346,0.70...
2020.0372479562998	2019.0	[0.076,0.346,0.71...
2020.1604505561693	2020.0	[0.076,0.346,0.73...
2019.9414770263909	2020.0	[0.082,0.154,0.62...
2020.1900975806977	2019.0	[0.082,0.154,0.64...
2018.0588073629103	2018.0	[0.087,0.308,0.66...
2019.83805965084	2021.0	[0.087,0.308,0.71...
2018.0836737924578	2018.0	[0.092,0.038,0.76...
2020.0407507845393	2020.0	[0.092,0.038,0.87...
2018.0295652629436	2018.0	[0.098,0.038,0.79...
2020.14525964841	2020.0	[0.098,0.038,0.84...
2017.2922637063189	2017.0	[0.103,0.231,0.89...
2018.0478293716926	2018.0	[0.103,0.231,0.92...
2019.8766633845219	2021.0	[0.103,0.231,0.93...
2019.6747709114036	2020.0	[0.103,0.231,0.96...
2017.1673214285715	2017.0	[0.109,0.154,0.69...
2020.1072549556307	2020.0	[0.109,0.154,0.81...
2018.0033849272481	2018.0	[0.12,0.154,0.801...
2020.1364658295647	2019.0	[0.12,0.154,0.802...
2020.1364658295647	2020.0	[0.12,0.154,0.802...
2020.01078711536	2019.0	[0.125,0.077,0.64...
2017.7877976190478	2017.0	[0.13,0.462,0.633...
2020.085874223584	2020.0	[0.13,0.462,0.814...
2019.9542207942886	2021.0	[0.141,0.769,0.69...

Fig 2.B, 10: Display of Prediction, Year and features columns

2019.9542207942886	2021.0	[0.141,0.769,0.69...
2020.1219962881023	2021.0	[0.147,0.385,0.76...
2020.211676576293	2019.0	[0.147,0.385,0.77...
2020.211676576293	2020.0	[0.147,0.385,0.79...
2020.197971607088	2020.0	[0.152,0.654,0.80...
2018.0185506939615	2018.0	[0.158,0.0,0.604,...
2020.0376238975143	2021.0	[0.158,0.0,0.647,...
2018.0470190948338	2018.0	[0.163,0.538,0.66...
2019.9190856591533	2021.0	[0.163,0.538,0.69...
2019.7713170737345	2019.0	[0.163,0.538,0.7,...
2019.9190856591533	2020.0	[0.163,0.538,0.71...
2017.4194444444445	2017.0	[0.174,0.0,0.605,...
2017.9643142180114	2018.0	[0.174,0.0,0.686,...
2020.0929569163145	2020.0	[0.174,0.0,0.756,...
2017.2045454545455	2017.0	[0.185,0.115,0.59...
2019.7157960279085	2019.0	[0.185,0.115,0.69...
2017.2	2017.0	[0.19,0.538,0.557...
2017.9696482377217	2018.0	[0.196,0.192,0.67...
2019.843842086851	2019.0	[0.196,0.192,0.7,...
2017.161929271709	2017.0	[0.201,0.346,0.68...
2020.2321952551451	2019.0	[0.212,0.385,0.95...
2019.8916510080915	2019.0	[0.217,0.615,0.63...
2018.052353114544	2018.0	[0.217,0.615,0.67...
2020.1874763067717	2020.0	[0.223,0.615,0.73...
2017.0	2017.0	[0.223,0.962,0.56...
2020.3563330946695	2019.0	[0.228,0.0,0.798,...
2020.3190315073675	2020.0	[0.228,0.0,0.817,...
2020.0008642763937	2021.0	[0.234,0.0,0.879,...
2019.8685422274543	2021.0	[0.239,0.231,0.66...
2019.8833366002764	2019.0	[0.239,0.231,0.72...
2018.1512283030813	2018.0	[0.239,0.231,0.74...
2018.1310060808592	2018.0	[0.245,0.192,0.78...
2019.8755607421867	2021.0	[0.25,0.846,0.623...
2018.0665943447552	2018.0	[0.255,0.154,0.63...
2020.0840683497997	2021.0	[0.255,0.154,0.74...
2017.0	2017.0	[0.261,0.346,0.56...
2019.9199084914314	2019.0	[0.261,0.346,0.61...
2019.9505840595314	2020.0	[0.261,0.346,0.64...
2017.9962037932776	2018.0	[0.266,0.038,0.67...
2020.0358721016037	2020.0	[0.266,0.038,0.74...
2019.9213271513302	2019.0	[0.272,0.192,0.61...

Fig 2.B, 11: Display of Prediction, Year and features columns



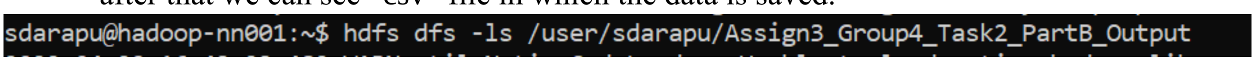
```
|2019.9213271513302|2019.0|[0.272,0.192,0.61...|
|2019.6383194659952|2020.0|[0.283,0.077,0.88...|
|2019.6383194659952|2021.0|[0.283,0.077,0.88...|
| 2020.045367149362|2019.0|[0.288,0.077,0.73...|
|2018.1692729161837|2018.0|[0.293,0.231,0.60...|
|2018.1310060808592|2018.0|[0.299,0.115,0.82...|
|2017.2045454545455|2017.0|[0.304,0.423,0.59...|
|2019.3221831129704|2019.0|[0.304,0.423,0.60...|
|2018.0707610114216|2018.0|[0.31,0.077,0.621...|
|2020.0075323502983|2019.0|[0.31,0.077,0.635...|
| 2018.111860072792|2018.0|[0.326,0.077,0.89...|
|2020.0978580072583|2019.0|[0.326,0.077,0.90...|
| 2017.309902597403|2017.0|[0.332,0.885,0.6,...|
|2018.3367291847578|2018.0|[0.332,0.885,0.61...|
|2017.1355436507936|2017.0|[0.337,0.077,0.66...|
|2020.0022112317843|2021.0|[0.342,0.077,0.64...|
|2019.7284385416824|2020.0|[0.342,0.077,0.65...|
|2017.9962037932776|2018.0|[0.348,0.038,0.66...|
| 2019.775411008557|2021.0|[0.348,0.038,0.70...|
|2020.0694607901191|2020.0|[0.353,0.0,0.812,...|
|2018.0449783030813|2018.0|[0.364,0.038,0.69...|
| 2018.108230884678|2018.0|[0.37,0.038,0.739...|
| 2019.915939148387|2021.0|[0.37,0.038,0.753...|
+------------------+------+--------------------+
only showing top 100 rows

2022-04-29 16:37:34,656 INFO datasources.FileSourceStrategy: Pruning directories with:
2022-04-29 16:37:34,657 INFO datasources.FileSourceStrategy: Pushed Filters:
2022-04-29 16:37:34,657 INFO datasources.FileSourceStrategy: Post-Scan Filters:
2022-04-29 16:37:34,658 INFO datasources.FileSourceStrategy: Output Data Schema: struct<INSTITUTE ID: double, NAME: double, STATE: double,
. 1 more field>
2022-04-29 16:37:34,756 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 2
2022-04-29 16:37:34,756 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders under output directory:false
```

Fig 2.B, 12: Display of Prediction, Year and features columns

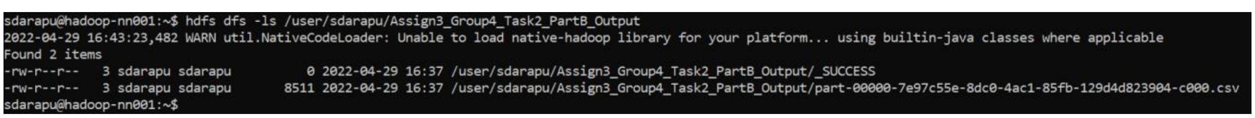
Output stored in the specified folder under HDFS:

i. To see the result stored in the specified folder ("Assign3_Group4_Task2_PartB_Output"), execute the below command as shown after that we can see "csv" file in which the data is saved.



```
sdarapu@hadoop-nn001:~$ hdfs dfs -ls /user/sdarapu/Assign3_Group4_Task2_PartB_Output
```

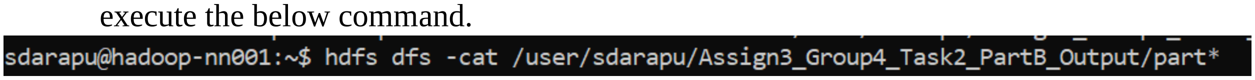
Fig 2.B,13: View list of contents



```
sdarapu@hadoop-nn001:~$ hdfs dfs -ls /user/sdarapu/Assign3_Group4_Task2_PartB_Output
2022-04-29 16:43:23,482 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Found 2 items
-rw-r--r--   3 sdarapu sdarapu          0 2022-04-29 16:37 /user/sdarapu/Assign3_Group4_Task2_PartB_Output/_SUCCESS
-rw-r--r--   3 sdarapu sdarapu       8511 2022-04-29 16:37 /user/sdarapu/Assign3_Group4_Task2_PartB_Output/part-00000-7e97c55e-8dc0-4ac1-85fb-129d4d823904-c000.csv
sdarapu@hadoop-nn001:~$
```

Fig 2.B,14: List of files in the respective folder

ii. To view the data in the folder "Assign3_Group4_Task2_PartB_Output" at once, execute the below command.



```
sdarapu@hadoop-nn001:~$ hdfs dfs -cat /user/sdarapu/Assign3_Group4_Task2_PartB_Output/part*
```

Fig 2.B,15: View the content in the file

Output Display

```
prediction,Year,features
2020.1276026413568,2019.0,"0.011,0.0,0.919,0.041,0.176"
2020.203992290115,2020.0,"0.011,0.0,0.965,0.05,0.176"
2017.2932738073287,2017.0,"0.016,0.0,0.877,0.017,0.879"
2018.0033849272481,2018.0,"0.022,0.115,0.984,0.012,0.557"
2018.058098731962,2018.0,"0.038,0.0,0.856,0.066,0.307"
2020.0168073610869,2021.0,"0.038,0.0,0.876,0.041,0.186"
2020.0168073610869,2019.0,"0.038,0.0,0.888,0.041,0.186"
2017.8982539682543,2017.0,"0.049,0.038,0.606,0.029,0.969"
2019.9220865825464,2021.0,"0.049,0.038,0.658,0.108,0.009"
2020.0016276403887,2019.0,"0.054,0.231,0.944,0.025,0.229"
2017.2745454545457,2017.0,"0.06,0.269,0.602,0.278,0.981"
2019.9931863827592,2019.0,"0.06,0.269,0.659,0.249,0.108"
2019.9517907367294,2021.0,"0.06,0.269,0.688,0.473,0.108"
2017.307568216942,2017.0,"0.076,0.346,0.693,0.079,1.0"
2018.0588073629103,2018.0,"0.076,0.346,0.708,0.232,0.353"
2020.0372479562998,2019.0,"0.076,0.346,0.71,0.062,0.046"
2020.1604505561693,2020.0,"0.076,0.346,0.734,0.062,0.046"
2019.9414770263909,2020.0,"0.082,0.154,0.629,0.129,0.059"
2020.1900975806977,2019.0,"0.082,0.154,0.644,0.402,0.059"
2018.0588073629103,2018.0,"0.087,0.308,0.661,0.207,0.44"
2019.83805965084,2021.0,"0.087,0.308,0.712,0.075,0.164"
2018.0836737924578,2018.0,"0.092,0.038,0.762,0.17,0.415"
2020.0407507845393,2020.0,"0.092,0.038,0.87,0.137,0.124"
2018.0295652629436,2018.0,"0.098,0.038,0.799,0.1,0.551"
2020.14525964841,2020.0,"0.098,0.038,0.841,0.054,0.127"
2017.2922637063189,2017.0,"0.103,0.231,0.89,0.033,0.817"
2018.0478293716926,2018.0,"0.103,0.231,0.926,0.041,0.529"
2019.8766633845219,2021.0,"0.103,0.231,0.936,0.008,0.232"
2019.6747709114036,2020.0,"0.103,0.231,0.961,0.004,0.232"
2017.1673214285715,2017.0,"0.109,0.154,0.699,0.452,0.839"
2020.1072549556307,2020.0,"0.109,0.154,0.818,0.34,0.062"
2018.0033849272481,2018.0,"0.12,0.154,0.801,0.008,0.359"
2020.1364658295647,2019.0,"0.12,0.154,0.802,0.008,0.068"
2020.1364658295647,2020.0,"0.12,0.154,0.802,0.008,0.068"
2020.01078711536,2019.0,"0.125,0.077,0.64,0.357,0.093"
2017.7877976190478,2017.0,"0.13,0.462,0.633,0.054,0.755"
2020.085874223584,2020.0,"0.13,0.462,0.814,0.183,0.149"
2019.9542207942886,2021.0,"0.141,0.769,0.69,0.295,0.146"
2020.1219962881023,2021.0,"0.147,0.385,0.765,0.095,0.037"
2020.211676576293,2019.0,"0.147,0.385,0.776,0.062,0.037"
```

Fig 2.B, 15: Output

```
2020.211676576293,2019.0,"0.147,0.385,0.776,0.062,0.037"
2020.211676576293,2020.0,"0.147,0.385,0.796,0.079,0.037"
2020.197971607088,2020.0,"0.152,0.654,0.806,0.033,0.053"
2018.0185506939615,2018.0,"0.158,0.0,0.604,0.29,0.461"
2020.0376238975143,2021.0,"0.158,0.0,0.647,0.506,0.195"
2018.0470190948338,2018.0,"0.163,0.538,0.661,0.112,0.539"
2019.9190856591533,2021.0,"0.163,0.538,0.692,0.232,0.155"
2019.7713170737345,2019.0,"0.163,0.538,0.7,0.1,0.155"
2019.9190856591533,2020.0,"0.163,0.538,0.714,0.237,0.155"
2017.4194444444445,2017.0,"0.174,0.0,0.605,0.253,0.941"
2017.9643142180114,2018.0,"0.174,0.0,0.686,0.112,0.505"
2020.0929569163145,2020.0,"0.174,0.0,0.756,0.502,0.012"
2017.2045454545455,2017.0,"0.185,0.115,0.594,0.27,0.824"
2019.7157960279085,2019.0,"0.185,0.115,0.693,0.1,0.238"
2017.2,2017.0,"0.19,0.538,0.557,0.054,0.793"
2017.9696482377217,2018.0,"0.196,0.192,0.678,0.473,0.517"
2019.843842086851,2019.0,"0.196,0.192,0.7,0.095,0.04"
2017.161929271709,2017.0,"0.201,0.346,0.683,0.22,0.82"
2020.2321952551451,2019.0,"0.212,0.385,0.959,0.021,0.043"
2019.8916510080915,2019.0,"0.217,0.615,0.632,0.556,0.077"
2018.052353114544,2018.0,"0.217,0.615,0.674,0.552,0.56"
2020.1874763067717,2020.0,"0.223,0.615,0.732,0.278,0.08"
2017.0,2017.0,"0.223,0.962,0.569,0.759,0.759"
2020.3563330946695,2019.0,"0.228,0.0,0.798,0.058,0.006"
2020.3190315073675,2020.0,"0.228,0.0,0.817,0.058,0.006"
2020.0008642763937,2021.0,"0.234,0.0,0.879,0.029,0.207"
2019.8685422274543,2021.0,"0.239,0.231,0.668,0.68,0.235"
2019.8833366002764,2019.0,"0.239,0.231,0.721,0.178,0.235"
2018.1512283030813,2018.0,"0.239,0.231,0.742,0.274,0.486"
2018.1310060808592,2018.0,"0.245,0.192,0.784,0.174,0.598"
2019.8755607421867,2021.0,"0.25,0.846,0.623,0.075,0.084"
2018.0665943447552,2018.0,"0.255,0.154,0.633,0.104,0.52"
2020.0840683497997,2021.0,"0.255,0.154,0.748,0.141,0.056"
2017.0,2017.0,"0.261,0.346,0.569,0.589,0.709"
2019.9199084914314,2019.0,"0.261,0.346,0.615,0.328,0.248"
2019.9505840595314,2020.0,"0.261,0.346,0.641,0.357,0.248"
2017.9962037932776,2018.0,"0.266,0.038,0.674,0.552,0.502"
2020.0358721016037,2020.0,"0.266,0.038,0.746,0.137,0.0"
2019.9213271513302,2019.0,"0.272,0.192,0.617,0.12,0.028"
2019.6383194659952,2020.0,"0.283,0.077,0.886,0.0,0.087"
2019.6383194659952,2021.0,"0.283,0.077,0.886,0.0,0.087"
```

Fig 2.B, 16: Output

```
2019.6383194659952,2020.0,"0.283,0.077,0.886,0.0,0.087"
2019.6383194659952,2021.0,"0.283,0.077,0.886,0.0,0.087"
2020.045367149362,2019.0,"0.288,0.077,0.732,0.054,0.09"
2018.1692729161837,2018.0,"0.293,0.231,0.608,0.154,0.483"
2018.1310060808592,2018.0,"0.299,0.115,0.82,0.307,0.319"
2017.2045454545455,2017.0,"0.304,0.423,0.599,0.436,0.762"
2019.3221831129704,2019.0,"0.304,0.423,0.604,0.149,0.257"
2018.0707610114216,2018.0,"0.31,0.077,0.621,0.602,0.39"
2020.0075323502983,2019.0,"0.31,0.077,0.635,0.768,0.096"
2018.111860072792,2018.0,"0.326,0.077,0.89,0.041,0.393"
2020.0978580072583,2019.0,"0.326,0.077,0.908,0.054,0.099"
2017.309902597403,2017.0,"0.332,0.885,0.6,0.884,0.768"
2018.3367291847578,2018.0,"0.332,0.885,0.616,0.602,0.43"
2017.1355436507936,2017.0,"0.337,0.077,0.669,0.361,0.771"
2020.0022112317843,2021.0,"0.342,0.077,0.641,0.349,0.105"
2019.7284385416824,2020.0,"0.342,0.077,0.659,0.075,0.105"
2017.9962037932776,2018.0,"0.348,0.038,0.668,0.602,0.604"
2019.775411008557,2021.0,"0.348,0.038,0.708,0.087,0.022"
2020.0694607901191,2020.0,"0.353,0.0,0.812,0.133,0.204"
2018.0449783030813,2018.0,"0.364,0.038,0.696,0.232,0.421"
2018.108230884678,2018.0,"0.37,0.038,0.739,0.162,0.424"
2019.915939148387,2021.0,"0.37,0.038,0.753,0.137,0.142"
2017.9036517040622,2018.0,"0.375,0.231,0.59,0.411,0.477"
2020.1018348736573,2021.0,"0.375,0.231,0.625,0.473,0.226"
2020.006807100513,2021.0,"0.38,0.115,0.792,0.037,0.214"
2017.147876984127,2017.0,"0.386,0.577,0.66,0.411,0.876"
2018.3899736235223,2018.0,"0.391,0.346,0.607,0.411,0.347"
2019.789052984713,2021.0,"0.391,0.346,0.613,0.149,0.245"
2020.0547411455732,2019.0,"0.413,0.385,0.625,0.12,0.031"
2017.0,2017.0,"0.424,0.0,0.572,0.763,0.765"
2018.4394610617019,2020.0,"0.435,0.192,0.657,0.17,0.616"
2018.4938808800584,2019.0,"0.446,0.0,0.639,0.556,0.672"
2017.1938769841267,2017.0,"0.446,0.0,0.663,0.593,0.808"
2019.7261489290136,2021.0,"0.457,0.077,0.656,0.083,0.26"
2018.4422770496355,2019.0,"0.467,0.423,0.661,0.17,0.635"
2020.1000129074425,2020.0,"0.478,0.038,0.705,0.224,0.272"
2018.3396433159276,2018.0,"0.484,0.731,0.63,0.473,0.378"
2018.27850346473,2018.0,"0.489,0.808,0.726,0.274,0.471"
2020.2948935542179,2020.0,"0.505,0.423,0.632,0.253,0.254"
2020.2079879963853,2020.0,"0.516,0.0,0.665,0.402,0.291"
2018.0795890081376,2018.0,"0.522,0.0,0.604,0.162,0.542"
```

Fig 2.B, 17: Output

```
2018.0795890081376,2018.0,"0.522,0.0,0.604,0.162,0.542"
2020.2423193429502,2020.0,"0.527,0.192,0.759,0.112,0.034"
2020.2639161351126,2020.0,"0.538,0.308,0.651,0.224,0.282"
2019.7074502015607,2021.0,"0.543,0.0,0.676,0.083,0.192"
2020.0334280517932,2019.0,"0.549,0.0,0.607,0.224,0.294"
2017.0,2017.0,"0.56,0.077,0.0,0.095,0.774"
2018.4343602919512,2019.0,"0.576,0.308,0.697,0.423,0.653"
2018.1198456770865,2018.0,"0.576,0.308,0.71,0.261,0.443"
2018.3789785657013,2019.0,"0.582,0.923,0.602,0.826,0.622"
2020.3882325469772,2019.0,"0.592,0.0,0.737,0.286,0.019"
2018.9682749749695,2019.0,"0.598,0.192,0.629,0.095,0.619"
2018.851722549883,2019.0,"0.609,0.077,0.619,0.61,0.65"
2017.7536309523814,2017.0,"0.63,0.115,0.613,0.075,0.867"
2018.6614374648343,2021.0,"0.674,0.115,0.666,0.436,0.675"
2017.9563095238095,2017.0,"0.701,0.731,0.618,0.1,0.712"
2017.05,2017.0,"0.707,0.192,0.551,0.095,0.721"
2017.05,2017.0,"0.712,0.692,0.582,0.274,0.854"
2017.7052857142858,2017.0,"0.717,0.308,0.627,0.647,0.858"
2017.1386389452334,2017.0,"0.723,0.154,0.796,0.137,0.715"
2019.9053667440135,2021.0,"0.755,0.077,0.609,0.183,0.263"
2017.1764575163402,2017.0,"0.766,0.192,0.647,0.311,0.697"
2017.0,2017.0,"0.777,0.0,0.577,0.365,0.889"
2017.0,2017.0,"0.804,0.269,0.577,0.423,0.904"
2019.0748700045174,2019.0,"0.81,0.077,0.614,0.61,0.647"
2017.0,2017.0,"0.821,0.077,0.557,0.324,0.916"
2018.444991333127,2018.0,"0.826,1.0,0.597,0.689,0.433"
2018.792587206855,2021.0,"0.848,0.577,0.647,0.328,0.632"
2019.7208886491487,2021.0,"0.886,0.0,0.778,0.116,0.198"
2017.1588611674556,2017.0,"0.891,0.0,0.679,0.149,0.944"
2017.0,2017.0,"0.908,0.192,0.563,0.846,0.811"
2019.9894303612189,2020.0,"0.913,0.0,0.798,0.112,0.019"
2017.2,2017.0,"0.967,0.115,0.557,0.402,0.954"
2017.75,2017.0,"0.978,0.0,0.582,0.104,0.706"
2017.8,2017.0,"0.984,0.231,0.553,0.884,0.836"
sdarapu@hadoop-nn001:~$
```

Fig 2.B, 18: Output

Discussion Of Results

In this Task, we have performed Random Forest Model on the output generated from the Task 1 to predict the variable ("In which year, the universities got it's score and PR Rank"). We have also trained and tested the data on the basis of 70% and 30% respectively with seed value 80. We have also found out the statistics summary for both train and test data.