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 A:

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

Linear Regression: Linear regression is a type of supervised learning of machine learning algorithm. It carries out a regression task. Based on independent variables, regression models a goal prediction value. It is mostly utilized in forecasting and determining the link between variables. Different regression models differ in terms of the type of relationship they evaluate between dependent and independent variables, as well as the number of independent variables they employ.

Approach:

Below are the steps we have followed to complete Task2 of this assignment,

1. At first, we have created a python file("Assign3_Group4_Task2_PartA.py") in the Hadoop cluster. Refer to "Group_4_Task_2_Part_A_Code.pdf" for code and author comments.

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 LinearRegression – It is imported to perform Linear Regression on the training data.

   b. Created Spark Session

spark = SparkSession.builder.appName("Assign3_Group4_Task2_PartA").getOrCreate()
→ Here, we provided the name to our application by setting a string

"Assign3_Group4_Task2_PartA" 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 Linear Regression

```
lr=LinearRegression(featuresCol = 'features', labelCol='Year')
lr_model=lr.fit(trainingData)
```

- Now, the data(features and Year) is prepared and transformed into a format for LinearRegression.

k. Calculating coefficients and intercepts

```
c=round(lr_model.coefficients[0],2)
s=round(lr_model.intercept,2)
print(f"""the formula for linear regression is Year={c}*features+{s}""")
```

- We have calculated the coefficients and intercepts of the model “lr_model” and have printed the linear regression formula with those values.

l. Transforming the test data

```
lr_predictions = lr_model.transform(testData)
```

- Now, we have transformed the test data and stored the result in “lr_predictions”.

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

```
lr=lr_predictions.select("prediction","Year","features")

lr.show(100)
```

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

n. Converting datatype of features column

```
lr = lr.withColumn('features', vector_to_array('features'))
lr = lr.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.

o. Store result into specified hdfs folder

```
lr.coalesce(1).write.mode("overwrite").option("header","true").csv("hdfs:////user/sdarapu/Assign3_Group4_Task2_PartA_Output")
```

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

3. Steps to execute the code:

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

```
sdarapu@hadoop-nn001:~$ spark-submit /home/sdarapu/Assign3_Group4_Task2_PartA.py
```

Fig 2.A, 1: Command to execute

ii. The above command executes as follows.

```
darapu@hadoop-nn001:~$ spark-submit /home/sdarapu/Assign3_Group4_Task2_PartA.py
ARNING: An illegal reflective access operation has occurred
ARNING: 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
irectByteBuffer(long,int)
ARNING: Please consider reporting this to the maintainers of org.apache.spark.unsafe.Platform
ARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations
ARNING: All illegal access operations will be denied in a future release
022-04-29 15:20:47,026 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applica
022-04-29 15:20:48,313 INFO spark.SparkContext: Running Spark version 3.0.1
022-04-29 15:20:48,364 INFO resource.ResourceUtils: ==============================================================
022-04-29 15:20:48,365 INFO resource.ResourceUtils: Resources for spark.driver:

022-04-29 15:20:48,366 INFO resource.ResourceUtils: ==============================================================
022-04-29 15:20:48,366 INFO spark.SparkContext: Submitted application: Assignment3_Group4_Task2_PartA
022-04-29 15:20:48,437 INFO spark.SecurityManager: Changing view acls to: sdarapu
022-04-29 15:20:48,438 INFO spark.SecurityManager: Changing modify acls to: sdarapu
022-04-29 15:20:48,438 INFO spark.SecurityManager: Changing view acls groups to:
022-04-29 15:20:48,438 INFO spark.SecurityManager: Changing modify acls groups to:
022-04-29 15:20:48,438 INFO spark.SecurityManager: SecurityManager: authentication disabled; ui acls disabled; users  with view permissions: Set(sdar
permissions: Set(); users  with modify permissions: Set(sdarapu); groups with modify permissions: Set()
022-04-29 15:20:48,734 INFO util.Utils: Successfully started service 'sparkDriver' on port 39613.
022-04-29 15:20:48,771 INFO spark.SparkEnv: Registering MapOutputTracker
022-04-29 15:20:48,807 INFO spark.SparkEnv: Registering BlockManagerMaster
022-04-29 15:20:48,828 INFO storage.BlockManagerMasterEndpoint: Using org.apache.spark.storage.DefaultTopologyMapper for getting topology informatio
022-04-29 15:20:48,836 INFO storage.BlockManagerMasterEndpoint: BlockManagerMasterEndpoint up
022-04-29 15:20:48,898 INFO spark.SparkEnv: Registering BlockManagerMasterHeartbeat
022-04-29 15:20:48,917 INFO storage.DiskBlockManager: Created local directory at /tmp/blockmgr-49050c81-7fb5-46ba-bde3-bbeec558994c
```

Fig 2.A, 2: Execution Results

```
2022-04-29 15:20:49,304 INFO handler.ContextHandler: Started o.s.j.s.ServletContextHandler@7efc41a8{/api,null,AVAILABLE,@Spark}
2022-04-29 15:20:49,304 INFO handler.ContextHandler: Started o.s.j.s.ServletContextHandler@644be1b2{/jobs/job/kill,null,AVAILABLE,@Spark}
2022-04-29 15:20:49,305 INFO handler.ContextHandler: Started o.s.j.s.ServletContextHandler@13270404{/stages/stage/kill,null,AVAILABLE,@Spark}
2022-04-29 15:20:49,308 INFO ui.SparkUI: Bound SparkUI to 0.0.0.0, and started at http://hadoop-nn001:4041
2022-04-29 15:20:49,709 INFO client.RMProxy: Connecting to ResourceManager at hadoop-nn001.cs.okstate.edu/192.168.122.2:8032
2022-04-29 15:20:50,016 INFO yarn.Client: Requesting a new application from cluster with 12 NodeManagers
2022-04-29 15:20:50,445 INFO conf.Configuration: resource-types.xml not found
2022-04-29 15:20:50,445 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
2022-04-29 15:20:50,461 INFO yarn.Client: Verifying our application has not requested more than the maximum memory capability of the cluster (8
2022-04-29 15:20:50,466 INFO yarn.Client: Will allocate AM container, with 896 MB memory including 384 MB overhead
2022-04-29 15:20:50,467 INFO yarn.Client: Setting up container launch context for our AM
2022-04-29 15:20:50,472 INFO yarn.Client: Setting up the launch environment for our AM container
2022-04-29 15:20:50,483 INFO yarn.Client: Preparing resources for our AM container
2022-04-29 15:20:50,580 WARN yarn.Client: Neither spark.yarn.jars nor spark.yarn.archive is set, falling back to uploading libraries under SPAR
2022-04-29 15:20:53,510 INFO yarn.Client: Uploading resource file:/tmp/spark-e5d53703-1e7f-4043-9142-95918adcada2/__spark_libs__88286699292043
01.cs.okstate.edu:9000/user/sdarapu/.sparkStaging/application_1647031195237_1352/__spark_libs__8828669929204300882.zip
2022-04-29 15:20:55,985 INFO yarn.Client: Uploading resource file:/usr/local/spark/python/lib/pyspark.zip -> hdfs://hadoop-nn001.cs.okstate.edu
ing/application_1647031195237_1352/pyspark.zip
2022-04-29 15:20:56,038 INFO yarn.Client: Uploading resource file:/usr/local/spark/python/lib/py4j-0.10.9-src.zip -> hdfs://hadoop-nn001.cs.oks
parkStaging/application_1647031195237_1352/py4j-0.10.9-src.zip
2022-04-29 15:20:56,279 INFO yarn.Client: Uploading resource file:/tmp/spark-e5d53703-1e7f-4043-9142-95918adcada2/__spark_conf__798537475658488
01.cs.okstate.edu:9000/user/sdarapu/.sparkStaging/application_1647031195237_1352/__spark_conf__.zip
2022-04-29 15:20:56,339 INFO spark.SecurityManager: Changing view acls to: sdarapu
2022-04-29 15:20:56,339 INFO spark.SecurityManager: Changing modify acls to: sdarapu
2022-04-29 15:20:56,339 INFO spark.SecurityManager: Changing view acls groups to:
2022-04-29 15:20:56,339 INFO spark.SecurityManager: Changing modify acls groups to:
2022-04-29 15:20:56,340 INFO spark.SecurityManager: SecurityManager: authentication disabled; ui acls disabled; users  with view permissions: S
 permissions: Set(); users  with modify permissions: Set(sdarapu); groups with modify permissions: Set()
2022-04-29 15:20:56,366 INFO yarn.Client: Submitting application application_1647031195237_1352 to ResourceManager
2022-04-29 15:20:56,402 INFO impl.YarnClientImpl: Submitted application application_1647031195237_1352
2022-04-29 15:20:57,407 INFO yarn.Client: Application report for application_1647031195237_1352 (state: ACCEPTED)
2022-04-29 15:20:57,413 INFO yarn.Client:
```

Fig 2.A, 3: Execution Results

```
        start time: 1651263656380
        final status: UNDEFINED
        tracking URL: http://hadoop-nn001.cs.okstate.edu:8088/proxy/application_1647031195237_1352/
        user: sdarapu
2022-04-29 15:21:01,433 INFO cluster.YarnClientSchedulerBackend: Application application_1647031195237_1352 has started running.
2022-04-29 15:21:01,449 INFO util.Utils: Successfully started service 'org.apache.spark.network.netty.NettyBlockTransferService' on port 46527.
2022-04-29 15:21:01,450 INFO netty.NettyBlockTransferService: Server created on hadoop-nn001:46527
2022-04-29 15:21:01,453 INFO storage.BlockManager: Using org.apache.spark.storage.RandomBlockReplicationPolicy for block replication policy
2022-04-29 15:21:01,466 INFO storage.BlockManagerMaster: Registering BlockManager BlockManagerId(driver, hadoop-nn001, 46527, None)
2022-04-29 15:21:01,472 INFO storage.BlockManagerMasterEndpoint: Registering block manager hadoop-nn001:46527 with 434.4 MiB RAM, BlockManagerId(driver, hadoop-nn0
, None)
2022-04-29 15:21:01,477 INFO storage.BlockManagerMaster: Registered BlockManager BlockManagerId(driver, hadoop-nn001, 46527, None)
2022-04-29 15:21:01,479 INFO storage.BlockManager: Initialized BlockManager: BlockManagerId(driver, hadoop-nn001, 46527, None)
2022-04-29 15:21:01,670 INFO ui.ServerInfo: Adding filter to /metrics/json: org.apache.hadoop.yarn.server.webproxy.amfilter.AmIpFilter
2022-04-29 15:21:01,673 INFO handler.ContextHandler: Started o.s.j.s.ServletContextHandler@4d64580c{/metrics/json,null,AVAILABLE,@Spark}
2022-04-29 15:21:01,971 INFO cluster.YarnSchedulerBackend$YarnSchedulerEndpoint: ApplicationMaster registered as NettyRpcEndpointRef(spark-client://YarnAM)
2022-04-29 15:21:06,503 INFO resource.ResourceProfile: Default ResourceProfile created, executor resources: Map(cores -> name: cores, amount: 1, script: , vendor:
-> name: memory, amount: 1024, script: , vendor: ), task resources: Map(cpus -> name: cpus, amount: 1.0)
2022-04-29 15:21:07,196 INFO cluster.YarnSchedulerBackend$YarnDriverEndpoint: Registered executor NettyRpcEndpointRef(spark-client://Executor) (10.221.247.13:34738
 1
2022-04-29 15:21:07,229 INFO cluster.YarnSchedulerBackend$YarnDriverEndpoint: Registered executor NettyRpcEndpointRef(spark-client://Executor) (10.221.247.18:52072
 2
2022-04-29 15:21:07,327 INFO cluster.YarnClientSchedulerBackend: SchedulerBackend is ready for scheduling beginning after reached minRegisteredResourcesRatio: 0.8
2022-04-29 15:21:07,345 INFO storage.BlockManagerMasterEndpoint: Registering block manager hadoop-dn001.cs.okstate.edu:39135 with 434.4 MiB RAM, BlockManagerId(1, h
001.cs.okstate.edu, 39135, None)
```

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

```
+-------+------------------+
|summary|              Year|
+-------+------------------+
|  count|               349|
|   mean|2019.0601719197707|
| stddev|1.4180044964555496|
|    min|            2017.0|
|    max|            2021.0|
+-------+------------------+
```

Fig 2.A, 6: Statistics summary of train data

```
+-------+------------------+
|summary|              Year|
+-------+------------------+
|  count|               152|
|   mean|2018.8486842105262|
| stddev|1.4083942687234334|
|    min|            2017.0|
|    max|            2021.0|
+-------+------------------+
```

Fig 2.A, 7: Statistics summary of test data

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

Fig 2.A, 8: Displays number of train records

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

Fig 2.A, 9: Displays number of test records

```
the formula for linear regression is Year=0.72*features+2019.41
```

Fig 2.A,10: Formula for Linear regression

prediction	Year	features
2019.6599680249049	2019.0	[0.011,0.0,0.919,...
2019.7068695042635	2020.0	[0.011,0.0,0.965,...
2016.9526883493056	2017.0	[0.016,0.0,0.877,...
2018.3079382199621	2018.0	[0.022,0.115,0.98...
2019.124822827811	2018.0	[0.038,0.0,0.856,...
2019.5989792076111	2021.0	[0.038,0.0,0.876,...
2019.6108093721991	2019.0	[0.038,0.0,0.888,...
2016.3784155412097	2017.0	[0.049,0.038,0.60...
2020.0829923571039	2021.0	[0.049,0.038,0.65...
2019.5629441635174	2019.0	[0.054,0.231,0.94...
2016.4310669178797	2017.0	[0.06,0.269,0.602...
2019.7920963222393	2019.0	[0.06,0.269,0.659...
2019.8593262626084	2021.0	[0.06,0.269,0.688...
2016.4429517220756	2017.0	[0.076,0.346,0.69...
2018.9371283132239	2018.0	[0.076,0.346,0.70...
2020.0737156980294	2019.0	[0.076,0.346,0.71...
2020.0973760272054	2020.0	[0.076,0.346,0.73...
2019.9178120161619	2020.0	[0.082,0.154,0.62...
2019.9796926800836	2019.0	[0.082,0.154,0.64...
2018.5560702047862	2018.0	[0.087,0.308,0.66...
2019.629987659562	2021.0	[0.087,0.308,0.71...
2018.6876945553227	2018.0	[0.092,0.038,0.76...
2019.8917530383087	2020.0	[0.092,0.038,0.87...
2018.2007661604232	2018.0	[0.098,0.038,0.79...
2019.8417642613454	2020.0	[0.098,0.038,0.84...
2017.3168362396666	2017.0	[0.103,0.231,0.89...
2018.4456122878844	2018.0	[0.103,0.231,0.92...
2019.5758057921732	2021.0	[0.103,0.231,0.93...
2019.5997619616849	2020.0	[0.103,0.231,0.96...
2017.1046111707344	2017.0	[0.109,0.154,0.69...
2020.1484769391539	2020.0	[0.109,0.154,0.81...
2018.9562888714165	2018.0	[0.12,0.154,0.801...
2020.060554275545	2019.0	[0.12,0.154,0.802...
2020.060554275545	2020.0	[0.12,0.154,0.802...
2019.8527542303623	2019.0	[0.125,0.077,0.64...
2017.3727435070696	2017.0	[0.13,0.462,0.633...
2019.8709857820218	2020.0	[0.13,0.462,0.814...
2019.8554403994342	2021.0	[0.141,0.769,0.69...

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

2019.8554403994342	2021.0	[0.141,0.769,0.69...
2020.2272015052897	2021.0	[0.147,0.385,0.76...
2020.2323532674434	2019.0	[0.147,0.385,0.77...
2020.2550027369552	2020.0	[0.147,0.385,0.79...
2020.2595439477984	2020.0	[0.152,0.654,0.80...
2018.4170124462992	2018.0	[0.158,0.0,0.604,...
2019.505160321904	2021.0	[0.158,0.0,0.647,...
2018.269757449315	2018.0	[0.163,0.538,0.66...
2019.7768929639376	2021.0	[0.163,0.538,0.69...
2019.7620095187888	2019.0	[0.163,0.538,0.7,...
2019.7994441074072	2020.0	[0.163,0.538,0.71...
2016.6032196770363	2017.0	[0.174,0.0,0.605,...
2018.3117742176364	2018.0	[0.174,0.0,0.686,...
2020.317189136891	2020.0	[0.174,0.0,0.756,...
2017.0722872382669	2017.0	[0.185,0.115,0.59...
2019.3622852796952	2019.0	[0.185,0.115,0.69...
2017.213540225763	2017.0	[0.19,0.538,0.557...
2018.3790161631598	2018.0	[0.196,0.192,0.67...
2020.1439677161513	2019.0	[0.196,0.192,0.7,...
2017.2292828432921	2017.0	[0.201,0.346,0.68...
2020.4294441402906	2019.0	[0.212,0.385,0.95...
2020.1250785650102	2019.0	[0.217,0.615,0.63...
2018.3345775499236	2018.0	[0.217,0.615,0.67...
2020.168626230091	2020.0	[0.223,0.615,0.73...
2017.5936011215204	2017.0	[0.223,0.962,0.56...
2020.3433839851793	2019.0	[0.228,0.0,0.798,...
2020.3621150791103	2020.0	[0.228,0.0,0.817,...
2019.6604684147796	2021.0	[0.234,0.0,0.879,...
2019.5134412652421	2021.0	[0.239,0.231,0.66...
2019.4790953163122	2019.0	[0.239,0.231,0.72...
2018.564732255747	2018.0	[0.239,0.231,0.74...
2018.1598944647476	2018.0	[0.245,0.192,0.78...
2020.081570131645	2021.0	[0.25,0.846,0.623...
2018.293401163434	2018.0	[0.255,0.154,0.63...
2020.1723372856	2021.0	[0.255,0.154,0.74...
2017.6443124210725	2017.0	[0.261,0.346,0.56...
2019.39244558066	2019.0	[0.261,0.346,0.61...
2019.4230801526057	2020.0	[0.261,0.346,0.64...
2018.461470032169	2018.0	[0.266,0.038,0.67...
2020.364114842615	2020.0	[0.266,0.038,0.74...
2020.1663219595391	2019.0	[0.272,0.192,0.61...

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

```
|2020.1663219595391|2019.0|[0.272,0.192,0.61...|
|2020.1694718647375|2020.0|[0.283,0.077,0.88...|
|2020.1694718647375|2021.0|[0.283,0.077,0.88...|
|2020.0191695294468|2019.0|[0.288,0.077,0.73...|
| 2018.461934551142|2018.0|[0.293,0.231,0.60...|
|2019.2976536823496|2018.0|[0.299,0.115,0.82...|
| 2017.494406109878|2017.0|[0.304,0.423,0.59...|
| 2019.364453398921|2019.0|[0.304,0.423,0.60...|
|2018.8826087784796|2018.0|[0.31,0.077,0.621...|
| 2020.039699488902|2019.0|[0.31,0.077,0.635...|
|2019.0511006912238|2018.0|[0.326,0.077,0.89...|
|2020.1857420330562|2019.0|[0.326,0.077,0.90...|
|2017.6724929418676|2017.0|[0.332,0.885,0.6,...|
|2018.9210934965931|2018.0|[0.332,0.885,0.61...|
|2017.4631724863552|2017.0|[0.337,0.077,0.66...|
|2019.9621073481455|2021.0|[0.342,0.077,0.64...|
|2019.9325871351623|2020.0|[0.342,0.077,0.65...|
|  2018.13612687311|2018.0|[0.348,0.038,0.66...|
| 2020.293281504884|2021.0|[0.348,0.038,0.70...|
| 2019.708864302203|2020.0|[0.353,0.0,0.812,...|
| 2018.805166462856|2018.0|[0.364,0.038,0.69...|
|2018.8284011952387|2018.0|[0.37,0.038,0.739...|
|2019.9070480208886|2021.0|[0.37,0.038,0.753...|
+------------------+------+--------------------+
only showing top 100 rows

2022-04-29 15:21:32,180 INFO datasources.FileSourceStrategy: Pruning directories with:
2022-04-29 15:21:32,180 INFO datasources.FileSourceStrategy: Pushed Filters:
2022-04-29 15:21:32,180 INFO datasources.FileSourceStrategy: Post-Scan Filters:
2022-04-29 15:21:32,180 INFO datasources.FileSourceStrategy: Output Data Schema: struct<INSTITUTE ID: double, NAME: double, STATE:
```

Fig 2.A, 13: 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_PartA_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_PartA_Output
```

Fig 2.A,14: View list of contents

```
sdarapu@hadoop-nn001:~$ hdfs dfs -ls /user/sdarapu/Assign3_Group4_Task2_PartA_Output
2022-04-29 15:41:09,710 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 15:21 /user/sdarapu/Assign3_Group4_Task2_PartA_Output/_SUCCESS
-rw-r--r--   3 sdarapu sdarapu       8668 2022-04-29 15:21 /user/sdarapu/Assign3_Group4_Task2_PartA_Output/part-00000-06477ac9-bfd5-4f53-b5df-ec8d5d9cf7a8-c000.csv
sdarapu@hadoop-nn001:~$
```

Fig 2.A,15: List of files in the respective folder

ii. To view the data in the folder "Assign3_Group4_Task2_PartA_Output" at once, execute the below command.

```
sdarapu@hadoop-nn001:~$ hdfs dfs -cat /user/sdarapu/Assign3_Group4_Task2_PartA_Output/part*
```

Fig 2.A,16: Command to view data

Output Display

```
sdarapu@hadoop-nn001:~$ hdfs dfs -cat /user/sdarapu/Assign3_Group4_Task2_PartA_Output/part*
2022-04-29 15:43:22,756 WARN util.NativeCodeLoader: Unable to load native-hadoop library for
prediction,Year,features
2019.6599680249049,2019.0,"0.011,0.0,0.919,0.041,0.176"
2019.7068695042635,2020.0,"0.011,0.0,0.965,0.05,0.176"
2016.9526883493056,2017.0,"0.016,0.0,0.877,0.017,0.879"
2018.3079382199621,2018.0,"0.022,0.115,0.984,0.012,0.557"
2019.124822827811,2018.0,"0.038,0.0,0.856,0.066,0.307"
2019.5989792076111,2021.0,"0.038,0.0,0.876,0.041,0.186"
2019.6108093721991,2019.0,"0.038,0.0,0.888,0.041,0.186"
2016.3784155412097,2017.0,"0.049,0.038,0.606,0.029,0.969"
2020.0829923571039,2021.0,"0.049,0.038,0.658,0.108,0.009"
2019.5629441635174,2019.0,"0.054,0.231,0.944,0.025,0.229"
2016.4310669178797,2017.0,"0.06,0.269,0.602,0.278,0.981"
2019.7920963222393,2019.0,"0.06,0.269,0.659,0.249,0.108"
2019.8593262626084,2021.0,"0.06,0.269,0.688,0.473,0.108"
2016.4429517220756,2017.0,"0.076,0.346,0.693,0.079,1.0"
2018.9371283132239,2018.0,"0.076,0.346,0.708,0.232,0.353"
2020.0737156980294,2019.0,"0.076,0.346,0.71,0.062,0.046"
2020.0973760272054,2020.0,"0.076,0.346,0.734,0.062,0.046"
2019.9178120161619,2020.0,"0.082,0.154,0.629,0.129,0.059"
2019.9796926800836,2019.0,"0.082,0.154,0.644,0.402,0.059"
2018.5560702047862,2018.0,"0.087,0.308,0.661,0.207,0.44"
2019.629987659562,2021.0,"0.087,0.308,0.712,0.075,0.164"
2018.6876945553227,2018.0,"0.092,0.038,0.762,0.17,0.415"
2019.8917530383087,2020.0,"0.092,0.038,0.87,0.137,0.124"
2018.2007661604232,2018.0,"0.098,0.038,0.799,0.1,0.551"
2019.8417642613454,2020.0,"0.098,0.038,0.841,0.054,0.127"
2017.3168362396666,2017.0,"0.103,0.231,0.89,0.033,0.817"
2018.4456122878844,2018.0,"0.103,0.231,0.926,0.041,0.529"
2019.5758057921732,2021.0,"0.103,0.231,0.936,0.008,0.232"
2019.5997619616849,2020.0,"0.103,0.231,0.961,0.004,0.232"
2017.1046111707344,2017.0,"0.109,0.154,0.699,0.452,0.839"
2020.1484769391539,2020.0,"0.109,0.154,0.818,0.34,0.062"
2018.9562888714165,2018.0,"0.12,0.154,0.801,0.008,0.359"
2020.060554275545,2019.0,"0.12,0.154,0.802,0.008,0.068"
2020.060554275545,2020.0,"0.12,0.154,0.802,0.008,0.068"
2019.8527542303623,2019.0,"0.125,0.077,0.64,0.357,0.093"
2017.3727435070696,2017.0,"0.13,0.462,0.633,0.054,0.755"
2019.8709857820218,2020.0,"0.13,0.462,0.814,0.183,0.149"
2019.8554403994342,2021.0,"0.141,0.769,0.69,0.295,0.146"
```

Fig 2.A, 17: Output

```
2019.8554403994342,2021.0,"0.141,0.769,0.69,0.295,0.146"
2020.2272015052897,2021.0,"0.147,0.385,0.765,0.095,0.037"
2020.2323532674434,2019.0,"0.147,0.385,0.776,0.062,0.037"
2020.2550027369552,2020.0,"0.147,0.385,0.796,0.079,0.037"
2020.2595439477984,2020.0,"0.152,0.654,0.806,0.033,0.053"
2018.4170124462992,2018.0,"0.158,0.0,0.604,0.29,0.461"
2019.505160321904,2021.0,"0.158,0.0,0.647,0.506,0.195"
2018.269757449315,2018.0,"0.163,0.538,0.661,0.112,0.539"
2019.7768929639376,2021.0,"0.163,0.538,0.692,0.232,0.155"
2019.7620095187888,2019.0,"0.163,0.538,0.7,0.1,0.155"
2019.7994441074072,2020.0,"0.163,0.538,0.714,0.237,0.155"
2016.6032196770363,2017.0,"0.174,0.0,0.605,0.253,0.941"
2018.3117742176364,2018.0,"0.174,0.0,0.686,0.112,0.505"
2020.317189136891,2020.0,"0.174,0.0,0.756,0.502,0.012"
2017.0722872382669,2017.0,"0.185,0.115,0.594,0.27,0.824"
2019.3622852796952,2019.0,"0.185,0.115,0.693,0.1,0.238"
2017.213540225763,2017.0,"0.19,0.538,0.557,0.054,0.793"
2018.3790161631598,2018.0,"0.196,0.192,0.678,0.473,0.517"
2020.1439677161513,2019.0,"0.196,0.192,0.7,0.095,0.04"
2017.2292828432921,2017.0,"0.201,0.346,0.683,0.22,0.82"
2020.4294441402906,2019.0,"0.212,0.385,0.959,0.021,0.043"
2020.1250785650102,2019.0,"0.217,0.615,0.632,0.556,0.077"
2018.3345775499236,2018.0,"0.217,0.615,0.674,0.552,0.56"
2020.168626230091,2020.0,"0.223,0.615,0.732,0.278,0.08"
2017.5936011215204,2017.0,"0.223,0.962,0.569,0.759,0.759"
2020.3433839851793,2019.0,"0.228,0.0,0.798,0.058,0.006"
2020.3621150791103,2020.0,"0.228,0.0,0.817,0.058,0.006"
2019.6604684147796,2021.0,"0.234,0.0,0.879,0.029,0.207"
2019.5134412652421,2021.0,"0.239,0.231,0.668,0.68,0.235"
2019.4790953163122,2019.0,"0.239,0.231,0.721,0.178,0.235"
2018.564732255747,2018.0,"0.239,0.231,0.742,0.274,0.486"
2018.1598944647476,2018.0,"0.245,0.192,0.784,0.174,0.598"
2020.081570131645,2021.0,"0.25,0.846,0.623,0.075,0.084"
2018.293401163434,2018.0,"0.255,0.154,0.633,0.104,0.52"
2020.1723372856,2021.0,"0.255,0.154,0.748,0.141,0.056"
2017.6443124210725,2017.0,"0.261,0.346,0.569,0.589,0.709"
2019.39244558066,2019.0,"0.261,0.346,0.615,0.328,0.248"
2019.4230801526057,2020.0,"0.261,0.346,0.641,0.357,0.248"
2018.461470032169,2018.0,"0.266,0.038,0.674,0.552,0.502"
2020.364114842615,2020.0,"0.266,0.038,0.746,0.137,0.0"
2020.1663219595391,2019.0,"0.272,0.192,0.617,0.12,0.028"
```

Fig 2.A, 18: Output

```
2020.1663219595391,2019.0,"0.272,0.192,0.617,0.12,0.028"
2020.1694718647375,2020.0,"0.283,0.077,0.886,0.0,0.087"
2020.1694718647375,2021.0,"0.283,0.077,0.886,0.0,0.087"
2020.0191695294468,2019.0,"0.288,0.077,0.732,0.054,0.09"
2018.461934551142,2018.0,"0.293,0.231,0.608,0.154,0.483"
2019.2976536823496,2018.0,"0.299,0.115,0.82,0.307,0.319"
2017.494406109878,2017.0,"0.304,0.423,0.599,0.436,0.762"
2019.364453398921,2019.0,"0.304,0.423,0.604,0.149,0.257"
2018.8826087784796,2018.0,"0.31,0.077,0.621,0.602,0.39"
2020.039699488902,2019.0,"0.31,0.077,0.635,0.768,0.096"
2019.0511006912238,2018.0,"0.326,0.077,0.89,0.041,0.393"
2020.1857420330562,2019.0,"0.326,0.077,0.908,0.054,0.099"
2017.6724929418676,2017.0,"0.332,0.885,0.6,0.884,0.768"
2018.9210934965931,2018.0,"0.332,0.885,0.616,0.602,0.43"
2017.4631724863552,2017.0,"0.337,0.077,0.669,0.361,0.771"
2019.9621073481455,2021.0,"0.342,0.077,0.641,0.349,0.105"
2019.9325871351623,2020.0,"0.342,0.077,0.659,0.075,0.105"
2018.13612687311,2018.0,"0.348,0.038,0.668,0.602,0.604"
2020.293281504884,2021.0,"0.348,0.038,0.708,0.087,0.022"
2019.708864302203,2020.0,"0.353,0.0,0.812,0.133,0.204"
2018.805166462856,2018.0,"0.364,0.038,0.696,0.232,0.421"
2018.8284011952387,2018.0,"0.37,0.038,0.739,0.162,0.424"
2019.9070480208886,2021.0,"0.37,0.038,0.753,0.137,0.142"
2018.5699336527966,2018.0,"0.375,0.231,0.59,0.411,0.477"
2019.5667594132833,2021.0,"0.375,0.231,0.625,0.473,0.226"
2019.679513051622,2021.0,"0.38,0.115,0.792,0.037,0.214"
2017.2108599681364,2017.0,"0.386,0.577,0.66,0.411,0.876"
2019.1165368414374,2018.0,"0.391,0.346,0.607,0.411,0.347"
2019.4639730297868,2021.0,"0.391,0.346,0.613,0.149,0.245"
2020.306541790496,2019.0,"0.413,0.385,0.625,0.12,0.031"
2017.5047899726155,2017.0,"0.424,0.0,0.572,0.763,0.765"
2018.1016851422137,2020.0,"0.435,0.192,0.657,0.17,0.616"
2017.9034673320828,2019.0,"0.446,0.0,0.639,0.556,0.672"
2017.4178881623166,2017.0,"0.446,0.0,0.663,0.593,0.808"
2019.425623931968,2021.0,"0.457,0.077,0.656,0.083,0.26"
2018.1077545564224,2019.0,"0.467,0.423,0.661,0.17,0.635"
2019.4591249502341,2020.0,"0.478,0.038,0.705,0.224,0.272"
2019.1843551798006,2018.0,"0.484,0.731,0.63,0.473,0.378"
2018.9127406781276,2018.0,"0.489,0.808,0.726,0.274,0.471"
2019.5651681410975,2020.0,"0.505,0.423,0.632,0.253,0.254"
2019.3971126350173,2020.0,"0.516,0.0,0.665,0.402,0.291"
```

Fig 2.A, 19: Output

```
2019.3971126350173,2020.0,"0.516,0.0,0.665,0.402,0.291"
2018.3482419953243,2018.0,"0.522,0.0,0.604,0.162,0.542"
2020.4646130144788,2020.0,"0.527,0.192,0.759,0.112,0.034"
2019.4708243929613,2020.0,"0.538,0.308,0.651,0.224,0.282"
2019.7475874395482,2021.0,"0.543,0.0,0.676,0.083,0.192"
2019.3214626268452,2019.0,"0.549,0.0,0.607,0.224,0.294"
2016.9059171392691,2017.0,"0.56,0.077,0.0,0.095,0.774"
2018.1711000105386,2019.0,"0.576,0.308,0.697,0.423,0.653"
2018.95215187395,2018.0,"0.576,0.308,0.71,0.261,0.443"
2018.4052805384872,2019.0,"0.582,0.923,0.602,0.826,0.622"
2020.5336984922253,2019.0,"0.592,0.0,0.737,0.286,0.019"
2018.1663811459757,2019.0,"0.598,0.192,0.629,0.095,0.619"
2018.1101757734057,2019.0,"0.609,0.077,0.619,0.61,0.65"
2017.2127091167338,2017.0,"0.63,0.115,0.613,0.075,0.867"
2018.0866470575181,2021.0,"0.674,0.115,0.666,0.436,0.675"
2017.9971181170654,2017.0,"0.701,0.731,0.618,0.1,0.712"
2017.7807479081266,2017.0,"0.707,0.192,0.551,0.095,0.721"
2017.4524865912626,2017.0,"0.712,0.692,0.582,0.274,0.854"
2017.464379068896,2017.0,"0.717,0.308,0.627,0.647,0.858"
2018.0552862837444,2017.0,"0.723,0.154,0.796,0.137,0.715"
2019.5983565987806,2021.0,"0.755,0.077,0.609,0.183,0.263"
2018.045850736756,2017.0,"0.766,0.192,0.647,0.311,0.697"
2017.2234762618277,2017.0,"0.777,0.0,0.577,0.365,0.889"
2017.2556294723902,2017.0,"0.804,0.269,0.577,0.423,0.904"
2018.2604173884179,2019.0,"0.81,0.077,0.614,0.61,0.647"
2017.14288799908,2017.0,"0.821,0.077,0.557,0.324,0.916"
2019.2849304031936,2018.0,"0.826,1.0,0.597,0.689,0.433"
2018.4393310526539,2021.0,"0.848,0.577,0.647,0.328,0.632"
2020.0764730092505,2021.0,"0.886,0.0,0.778,0.116,0.198"
2017.1598050841067,2017.0,"0.891,0.0,0.679,0.149,0.944"
2017.724703185782,2017.0,"0.908,0.192,0.563,0.846,0.811"
2020.7934655606293,2020.0,"0.913,0.0,0.798,0.112,0.019"
2017.1251554699008,2017.0,"0.967,0.115,0.557,0.402,0.954"
2018.0209943724253,2017.0,"0.978,0.0,0.582,0.104,0.706"
2017.6896429636656,2017.0,"0.984,0.231,0.553,0.884,0.836"
sdarapu@hadoop-nn001:~$
```

Fig 2.A, 20: Output

Discussion Of Results

In this Task, we have performed Linear Regression 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 as well as found out the formula for linear regression for the prediction.