CS 5433: Bigdata Management
Programming Assignment 3
Task3 – Measure Accuracy for Random Forest

Group 4

Task 3: Measure the accuracy of both predictions. You may use RMSE (Root Mean Square Error) or some other metric.

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.

→ In this Task, we have used RMSE and R2 measures for finding the accuracy of the model.

RMSE: It is also called as Root Mean Squared Error. The standard deviation of the errors that occur when making a prediction on a dataset is known as the RMSE. This is the same as MSE (Mean Squared Error), but the root of the number is taken into account when calculating the model's accuracy.

R2: The R2 score is a critical indicator for assessing the effectiveness of a regression-based machine learning model. It's also known as the coefficient of determination and is called as R squared. It operates by calculating the amount of variation in the dataset-explained predictions.

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[ind
exx]+"_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(20)
```

➔ Now, we have retrieved the columns “prediction”, “Year”, “features” from rf_predictions dataframe and stored the result into “rf”. After that, displayed the first 20 rows on the console by using show() method.

m. Calculation of RMSE and R2 Value

```
evaluator = RegressionEvaluator(labelCol="Year", predictionCol="prediction", metricName="rmse")
print("RMSE:",evaluator.evaluate(rf_predictions))
evaluator = RegressionEvaluator(labelCol="Year", predictionCol="prediction", metricName="r2")
print("R2:",evaluator.evaluate(rf_predictions))
```

➔ RegressionEvaluator function is used on prediction col and label col for calculating rmse and r2 value for measuring the accuracy.

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_Task3_PartB.py
```

Fig 3.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 3.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://ha
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/.sp
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/sda
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://ha
001.cs.okstate.edu:9000/user/sdarapu/.sparkStaging/application_1647031195237_1387/__spark_conf__.zip
```

Fig 3.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 3.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 3.B, 5: Schema of the data frame

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

Fig 3.B, 6: Displays number of train records

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

Fig 3.B, 7: Displays number of test records

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

Fig 3.B, 8: Statistics summary of train data

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

Fig 3.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...

only showing top 20 rows

Fig 3.B, 10: Display of Prediction, Year and features columns

R2 Value: 0.7419719488388121

Fig 3.B, 11: R2 Value

RMSE Value: 0.7130572335377876

Fig 3.B, 12: RMSE Value

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.

At last, we found out the accuracy of the model by calculating RMSE and R2 values.