CS 5433: Bigdata Management
Programming Assignment 3
Task 3– Measure Accuracy for Linear Regression

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 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.

→ 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.

<u>Approach:</u>

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_PartA.py") in the Hadoop cluster. Refer to "Group_4_Task_3_Part_A_Code.pdf" for code and author comments.

2. **<u>Code Explanation:</u>**

    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_Task3_PartA").getOrCreate() → Here, we provided the name to our application by setting a string "Assign3_Group4_Task3_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(20)
```

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

n. Calculation of RMSE and R2 Value

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

```
evaluator = RegressionEvaluator(labelCol="Year", predictionCol="prediction",
metricName="r2")
print("R2:",evaluator.evaluate(lr_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_PartA.py
```

Fig 3.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
ermissions: 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 3.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__882866992920430
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 3.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, 
001.cs.okstate.edu, 39135, None)
```

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

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

Fig 3.A, 6: Statistics summary of train data

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

Fig 3.A, 7: Statistics summary of test data

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

Fig 3.A, 8: Displays number of train records

Fig 3.A, 9: Displays number of test records

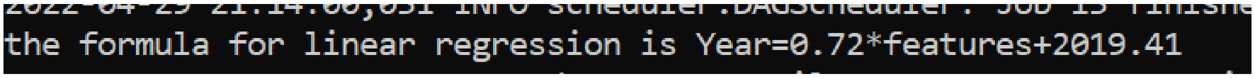

Fig 3.A,10: Formula for Linear regression

```
+------------------+------+--------------------+
|        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...|
|2019.859326262608 4|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...|
+------------------+------+--------------------+
only showing top 20 rows
```

Fig 3.A, 11: Display of Prediction, Year and features columns

```
R2: 0.6169654641635698
```

Fig 3.A, 12: R2 value

```
RMSE: 0.868780161820877
```

Fig 3.A, 13: RMSE Value

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.

At last, we found out the accuracy of the model in terms of rmse and r2 values.