CIS5560 Term Project Tutorial

Authors: Adnan Elahi; Jinhui Liu; Shilpa Konde Deshmukh; Siying Chen

Instructor: Jongwook Woo

Date: 05/18/2021

Lab Tutorial

Adnan Elahi (aelahi@calstatela.edu)

Jinhui Liu (jliu2@calstatela.edu)

Shilpa Konde Deshmukh (skonded@calstatela.edu)

Siying Chen (schen112@calstatela.edu)

05/18/2021

Fare Prediction for New York taxi

Objectives

The aim of this tutorial is to predict the fare by giving trip distance, trip time, and number of passengers. In this hands-on lab, you will learn how to:

- Get data manually
- AzureML Model
- Create Spark cluster
- Spark ML Model
- Visualization
- https://gallery.cortanaintelligence.com/Experiment/Kaggle-Final

- https://gallery.cortanaintelligence.com/Experiment/FareTrip-Final

Platform Spec

- Microsoft Azure Machine Learning
- Databricks Community Edition
- # of CPU cores: 1(AZML)/2(Databricks)
- # of nodes: 1
- Total Memory Size: 10GB(AZML)/15.3GB(Databricks)

Task 1: Get data manually and modify the data

This step is to get data manually, and prepare the datasets for azure ml and databricks

1. Download the first dataset from Kaggle: https://www.kaggle.com/microize/newyork-yellow-taxi-trip-data-2020-2019 (we need the data from 2019/11 to 2020/02 here).

2. Download the second dataset from Chris Whong: https://chriswhong.com/open-data/foil_nyc_taxi/ (we need both Fare and Trip data).

3. Put all the Kaggle datasets in a new folder and combine the Kaggle datasets by following cmd code:

```
cat *.csv > KaggleTaxi.csv
```

4. Copy the first 113400 rows from KaggleTaxi.csv for a 10MB sample dataset by following cmd

   code:

```
head -n 113400 KaggleTaxi.csv > KaggleTaxiSample.csv
```

5. Do the same thing for fare and trip sample. 92530 rows for fareSample.csv and 62700 rows for

   tripSample.csv

Task 2: Azure Machine Learning

This step will introduce Azure ML

1. Open a browser and sign in using the Microsoft account associated with your Azure ML account at https://studio.azureml.net/

2. Click New at bottom left -> Dataset -> From Local File -> select **KaggleTaxiSample.csv** to upload the sample dataset to Azure.

3. Click New -> Experiment -> Blank Experiment and rename it to KaggleTaxi.

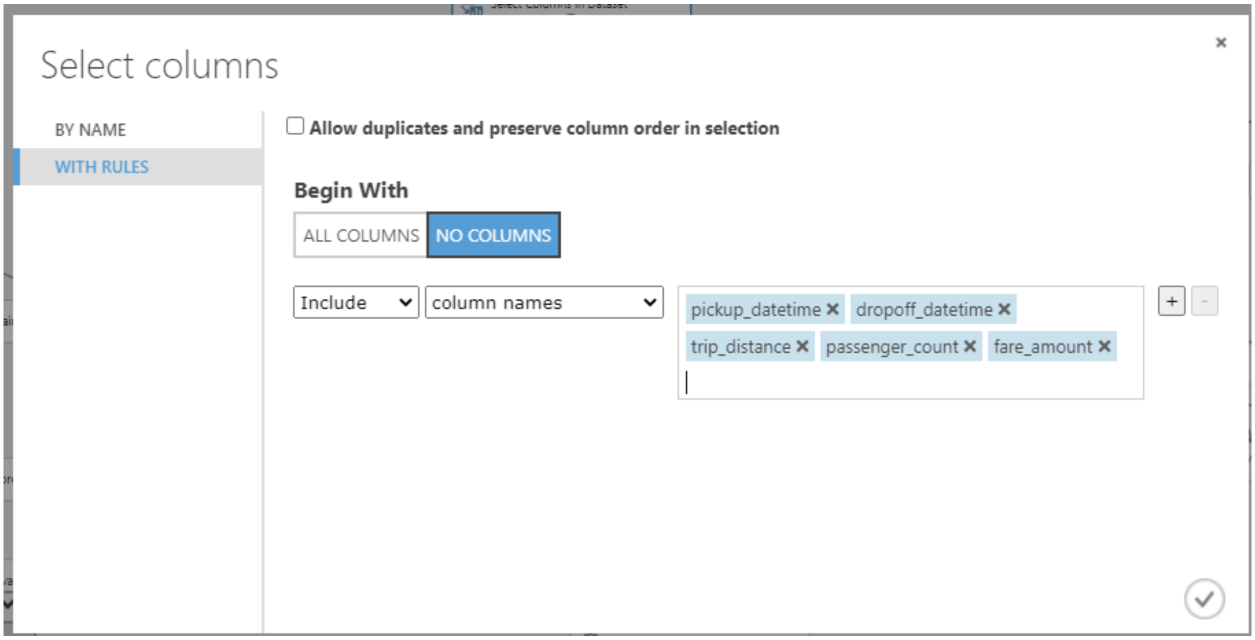
4. Click Saved Datasets on the left -> My Datasets and drag KaggleTaxiSample.csv to the canvas.

5. Search for "**Select Columns in Dataset**" and drag it to the canvas under Dataset module.

6. Link the Dataset output to the input of **Select Columns** in Dataset module.

7. Click **Select Columns** in Dataset module and click **Launch Column Selector**.

8. Choose **With Rules**, **No Columns** and select pickup_datetime, dropoff_datetime, trip_distance, passenger_count and fare_amount.



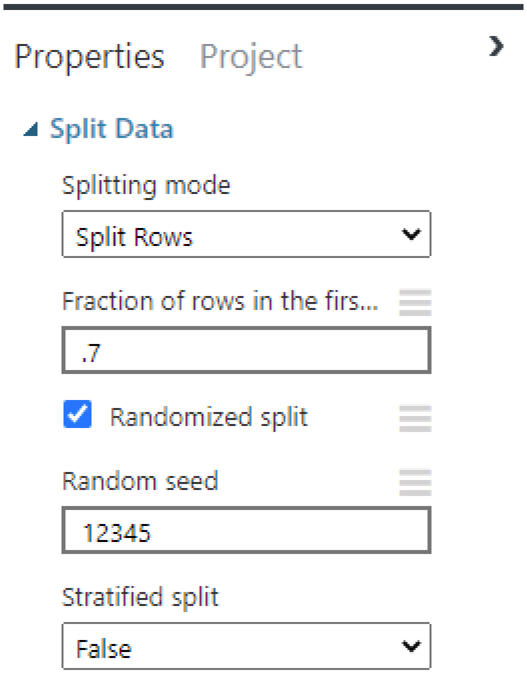
Select columns

□ Allow duplicates and preserve column order in selection

Begin With

Include | column names | pickup_datetime ✕ dropoff_datetime ✕ trip_distance ✕ passenger_count ✕ fare_amount ✕

9. Search and drag **Split Data** to the canvas, connect it to the select columns module. And change the properties as in the picture.



Properties Project

Split Data

Splitting mode

Split Rows

Fraction of rows in the firs...

.7

☑ Randomized split

Random seed

12345

Stratified split

False

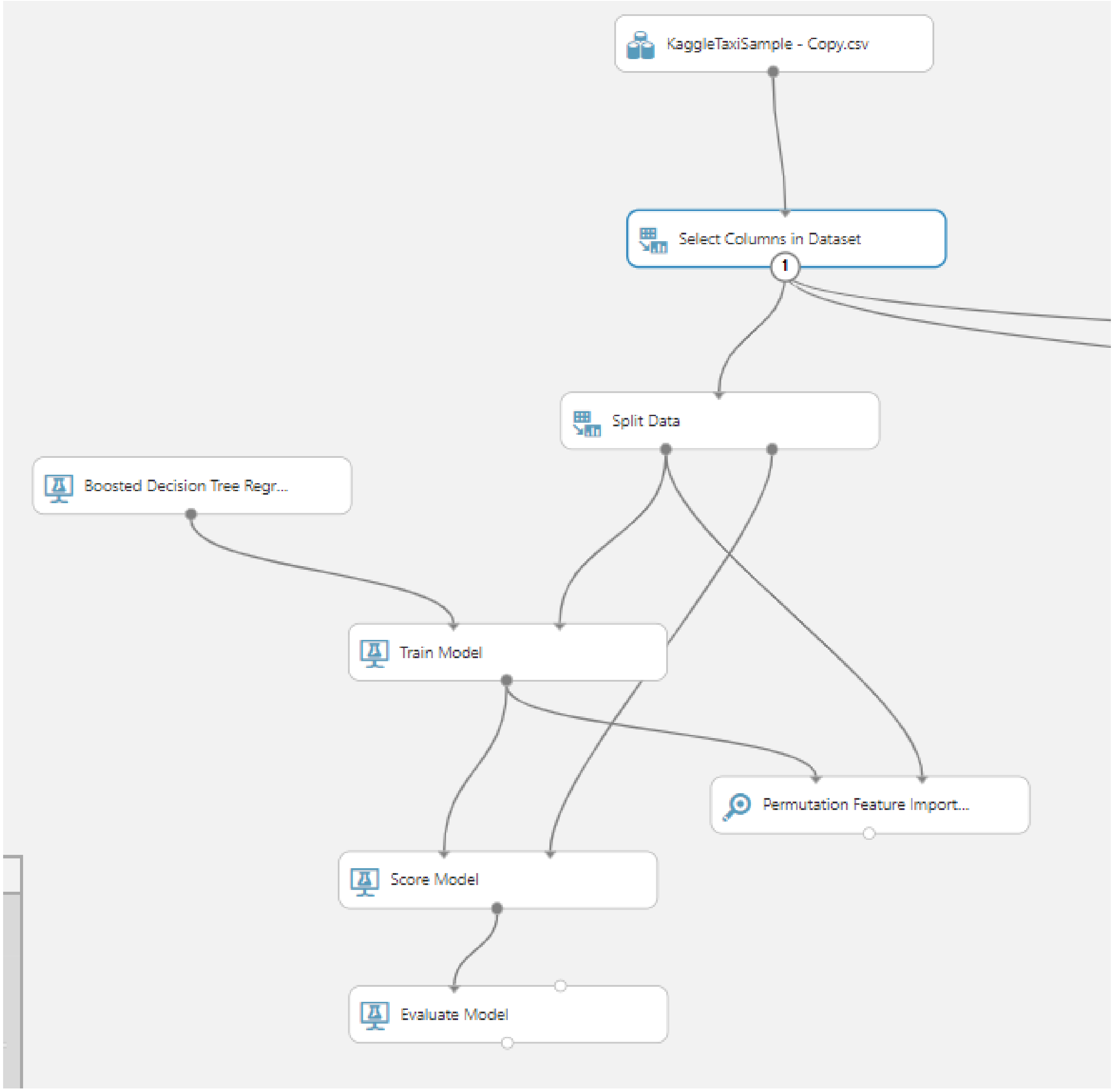
10. Search and drag **Train Model** under **Split Data**. Select “fare_amount” by using column selector. Link the Results dataset1(left) from Split Data to Dataset(right) of Train Model.

11. Search and drag **Boosted Decision Tree Regression**. Set up the properties as following: Create trainer mode -> Single Parameter. Maximum number of leaves per tree ->20. Minimum number of Samples per leaf node -> 10. Learning rate -> 0.2. Total number of trees constructed ->100. Random number seed ->12345. Allow unknown categorical levels -> checked. And link it to Untrained model(left) of Train Model.

12. Search and drag **Score Model** under **Train Model**. Link Trained Model(left) from Trained Model of Train Model, and Dataset(right) from Results Dataset2(right) from Split Data. Append score columns to output -> checked.

13. Search and drag **Permutation Feature Importance** next to **Score Model**. And link Trained Model(left) from Trained Model, Test Data(right) from Results dataset1 of Split Data. Set the properties as Random seed -> 12345, Metric for measuring performance -> Regression-Root Mean Squared Error.

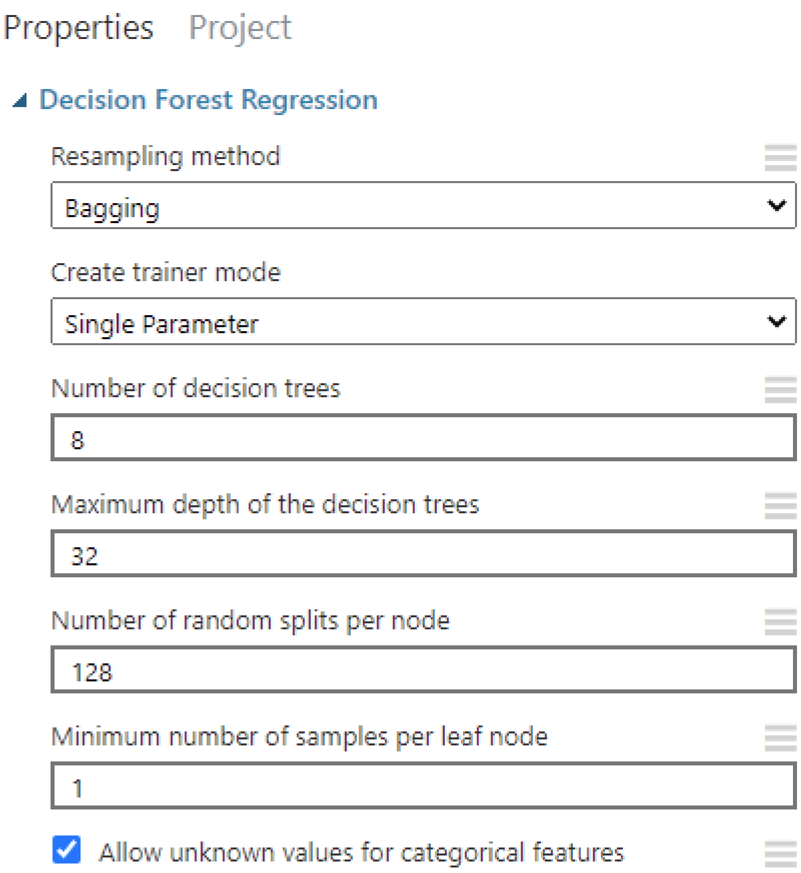
14. Search and drag **Evaluate Model** under **Score Model**, link Scored data(left) from Scored datasets of Score Model. The general view should be like this:


KaggleTaxiSample - Copy.csv
Select Columns in Dataset
1
Split Data
Boosted Decision Tree Regr...
Train Model
Permutation Feature Import...
Score Model
Evaluate Model

15. Save and Run it.

16. Repeat from step 9 to 15 for **Linear Regression**. (Search and drag **Linear Regression** instead of **Boosted Decision Tree Regression**) and Setup the properties of Linear Regression as following: Solution method -> Ordinary Least Squares. L2 regularization weight -> 0.001. Include intercept term -> checked. Random number seed -> 12345. Allow unknow categorical levels -> checked.

17. Repeat from step 9 to 15 for **Decision Forest Regression**. And setup the properties as following:



Properties Project

Decision Forest Regression

Resampling method

Bagging

Create trainer mode

Single Parameter

Number of decision trees

8

Maximum depth of the decision trees

32

Number of random splits per node

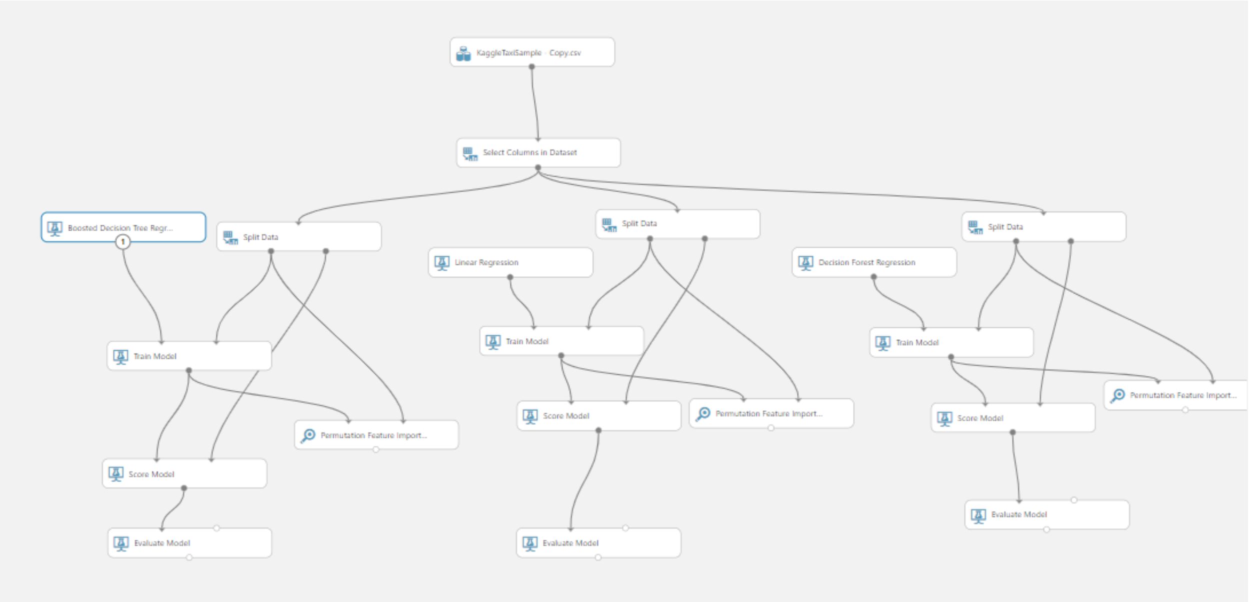
128

Minimum number of samples per leaf node

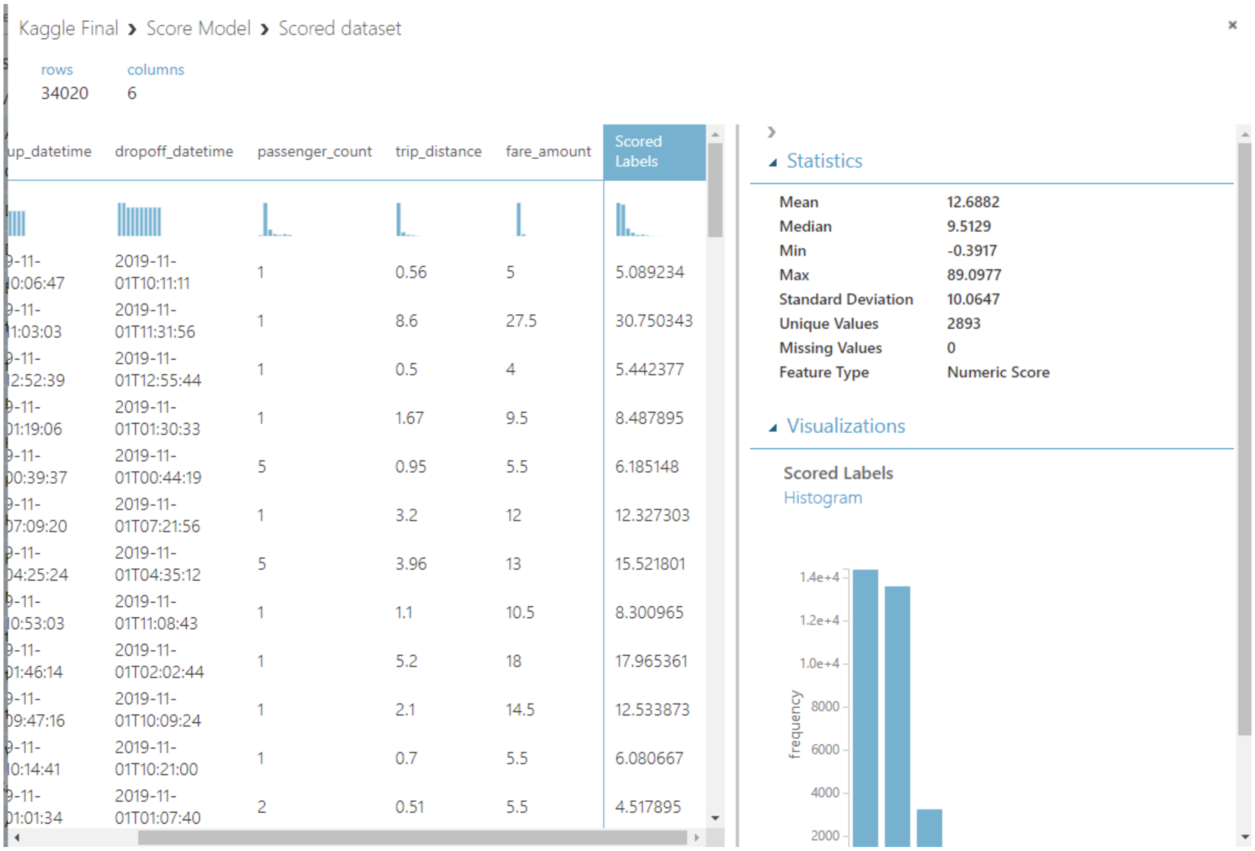
1

☑ Allow unknown values for categorical features

18. The final view will be like this:


KaggleTaxiSample - Copy.csv
Select Columns in Dataset
Boosted Decision Tree Regr...
Split Data
Split Data
Split Data
Linear Regression
Decision Forest Regression
Train Model
Train Model
Train Model
Permutation Feature Import...
Permutation Feature Import...
Permutation Feature Import...
Score Model
Score Model
Score Model
Evaluate Model
Evaluate Model
Evaluate Model

19. Right click **Scored dataset** of **Score Model** to Visualize the output. The Score Labels are the predicted values.



Kaggle Final › Score Model › Scored dataset

rows	columns
34020	6

up_datetime	dropoff_datetime	passenger_count	trip_distance	fare_amount	Scored Labels
9-11- 0:06:47	2019-11- 01T10:11:11	1	0.56	5	5.089234
9-11- 1:03:03	2019-11- 01T11:31:56	1	8.6	27.5	30.750343
9-11- 2:52:39	2019-11- 01T12:55:44	1	0.5	4	5.442377
9-11- 01:19:06	2019-11- 01T01:30:33	1	1.67	9.5	8.487895
9-11- 00:39:37	2019-11- 01T00:44:19	5	0.95	5.5	6.185148
9-11- 07:09:20	2019-11- 01T07:21:56	1	3.2	12	12.327303
9-11- 04:25:24	2019-11- 01T04:35:12	5	3.96	13	15.521801
9-11- 0:53:03	2019-11- 01T11:08:43	1	1.1	10.5	8.300965
9-11- 01:46:14	2019-11- 01T02:02:44	1	5.2	18	17.965361
9-11- 09:47:16	2019-11- 01T10:09:24	1	2.1	14.5	12.533873
9-11- 0:14:41	2019-11- 01T10:21:00	1	0.7	5.5	6.080667
9-11- 01:01:34	2019-11- 01T01:07:40	2	0.51	5.5	4.517895

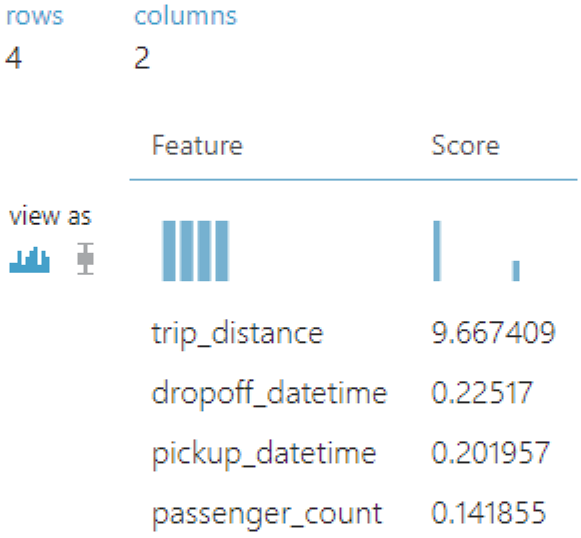
Statistics

Mean	12.6882
Median	9.5129
Min	-0.3917
Max	89.0977
Standard Deviation	10.0647
Unique Values	2893
Missing Values	0
Feature Type	Numeric Score

Visualizations

Scored Labels
Histogram

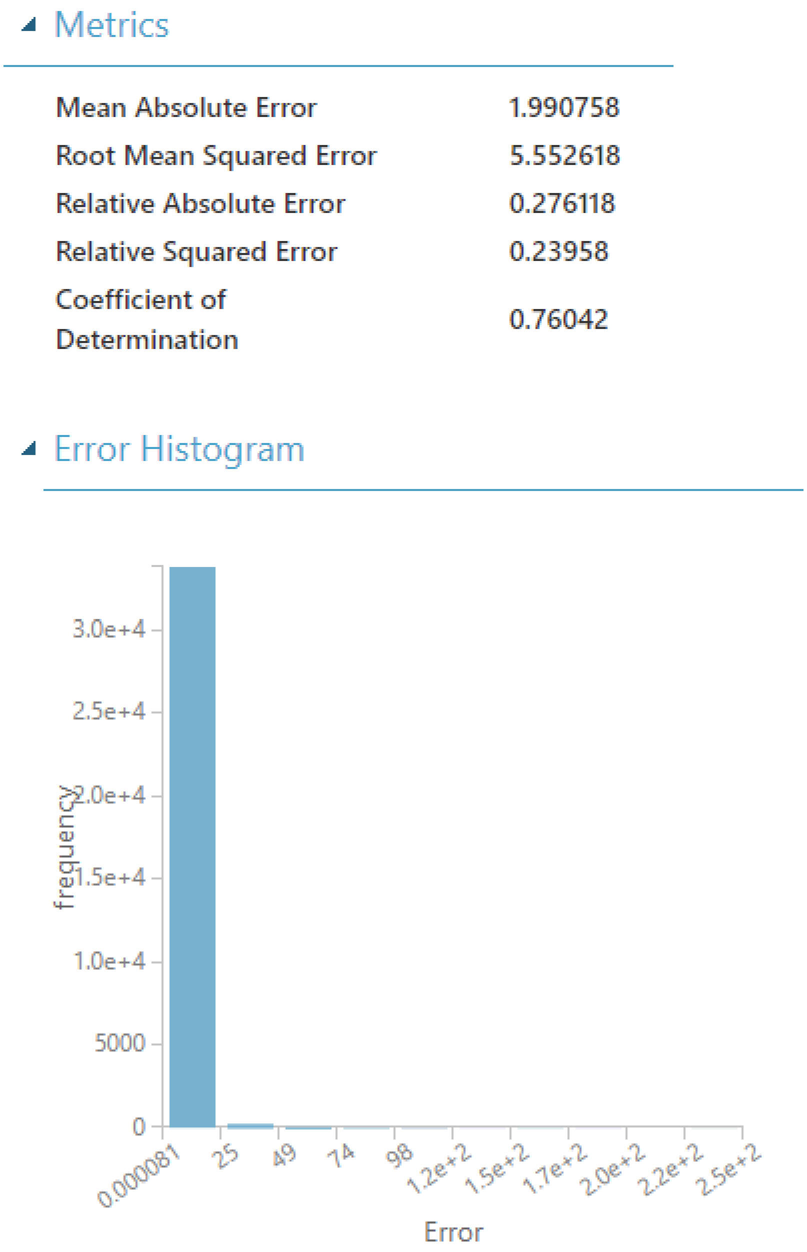
20. Right click **Feature importance** of **Permutation Feature Importance** to visualize the importance of each features.



rows	columns
4	2

Feature	Score
trip_distance	9.667409
dropoff_datetime	0.22517
pickup_datetime	0.201957
passenger_count	0.141855

21. Right click **Evaluation Results** of **Evaluate Model** to visualize the **RMSE** and **Coefficient of Determination** or other evaluation results.



◢ Metrics

Mean Absolute Error	1.990758
Root Mean Squared Error	5.552618
Relative Absolute Error	0.276118
Relative Squared Error	0.23958
Coefficient of Determination	0.76042

◢ Error Histogram

22. Do the same thing from step 19 to 21 for all three algorithms and compare the RMSE and Coefficient of Determination of them. Think which algorithm is the best(Boosted Decision Tree Regression is the best in this case.

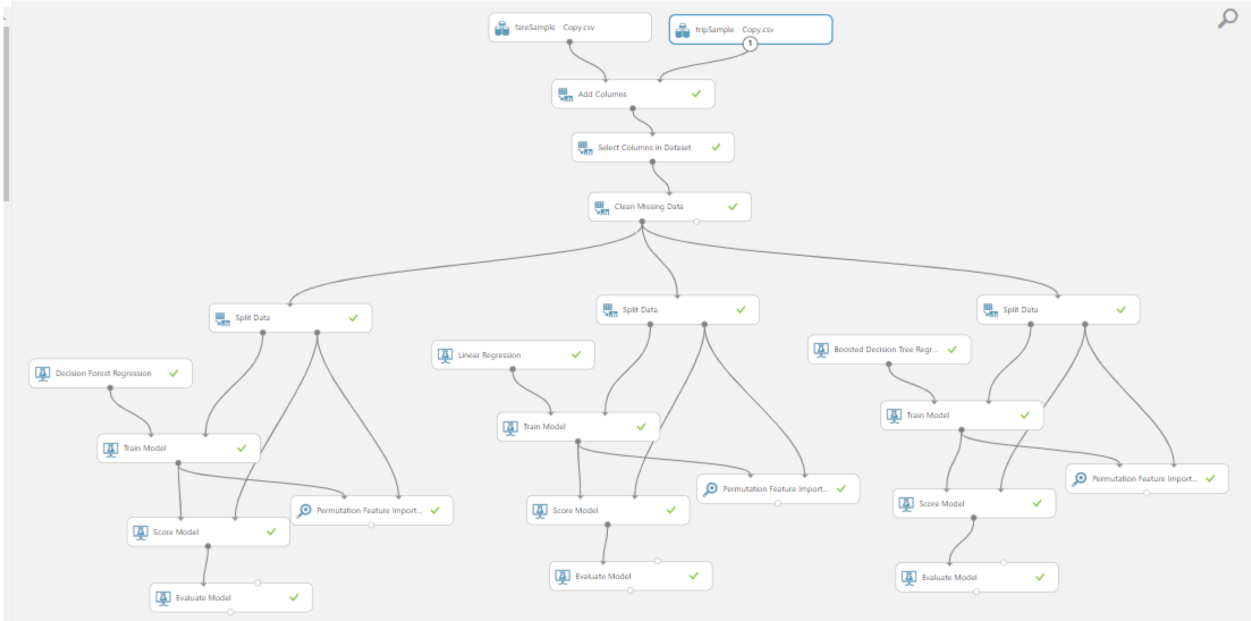
23. Upload **fareSample.csv** and **tripSample.csv** to Azure and drag them to a new canvas named as **FareTrip Final**.

24. Search and drag **Add Columns** and connect both datasets to it.

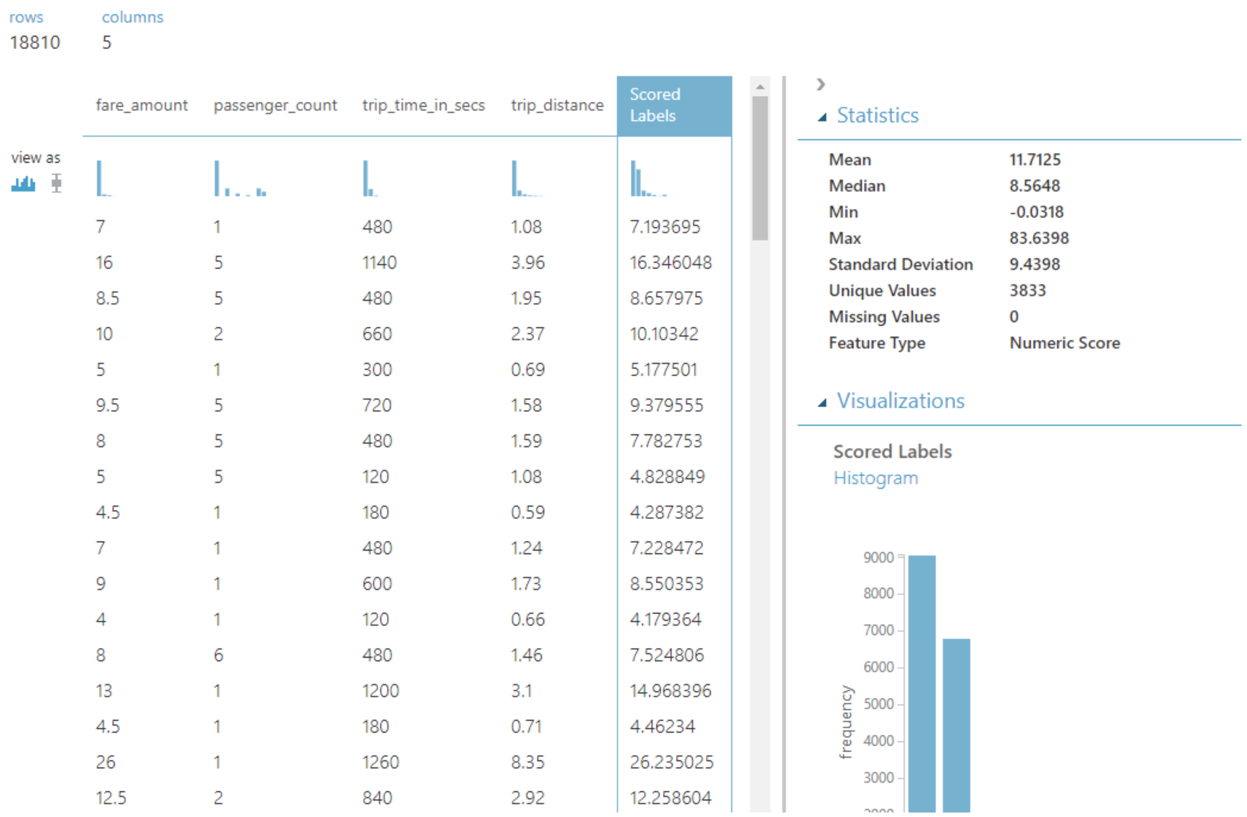
25. Drag **Select Columns** in Dataset under **Add Columns**. And select fare_amount, trip_time_in_secs, trip_distance and passenger_count by using column selector.

26. Drag **Clean Missing Data** under **Select Columns**, and set the properties as following: Selected columns -> All columns. Minimum missing value ratio -> 0. Maximum missing value ratio -> 1. Cleaning mode -> Remove entire row.

27. Repeat step 9 to 17 for all three algorithms, save and run the experiment.



28. Repeat step 19 to 22. to compare the results for all three algorithms and also compare the all the results from two different datasets/experiments.(following is results for BDTR of FareTrip Final)



rows 18810 columns 5

view as

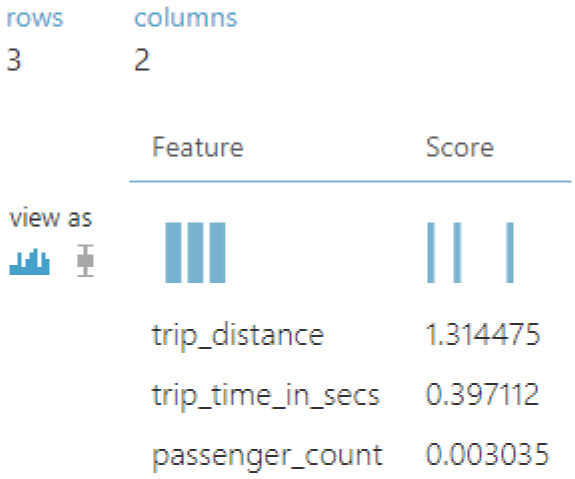
fare_amount	passenger_count	trip_time_in_secs	trip_distance	Scored Labels
7	1	480	1.08	7.193695
16	5	1140	3.96	16.346048
8.5	5	480	1.95	8.657975
10	2	660	2.37	10.10342
5	1	300	0.69	5.177501
9.5	5	720	1.58	9.379555
8	5	480	1.59	7.782753
5	5	120	1.08	4.828849
4.5	1	180	0.59	4.287382
7	1	480	1.24	7.228472
9	1	600	1.73	8.550353
4	1	120	0.66	4.179364
8	6	480	1.46	7.524806
13	1	1200	3.1	14.968396
4.5	1	180	0.71	4.46234
26	1	1260	8.35	26.235025
12.5	2	840	2.92	12.258604

Statistics

Mean	11.7125
Median	8.5648
Min	-0.0318
Max	83.6398
Standard Deviation	9.4398
Unique Values	3833
Missing Values	0
Feature Type	Numeric Score

Visualizations

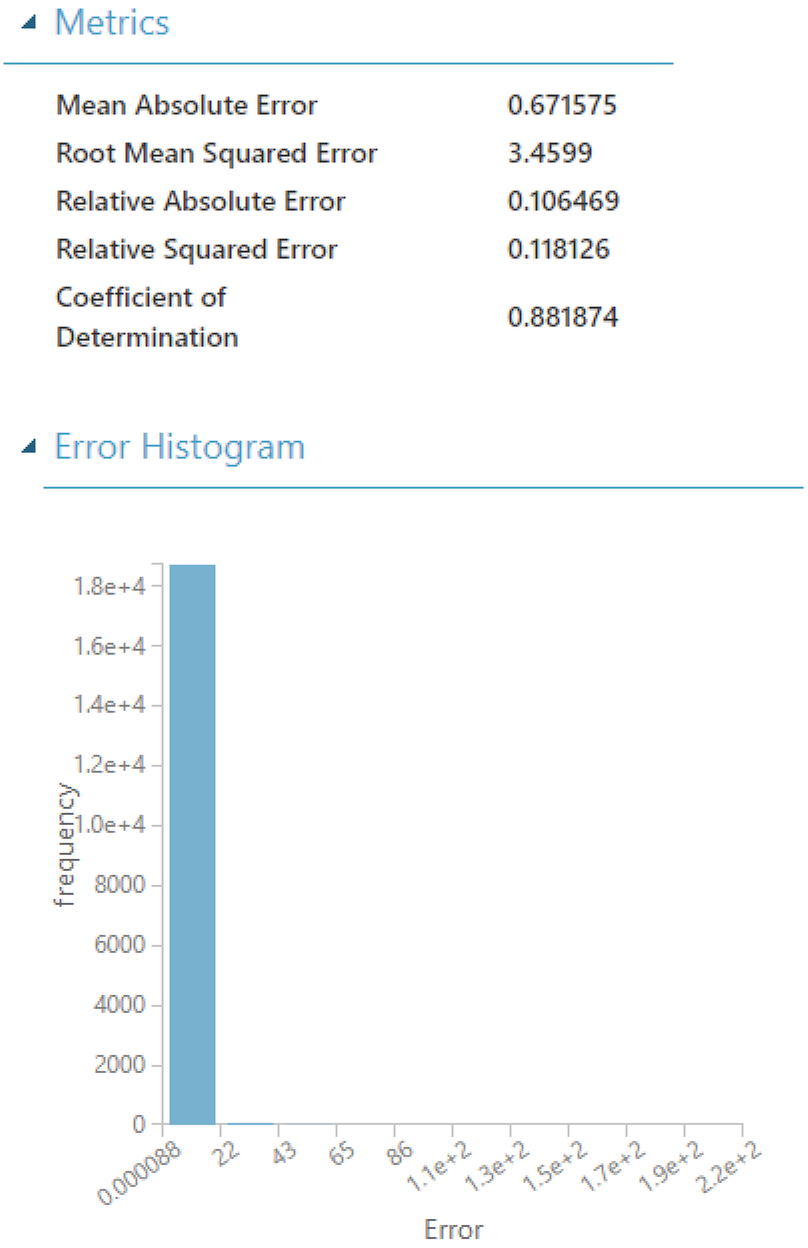
Scored Labels
Histogram



rows 3 columns 2

view as

Feature	Score
trip_distance	1.314475
trip_time_in_secs	0.397112
passenger_count	0.003035



◢ Metrics

Mean Absolute Error	0.671575
Root Mean Squared Error	3.4599
Relative Absolute Error	0.106469
Relative Squared Error	0.118126
Coefficient of Determination	0.881874

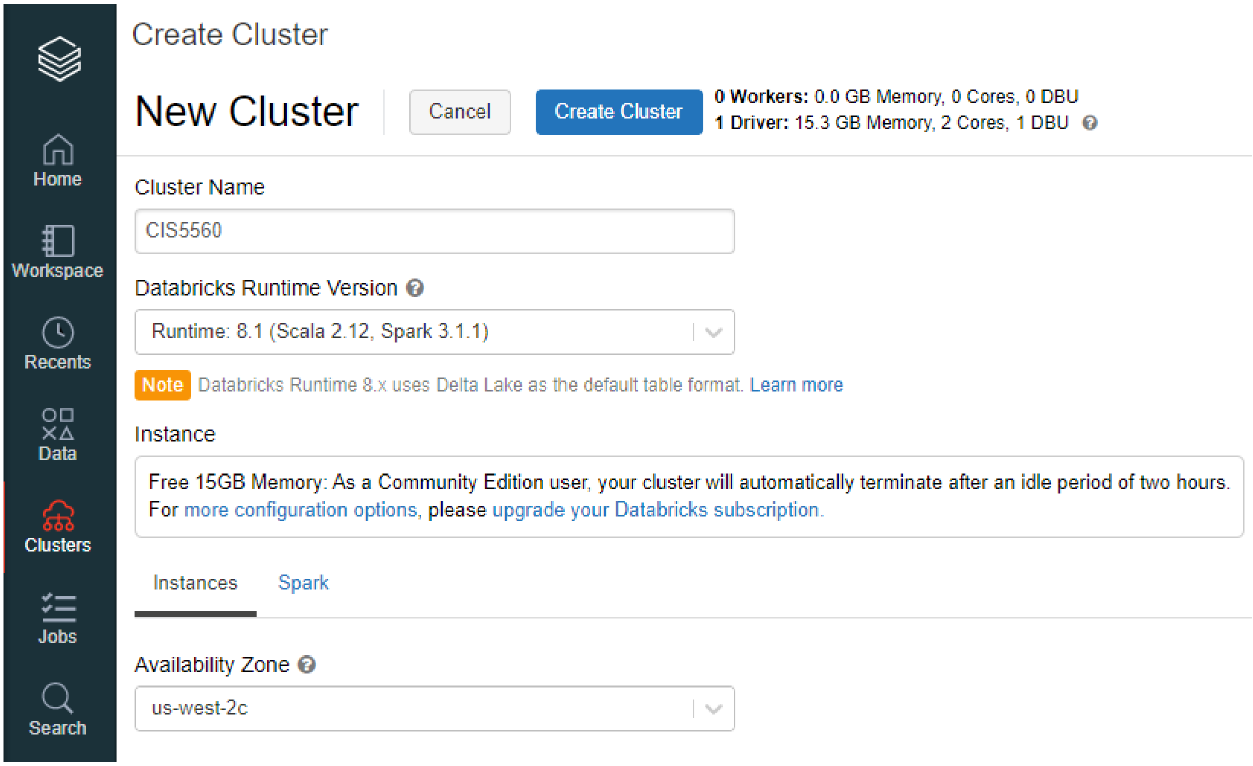
◢ Error Histogram

Task 3: Databricks

This step will introduce Databricks

1. Open a browser and sign in using the Databricks account at https://community.cloud.databricks.com/

2. Once sign in to Databricks community edition, click **Clusters** on the left side -> Create Cluster and name the Cluster Name as **CIS5560**. Choose Databricks Runtime Version as Runtime: **8.1 (Scala 2.12, Spark 3.1.1)**



Create Cluster

New Cluster

0 Workers: 0.0 GB Memory, 0 Cores, 0 DBU
1 Driver: 15.3 GB Memory, 2 Cores, 1 DBU

Cluster Name: CIS5560

Databricks Runtime Version: Runtime: 8.1 (Scala 2.12, Spark 3.1.1)

Note Databricks Runtime 8.x uses Delta Lake as the default table format. Learn more

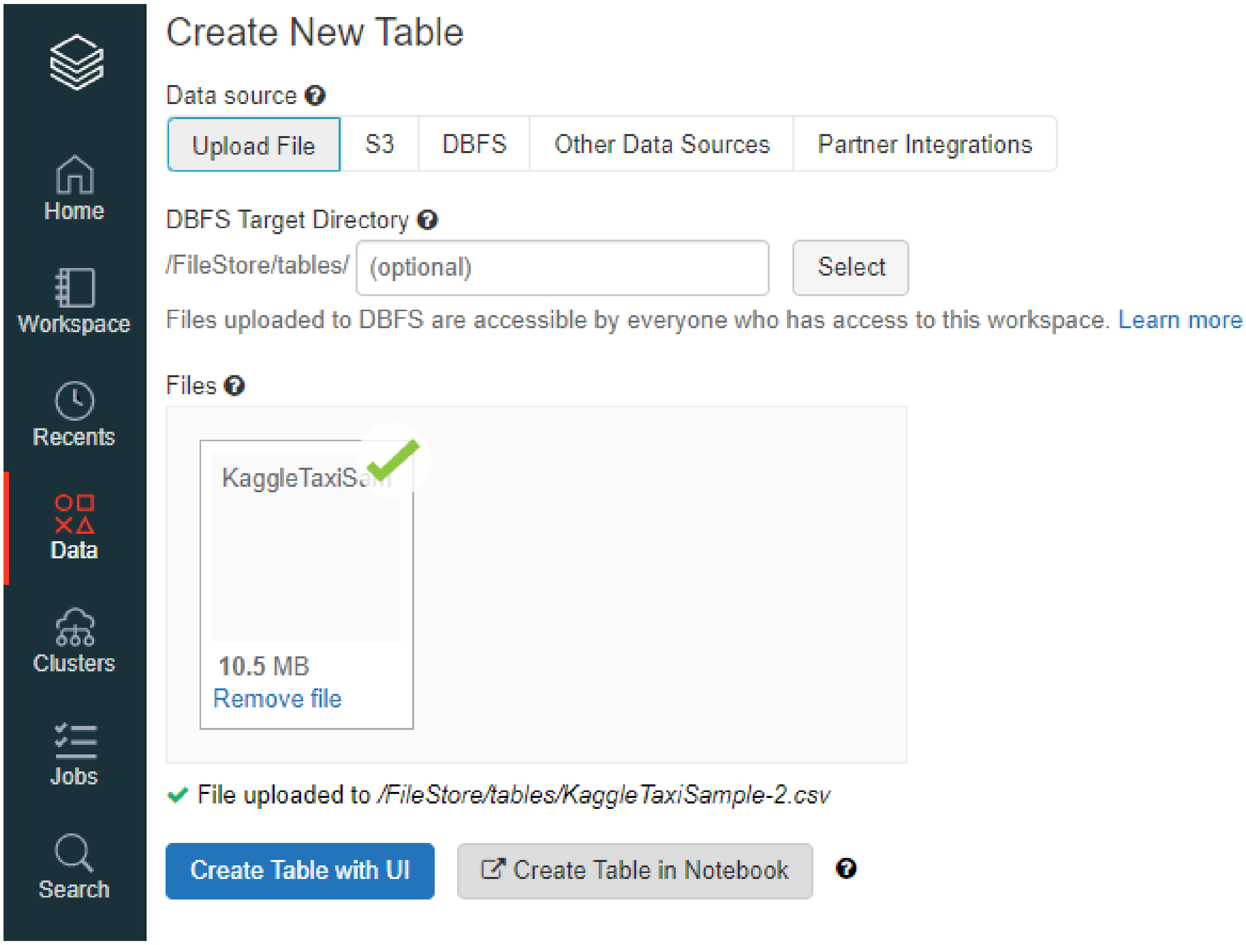
Instance

Free 15GB Memory: As a Community Edition user, your cluster will automatically terminate after an idle period of two hours. For more configuration options, please upgrade your Databricks subscription.

Availability Zone: us-west-2c

3. Click the symbol on the left top to back the main page.

4. Click **Import & Explore Data** -> Drop files to upload, or click to browse to upload the KaggleTaxiSample.csv and choose **Create Table in Notebook**



Create New Table

Data source

DBFS Target Directory

/FileStore/tables/ (optional)

Files uploaded to DBFS are accessible by everyone who has access to this workspace. Learn more

Files

10.5 MB

✓ File uploaded to */FileStore/tables/KaggleTaxiSample-2.csv*

5. Once finished uploading the dataset, click **Workspace** on the left side and open the dataset (if the dataset does not show up automatically) rename it as KTSample.

6. Click Workspace -> left click cis5560 folder if have -> right click on the blank space -> Create -> Notebook. Give the name as KaggleTSampleTest, default language as Python, Cluster using the CIS5560 which just created.

7. Insert following code for **apply packages**:

```
from pyspark.sql.types import *
from pyspark.sql.functions import *

from pyspark.context import SparkContext
from pyspark.sql.session import SparkSession

from pyspark.ml import Pipeline
from pyspark.ml.regression import GBTRegressor
from pyspark.ml.regression import LinearRegression
from pyspark.ml.feature import VectorAssembler
from pyspark.ml.tuning import ParamGridBuilder, TrainValidationSplit
from pyspark.ml.evaluation import BinaryClassificationEvaluator,
RegressionEvaluator
```

```
from pyspark.ml.tuning import CrossValidator, ParamGridBuilder
from pyspark.ml.regression import DecisionTreeRegressor
```

8. Insert a new cell by enter following code:

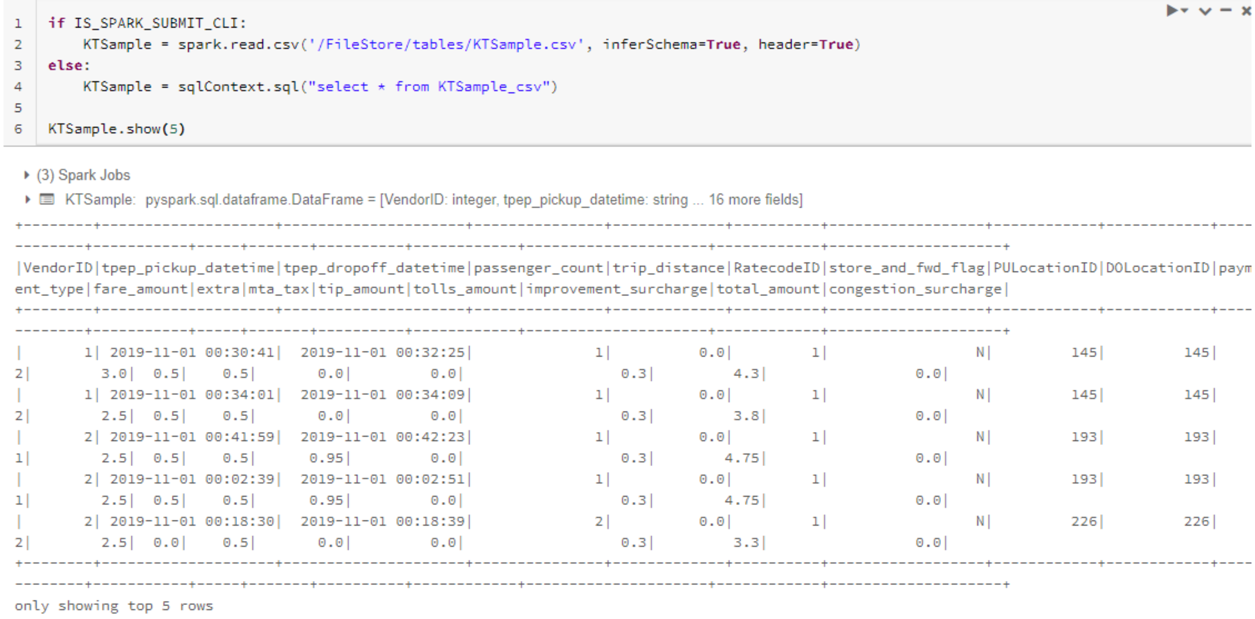
```
IS_SPARK_SUBMIT_CLI = True

if IS_SPARK_SUBMIT_CLI:
    sc = SparkContext.getOrCreate()
 spark = SparkSession(sc)
```

9. Insert a new cell by enter following code for **load data**:

```
file_location = "/FileStore/tables/KTSample.csv"
file_type = "csv"

# CSV options
infer_schema = "true"
first_row_is_header = "true"
delimiter = ","

# The applied options are for CSV files. For other file types, these
will be ignored.
df1 = spark.read.format(file_type) \
  .option("inferSchema", infer_schema) \
  .option("header", first_row_is_header) \
  .option("sep", delimiter) \
  .load(file_location)
```

10. Insert a new cell by enter following code for **load data** and check the first 5 lines of the data:

```
temp_table_name1 = "KTSample_csv"

df1.createOrReplaceTempView(temp_table_name1)
if IS_SPARK_SUBMIT_CLI:
    KTSample    =    spark.read.csv('/FileStore/tables/KTSample.csv',
inferSchema=True, header=True)
else:
    KTSample = sqlContext.sql("select * from KTSample_csv")

KTSample.show(5)
```



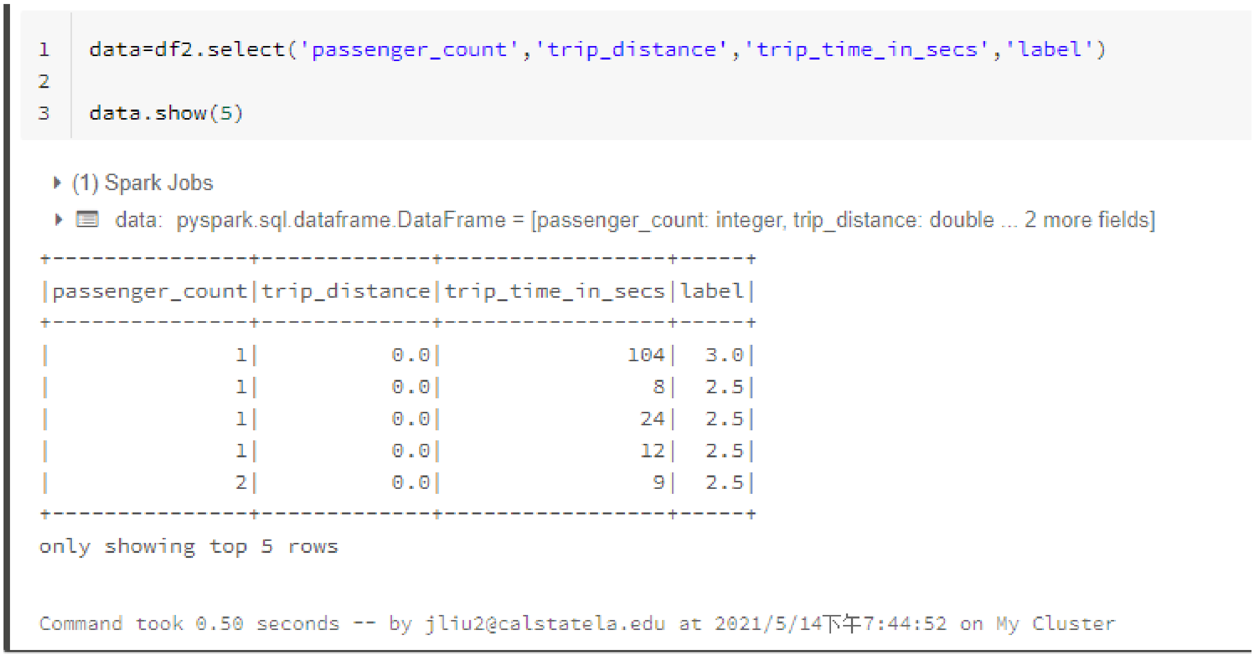
```
if IS_SPARK_SUBMIT_CLI:
    KTSample = spark.read.csv('/FileStore/tables/KTSample.csv', inferSchema=True, header=True)
else:
    KTSample = sqlContext.sql("select * from KTSample_csv")

KTSample.show(5)
```

```
KTSample:  pyspark.sql.dataframe.DataFrame = [VendorID: integer, tpep_pickup_datetime: string ... 16 more fields]
```

```
+--------+--------------------+---------------------+---------------+-------------+----------+------------------+------------+------------+----
--------+-----------+-----+-------+----------+------------+---------------------+------------+--------------------+
|VendorID|tpep_pickup_datetime|tpep_dropoff_datetime|passenger_count|trip_distance|RatecodeID|store_and_fwd_flag|PULocationID|DOLocationID|paym
ent_type|fare_amount|extra|mta_tax|tip_amount|tolls_amount|improvement_surcharge|total_amount|congestion_surcharge|
+--------+--------------------+---------------------+---------------+-------------+----------+------------------+------------+------------+----
--------+-----------+-----+-------+----------+------------+---------------------+------------+--------------------+
|       1| 2019-11-01 00:30:41|  2019-11-01 00:32:25|              1|          0.0|         1|                 N|         145|         145|
2|        3.0|  0.5|    0.5|       0.0|         0.0|                  0.3|         4.3|                 0.0|
|       1| 2019-11-01 00:34:01|  2019-11-01 00:34:09|              1|          0.0|         1|                 N|         145|         145|
2|        2.5|  0.5|    0.5|       0.0|         0.0|                  0.3|         3.8|                 0.0|
|       2| 2019-11-01 00:41:59|  2019-11-01 00:42:23|              1|          0.0|         1|                 N|         193|         193|
1|        2.5|  0.5|    0.5|      0.95|         0.0|                  0.3|        4.75|                 0.0|
|       2| 2019-11-01 00:02:39|  2019-11-01 00:02:51|              1|          0.0|         1|                 N|         193|         193|
1|        2.5|  0.5|    0.5|      0.95|         0.0|                  0.3|        4.75|                 0.0|
|       2| 2019-11-01 00:18:30|  2019-11-01 00:18:39|              2|          0.0|         1|                 N|         226|         226|
2|        2.5|  0.0|    0.5|       0.0|         0.0|                  0.3|         3.3|                 0.0|
+--------+--------------------+---------------------+---------------+-------------+----------+------------------+------------+------------+----
--------+-----------+-----+-------+----------+------------+---------------------+------------+--------------------+
only showing top 5 rows
```

11. Insert a new cell by enter following code for **select data** and **calculate the trip time in sec**:

```
timediff=KTSample.select('tpep_pickup_datetime',
'tpep_dropoff_datetime','passenger_count','trip_distance',col('fare_
amount').alias('label'))

df2=timediff.withColumn('tpep_pickup_datetime',to_timestamp(col('tpe
p_pickup_datetime')))\
  .withColumn('tpep_dropoff_datetime',
to_timestamp(col('tpep_dropoff_datetime')))\
  .withColumn('trip_time_in_secs',col("tpep_dropoff_datetime").cast(
"long") - col('tpep_pickup_datetime').cast("long"))

data=df2.select('passenger_count','trip_distance','trip_time_in_secs
','label')

data.show(5)
```



```
data=df2.select('passenger_count','trip_distance','trip_time_in_secs','label')

data.show(5)
```

▸ (1) Spark Jobs
▸ data: pyspark.sql.dataframe.DataFrame = [passenger_count: integer, trip_distance: double ... 2 more fields]

```
+---------------+-------------+-----------------+-----+
|passenger_count|trip_distance|trip_time_in_secs|label|
+---------------+-------------+-----------------+-----+
|              1|          0.0|              104|  3.0|
|              1|          0.0|                8|  2.5|
|              1|          0.0|               24|  2.5|
|              1|          0.0|               12|  2.5|
|              2|          0.0|                9|  2.5|
+---------------+-------------+-----------------+-----+
only showing top 5 rows

Command took 0.50 seconds -- by jliu2@calstatela.edu at 2021/5/14下午7:44:52 on My Cluster
```

12. Insert a new cell to setup **train** and **test** datasets by following code:

```
splits = data.randomSplit([0.7, 0.3])
train = splits[0]
test = splits[1].withColumnRenamed("label", "trueLabel")
```

13. Insert a new cell to setup **assembler** for GBT-Regression:

```
assembler = VectorAssembler(inputCols = ['passenger_count', 'trip_time_in_secs', 'trip_distance'], outputCol="features")
gbt = GBTRegressor(labelCol="label")
```

14. Insert a new cell to setup **ParamGrid** and **CrossValidator**:

```
paramGrid = ParamGridBuilder()\
  .addGrid(gbt.maxDepth, [2, 3])\
  .addGrid(gbt.maxIter, [10, 20])\
  .build()

cv = CrossValidator(estimator=gbt, evaluator=RegressionEvaluator(), estimatorParamMaps=paramGrid)
```

15. Insert a new cell to setup **pipeline** and **train** the model:

```
pipeline = Pipeline(stages=[assembler, cv])
pipelineModel = pipeline.fit(train)
```

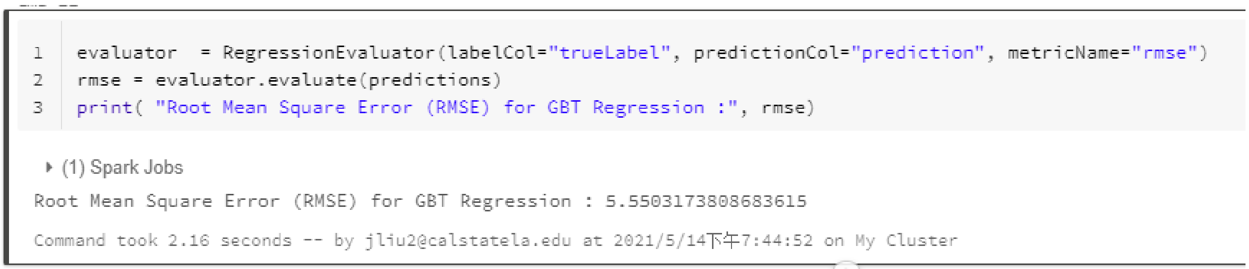
16. Insert a new cell for **prediction**:

```
predictions = pipelineModel.transform(test)
predicted = predictions.select("features", "prediction", "trueLabel")
```

```
predicted.createOrReplaceTempView("regressionPredictions")
```

17. Insert a new cell for **RMSE**:

```
evaluator = RegressionEvaluator(labelCol="trueLabel",
predictionCol="prediction", metricName="rmse")
rmse = evaluator.evaluate(predictions)
print( "Root Mean Square Error (RMSE) for GBT Regression :", rmse)
```



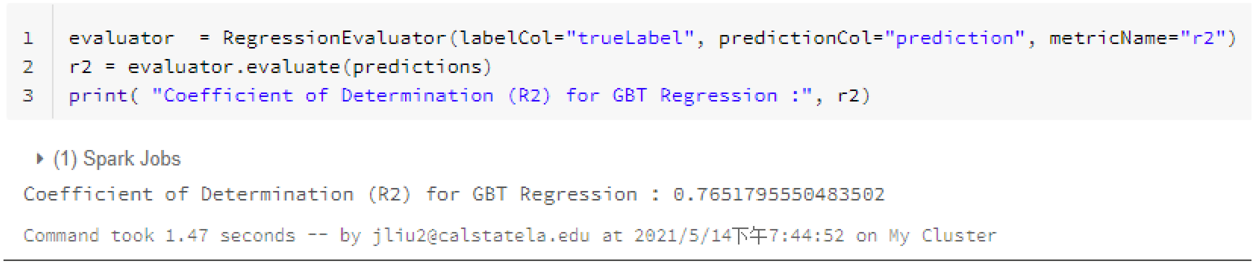
```
evaluator  = RegressionEvaluator(labelCol="trueLabel", predictionCol="prediction", metricName="rmse")
rmse = evaluator.evaluate(predictions)
print( "Root Mean Square Error (RMSE) for GBT Regression :", rmse)
```

▸ (1) Spark Jobs

```
Root Mean Square Error (RMSE) for GBT Regression : 5.5503173808683615

Command took 2.16 seconds -- by jliu2@calstatela.edu at 2021/5/14下午7:44:52 on My Cluster
```

18. Insert a new cell for **R2**(Coefficient of Determination):

```
evaluator = RegressionEvaluator(labelCol="trueLabel",
predictionCol="prediction", metricName="r2")
r2 = evaluator.evaluate(predictions)
print( "Coefficient of Determination (R2) for GBT Regression :", r2)
```



```
evaluator  = RegressionEvaluator(labelCol="trueLabel", predictionCol="prediction", metricName="r2")
r2 = evaluator.evaluate(predictions)
print( "Coefficient of Determination (R2) for GBT Regression :", r2)
```

▸ (1) Spark Jobs

```
Coefficient of Determination (R2) for GBT Regression : 0.7651795550483502

Command took 1.47 seconds -- by jliu2@calstatela.edu at 2021/5/14下午7:44:52 on My Cluster
```

19. Now we need to setup **Linear Regression**. Insert the following code for **assembler, lr and pipeline1**:

```
assembler = VectorAssembler(inputCols = ['passenger_count',
'trip_time_in_secs', 'trip_distance'], outputCol="features")
lr = LinearRegression(labelCol="label",featuresCol="features",
maxIter=10, regParam=0.3)
pipeline1 = Pipeline(stages=[assembler, lr])
```

20. Code for **paramGrid1**:

```
paramGrid1 = ParamGridBuilder().addGrid(lr.regParam, [0.3,
0.01]).addGrid(lr.maxIter, [10, 5]).build()
trainval = TrainValidationSplit(estimator=pipeline1,
evaluator=RegressionEvaluator(), estimatorParamMaps=paramGrid1,
trainRatio=0.8)
```

21. **Tran** the model:

```
pipelineModel = trainval.fit(train)
```

22. **Transform** the model with test dataset:

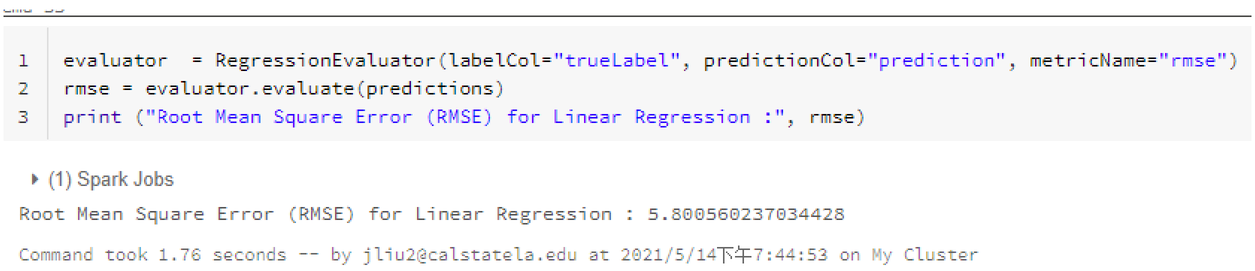
```
predictions = pipelineModel.transform(test)
```

23. Setup for **prediction**:

```
predicted = predictions.select("features", "prediction", "trueLabel")
predicted.createOrReplaceTempView("regressionPredictions")
```

24. Setup **RMSE** for LR:

```
evaluator = RegressionEvaluator(labelCol="trueLabel",
predictionCol="prediction", metricName="rmse")
rmse = evaluator.evaluate(predictions)
print ("Root Mean Square Error (RMSE) for Linear Regression :", rmse)
```

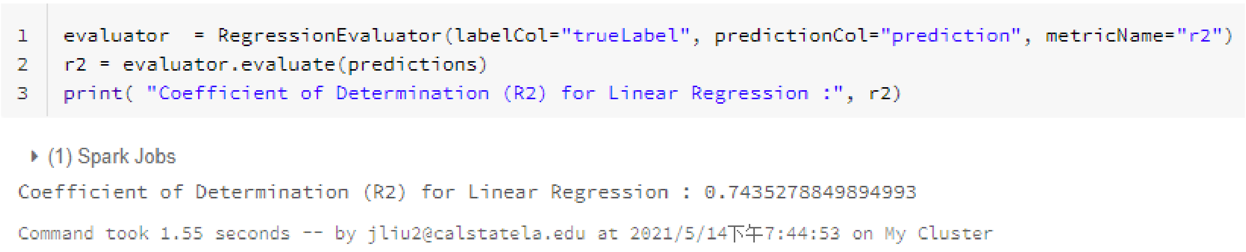


```
evaluator  = RegressionEvaluator(labelCol="trueLabel", predictionCol="prediction", metricName="rmse")
rmse = evaluator.evaluate(predictions)
print ("Root Mean Square Error (RMSE) for Linear Regression :", rmse)
```

```
Root Mean Square Error (RMSE) for Linear Regression : 5.800560237034428

Command took 1.76 seconds -- by jliu2@calstatela.edu at 2021/5/14下午7:44:53 on My Cluster
```

25. Setup **R2** for LR:

```
evaluator = RegressionEvaluator(labelCol="trueLabel",
predictionCol="prediction", metricName="r2")
r2 = evaluator.evaluate(predictions)
print( "Coefficient of Determination (R2) for Linear Regression :",
r2)
```



```
evaluator  = RegressionEvaluator(labelCol="trueLabel", predictionCol="prediction", metricName="r2")
r2 = evaluator.evaluate(predictions)
print( "Coefficient of Determination (R2) for Linear Regression :", r2)
```

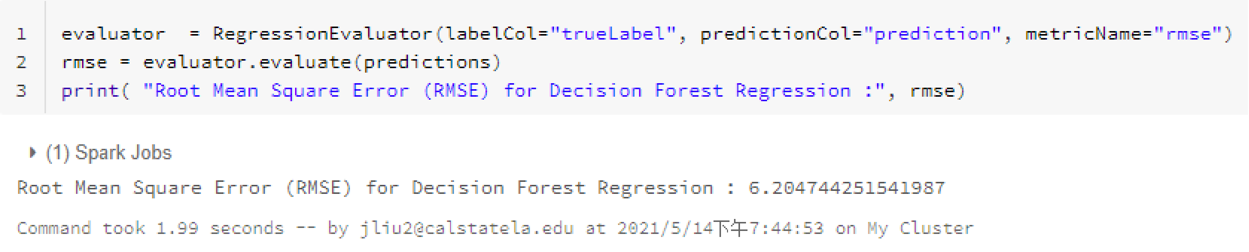
```
Coefficient of Determination (R2) for Linear Regression : 0.7435278849894993

Command took 1.55 seconds -- by jliu2@calstatela.edu at 2021/5/14下午7:44:53 on My Cluster
```

26. Set up **Decision Forest Regression** by following code:

```
assembler = VectorAssembler(inputCols = ['passenger_count',
'trip_time_in_secs', 'trip_distance'], outputCol="features")
dt = DecisionTreeRegressor(labelCol="label",featuresCol="features")
paramGrid2 = ParamGridBuilder()\
  .addGrid(dt.maxDepth, [2,3])\
  .addGrid(dt.maxBins, [10,20])\
  .build()
dtcv = CrossValidator(estimator = dt, estimatorParamMaps = paramGrid2,
evaluator = RegressionEvaluator(), numFolds=2)
pipeline2 = Pipeline(stages=[assembler, dtcv])
pipelineModel = pipeline2.fit(train)
predictions = pipelineModel.transform(test)
predicted = predictions.select("features","prediction","truelabel")
predicted.createOrReplaceTempView("regressionPredictions")
```

27. Setup **RMSE**:

```
evaluator = RegressionEvaluator(labelCol="trueLabel",
predictionCol="prediction", metricName="rmse")
rmse = evaluator.evaluate(predictions)
print( "Root Mean Square Error (RMSE) for Decision Forest
Regression :", rmse)
```



```
evaluator  = RegressionEvaluator(labelCol="trueLabel", predictionCol="prediction", metricName="rmse")
rmse = evaluator.evaluate(predictions)
print( "Root Mean Square Error (RMSE) for Decision Forest Regression :", rmse)
```

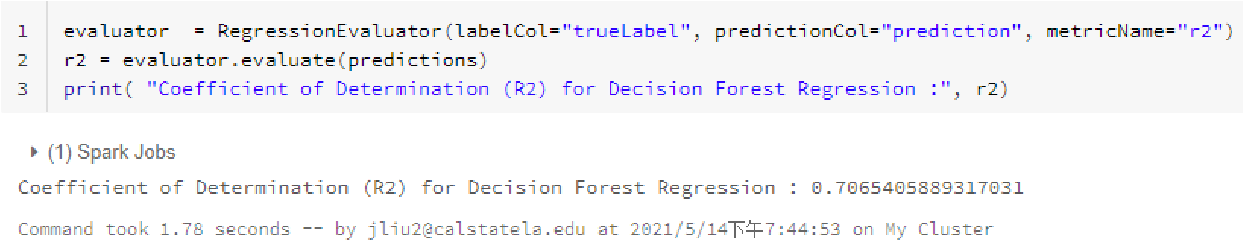
▸ (1) Spark Jobs

```
Root Mean Square Error (RMSE) for Decision Forest Regression : 6.204744251541987
```

Command took 1.99 seconds -- by jliu2@calstatela.edu at 2021/5/14下午7:44:53 on My Cluster

28. Setup **R2**:

```
evaluator = RegressionEvaluator(labelCol="trueLabel",
predictionCol="prediction", metricName="r2")
r2 = evaluator.evaluate(predictions)
print( "Coefficient of Determination (R2) for Decision Forest
Regression :", r2)
```



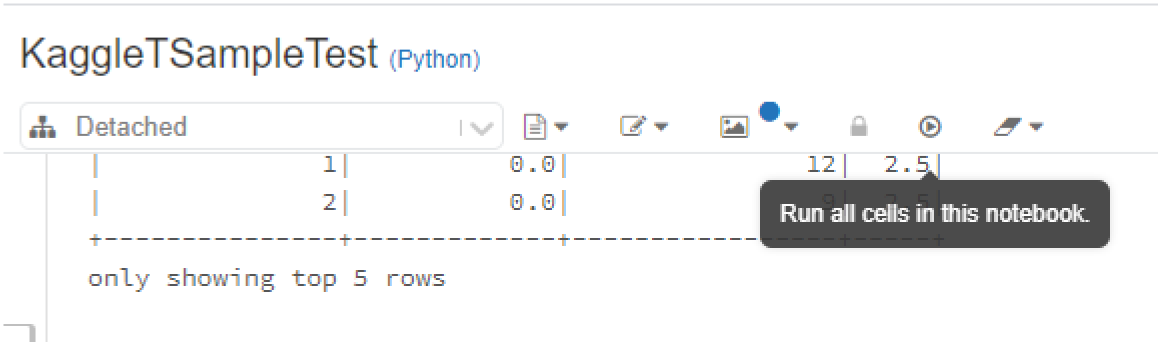
```
evaluator  = RegressionEvaluator(labelCol="trueLabel", predictionCol="prediction", metricName="r2")
r2 = evaluator.evaluate(predictions)
print( "Coefficient of Determination (R2) for Decision Forest Regression :", r2)
```

▸ (1) Spark Jobs

```
Coefficient of Determination (R2) for Decision Forest Regression : 0.7065405889317031
```

Command took 1.78 seconds -- by jliu2@calstatela.edu at 2021/5/14下午7:44:53 on My Cluster

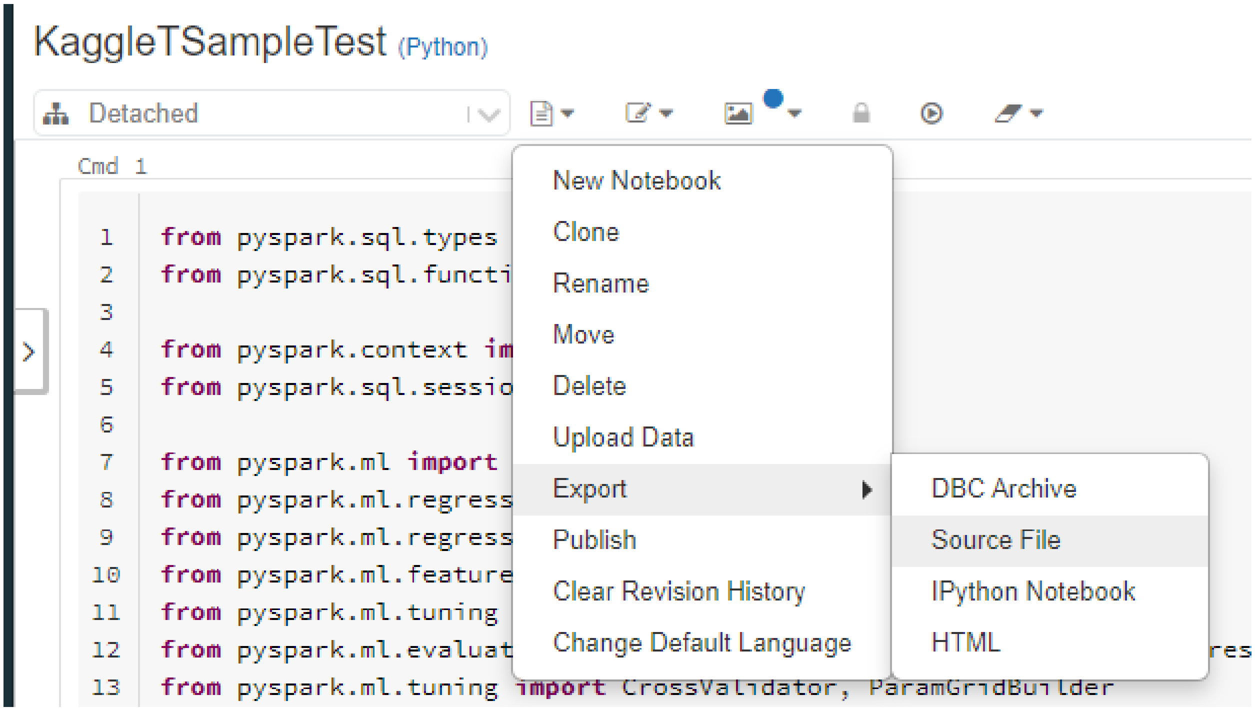
29. Click **Run all** to run all the codes



```
|               1|           0.0|               12|   2.5|
|               2|           0.0|
+---------------+--------------+----------------
only showing top 5 rows
```

30. Now you can compare the results for all three different models. For the **paramGrid** in GBT-Regression, we using **[2,3] for maxDepth and [10,20] for maxIter**. You may want to using **[2,5] for maxDepth and [10,100] for maxIter** to have more accuracy results.

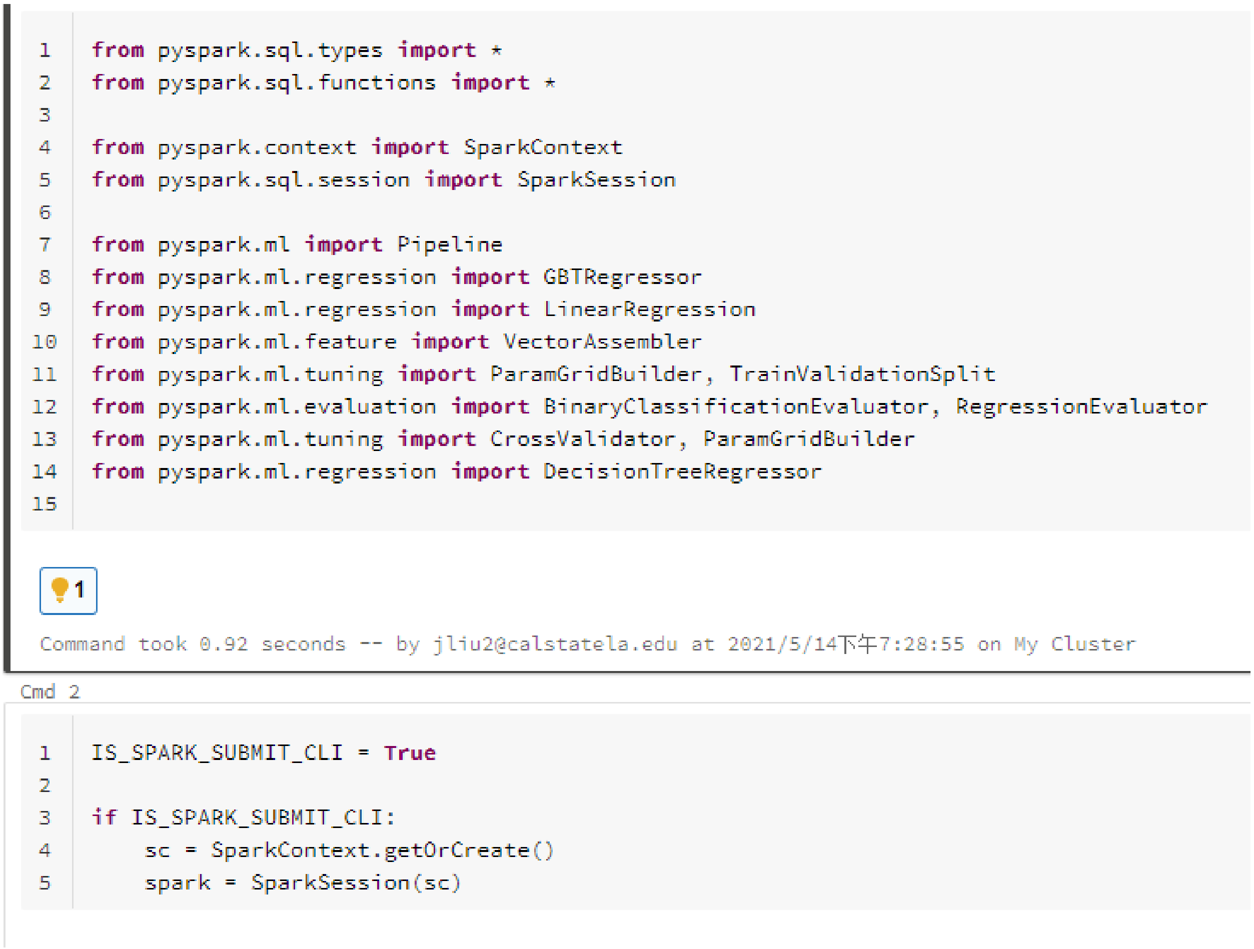
31. You may also export the code as a **py** file for **Hadoop Spark** and **ipynb** file for **IPythonNotebook** as shown in the picture:



```
from pyspark.sql.types
from pyspark.sql.functi

from pyspark.context im
from pyspark.sql.sessio

from pyspark.ml import
from pyspark.ml.regress
from pyspark.ml.regress
from pyspark.ml.feature
from pyspark.ml.tuning
from pyspark.ml.evaluat                                                            res
from pyspark.ml.tuning import CrossValidator, ParamGridBuilder
```

32. Now repeat step 3 to 6 for upload fareSample.csv and tripSample.csv, renamed them as **fareSample** and **tripSample**. And create a new notebook named **FareTripSampleTest.**

33. Repeat step 7 and 8 to setup the environment:



```
from pyspark.sql.types import *
from pyspark.sql.functions import *

from pyspark.context import SparkContext
from pyspark.sql.session import SparkSession

from pyspark.ml import Pipeline
from pyspark.ml.regression import GBTRegressor
from pyspark.ml.regression import LinearRegression
from pyspark.ml.feature import VectorAssembler
from pyspark.ml.tuning import ParamGridBuilder, TrainValidationSplit
from pyspark.ml.evaluation import BinaryClassificationEvaluator, RegressionEvaluator
from pyspark.ml.tuning import CrossValidator, ParamGridBuilder
from pyspark.ml.regression import DecisionTreeRegressor

```

💡1

```
Command took 0.92 seconds -- by jliu2@calstatela.edu at 2021/5/14下午7:28:55 on My Cluster
```

Cmd 2

```
IS_SPARK_SUBMIT_CLI = True

if IS_SPARK_SUBMIT_CLI:
    sc = SparkContext.getOrCreate()
    spark = SparkSession(sc)
```

34. **Load data** by following codes:

```
file_location = "/FileStore/tables/fareSample.csv"
file_type = "csv"

# CSV options
infer_schema = "true"
first_row_is_header = "true"
delimiter = ","

# The applied options are for CSV files. For other file types, these
will be ignored.
df1 = spark.read.format(file_type) \
  .option("inferSchema", infer_schema) \
  .option("header", first_row_is_header) \
  .option("sep", delimiter) \
  .load(file_location)

file_location = "/FileStore/tables/tripSample.csv"
file_type = "csv"
```

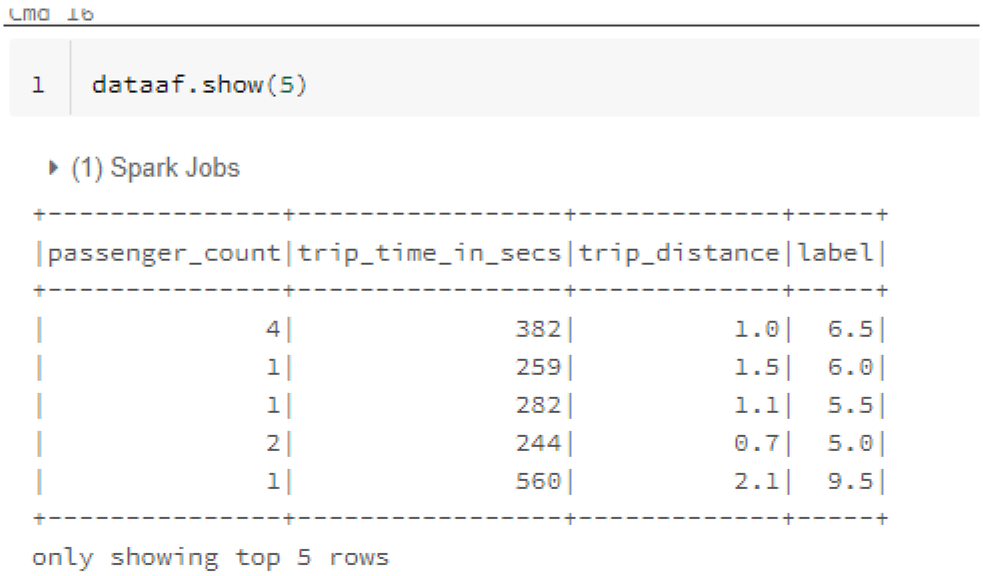
```
# CSV options
infer_schema = "true"
first_row_is_header = "true"
delimiter = ","

# The applied options are for CSV files. For other file types, these
will be ignored.
df2 = spark.read.format(file_type) \
  .option("inferSchema", infer_schema) \
  .option("header", first_row_is_header) \
  .option("sep", delimiter) \
  .load(file_location)

temp_table_name1 = "fareSample_csv"
df1.createOrReplaceTempView(temp_table_name1)

temp_table_name2 = "tripSample_csv"
df2.createOrReplaceTempView(temp_table_name2)

if IS_SPARK_SUBMIT_CLI:
    fareSample = spark.read.csv('/FileStore/tables/fareSample.csv',
inferSchema=True, header=True)
else:
  fareSample = sqlContext.sql("select * from fareSample_csv")
if IS_SPARK_SUBMIT_CLI:
    tripSample = spark.read.csv('/FileStore/tables/tripSample.csv',
inferSchema=True, header=True)
else:
    tripSample = sqlContext.sql("select * from tripSample_csv")
```

35. Now we need to **combine** the two datasets by insert an **id** column to each dataset and **inner join** them:

```
fareSample = fareSample.withColumn('id',
monotonically_increasing_id())
tripSample = tripSample.withColumn('id',
monotonically_increasing_id())
data=fareSample.join(tripSample,"id")
```

36. Select the data, **clean** the **null** rows and show first **5** lines:

```
data1 = data.select('passenger_count', 'trip_time_in_secs',
'trip_distance', col(' fare_amount').alias('label'))
dataaf = data1.dropna(how = 'any')
dataaf.show(5)
```



```
dataaf.show(5)
```

```
+---------------+----------------+-------------+-----+
|passenger_count|trip_time_in_secs|trip_distance|label|
+---------------+----------------+-------------+-----+
|              4|             382|          1.0|  6.5|
|              1|             259|          1.5|  6.0|
|              1|             282|          1.1|  5.5|
|              2|             244|          0.7|  5.0|
|              1|             560|          2.1|  9.5|
+---------------+----------------+-------------+-----+
only showing top 5 rows
```

37. Setup **train** and **test** datasets:

```
splits = dataaf.randomSplit([0.7, 0.3])
train = splits[0]
test = splits[1].withColumnRenamed("label", "trueLabel")
```

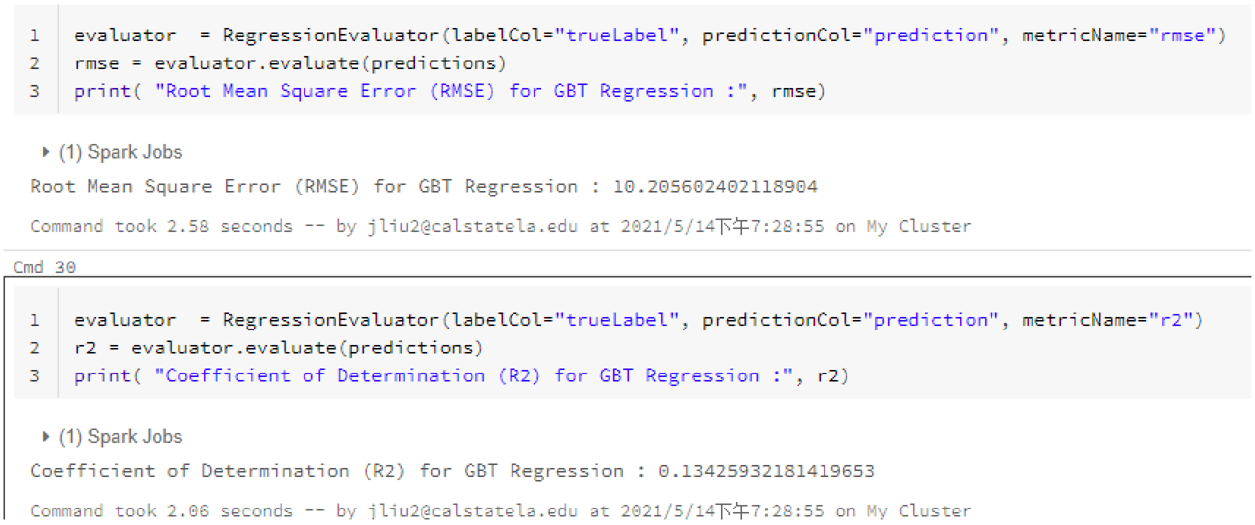
38. Now we need to setup **GBT-Regression** which will be using in KaggleTaxi Sample by following codes:

```
assembler = VectorAssembler(inputCols = ['passenger_count',
'trip_time_in_secs', 'trip_distance'], outputCol="features")
gbt = GBTRegressor(labelCol="label")
paramGrid = ParamGridBuilder()\
  .addGrid(gbt.maxDepth, [2, 3])\
  .addGrid(gbt.maxIter, [10, 20])\
  .build()

cv = CrossValidator(estimator=gbt, evaluator=RegressionEvaluator(),
estimatorParamMaps=paramGrid)

pipeline = Pipeline(stages=[assembler, cv])
pipelineModel = pipeline.fit(train)
predictions = pipelineModel.transform(test)
predicted = predictions.select("features", "prediction", "trueLabel")
predicted.createOrReplaceTempView("regressionPredictions")
```

39. Setup **RMSE** and **R2**:

```
evaluator = RegressionEvaluator(labelCol="trueLabel",
predictionCol="prediction", metricName="rmse")
rmse = evaluator.evaluate(predictions)
print( "Root Mean Square Error (RMSE) for GBT Regression :", rmse)
```

```
evaluator = RegressionEvaluator(labelCol="trueLabel",
predictionCol="prediction", metricName="r2")
r2 = evaluator.evaluate(predictions)
print( "Coefficient of Determination (R2) for GBT Regression :", r2)
```



```
evaluator  = RegressionEvaluator(labelCol="trueLabel", predictionCol="prediction", metricName="rmse")
rmse = evaluator.evaluate(predictions)
print( "Root Mean Square Error (RMSE) for GBT Regression :", rmse)
```

```
Root Mean Square Error (RMSE) for GBT Regression : 10.205602402118904
```

Command took 2.58 seconds -- by jliu2@calstatela.edu at 2021/5/14下午7:28:55 on My Cluster

```
evaluator  = RegressionEvaluator(labelCol="trueLabel", predictionCol="prediction", metricName="r2")
r2 = evaluator.evaluate(predictions)
print( "Coefficient of Determination (R2) for GBT Regression :", r2)
```

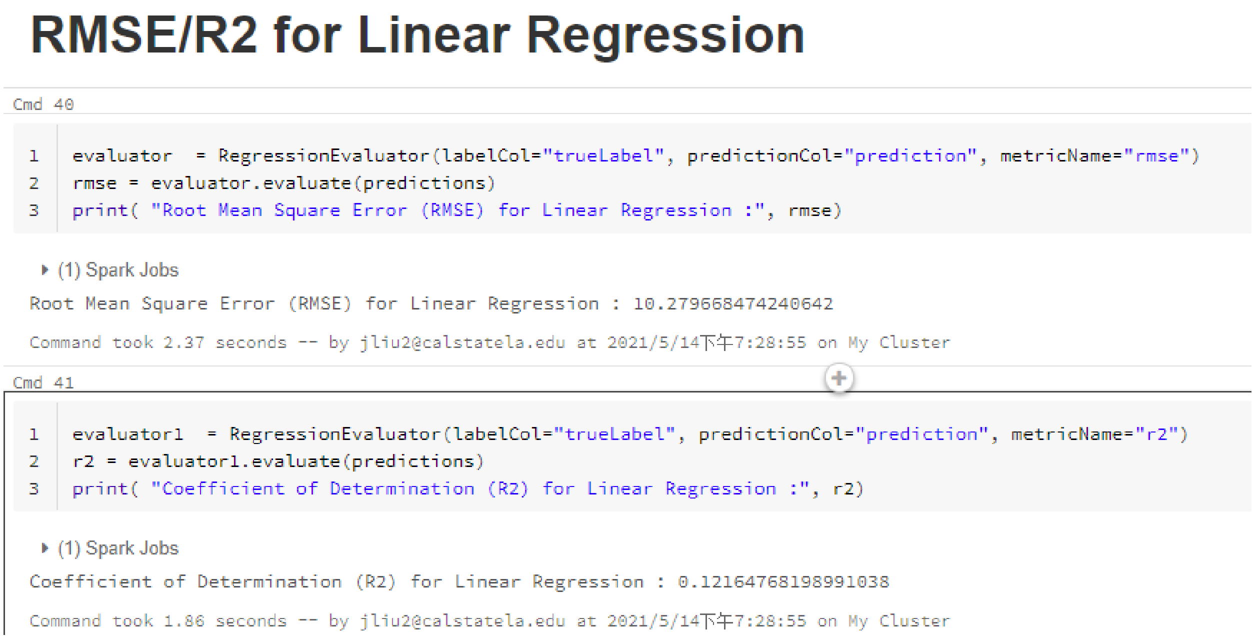
```
Coefficient of Determination (R2) for GBT Regression : 0.13425932181419653
```

Command took 2.06 seconds -- by jliu2@calstatela.edu at 2021/5/14下午7:28:55 on My Cluster

40. Setup **Linear Regression**, **RMSE** and **R2** for LR:

```
assembler = VectorAssembler(inputCols = ['passenger_count',
'trip_time_in_secs', 'trip_distance'], outputCol="features")

lr = LinearRegression(labelCol="label",featuresCol="features",
maxIter=10, regParam=0.3)

pipeline1 = Pipeline(stages=[assembler, lr])
paramGrid1 = ParamGridBuilder().addGrid(lr.regParam, [0.3,
0.01]).addGrid(lr.maxIter, [10, 5]).build()

trainval = TrainValidationSplit(estimator=pipeline1,
evaluator=RegressionEvaluator(), estimatorParamMaps=paramGrid1,
trainRatio=0.8)

pipelineModel = trainval.fit(train)
predictions = pipelineModel.transform(test)
predicted = predictions.select("features", "prediction", "trueLabel")
predicted.createOrReplaceTempView("regressionPredictions")

evaluator = RegressionEvaluator(labelCol="trueLabel",
predictionCol="prediction", metricName="rmse")
rmse = evaluator.evaluate(predictions)
print( "Root Mean Square Error (RMSE) for Linear Regression :", rmse)

evaluator1 = RegressionEvaluator(labelCol="trueLabel",
predictionCol="prediction", metricName="r2")
r2 = evaluator1.evaluate(predictions)
```

```
print( "Coefficient of Determination (R2) for Linear Regression :",
r2)
```



RMSE/R2 for Linear Regression

Cmd 40

```
evaluator  = RegressionEvaluator(labelCol="trueLabel", predictionCol="prediction", metricName="rmse")
rmse = evaluator.evaluate(predictions)
print( "Root Mean Square Error (RMSE) for Linear Regression :", rmse)
```

```
Root Mean Square Error (RMSE) for Linear Regression : 10.279668474240642
```

Command took 2.37 seconds -- by jliu2@calstatela.edu at 2021/5/14下午7:28:55 on My Cluster

Cmd 41

```
evaluator1  = RegressionEvaluator(labelCol="trueLabel", predictionCol="prediction", metricName="r2")
r2 = evaluator1.evaluate(predictions)
print( "Coefficient of Determination (R2) for Linear Regression :", r2)
```

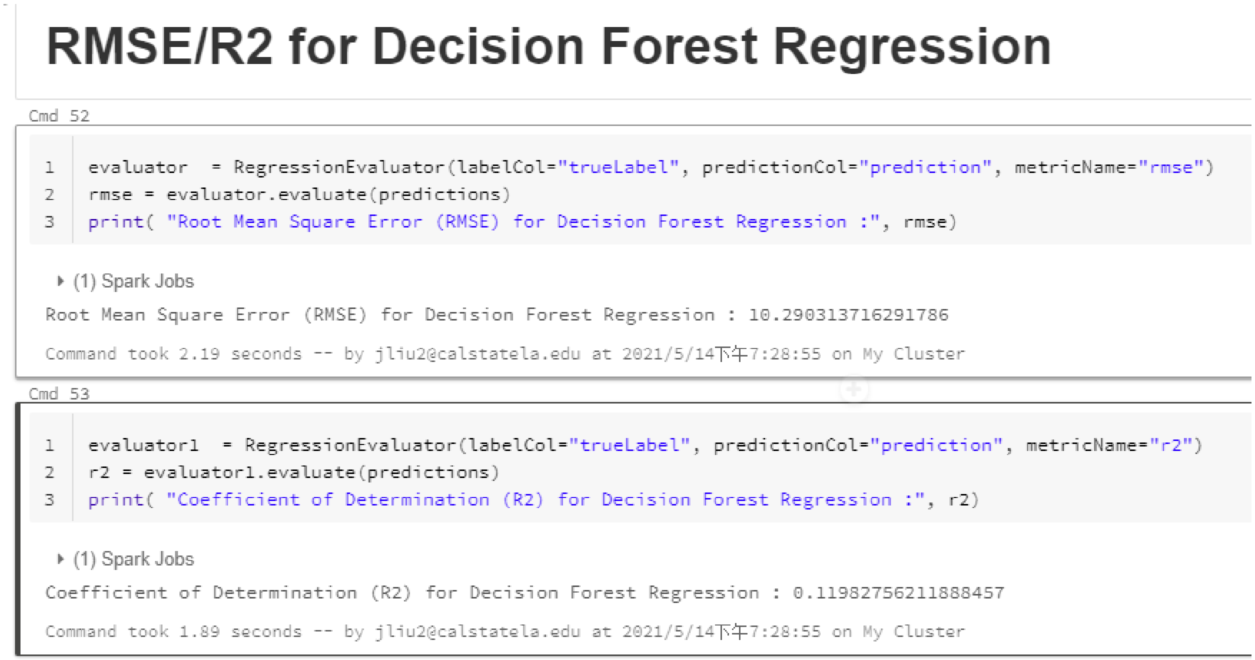
```
Coefficient of Determination (R2) for Linear Regression : 0.12164768198991038
```

Command took 1.86 seconds -- by jliu2@calstatela.edu at 2021/5/14下午7:28:55 on My Cluster

41. Setup **Decision Forest Regression**, **RMSE** and **R2** as before by following codes:

```
assembler    =    VectorAssembler(inputCols    =    ['passenger_count',
'trip_time_in_secs', 'trip_distance'], outputCol="features")
dt = DecisionTreeRegressor(labelCol="label",featuresCol="features")

paramGrid2 = ParamGridBuilder()\
  .addGrid(dt.maxDepth, [2,3])\
  .addGrid(dt.maxBins, [10,20])\
  .build()

dtcv = CrossValidator(estimator = dt, estimatorParamMaps = paramGrid2,
evaluator = RegressionEvaluator(), numFolds=2)

pipeline2 = Pipeline(stages=[assembler, dtcv])
pipelineModel = pipeline2.fit(train)
predictions = pipelineModel.transform(test)
predicted = predictions.select("features","prediction","truelabel")
predicted.createOrReplaceTempView("regressionPredictions")

evaluator             =      RegressionEvaluator(labelCol="trueLabel",
predictionCol="prediction", metricName="rmse")
rmse = evaluator.evaluate(predictions)
print(  "Root  Mean  Square  Error  (RMSE)  for  Decision  Forest
Regression :", rmse)
```

```
evaluator1 = RegressionEvaluator(labelCol="trueLabel", predictionCol="prediction", metricName="r2")
r2 = evaluator1.evaluate(predictions)
print( "Coefficient of Determination (R2) for Decision Forest Regression :", r2)
```



RMSE/R2 for Decision Forest Regression

Cmd 52

```
evaluator  = RegressionEvaluator(labelCol="trueLabel", predictionCol="prediction", metricName="rmse")
rmse = evaluator.evaluate(predictions)
print( "Root Mean Square Error (RMSE) for Decision Forest Regression :", rmse)
```

▸ (1) Spark Jobs

```
Root Mean Square Error (RMSE) for Decision Forest Regression : 10.290313716291786
```

Command took 2.19 seconds -- by jliu2@calstatela.edu at 2021/5/14下午7:28:55 on My Cluster

Cmd 53

```
evaluator1  = RegressionEvaluator(labelCol="trueLabel", predictionCol="prediction", metricName="r2")
r2 = evaluator1.evaluate(predictions)
print( "Coefficient of Determination (R2) for Decision Forest Regression :", r2)
```

▸ (1) Spark Jobs

```
Coefficient of Determination (R2) for Decision Forest Regression : 0.11982756211888457
```

Command took 1.89 seconds -- by jliu2@calstatela.edu at 2021/5/14下午7:28:55 on My Cluster

42. **Run** all the code and now you can compare the results for all three different models. For the **paramGrid** in GBT-Regression, we using **[2,3] for maxDepth and [10,20] for maxIter**. You may want to using **[2,5] for maxDepth and [10,100] for maxIter** to have more accuracy results.

43. You may also export the code as a **py** file for **Hadoop Spark** and **ipynb** file for **IPythonNotebook.**

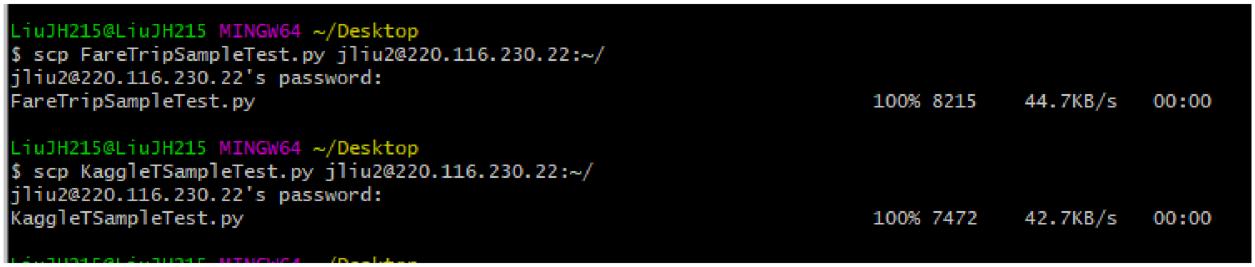
44. As we can see here, the GBT-Regression is the best model for both datasets based on the RMSE and R2. However, the Fare and Trip Samples return a different R2 compare to KaggleTaxi Sample. The reason could be the way how we combine the two(fare and trip) datasets into one for train and test.

Task 4: Hadoop Spark(optional)

This step will introduce Hadoop Spark

1. Open a Git Bash window. And upload the sample datasets, py file to Hadoop(you need to do this for three sample datasets and two py files. And using your user name instead of jliu2):

```
scp FareTripSampleTest.py jliu2@220.116.230.22:~/
```

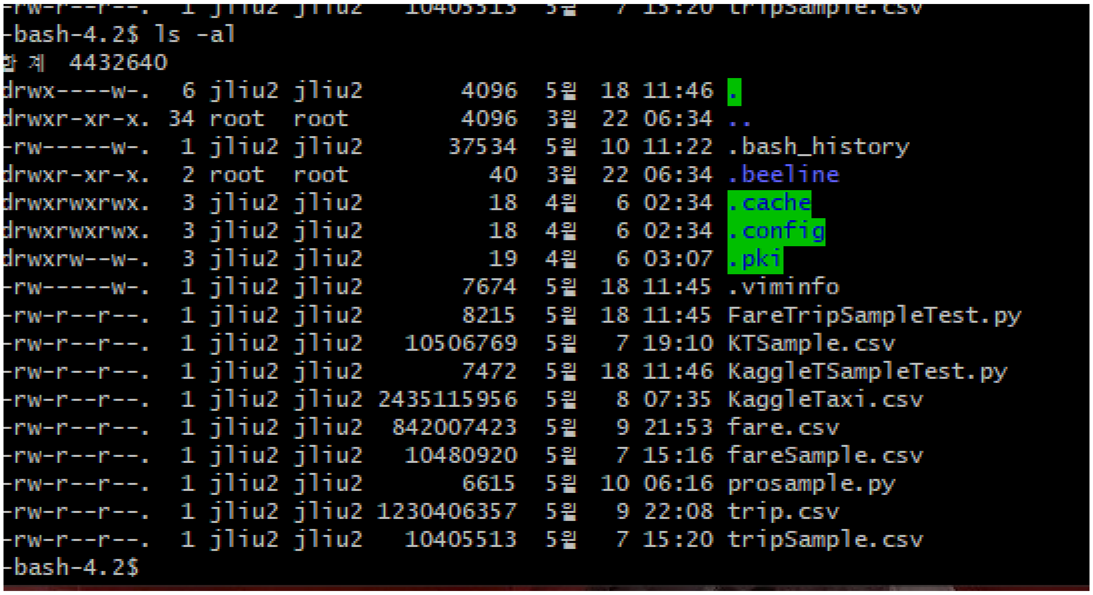


```
LiuJH215@LiuJH215 MINGW64 ~/Desktop
$ scp FareTripSampleTest.py jliu2@220.116.230.22:~/
jliu2@220.116.230.22's password:
FareTripSampleTest.py                                                     100% 8215     44.7KB/s    00:00

LiuJH215@LiuJH215 MINGW64 ~/Desktop
$ scp KaggleTSampleTest.py jliu2@220.116.230.22:~/
jliu2@220.116.230.22's password:
KaggleTSampleTest.py                                                      100% 7472     42.7KB/s    00:00

LiuJH215@LiuJH215 MINGW64 ~/Desktop
```

2. Once finish upload, open a new Git Bush window and login into Hadoop by following code using your username and password:

```
ssh jliu2@220.116.230.22
```

3. using **ls -al** to check all the files uploaded correctly.



```
-rw-r--r--.  1 jliu2 jliu2   10405513  5월   7 15:20 tripSample.csv
-bash-4.2$ ls -al
합 계  4432640
drwx----w-.  6 jliu2 jliu2       4096  5월  18 11:46 .
drwxr-xr-x. 34 root  root        4096  3월  22 06:34 ..
-rw-----w-.  1 jliu2 jliu2      37534  5월  10 11:22 .bash_history
drwxr-xr-x.  2 root  root          40  3월  22 06:34 .beeline
drwxrwxrwx.  3 jliu2 jliu2         18  4월   6 02:34 .cache
drwxrwxrwx.  3 jliu2 jliu2         18  4월   6 02:34 .config
drwxrw--w-.  3 jliu2 jliu2         19  4월   6 03:07 .pki
-rw-----w-.  1 jliu2 jliu2       7674  5월  18 11:45 .viminfo
-rw-r--r--.  1 jliu2 jliu2       8215  5월  18 11:45 FareTripSampleTest.py
-rw-r--r--.  1 jliu2 jliu2   10506769  5월   7 19:10 KTSample.csv
-rw-r--r--.  1 jliu2 jliu2       7472  5월  18 11:46 KaggleTSampleTest.py
-rw-r--r--.  1 jliu2 jliu2 2435115956  5월   8 07:35 KaggleTaxi.csv
-rw-r--r--.  1 jliu2 jliu2  842007423  5월   9 21:53 fare.csv
-rw-r--r--.  1 jliu2 jliu2   10480920  5월   7 15:16 fareSample.csv
-rw-r--r--.  1 jliu2 jliu2       6615  5월  10 06:16 prosample.py
-rw-r--r--.  1 jliu2 jliu2 1230406357  5월   9 22:08 trip.csv
-rw-r--r--.  1 jliu2 jliu2   10405513  5월   7 15:20 tripSample.csv
-bash-4.2$
```

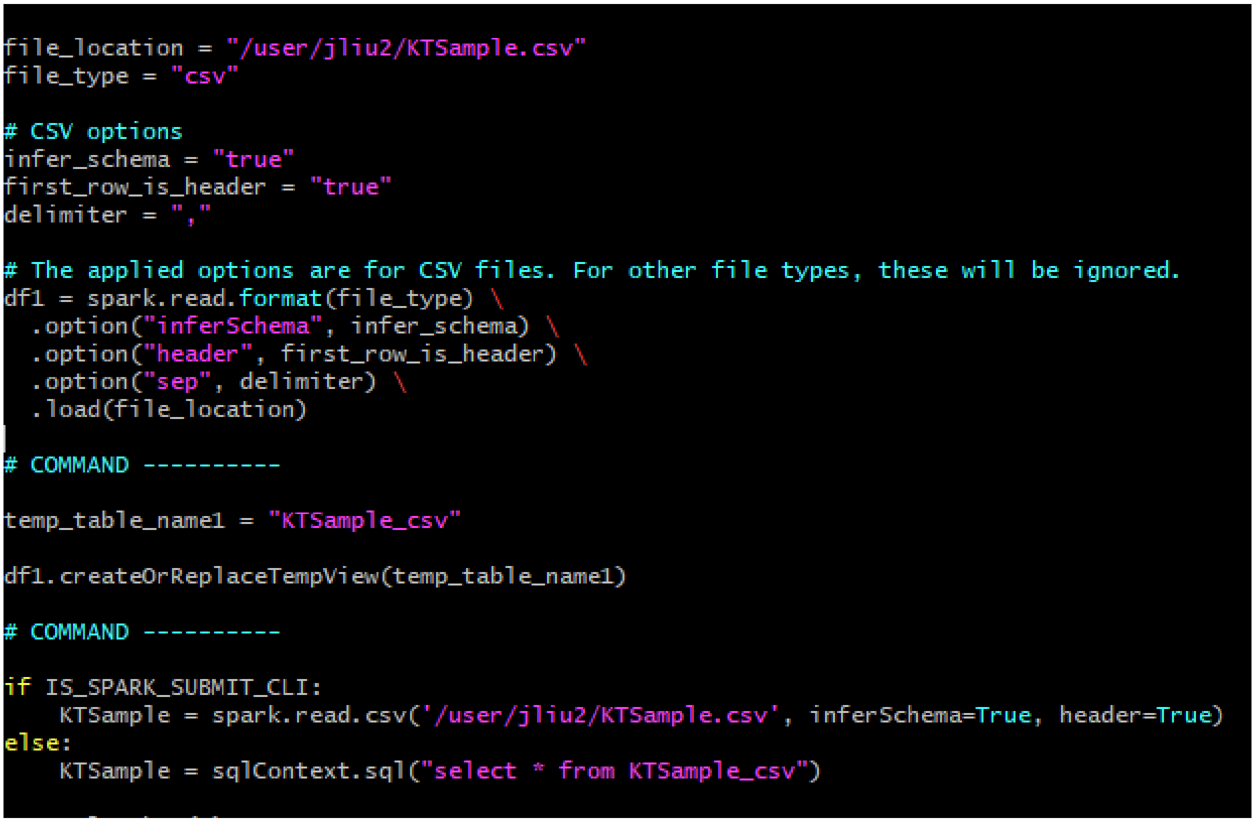
4. Using following code to put all the dataset to **HDFS** and give permission to read/write:

```
hdfs dfs -put KTSample.csv

hdfs dfs -chmod -R o+w .
```

5. edit KaggleTSampleTest.py by using **vi**:

```
vi KaggleTSampleTest.py
```

6. Change file location and save it(change **jliu2** to your own user name).

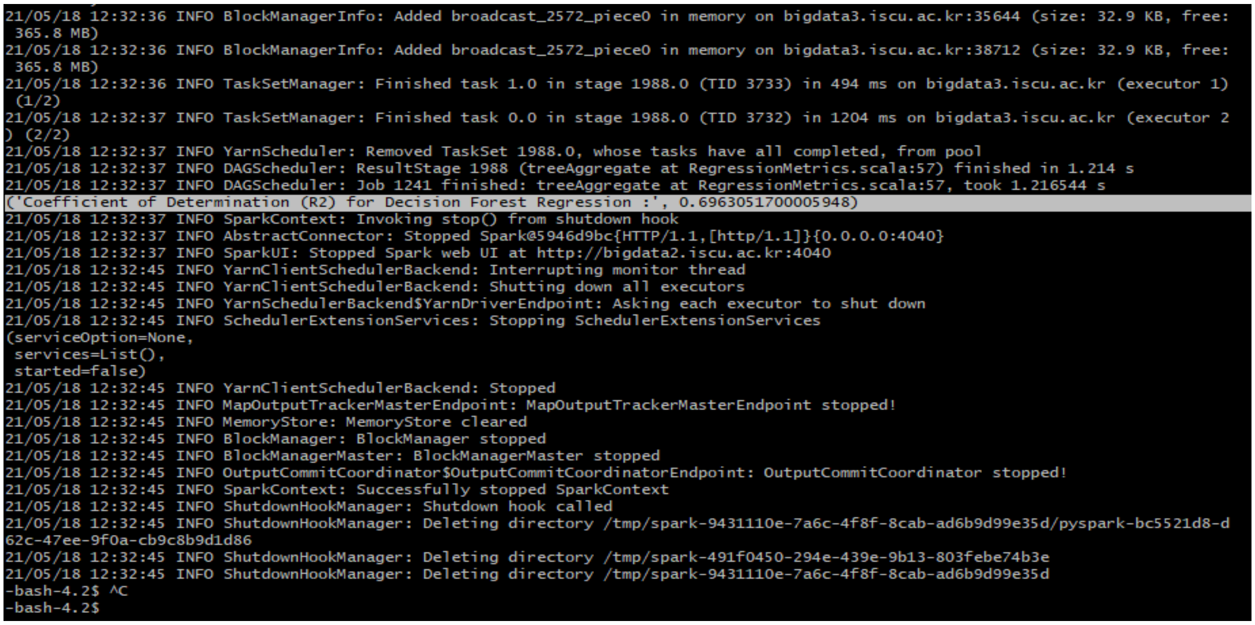


```
file_location = "/user/jliu2/KTSample.csv"
file_type = "csv"

# CSV options
infer_schema = "true"
first_row_is_header = "true"
delimiter = ","

# The applied options are for CSV files. For other file types, these will be ignored.
df1 = spark.read.format(file_type) \
  .option("inferSchema", infer_schema) \
  .option("header", first_row_is_header) \
  .option("sep", delimiter) \
  .load(file_location)

# COMMAND ----------

temp_table_name1 = "KTSample_csv"

df1.createOrReplaceTempView(temp_table_name1)

# COMMAND ----------

if IS_SPARK_SUBMIT_CLI:
    KTSample = spark.read.csv('/user/jliu2/KTSample.csv', inferSchema=True, header=True)
else:
    KTSample = sqlContext.sql("select * from KTSample_csv")
```

7. run the py file by using following code:

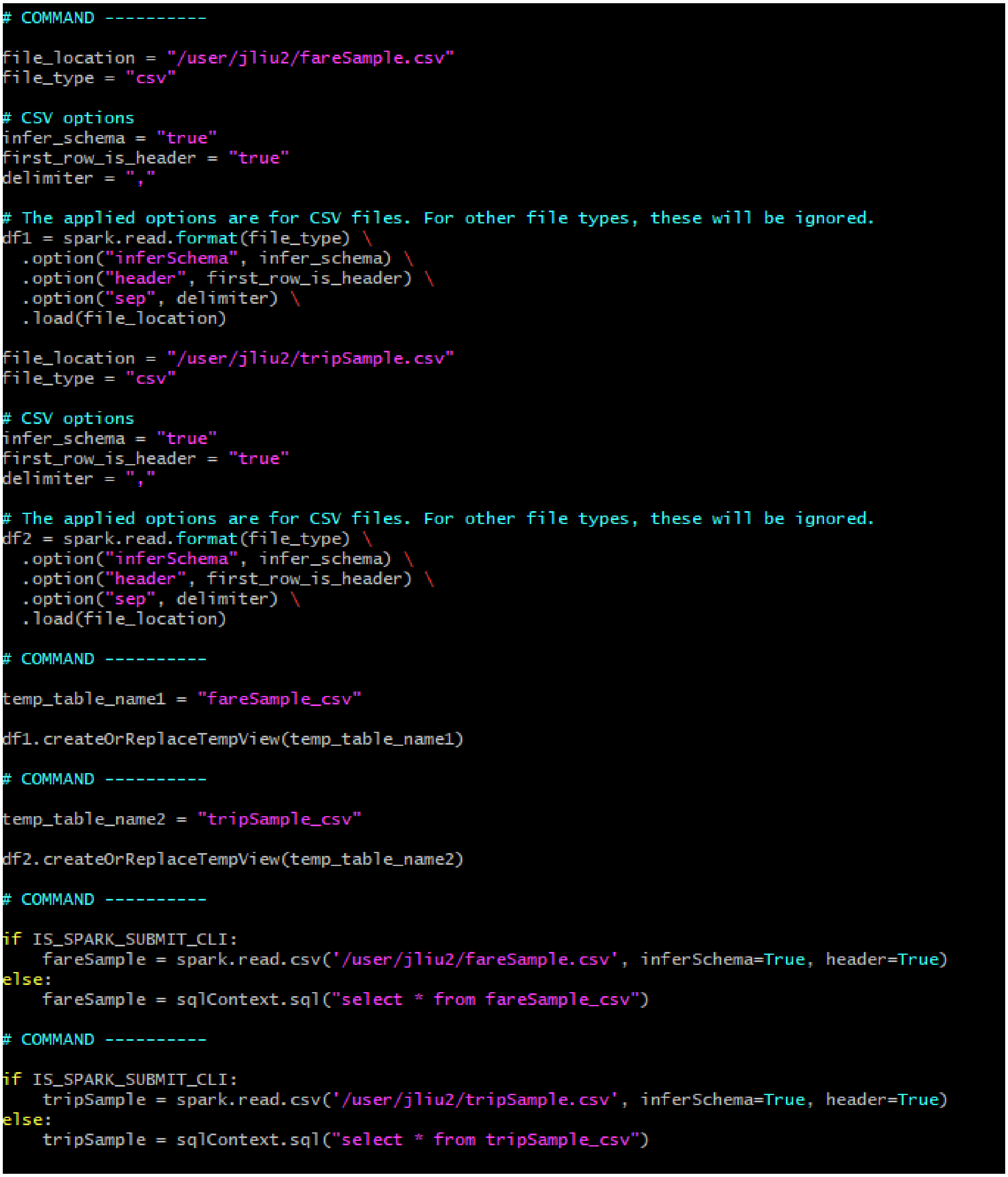
```
spark-submit KaggleTSampleTest.py
```

8. After finished, you will see the **RMSE** and **R2** like the following, and you may also want to find the **RMSE** and **R2** for all **three** models, or you can modify the py file to display all results at the end.



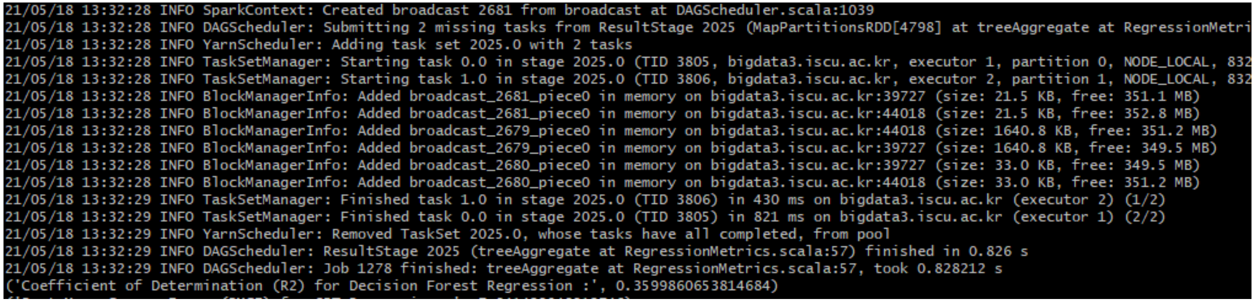
```
21/05/18 12:32:36 INFO BlockManagerInfo: Added broadcast_2572_piece0 in memory on bigdata3.iscu.ac.kr:35644 (size: 32.9 KB, free:
 365.8 MB)
21/05/18 12:32:36 INFO BlockManagerInfo: Added broadcast_2572_piece0 in memory on bigdata3.iscu.ac.kr:38712 (size: 32.9 KB, free:
 365.8 MB)
21/05/18 12:32:36 INFO TaskSetManager: Finished task 1.0 in stage 1988.0 (TID 3733) in 494 ms on bigdata3.iscu.ac.kr (executor 1)
 (1/2)
21/05/18 12:32:37 INFO TaskSetManager: Finished task 0.0 in stage 1988.0 (TID 3732) in 1204 ms on bigdata3.iscu.ac.kr (executor 2
) (2/2)
21/05/18 12:32:37 INFO YarnScheduler: Removed TaskSet 1988.0, whose tasks have all completed, from pool
21/05/18 12:32:37 INFO DAGScheduler: ResultStage 1988 (treeAggregate at RegressionMetrics.scala:57) finished in 1.214 s
21/05/18 12:32:37 INFO DAGScheduler: Job 1241 finished: treeAggregate at RegressionMetrics.scala:57, took 1.216544 s
('Coefficient of Determination (R2) for Decision Forest Regression :', 0.6963051700005948)
21/05/18 12:32:37 INFO SparkContext: Invoking stop() from shutdown hook
21/05/18 12:32:37 INFO AbstractConnector: Stopped Spark@5946d9bc{HTTP/1.1,[http/1.1]}{0.0.0.0:4040}
21/05/18 12:32:37 INFO SparkUI: Stopped Spark web UI at http://bigdata2.iscu.ac.kr:4040
21/05/18 12:32:45 INFO YarnClientSchedulerBackend: Interrupting monitor thread
21/05/18 12:32:45 INFO YarnClientSchedulerBackend: Shutting down all executors
21/05/18 12:32:45 INFO YarnSchedulerBackend$YarnDriverEndpoint: Asking each executor to shut down
21/05/18 12:32:45 INFO SchedulerExtensionServices: Stopping SchedulerExtensionServices
(serviceOption=None,
 services=List(),
 started=false)
21/05/18 12:32:45 INFO YarnClientSchedulerBackend: Stopped
21/05/18 12:32:45 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
21/05/18 12:32:45 INFO MemoryStore: MemoryStore cleared
21/05/18 12:32:45 INFO BlockManager: BlockManager stopped
21/05/18 12:32:45 INFO BlockManagerMaster: BlockManagerMaster stopped
21/05/18 12:32:45 INFO OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: OutputCommitCoordinator stopped!
21/05/18 12:32:45 INFO SparkContext: Successfully stopped SparkContext
21/05/18 12:32:45 INFO ShutdownHookManager: Shutdown hook called
21/05/18 12:32:45 INFO ShutdownHookManager: Deleting directory /tmp/spark-9431110e-7a6c-4f8f-8cab-ad6b9d99e35d/pyspark-bc5521d8-d
62c-47ee-9f0a-cb9c8b9d1d86
21/05/18 12:32:45 INFO ShutdownHookManager: Deleting directory /tmp/spark-491f0450-294e-439e-9b13-803febe74b3e
21/05/18 12:32:45 INFO ShutdownHookManager: Deleting directory /tmp/spark-9431110e-7a6c-4f8f-8cab-ad6b9d99e35d
-bash-4.2$ ^C
-bash-4.2$
```

9. Repeat step 5 to 7 for **FareTripSampleTest.py**



```
# COMMAND ----------

file_location = "/user/jliu2/fareSample.csv"
file_type = "csv"

# CSV options
infer_schema = "true"
first_row_is_header = "true"
delimiter = ","

# The applied options are for CSV files. For other file types, these will be ignored.
df1 = spark.read.format(file_type) \
  .option("inferSchema", infer_schema) \
  .option("header", first_row_is_header) \
  .option("sep", delimiter) \
  .load(file_location)

file_location = "/user/jliu2/tripSample.csv"
file_type = "csv"

# CSV options
infer_schema = "true"
first_row_is_header = "true"
delimiter = ","

# The applied options are for CSV files. For other file types, these will be ignored.
df2 = spark.read.format(file_type) \
  .option("inferSchema", infer_schema) \
  .option("header", first_row_is_header) \
  .option("sep", delimiter) \
  .load(file_location)

# COMMAND ----------

temp_table_name1 = "fareSample_csv"

df1.createOrReplaceTempView(temp_table_name1)

# COMMAND ----------

temp_table_name2 = "tripSample_csv"

df2.createOrReplaceTempView(temp_table_name2)

# COMMAND ----------

if IS_SPARK_SUBMIT_CLI:
    fareSample = spark.read.csv('/user/jliu2/fareSample.csv', inferSchema=True, header=True)
else:
    fareSample = sqlContext.sql("select * from fareSample_csv")

# COMMAND ----------

if IS_SPARK_SUBMIT_CLI:
    tripSample = spark.read.csv('/user/jliu2/tripSample.csv', inferSchema=True, header=True)
else:
    tripSample = sqlContext.sql("select * from tripSample_csv")
```

10. After finished, you will see the **RMSE** and **R2** like the following, and you may also want to find the **RMSE** and **R2** for all **three** models, or you can modify the py file to display all results at the end.



```
21/05/18 13:32:28 INFO SparkContext: Created broadcast 2681 from broadcast at DAGScheduler.scala:1039
21/05/18 13:32:28 INFO DAGScheduler: Submitting 2 missing tasks from ResultStage 2025 (MapPartitionsRDD[4798] at treeAggregate at RegressionMetri
21/05/18 13:32:28 INFO YarnScheduler: Adding task set 2025.0 with 2 tasks
21/05/18 13:32:28 INFO TaskSetManager: Starting task 0.0 in stage 2025.0 (TID 3805, bigdata3.iscu.ac.kr, executor 1, partition 0, NODE_LOCAL, 832
21/05/18 13:32:28 INFO TaskSetManager: Starting task 1.0 in stage 2025.0 (TID 3806, bigdata3.iscu.ac.kr, executor 2, partition 1, NODE_LOCAL, 832
21/05/18 13:32:28 INFO BlockManagerInfo: Added broadcast_2681_piece0 in memory on bigdata3.iscu.ac.kr:39727 (size: 21.5 KB, free: 351.1 MB)
21/05/18 13:32:28 INFO BlockManagerInfo: Added broadcast_2681_piece0 in memory on bigdata3.iscu.ac.kr:44018 (size: 21.5 KB, free: 352.8 MB)
21/05/18 13:32:28 INFO BlockManagerInfo: Added broadcast_2679_piece0 in memory on bigdata3.iscu.ac.kr:44018 (size: 1640.8 KB, free: 351.2 MB)
21/05/18 13:32:28 INFO BlockManagerInfo: Added broadcast_2679_piece0 in memory on bigdata3.iscu.ac.kr:39727 (size: 1640.8 KB, free: 349.5 MB)
21/05/18 13:32:28 INFO BlockManagerInfo: Added broadcast_2680_piece0 in memory on bigdata3.iscu.ac.kr:39727 (size: 33.0 KB, free: 349.5 MB)
21/05/18 13:32:28 INFO BlockManagerInfo: Added broadcast_2680_piece0 in memory on bigdata3.iscu.ac.kr:44018 (size: 33.0 KB, free: 351.2 MB)
21/05/18 13:32:29 INFO TaskSetManager: Finished task 1.0 in stage 2025.0 (TID 3806) in 430 ms on bigdata3.iscu.ac.kr (executor 2) (1/2)
21/05/18 13:32:29 INFO TaskSetManager: Finished task 0.0 in stage 2025.0 (TID 3805) in 821 ms on bigdata3.iscu.ac.kr (executor 1) (2/2)
21/05/18 13:32:29 INFO YarnScheduler: Removed TaskSet 2025.0, whose tasks have all completed, from pool
21/05/18 13:32:29 INFO DAGScheduler: ResultStage 2025 (treeAggregate at RegressionMetrics.scala:57) finished in 0.826 s
21/05/18 13:32:29 INFO DAGScheduler: Job 1278 finished: treeAggregate at RegressionMetrics.scala:57, took 0.828212 s
('Coefficient of Determination (R2) for Decision Forest Regression :', 0.3599860653814684)
```

11. According to all the results, the GBT Regression will give the best prediction. And compare to the databricks, R2 in Hadoop increased.

12. (**Optional**) you can also try the original dataset for Kaggle Taxi but be aware, it will take more than **6** hours to return the results.

References

1. **URL of Data Source:** https://www.kaggle.com/microize/newyork-yellow-taxi-trip-data-2020-2019 and https://chriswhong.com/open-data/foil_nyc_taxi/

2. **GitHub URL:** https://github.com/liujh215/CIS5560.git

3. **Other Reference:**

- https://sparkbyexamples.com/spark/spark-difference-between-two-timestamps-in-seconds-minutes-and-hours/
- https://spark.apache.org/docs/latest/mllib-decision-tree.html
- https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.sql.DataFrame.dropna.html
- https://spark.apache.org/docs/2.1.1/api/R/spark.gbt.html
- https://gist.github.com/colbyford/daa4508f6d8d94a405e7bd3a50c5ed77
- https://docs.rapidminer.com/latest/studio/operators/modeling/predictive/trees/gradient_boosted_trees.html
- https://docs.azuredatabricks.net/_static/notebooks/gbt-regression.html