# [C:\Users\jwoo5\AppData\Local\Temp\templateTermTutorial.html](http://www.calstatela.edu/centers/hipic) CIS5560 Term Project Tutorial https://avatars2.githubusercontent.com/u/4156894?v=3&s=100

California State University, Los Angeles

#### Authors: Ashish Solanki, Vinayak Gaur, Rakshit Makkar

#### Instructor: [Jongwook Woo](https://www.linkedin.com/in/jongwook-woo-7081a85)

#### Date: 05/19/2019

**Lab Tutorial**

Ashish Solanki ([asolank5@calstatela.edu](mailto:asolank5@calstatela.edu))

Rakshit Makkar ([rmakkar@calstatela.edu](mailto:rmakkar@calstatela.edu))

Vinayak Gaur ([vgaur@calstatela.edu](mailto:vgaur@calstatela.edu))

05/19/2019

**Safecast Radiation Measurement**

**Objectives**

**List what your objectives are.** In this hands-on lab, you will learn how to:

* Get data manually from Kaggle.com
* Create Spark cluster
* Add Data and create table
* Using Regression for prediction

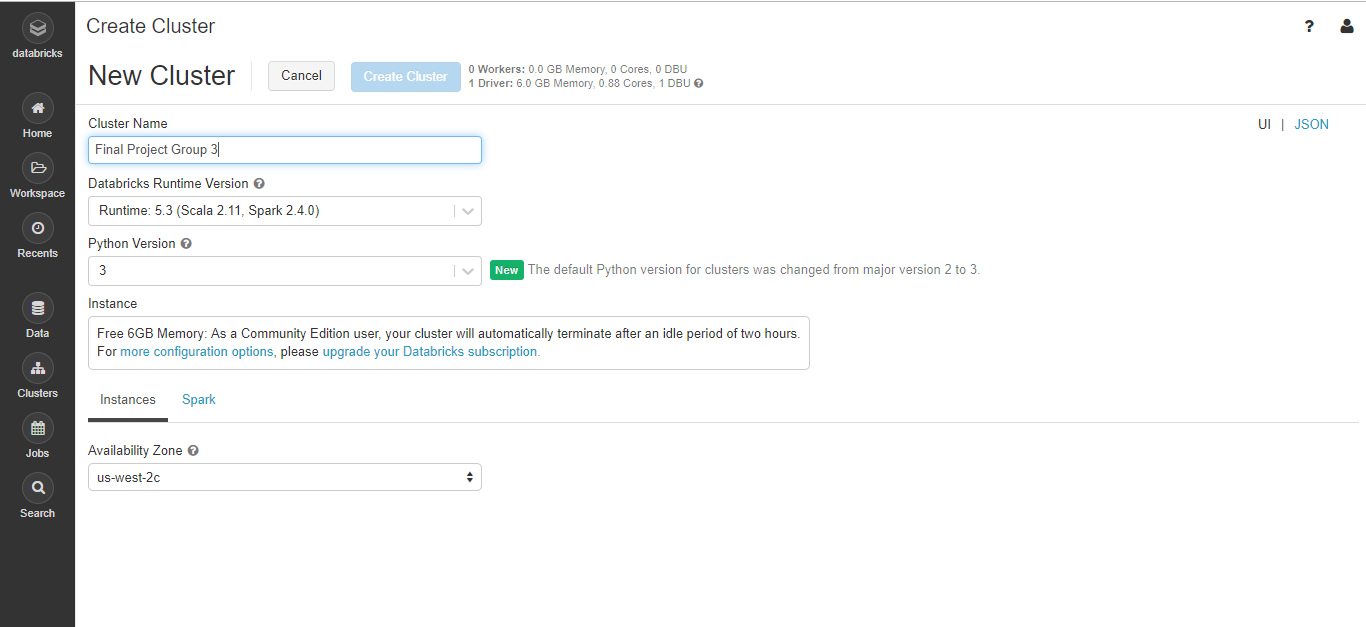
**Platform Spec**

* Databricks
* Execution: Single Node
* Max Storage 2 GB Memory, 0.88 Cores, 1 DBU
* Databricks Runtime Version 4.0
* Python Version 3
* Includes Apache Spark 2.3.0, Scala 2.11

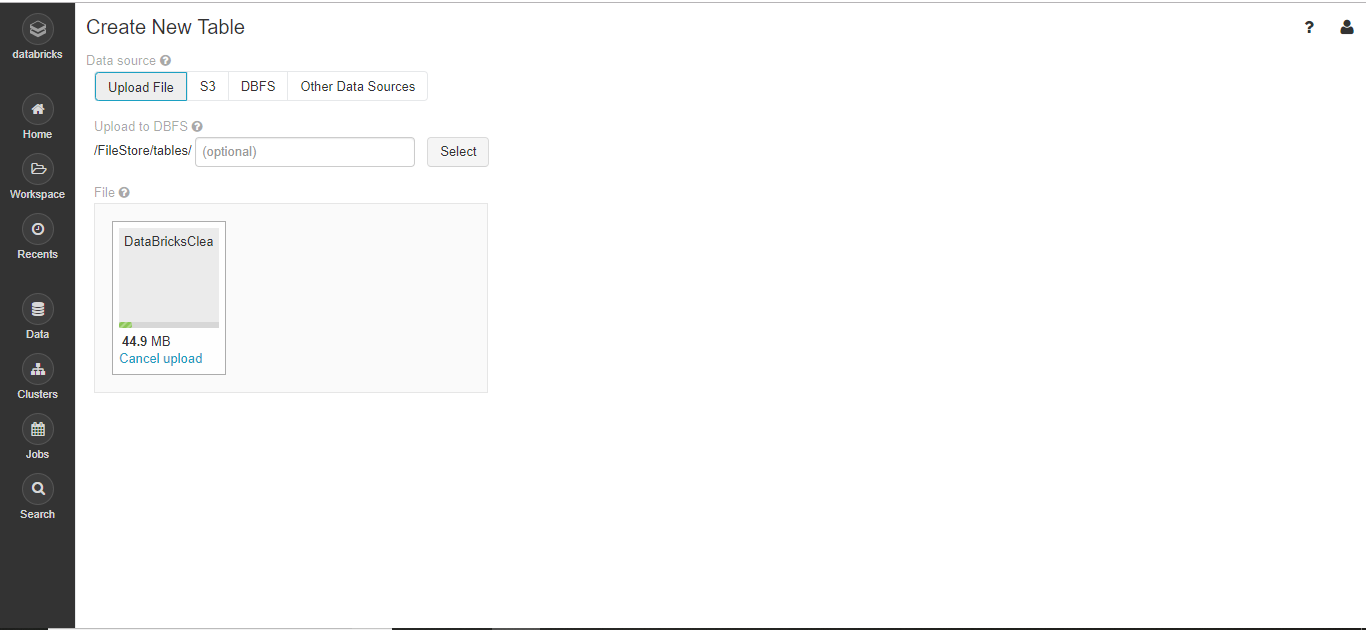
Step 1: Get data manually

This step is to get data manually for the cluster

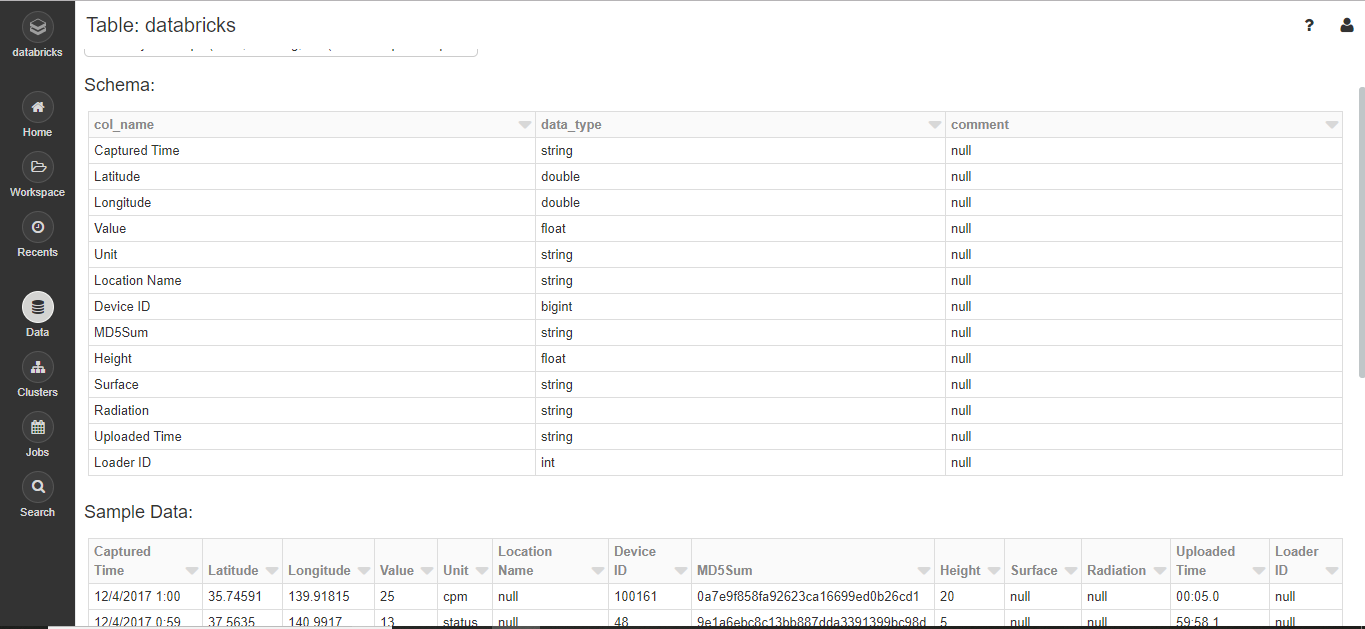
1. Sign into your databricks account.
2. Go to Clusters option on the left and click on create cluster.
3. Give the cluster name and click create cluster.



1. Under tables section click on create table and select the file to upload.



After creation, databricks table looks like below



Step 2: Train Model

**Here, we are explaining every single step we implemented in Databricks for prediction of radiation data of the year 2017 and predicting the outcome of future months based on the regression model**

1. Import Spark SQL and Spark ML libraries

*# Import Spark SQL and Spark ML libraries*

*from pyspark.sql.types import \**

*from pyspark.sql.functions import \**

*from pyspark.ml import Pipeline*

*from pyspark.ml.classification import LogisticRegression*

*from pyspark.ml.feature import VectorAssembler*

*from pyspark.ml.tuning import ParamGridBuilder, TrainValidationSplit, CrossValidator*

*from pyspark.ml.evaluation import BinaryClassificationEvaluator*

*from pyspark.ml.evaluation import RegressionEvaluator*

*from pyspark.ml.regression import LinearRegression*

1. DataFrame Schema

*# DataFrame Schema that should be a Table schema*

*flightSchema = StructType([*

*StructField("DayofMonth", IntegerType(), False),*

*StructField("DayOfWeek", IntegerType(), False),*

*StructField("Carrier", StringType(), False),*

*StructField("OriginAirportID", IntegerType(), False),*

*StructField("DestAirportID", IntegerType(), False),*

*StructField("DepDelay", IntegerType(), False),*

*StructField("ArrDelay", IntegerType(), False),*

*])*

1. # Load the source data

*# Load the source data*

*csv = spark.sql("SELECT \* FROM databricks")*

1. Select Features and Labels

*# Select features and label*

*data = csv.select(col("Height").cast("Float"), col("Value").alias("label").cast("Float"),col("Latitude").cast("Double"),col("Longitude").cast("Double"))*

*# Split the data*

*splits = data.randomSplit([0.7, 0.3])*

*train = splits[0]*

*test = splits[1].withColumnRenamed("label", "trueLabel")*

1. Define the pipeline using linear regression

*# Define the pipeline*

*lr = []*

*pipeline = []*

*assembler = []*

*for i in range(3):*

*assembler.insert(i, VectorAssembler(inputCols = ["Height"], outputCol="features"))*

*lr.insert(i, LinearRegression(labelCol="label", featuresCol="features"))*

*pipeline.insert(i, Pipeline(stages=[assembler[i], lr[i]]))*

1. Model

*# define list of models made from Train Validation Split and Cross Validation*

*model = []*

1. ParamGrid

*# params refered to the reference above*

*paramGrid = (ParamGridBuilder() \*

*.addGrid(lr[0].regParam, [0.01, 0.5, 2.0]) \*

*.addGrid(lr[0].maxIter, [1, 5]) \*

*.build())*

1. Tune Parameters using TrainValidationSplit

*tvs = TrainValidationSplit(estimator=pipeline[0], evaluator=RegressionEvaluator(), estimatorParamMaps=paramGrid, trainRatio=0.8)*

*# the first best model*

*model.insert(0, tvs.fit(train))*

1. ParamGrid2- combination of parameters to evaluate each combination of parameters defined in a ParameterGrid

*paramGrid2 = (ParamGridBuilder() \*

*.addGrid(lr[0].regParam, [0.01, 0.5, 2.0]) \*

*.addGrid(lr[0].elasticNetParam, [0.0, 0.5, 1]) \*

*.addGrid(lr[0].maxIter, [1, 5]) \*

*.build())*

1. TrainValidationSplit

*tvs2 = TrainValidationSplit(estimator=pipeline[1], evaluator=RegressionEvaluator(), estimatorParamMaps=paramGrid2, trainRatio=0.8)*

*# the second best model*

*model.insert(1, tvs2.fit(train))*

1. Params referred above

*paramGridCV = (ParamGridBuilder() \*

*.addGrid(lr[0].regParam, [0.01, 0.5, 2.0]) \*

*.addGrid(lr[0].elasticNetParam, [0.0, 0.5, 1]) \*

*.addGrid(lr[0].maxIter, [1, 5]) \*

*.build())*

1. CrossValidation to optimize parameters

*cv = CrossValidator(estimator=pipeline[2], evaluator=RegressionEvaluator(), \*

*estimatorParamMaps=paramGridCV, numFolds=5)*

*# the third best model*

*model.insert(2, cv.fit(train))*

1. Prediction

*# list prediction*

*prediction = []*

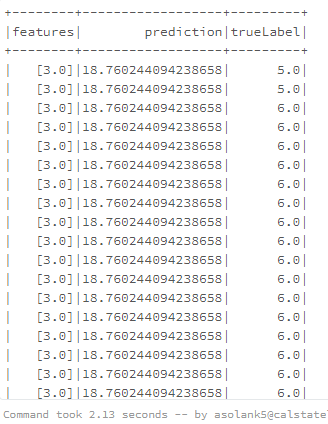
*predicted = []*

*for i in range(3):*

*prediction.insert(i, model[i].transform(test))*

*predicted.insert(i, prediction[i].select("features", "prediction", "trueLabel"))*

*predicted[i].show(30)*



1. RMSE

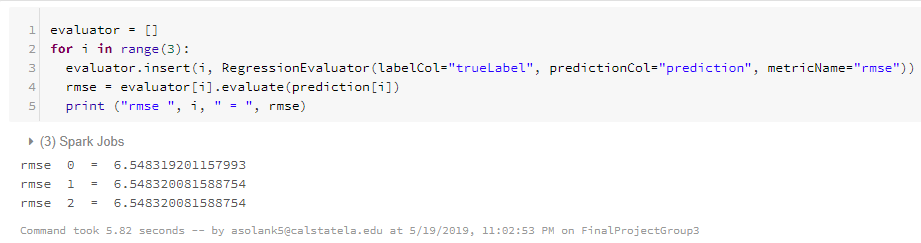
*evaluator = []*

*for i in range(3):*

*evaluator.insert(i, RegressionEvaluator(labelCol="trueLabel", predictionCol="prediction", metricName="rmse"))*

*aur = evaluator[i].evaluate(prediction[i])*

*print ("RMSE ", i, " = ", rmse)*



**Using Gradient-Boosted tree regression**

1. Importing GBTRegressor

from pyspark.ml.regression import GBTRegressor

2. Creating vector to put values in pipeline

assembler = VectorAssembler(inputCols = ["Latitude", "Longitude"], outputCol="features")

3. Gradient Boosted Tree

gbt = GBTRegressor(featuresCol="features", maxIter=10)

4. Parameters for cross validation

paramGrid = ParamGridBuilder()\

.addGrid(gbt.maxDepth, [2, 5])\

.addGrid(gbt.maxIter, [10, 100])\

.build()

5. Cross validator

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

6. Pipeline

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

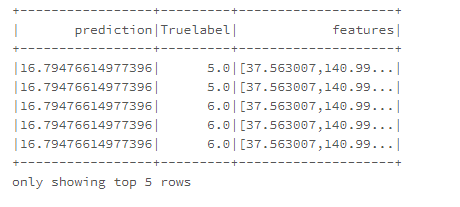
7. Model

model = pipeline.fit(train)

8. prediction

predictions = model.transform(test)

predictions.select("prediction", "Truelabel", "features").show(5)



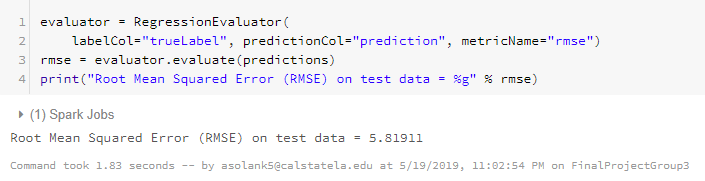
9. Evaluation

evaluator = RegressionEvaluator(

labelCol="trueLabel", predictionCol="prediction", metricName="rmse")

rmse = evaluator.evaluate(predictions)

print("Root Mean Squared Error (RMSE) on test data = %g" % rmse)



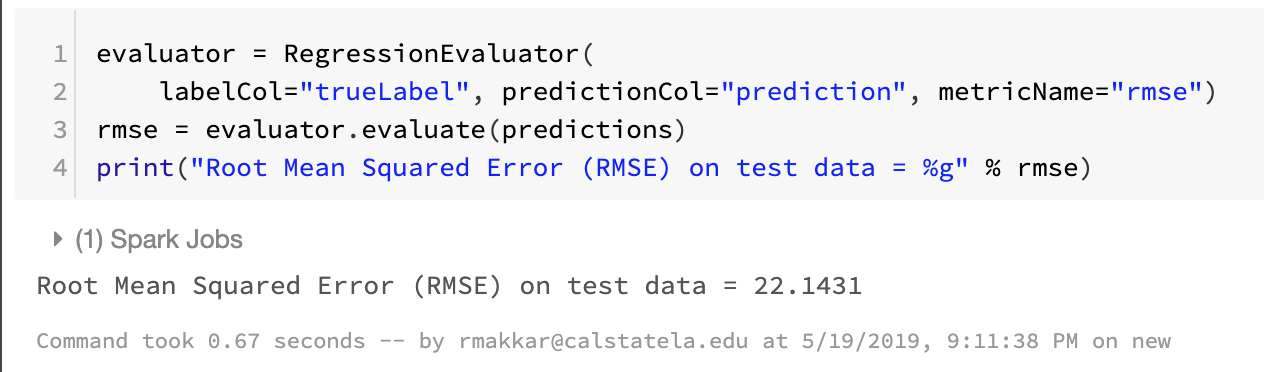
Step 3: Visualization

**In this step we are visualizing the predictions on Safecast Radiation Data using Regression**

* + - 1. USING LINEAR REGRESSION



* + - 1. USING GBT



References

* 1. URL of Data Source, <https://www.kaggle.com/safecast/safecast>
  2. URL of Github, <https://github.com/asolank5/CIS5560-Safecast-Radiation-Measurements>
  3. URL of Databricks, <https://community.cloud.databricks.com/?o=2715954072131794#notebook/1258431863633337/command/1258431863633355>
  4. URL of Azure ML, <https://gallery.cortanaintelligence.com/Experiment/Group3-Project>