**Safe Cast Radiation Measurement**

Vinayak Gaur, Ashish Solanki, Rakshit Makkar

Department of Information Systems, California State University

Los Angeles

e-mail : [vgaur@calstatela.edu](mailto:vgaur@calstatela.edu), [asolank5@calstatela.edu](mailto:asolank5@calstatela.edu), [rmakkar@calstatela.edu](mailto:rmakkar@calstatela.edu).

**Abstract:** This project will illustrate the data science models created with the sole purpose of predicting radiation from radio-active materials using Azure ML studio and Databrick. In this project, we have created, trained, tested and evaluate models with the values and features which have the highest impact on the intensity of the radiations. The data associated has been taken from the open source platform “Kaggle” which is shared by Safe Cast, a non-profit organization use to measure radiations from the radio active isotopes. The models had built using regression as the values are in the numerical format.

1. **Introduction**

Safecast is a volunteer-driven non-profit organization whose goal is to create useful, accessible, and granular environmental data for public information and research.

Radiation measurement is the data provided by this organization of one of the radioactive isotopes caesium-137 which is the most prevalent radioactive isotope and it emits radiation around nuclear weapons testing and areas such as Chernobyl and Fukushima. Continuous exposure to this isotope can have catastrophic results on the human body including burns, acute radiation, sickness, and even death.

Initially, the radiations have been measured in “sievert” unit using the standard devices. But later the measurement has converted into unit CPM (counts per minute), which is the standard unit for measurement using formulae:

334 CPM = 1  Sv/h

The vulnerability of this material can be expressed from the considered as exposure to 100 mSv a year to a human body can increase the chances of cancer by 100%.

The primary objective is to build a model from the data of the previous two instances gives a better understanding of the radiations with respect to time and the location of the instance. This will help us to predict the exposure of the radiations to the human body as time passing if any explosion during nuclear testing happens again in the future.

1. **Dataset Description**

The Data of Safe Cast radiation has been downloaded from the open source website Kaggle. The original size of the data set is 10 GB. The data is of the year 2015 and 2017 and in the CSV format.

The dataset comprises of 13 columns out of which we have used few of them, which are briefly described below:

Captured Time: this column defines the exact time at which the radiation has been captured by the devices.

Latitude: the latitude of that place where the isotope is placed after the explosion.

Longitude: The longitude of that place where the isotope is present after explosion.

Value: comprises of the value of the radiation from the isotope cesium-137

Unit: this column defines the unit in which the radiations have been captured. We have used the data in the CPM unit.

Location Name: this defines the name of the location where the isotope is present.

Height: this column defines the height of the isotope from the nearest sea level.

1. **Work Flow**

In this project we have started with the downloading the data from the open source platform Kaggle into our local machine in the CSV format. Afterward, the data has been uploaded in the Oracle cluster with the cluster size of 10GB using Spark query language. The further cleaning of data has been performed on the same Oracle cluster using a spark.

The data for year 2017 only with the value in CPM unit has been extracted into the local machine, which is more than 100 MB and used for further data modeling in Databricks. Another sample of radiation values between 10 to 40 CPM has been extracted for data modeling in Azure ML studio. This data sample is 33 MB.

Beside to this, the Big data of size 10 GB in cluster has also processed using Oracle pyspark platform. At the end the RMSE and coefficient of determination has been compared from every platform to find the best model.

1. **Platform Specification**

To begin our project, we have used Microsoft Azure ML Studio and Databricks as they are popular to built machine learning model on massive datasets.

**Microsoft Azure ML Studio:** We have used the free version of the Machine Learning Studio for which Azure subscription is not needed. Below is the Microsoft Azure ML Studio system details: -

* Max number of modules per experiment – 100
* Max storage space – 10 GB
* Execution/performance – Single Node

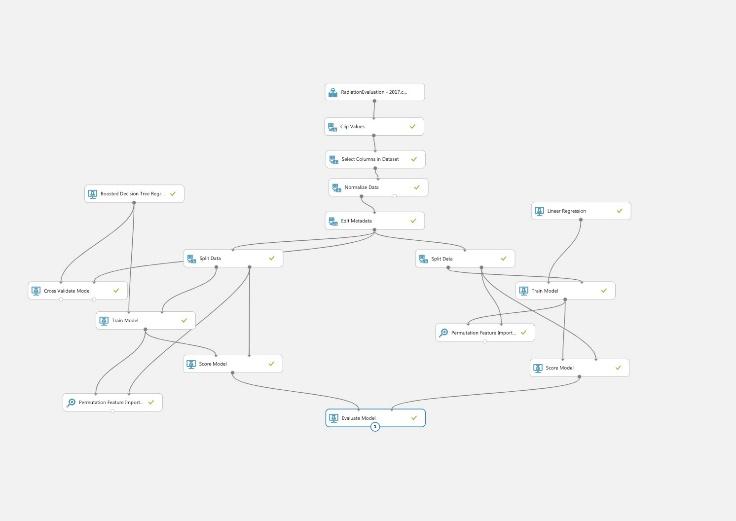
**Databricks:** We have used the Databricks community edition, a free version of the cloud-based Spark Platform. Below are the cluster details: -

* Memory – 2 GB Memory, 0.88 Cores, 1 DBU
* DataBricks Runtime Version – 4.0 (includes Apache Spark 2.3.0, Scala 2.11) Python Version – 2

1. **Azure ML Studio**

Microsoft Azure Machine Learning, or just Azure ML, is a cloud-based platform for designing, developing, testing and deploying predictive models. Those models can use many well-known machine learning algorithms, such as decision forests or neural networks, to ﬁnd useful patterns in your data. The heart of Azure ML toolkit is ML Studio, a web-based graphical environment where we uploaded and validated our data models, which Azure ML refers to as experiments. This is an apt name because an experiment can consist of multiple models and much additional logic, or other helpful calculations and data preparation tasks.

For building Machine learning model, we chose Boosted Decision Tree Regression and Linear Regression algorithms, as we have the numerical continuous data and do not want our model to be over fitted.



* 1. **Data Transformation Modules**

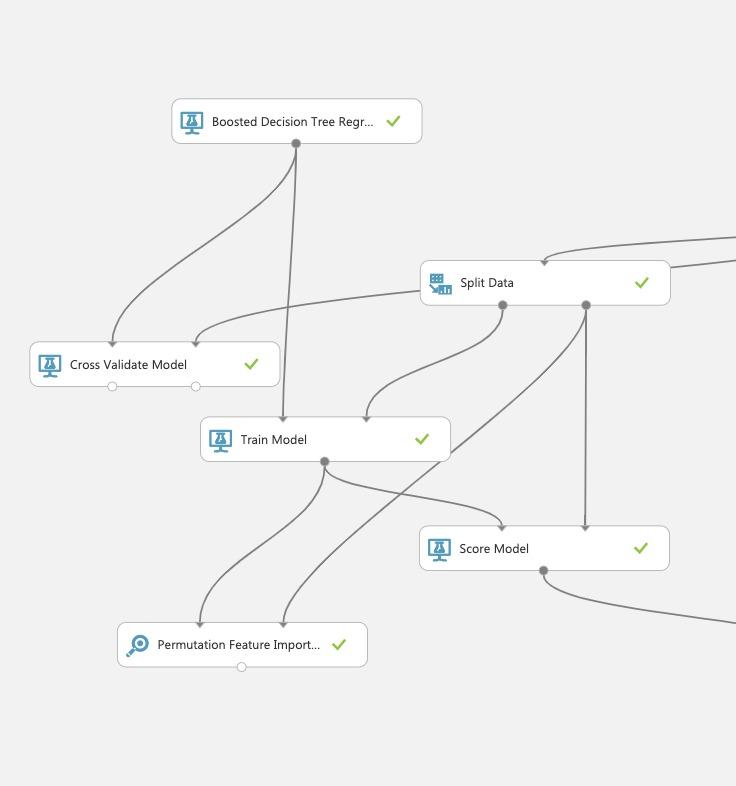
We have used several Data Transformation modules with properties to convert data into a dataset suitable to be used by the machine learning algorithms. Under this category, we have used different modules which are as follow:

* Clip Values: This module is used to detect the outliners and replace them with the values.
* Select Column in Dataset – This module was used to select a subset of columns from the original dataset.
* Normalize Data: This module was used to rescale our numeric data.
* Edit Meta Data: This module was used to modify the definition of a column, typically to meet requirements for a downstream module.
* Split Data: This module was used to split our dataset into two groups for training and testing purpose. We have used 70% of data for training and 30% for testing our model.



* 1. **Boosted Decision Tree Regression Model**

We have used boosted decision tree to avoid over-fitting and underfitting of the model as this helps us by fitting the residual of the trees that preceded it. Thus, boosting in a decision tree ensemble tends to improve accuracy with some small risk of less coverage.



We have used 20 number of leaves per tree and 10 leaves per node with a learning rate of 0.4 to obtain the best result.

Further, we have used Cross Validation to access both the variability of a dataset and the reliability of any model trained using “Value” column. After that, we have used Train Model over the value column and used Permutation Feature Importance to compute a set of feature importance scores for our dataset.

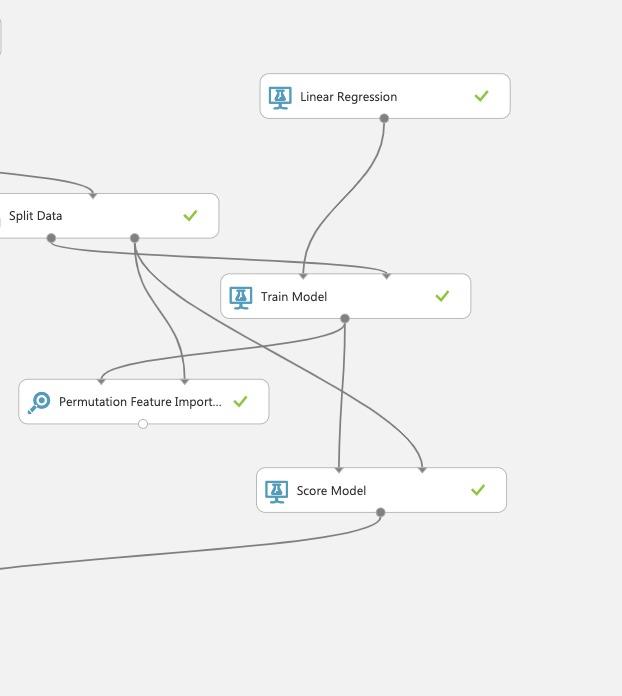
A screenshot of a cell phone

Description automatically generated

Then we selected the features and label for prediction. Feature is a parameter with which the model is trained and label is the value that must be trained. In our prediction, **features** such as **Latitude, Longitude, Height** werepassed into the Split Model and **Label** such as **Value** was passed into the trained model. The result of the trained data and original data where compared using the score model. The score model output depicts the accuracy of prediction in comparison with the features used for training. As per the above visual we can see that Score Label was very close to the Values.

* 1. **Linear Regression Model**

We have used the Linear Regression Model as an attempt to build a relationship between the value and the feature made from the Latitude, Longitude and Height column. Moreover, this model has helped us in fitting the values into the range and measuring the error.



The linear regression has been applied with the Ordinary Least Squares of Solution Method and with Random Seed of 1234.

The features created with three columns of Latitude, Longitude, and Height are passed through Split Model and Value column has trained.

The Score Label column in this model is similar to the previous as the data is in a particular range.

A screenshot of a cell phone

Description automatically generated

* 1. **Model Evaluation**

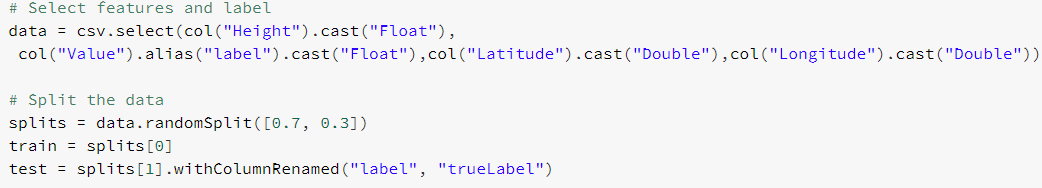
After running the steps discussed above we have evaluated our models to find the best fit. The Linear Regression model can provide us the more accurate and reliable value as compared to the Boosted Decision Forest regression model.

|  |  |  |
| --- | --- | --- |
|  | **Boosted Decision Forest Regression** | **Linear Regression** |
| **Root Mean Square Error** | 18.148 | 18.140 |
| **Relative Squared Error** | 0.4377 | 0.4392 |
| **Co-efficient of Determination** | 56.22% | 56.02% |

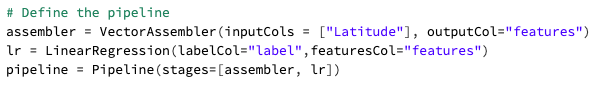
1. **Data Bricks ML Model**

We have used the free cloud-based version of Databricks containing the Spark platform. We have used IPython notebook environment to build our models. We have selected the Linear Regression Model to rebuild in Data Bricks. We have followed a certain number of steps to build our model in the Databricks. The steps are as follows:

* We first created the cluster and uploaded our CSV file into the database of the Databricks.
* After uploading the file, we have to read the read the CSV file using Spark SQL command.
* Data Manipulation was performed by selecting columns such as **Height, Latitude, Longitude, and Value** suitable for our model.



* Then, we have spilled our data into training (70%) and testing (30%) by using data.randomSplit() function as shown in the above picture.
* After splitting, we have defined our pipeline which created a vector of features using VectorAssembler function and trained our model.
* We used **linear regression** to train out model.

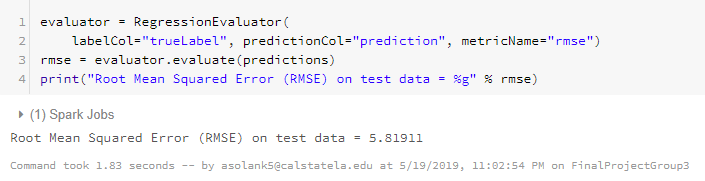


* Later, we have performed testing by inserting test data to our model.
* The table formed depicts the value of the Features, predicted values and the true label. The values are quite close to each other but farther as compared to the Azure ML because in data bricks we have used larger data set.



At the end, we have evaluated our model by retrieving the RMSE (Root Mean Square Error) value from the featured and the predicted value.

We trained the same data using **Gradient boosted tree regression** to see the comparison.



1. **Comparing Models in Azure ML and Data Bricks**

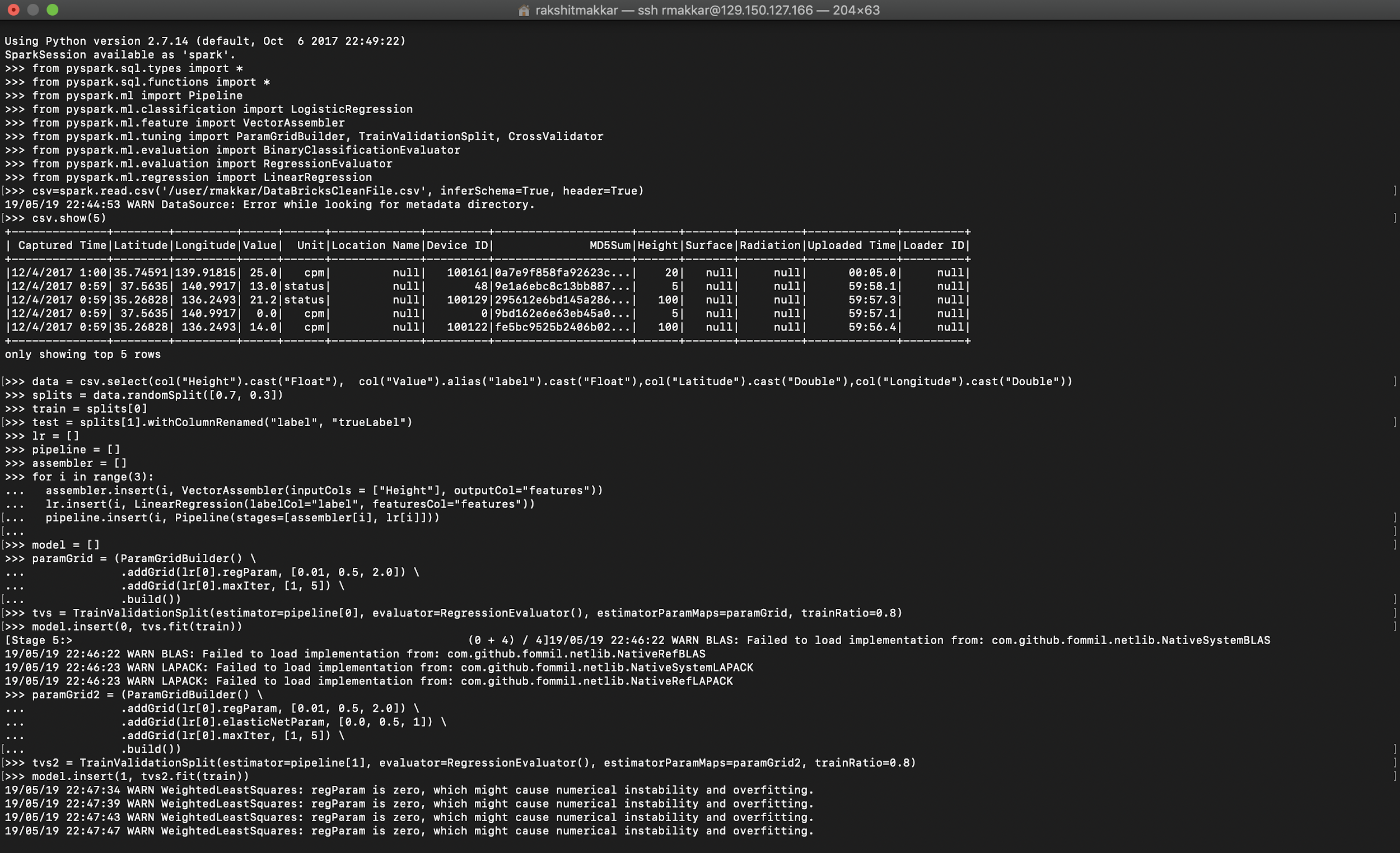
Below is the overview table of the measured metrics to evaluate our models both in Azure ML and Databricks.

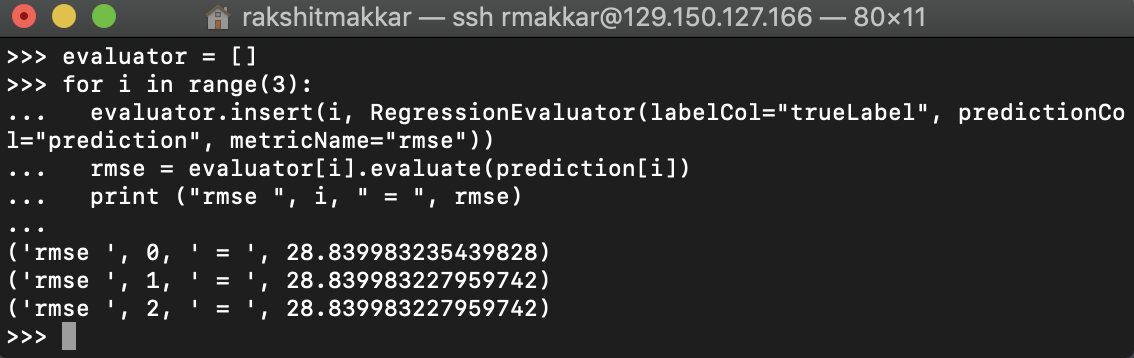
|  |  |  |
| --- | --- | --- |
|  | **Boosted Decision Forest Regression** | **Linear Regression** |
| **Root Mean Square Error** | 18.148 | 18.140 |
| **Relative Squared Error** | 0.4377 | 0.4392 |
| **Co-efficient of Determination** | 56.22% | 56.02% |

1. **Oracle**

We run the same code in oracle which we used in databricks to train the model. We used oracle to train the model using big data and predict the values for the same. Both linear regression and gradient boosted tree regression have been used to train the model.

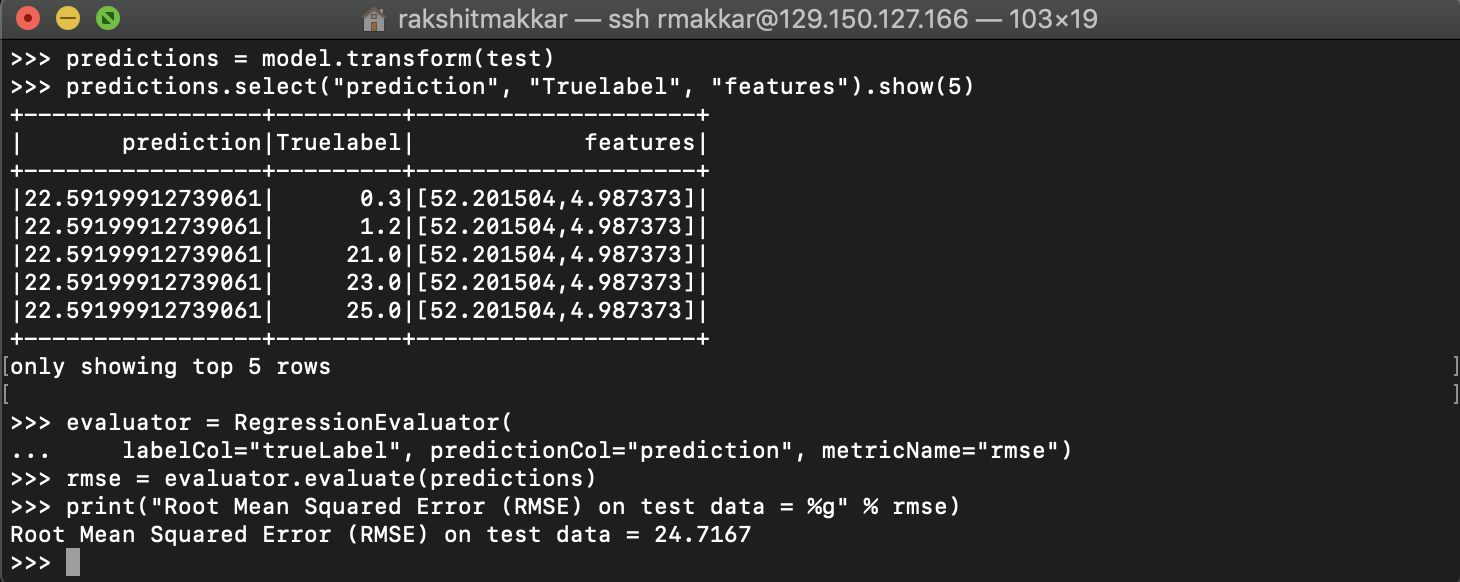
**8.1 Using Linear Regression**-

****

****

**8.2 Using Gradient Boosted Tree Regression-**

****

****

1. **Summary**

The Azure ML platform has more advanced features and supports more algorithms when compared with Spark ML. For Continuous values, Regression model can only give accurate results. Amongst the two-regression model we selected Linear Regression accuracy is more efficient when compared with Boosted Decision Tree Regression. The business advantage of this prediction is that it can be used to build an application that helps end user to decide the loan amount depending on the predicted installments that lies within their range.

1. **Dataset Link**

Safecast. “Safecast Radiation Measurements.” *Kaggle*, 7 Dec. 2017, www.kaggle.com/safecast/safecast.

1. **Git Hub Link**

<https://github.com/asolank5/CIS5560-Safecast-Radiation-Measurements.git>

1. **Azure ML Link**

<https://gallery.azure.ai/Experiment/Group3-Project>

1. **References**
2. Xiaoharper. “Machine Learning Studio: Algorithm and Module Help - Azure Machine Learning Studio.” *Machine Learning Studio: Algorithm and Module Help - Azure Machine Learning Studio | Microsoft Docs*, docs.microsoft.com/en-us/azure/machine-learning/studio-module-reference/machine-learning-studio-algorithm-and-module-help.
3. “Saving, Loading, and Deploying Models.” *Saving, Loading, and Deploying Models - Databricks Documentation*, docs.databricks.com/applications/mlflow/models.html.