## HU Extension Assignment 07 E63 Big Data Analytics

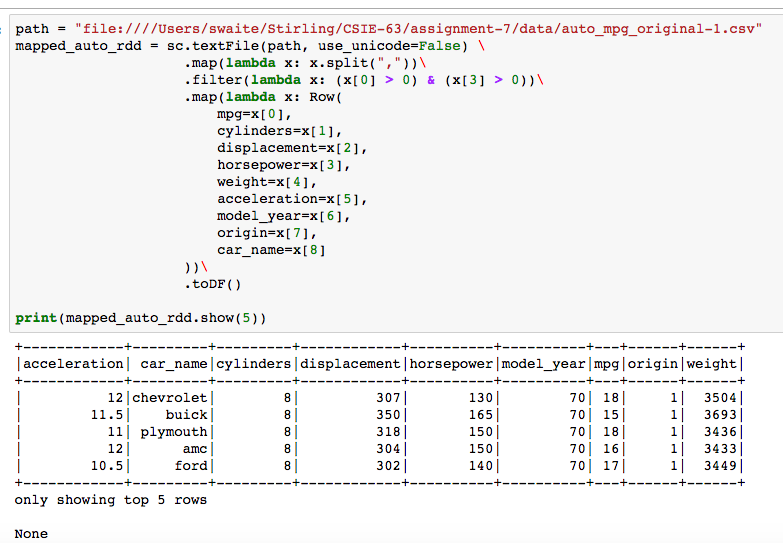
### Handed out: 10/13/2017 Due by 4:00 PM EST on Saturday, 10/21/2017

Attached file Regression Analysis with Spark MLlib.py contains all the code discussed in lecture. Be free to adopt that code to your needs**.** This code will work as is on your Cloudera VM with Spark 1.6. On your own VM with Spark 2.2, you might have to modify some functions with newer API calls.You are also welcome to switch from this mostly RDD based code to Data Frame based code offered in Spark 2.2. That is up to you.

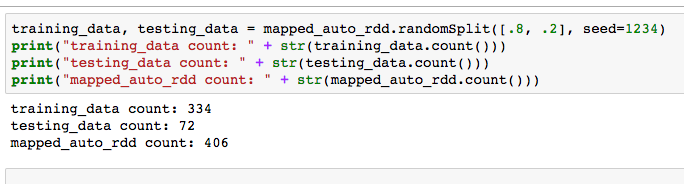
**Problem 1.** Attached file auto\_mpg\_original.csv contains a set of data on automobile characteristics and fuel consumption. File auto\_mpg\_description.csv contains the description of the data. Import data into Spark. Randomly select 10-20% of you data for testing and use remaining data for training. Find all null values in all numerical columns. Replace nulls, if any, with average values for respective columns using Spark Data Frame API.

(25%)

* **Attached file auto\_mpg\_original.csv contains a set of data on automobile characteristics and fuel consumption.**
* **File auto\_mpg\_description.csv contains the description of the data.**
* **Find all null values in all numerical columns. Replace nulls, if any, with average values for respective columns using Spark Data Frame API.**
* **Import data into Spark.**

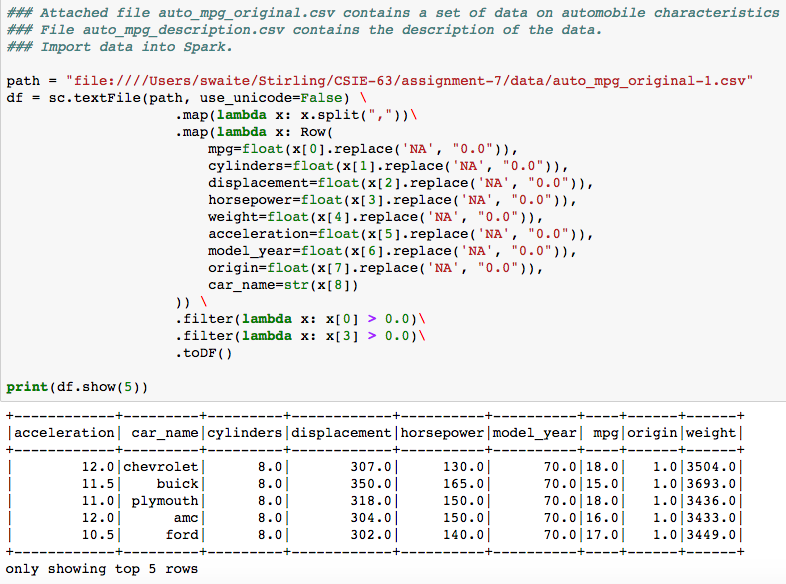
****

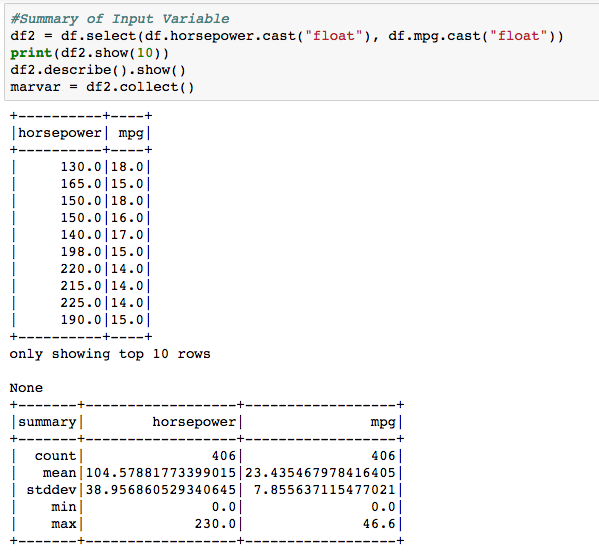
* **Randomly select 10-20% of you data for testing and use remaining data for training.**

****

**Problem 2.** Look initially at two variables in the data set from the previous problem: the horsepower and the mpg (miles per gallon). Treat mpg as a feature and horsepower as the target variable (label). Use MLlib linear regression to identify the model for the relationship. Use the test data to illustrate accuracy of the linear regression model and its ability to predict the relationship. Calculate two standard measures of model accuracy. Create a diagram using any technique of convenience to presents the model (straight line), and the original test data. Please label your axes and use different colors for original data and predicted data.

(25%)





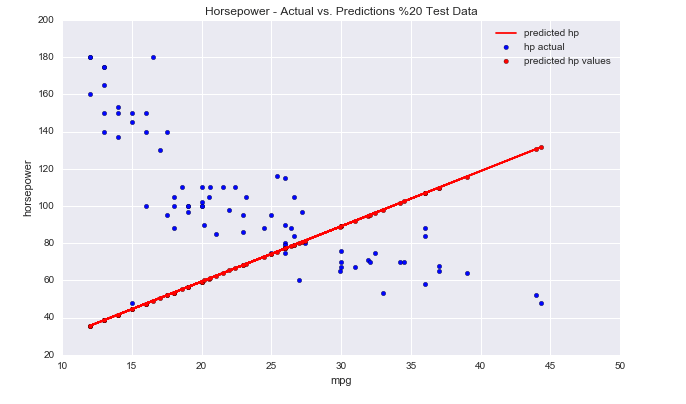
|  |
| --- |
| # Training the Model  linear\_model = LinearRegressionWithSGD.train(training\_data, iterations=1000, step=.0001)  print "\n"  print "--------------"  print "Linear model parameters"  print "--------------"  print(linear\_model)  print "\n"  print "--------------"  print "Predictions "  print "--------------"  true\_vs\_predicted = testing\_data.map(lambda p: (p.label, linear\_model.predict(p.features)))  print("Linear Model preditions: " + str(true\_vs\_predicted.take(5)))  print "\n"  print "--------------"  print "Model Metrics "  print "--------------"  #Gather Metrics  mse = true\_vs\_predicted.map(lambda (t,p): squared\_error(t,p)).mean()  mae = true\_vs\_predicted.map(lambda (t,p): abs\_error(t,p)).mean()  rmsle = np.sqrt(true\_vs\_predicted.map(lambda (t,p): squared\_log\_error(t,p)).mean())  print("Linear Model - Mean Squared Error: %2.4f" % mse)  print("Linear Model - Mean Absolute Error: %2.4f" % mae)  print("Linear Model - Root Mean Squared Log Error: %2.4f" % rmsle) |

**OUTPUT**

|  |
| --- |
| --------------  Linear model parameters  --------------  (weights=[2.96970534459], intercept=0.0)  --------------  Predictions  --------------  Linear Model predictions: [(115.0, 77.212338959343398), (85.0, 62.363812236392739), (95.0, 74.24263361475326), (48.0, 44.545580168851956), (100.0, 53.454696202622351)]  --------------  Model Metrics  --------------  Linear Model - Mean Squared Error: 4130.7654  Linear Model - Mean Absolute Error: 50.8381  Linear Model - Root Mean Squared Log Error: 0.7434 |

**Horsepower - Actual vs. Predictions 20% Test Data**

|  |
| --- |
| tvp = testing\_data.map(lambda p: (float(p.label), float(linear\_model.predict(p.features)), float(p.features[0]))).toDF().toPandas()  tvp.columns = ['horsepower', 'predicted\_horsepower', 'mpg']  plt.figure(1, figsize = (10, 6))  plt.scatter(tvp["mpg"], tvp["horsepower"], c='b', label="hp actual")  plt.plot(tvp["mpg"], tvp["predicted\_horsepower"], c='r', label="predicted hp")  plt.scatter(tvp["mpg"], tvp["predicted\_horsepower"], c='r', label="predicted hp values")  plt.xlabel("mpg")  plt.ylabel("horsepower")  plt.title("Horsepower - Actual vs. Predictions %20 Test Data")  plt.legend()  plt.show() |

****

**Horsepower - Actual vs. Predictions 20% Test Data**

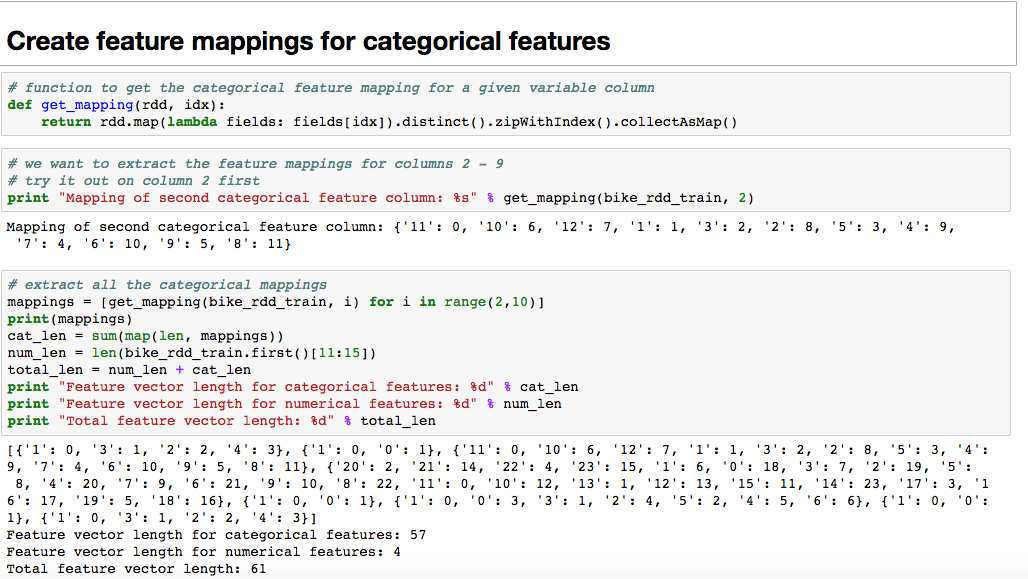
|  |
| --- |
| plt.figure(2, figsize = (12, 10))  tvp2 = training\_data.map(lambda p: (float(p.label), float(linear\_model.predict(p.features)), float(p.features[0]))).toDF().toPandas()  tvp2.columns = ['horsepower', 'predicted\_horsepower', 'mpg']  plt.scatter(tvp["mpg"], tvp["horsepower"], c='b', label="hp .2 sample data")  plt.scatter(tvp2["mpg"], tvp2["horsepower"], c='g', label="hp .8 training data")  plt.plot(tvp["mpg"], tvp["predicted\_horsepower"], c='r', label="predicted hp")  plt.scatter(tvp["mpg"], tvp["predicted\_horsepower"], c='m', label="predicted hp values")  plt.xlabel("mpg")  plt.ylabel("horsepower")  plt.title("Horsepower - Actual vs. Predictions 20% Test Data")  plt.legend()  plt.show() |

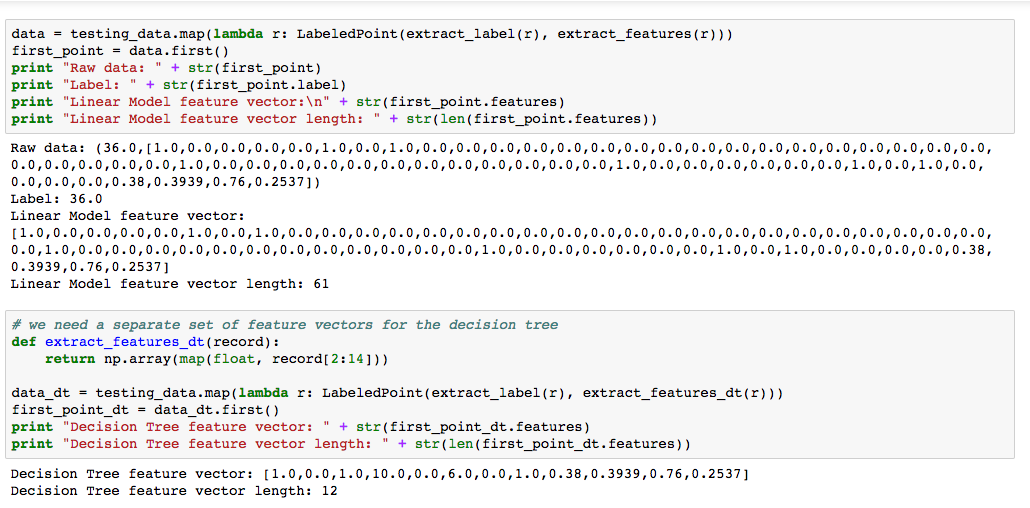
****

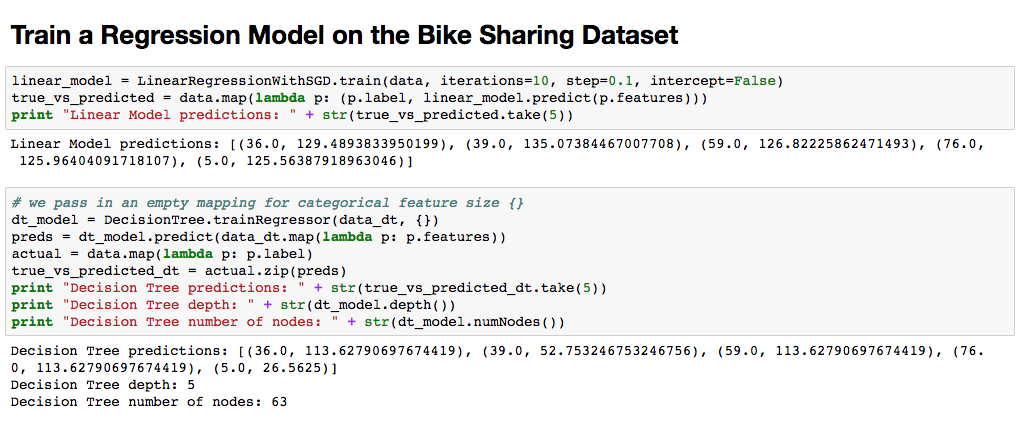
**Problem 3**. Consider attached file Bike-Sharing-Dataset.zip. This is the bike set discussed in class. Do not use all columns of the data set. Retain the following variables: season,yr,mnth,hr,holiday,weekday,workingday,weathersit,temp,atemp,hum,windspeed,cnt. Discard others. Regard cnt as the target variable and all other variables as features. Please note that some of those are categorical variables. Identify categorical variables and use 1-of-k binary encoding for those variables. If there are any null values in numerical columns, replace those with average values for those columns using Spark DataFrame API. Train your model using LinearRegressionSGD method. Use test data (15% of all) to assess the quality of prediction for cnt variable. Calculate at least two performance metrics of your model.

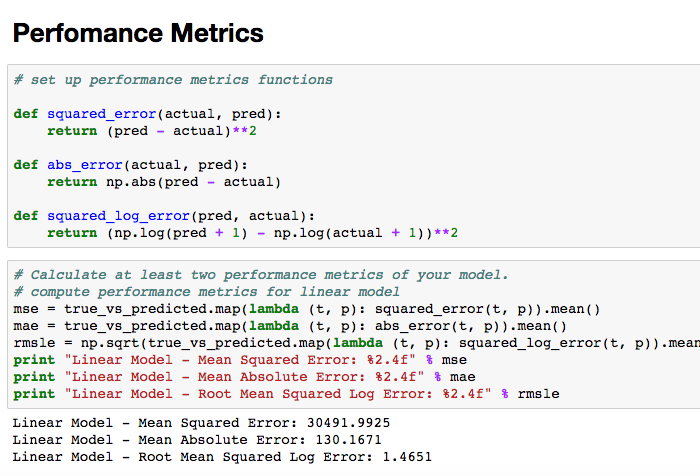
(25%)

****

****

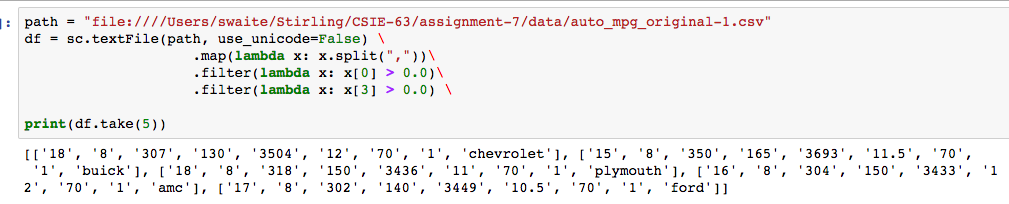
****

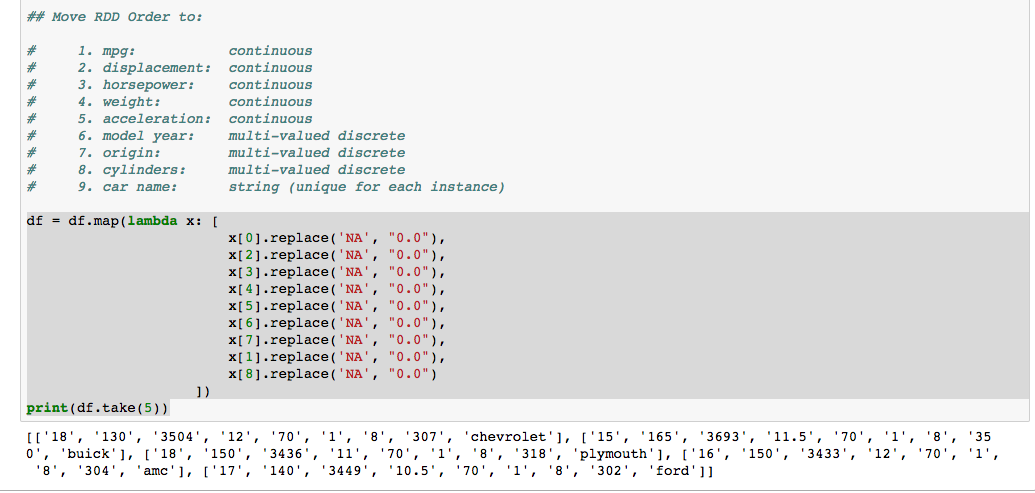
****

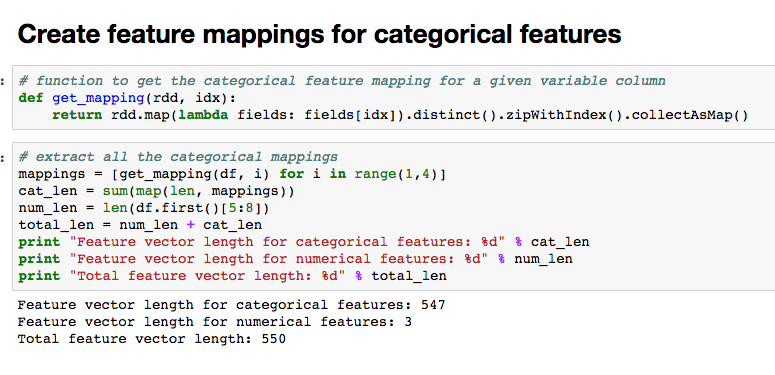
****

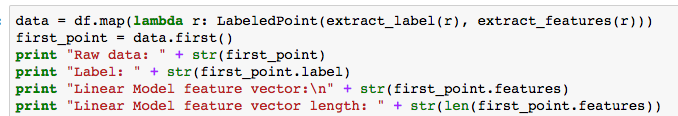
**Problem 4**. Use a Decision Tree model to predict mpg values in auto\_mpg\_original.txt data. Assess accuracy of your prediction using at least two performance metrics.

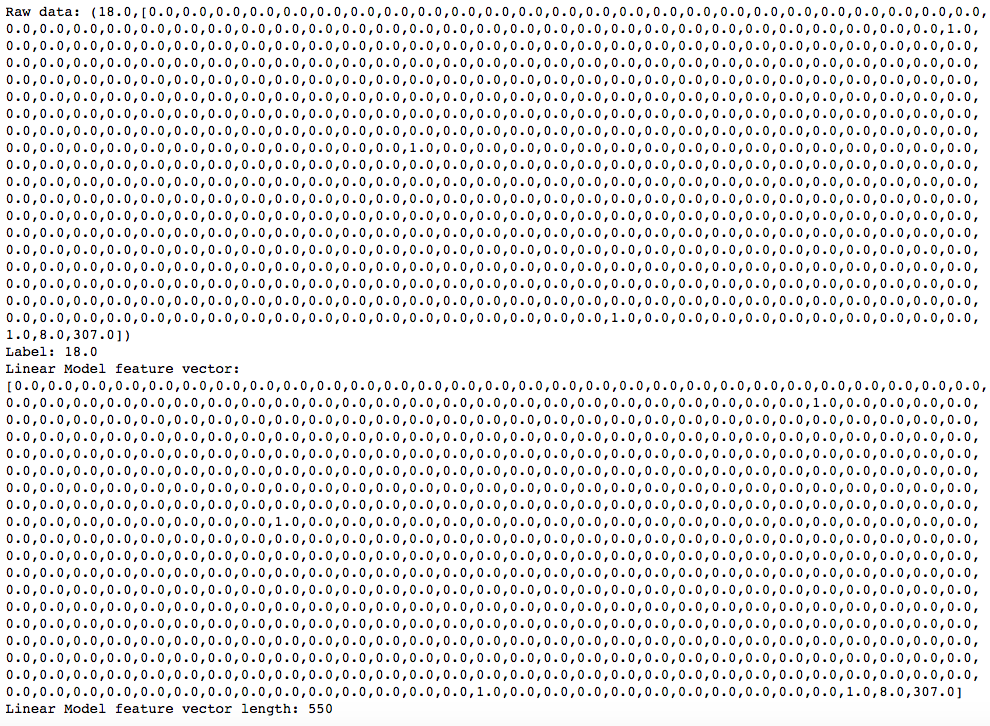
(25%)

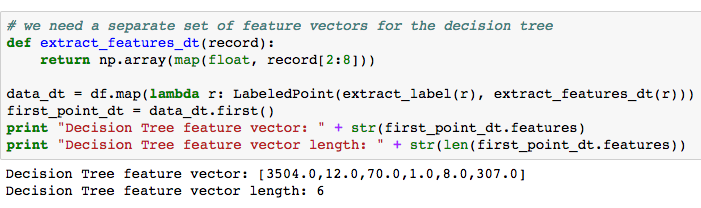


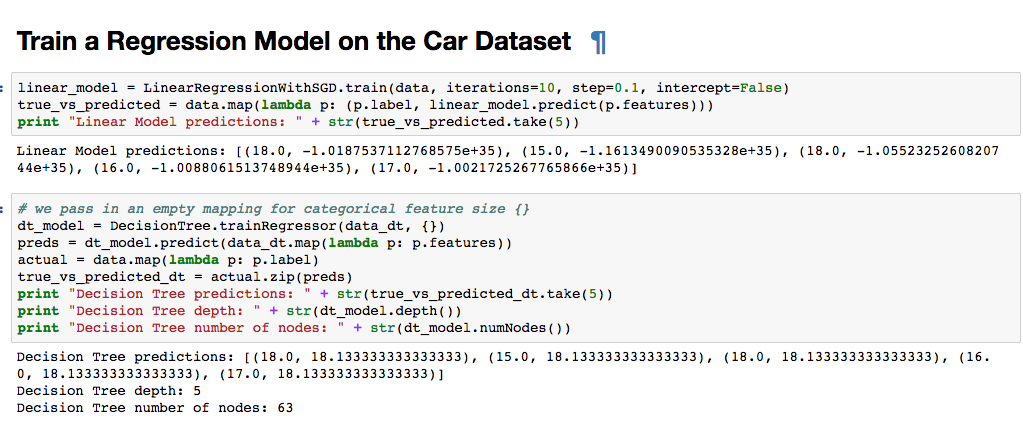


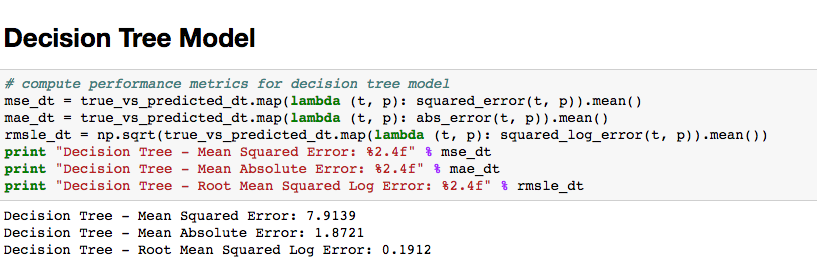












**Assessment:**

It appears given these results for MPG the MSE is the best result at 7.9

In your solution, please leave the text of every problem as presented here. Add your solution below the problem statement. It is important for us and we will take points if you ignore this request. Please make sure that you provide numeric or textual results of your calculations. If there are no results, we will treat the problem as not addressed. Just providing some code without results will give you 0 point.

You are welcome to implement your solution in any language of your choice.

You are welcome to follow any other instructions and use any other programming or scripting language to accomplish the above goals.

Please, describe every step of your work and present all intermediate and final results in a Word document. Please, copy past text version of all essential command and snippets of results into the Word document. We cannot retype text that is in JPG images. Please, always submit a separate copy of the original, working scripts and/or class files you used as separate files. Sometimes we need to run your code and retyping is too costly. Please include in your MS Word document only relevant portions of the console output or output files. Sometime either console output or the result file is too long and including it into the MS Word document makes that document too hard to read. PLEASE DO NOT EMBED files into your MS Word document. Please, submit to the class drop box. For issues and comments visit the class Discussion Board. You are not obliged to use Java or Eclipse. You are welcome to use any language and any IDE of your choice.