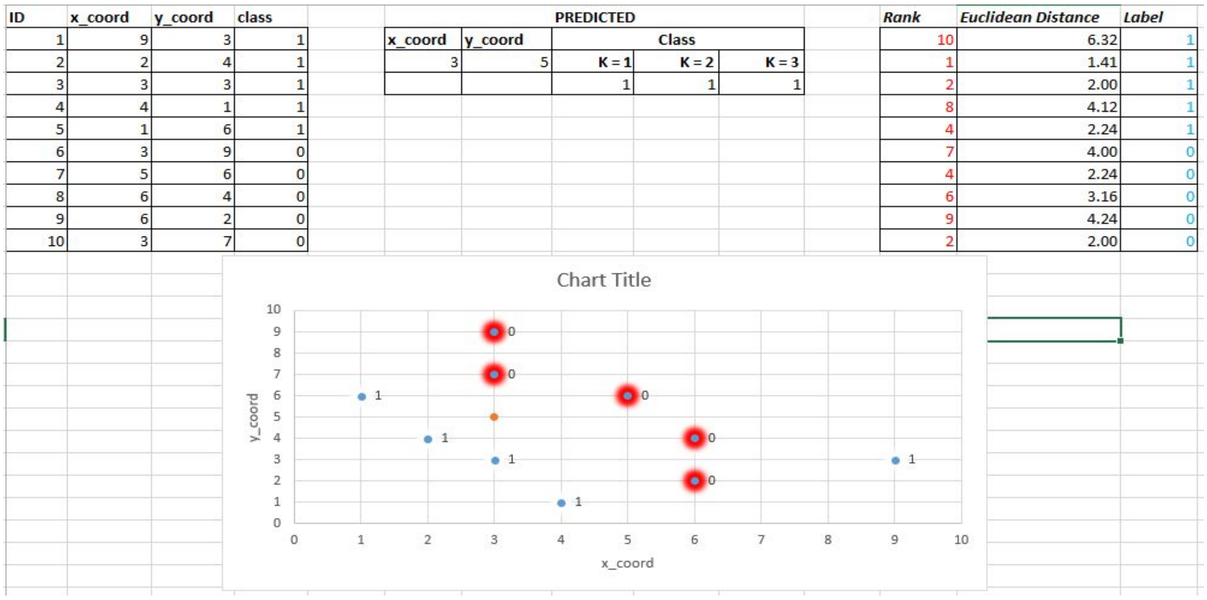
HOMEWORK 10

Kenigbolo Meya Stephen

1. Take the following data and and simulate K-NN algorithm to predict the class probabilities of points (3, 5) and (4, 6). Report the probabilities with K=1, K=2 and K=3.



ID	x_coord	y_coord	class
1	9	3	1
2	2	4	1
3	3	3	1
4	4	1	1
5	1	6	1
6	3	9	0
7	5	6	0
8	6	4	0
9	6	2	0
10	3	7	0

PREDICTED				
x_coord	y_coord	Class		
3	5	K = 1	K = 2	K = 3
		1	1	1

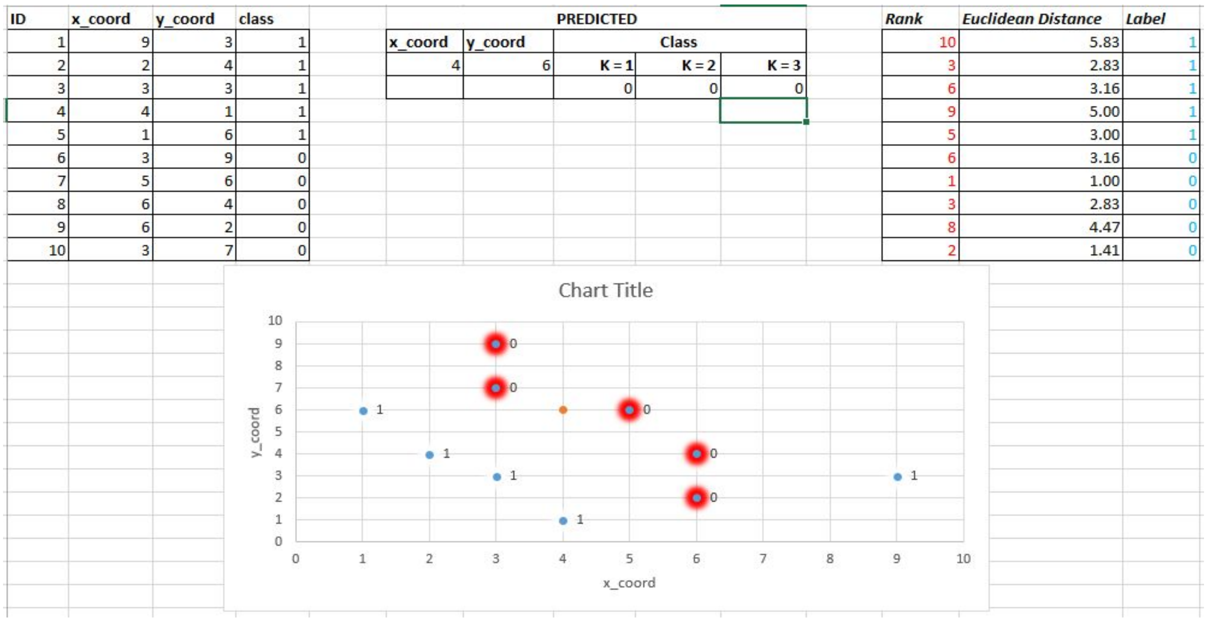
Rank	Euclidean Distance	Label
10	6.32	1
1	1.41	1
2	2.00	1
8	4.12	1
4	2.24	1
7	4.00	0
4	2.24	0
6	3.16	0
9	4.24	0
2	2.00	0

As can be seen in the image above, for coordinates (3, 5) the following classes are obtainable:

K = 1 Class 1

K = 2 Class 1

K = 3 Class 1



ID	x_coord	y_coord	class
1	9	3	1
2	2	4	1
3	3	3	1
4	4	1	1
5	1	6	1
6	3	9	0
7	5	6	0
8	6	4	0
9	6	2	0
10	3	7	0

PREDICTED				
x_coord	y_coord	Class		
4	6	K = 1	K = 2	K = 3
		0	0	0

Rank	Euclidean Distance	Label
10	5.83	1
3	2.83	1
6	3.16	1
9	5.00	1
5	3.00	1
6	3.16	0
1	1.00	0
3	2.83	0
8	4.47	0
2	1.41	0

As can be seen in the image above, for coordinates (4, 6) the following classes are obtainable:

K = 1 Class 0

K = 2 Class 0

K = 3 Class 0

2. In this task we use diabetes dataset to predict diabetes.

- Split the data randomly on 80% of training and 20% for testing.
- Fit the logistic regression on the training set to predict V9.
- Interpret the model. How the Plasma glucose concentration impacts the odds ratio of having diabetes. What about diabetes pedigree function? Which features do not affect (significantly) the risk of having diabetes?
- Now compute Accuracy, Precision, Recall and F1 score on the test set.

For this task I made use of Weka.

- Split the data randomly on 80% of training and 20% for testing.

To split the data into the test set and training set I simple did the following:

- training set:
  - Loaded the full diabetes dataset
  - selected the “RemovePercentage” filter in the preprocess panel
  - set the percentage for the split to 80%
  - applied the filter
  - saved the generated data as a new file
- test set:
  - Reverted the changes to the loaded dataset
  - set the “invertSelection” property to true in the “RemovePercentage” filter
  - applied the filter
  - saved the generated data as new file

- Fit the logistic regression on the training set to predict V9.

I began by uploading the training set and selected “use training set” from the test options in the classify tab. I then selected Logistic regression in the classifier section (After changing V9 from Numeric to Nominal). The output is as follows

```
Test mode:evaluate on training data
=== Classifier model (full training set) ===
Logistic Regression with ridge parameter of 1.0E-8
Coefficients...
              Class
Variable          0
====================
V1          -0.1217
V2          -0.0329
V3           0.0106
V4           0.0026
V5           0.0009
V6          -0.1026
V7          -1.0648
V8          -0.0096
Intercept     8.5879
Odds Ratios...
```

```
           Class
Variable        0
====================
V1          0.8854
V2          0.9676
V3          1.0106
V4          1.0026
V5          1.0009
V6          0.9024
V7          0.3448
V8          0.9905
Time taken to build model: 0.02 seconds
=== Evaluation on training set ===
=== Summary ===
Correctly Classified Instances       483             78.6645 %
Incorrectly Classified Instances      131             21.3355 %
Kappa statistic                       0.5046
Mean absolute error                     0.3046
Root mean squared error                   0.3894
Relative absolute error               67.2014 %
Root relative squared error             81.8171 %
Total Number of Instances               614
=== Detailed Accuracy By Class ===

           TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class

             0.893     0.413      0.803     0.893     0.845      0.837    0

             0.587     0.107      0.744     0.587     0.656      0.837    1

Weighted Avg.    0.787     0.307      0.782     0.787     0.78       0.837

=== Confusion Matrix ===

  a   b   <-- classified as

 358  43 |   a = 0

  88 125 |   b = 1
```

- Now compute Accuracy, Precision, Recall and F1 score on the test set.

To accomplish this I simply selected the option “supplied test set” in the test set options and added the test data from the first step. The output are as follows

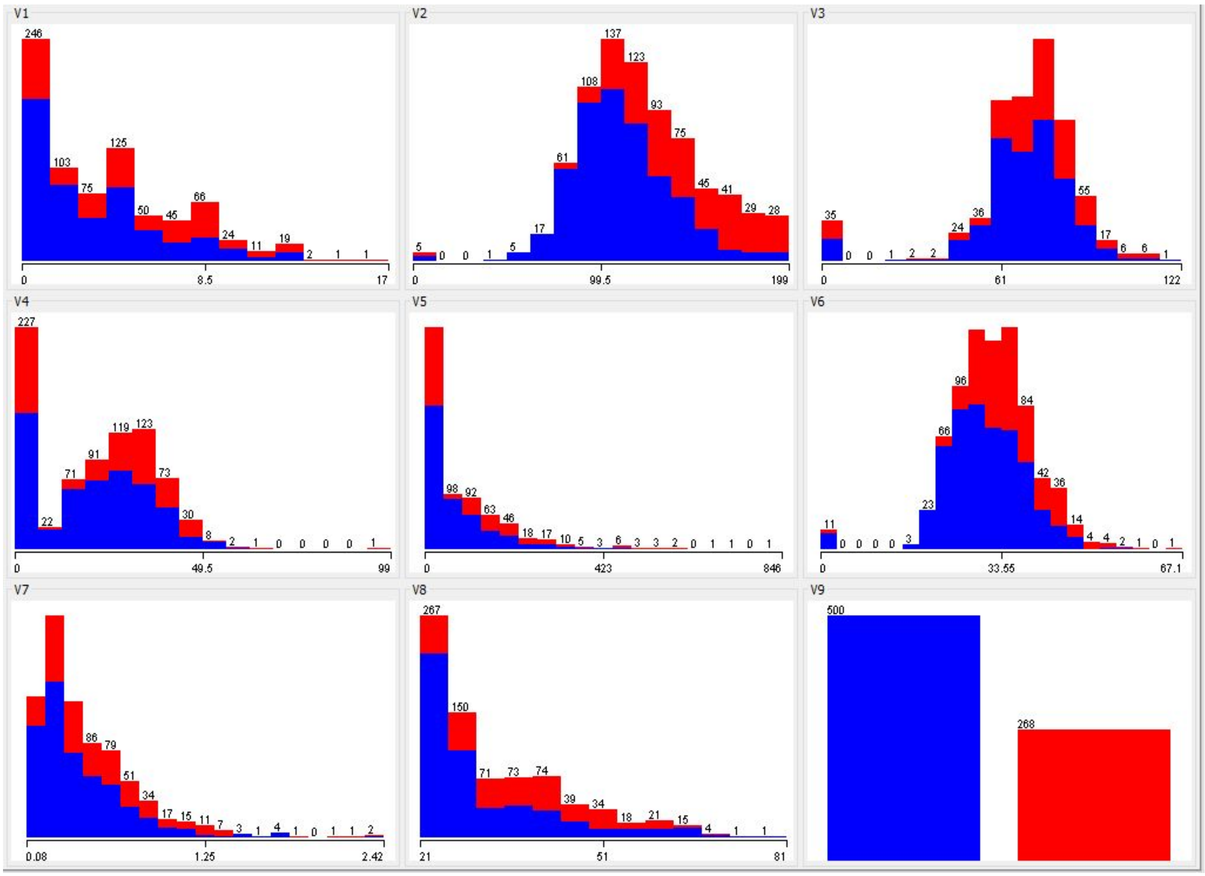
```
Test mode:user supplied test set: size unknown (reading incrementally)
=== Classifier model (full training set) ===
Logistic Regression with ridge parameter of 1.0E-8
Coefficients...
           Class
Variable        0
====================
V1         -0.1217
V2         -0.0329
V3          0.0106
```

```
V4           0.0026
V5           0.0009
V6          -0.1026
V7          -1.0648
V8          -0.0096
Intercept     8.5879

Odds Ratios...
              Class
Variable          0
====================
V1           0.8854
V2           0.9676
V3           1.0106
V4           1.0026
V5           1.0009
V6           0.9024
V7           0.3448
V8           0.9905
Time taken to build model: 0.02 seconds
=== Evaluation on test set ===
=== Summary ===
Correctly Classified Instances        119               77.2727 %
Incorrectly Classified Instances       35               22.7273 %
Kappa statistic                      0.4731
Mean absolute error                     0.3176
Root mean squared error                   0.3973
Relative absolute error                69.5879 %
Root relative squared error             82.8984 %
Total Number of Instances               154


=== Detailed Accuracy By Class ===

              TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class

               0.899     0.455      0.781     0.899     0.836      0.848     0

               0.545     0.101      0.75      0.545     0.632      0.848     1

Weighted Avg.    0.773     0.328      0.77      0.773     0.763      0.848

=== Confusion Matrix ===

  a  b   <-- classified as

 89 10 |  a = 0

 25 30 |  b = 1
```

- Interpret the model. How the Plasma glucose concentration impacts the odds ratio of having diabetes. What about diabetes pedigree function? Which features do not affect (significantly) the risk of having diabetes?

From the results I can conclude that a high plasma glucose concentration is a strong predictor for future diabetes. Also with a bit of manipulation of removing and adding various attributes I can conclude that the risk of diabetes increases with obesity and older age. The pedigree function does in fact, accurately help estimate the test results for diabetes.

Upon examining the distribution of class values, I noticed that there are 500 negative instances and 268 positive instances. The population is generally young (less than 50 years old) and I believe that some attributes wherein a zero value exists might be due to errors in the data, most especially in mass. Also Plasma glucose concentration,Blood pressure,Skinfold thickness and body mass are normally distributed in the dataset.



summary of the data in a histogram format

3. Run K-NN on the same data (also using the same setup) to predict diabetes.

- Try different K's (K=1, K=3).
- Report the same scores as before (for each K value).
- Compare the models with F score. Which model has better Accuracy and F score? (logistic or KNN K1 or KNN K3).
- *Optional: plot also roc curves to compare.*

For this task I maintained the train and test data from Task number 2. For obtaining my results I used Weka and implemented the K-NN algorithm in Weka called Ibk.

For K = 1 for the train data the following scores are obtainable

```
Test mode:evaluate on training data
=== Classifier model (full training set) ===
IB1 instance-based classifier
using 1 nearest neighbour(s) for classification
Time taken to build model: 0 seconds
=== Evaluation on training set ===
=== Summary ===
Correctly Classified Instances        614             100      %
Incorrectly Classified Instances        0               0      %
Kappa statistic                         1
Mean absolute error                     0.0016
Root mean squared error                 0.0016
Relative absolute error                 0.3581 %
Root relative squared error             0.3411 %
Total Number of Instances             614
=== Detailed Accuracy By Class ===

              TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class

                 1        0         1         1        1          1        0

                 1        0         1         1        1          1        1

Weighted Avg.    1        0         1         1        1          1

=== Confusion Matrix ===

   a   b   <-- classified as

 401   0 |   a = 0

   0 213 |   b = 1
```

For K = 1 using test data on model

```
Test mode:user supplied test set: size unknown (reading incrementally)
=== Classifier model (full training set) ===
IB1 instance-based classifier
using 1 nearest neighbour(s) for classification
Time taken to build model: 0 seconds
=== Evaluation on test set ===
=== Summary ===
Correctly Classified Instances        104              67.5325 %
Incorrectly Classified Instances       50              32.4675 %
Kappa statistic                         0.3042
Mean absolute error                     0.3252
Root mean squared error                 0.5689
Relative absolute error                71.2629 %
Root relative squared error           118.7004 %
Total Number of Instances             154
```

```
=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class

                 0.727     0.418      0.758     0.727     0.742      0.655    0

                 0.582     0.273      0.542     0.582     0.561      0.655    1

Weighted Avg.    0.675     0.366      0.681     0.675     0.678      0.655

=== Confusion Matrix ===
  a  b   <-- classified as

 72 27 |  a = 0

 23 32 |  b = 1
```

For K = 3 for the train data the following scores are obtainable

```
Test mode:evaluate on training data
=== Classifier model (full training set) ===
IB1 instance-based classifier
using 3 nearest neighbour(s) for classification
Time taken to build model: 0 seconds
=== Evaluation on training set ===
=== Summary ===
Correctly Classified Instances        534               86.9707 %
Incorrectly Classified Instances       80               13.0293 %
Kappa statistic                      0.7026
Mean absolute error                       0.1898
Root mean squared error                   0.3035
Relative absolute error                41.874  %
Root relative squared error              63.7637 %
Total Number of Instances              614

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class

                 0.938     0.258      0.872     0.938     0.904      0.938    0

                 0.742     0.062      0.863     0.742     0.798      0.938    1

Weighted Avg.    0.87      0.19       0.869     0.87      0.867      0.938

=== Confusion Matrix ===
   a   b   <-- classified as

 376  25 |   a = 0

  55 158 |   b = 1
```

For K = 3 for using the test data on the outcome of the train data the following scores are obtainable

```
Test mode:user supplied test set: size unknown (reading incrementally)
=== Classifier model (full training set) ===
IB1 instance-based classifier
using 3 nearest neighbour(s) for classification
Time taken to build model: 0 seconds
=== Evaluation on test set ===
=== Summary ===
Correctly Classified Instances        111             72.0779 %
Incorrectly Classified Instances       43             27.9221 %
Kappa statistic                     0.3944
Mean absolute error                      0.327
Root mean squared error                    0.472
Relative absolute error              71.6536 %
Root relative squared error             98.4927 %
Total Number of Instances              154
=== Detailed Accuracy By Class ===
             TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
               0.778     0.382      0.786     0.778     0.782      0.731    0
               0.618     0.222      0.607     0.618     0.613      0.731    1
Weighted Avg.    0.721     0.325      0.722     0.721     0.721      0.731
=== Confusion Matrix ===
  a  b   <-- classified as
 77 22 |  a = 0
 21 34 |  b = 1
```

Conclusion

From the data above one can comfortably conclude that model where K=3 is more accurate (72.0779%) than the model with K=1(65%) when comparing the number of correctly classified instances for both test data. The F-Score for K=3 is at 0.721 for the test data whereas when that of K = 1 is observed the result is 0.675.

4. In this task we are using diamonds data from the package ggplot2 (data(diamonds)). Build regression models predicting price from the rest of the features, where

A) model 1 has all the features

B) model 2 has all the features + 'carat' and 'depth' of degree 2

C) model 3 has all the features + 3rd degree polynomials of 'carat' and 'depth' (i.e. carat^3, carat^2, carat, depth^3,...)

D) model 4 has all the features + 3rd degree polynomials of 'carat' and 'depth' + 'x','y','z' of degree 2

- in R you can use `poly(x,d)` to evaluate a polynomial of degree *d*, e.g. `lm(price ~ poly(x,3) + ..., data=diamonds)`
- Use the regular 80% train / 20% test split.
- Measure the RMSE for all the models on the train and test set and plot a graph, where on x-axis models are sorted according to the complexity of the model and on y-axis RMSE for train and test split. What do you observe? Can we diagnose under- or overfitting problems?

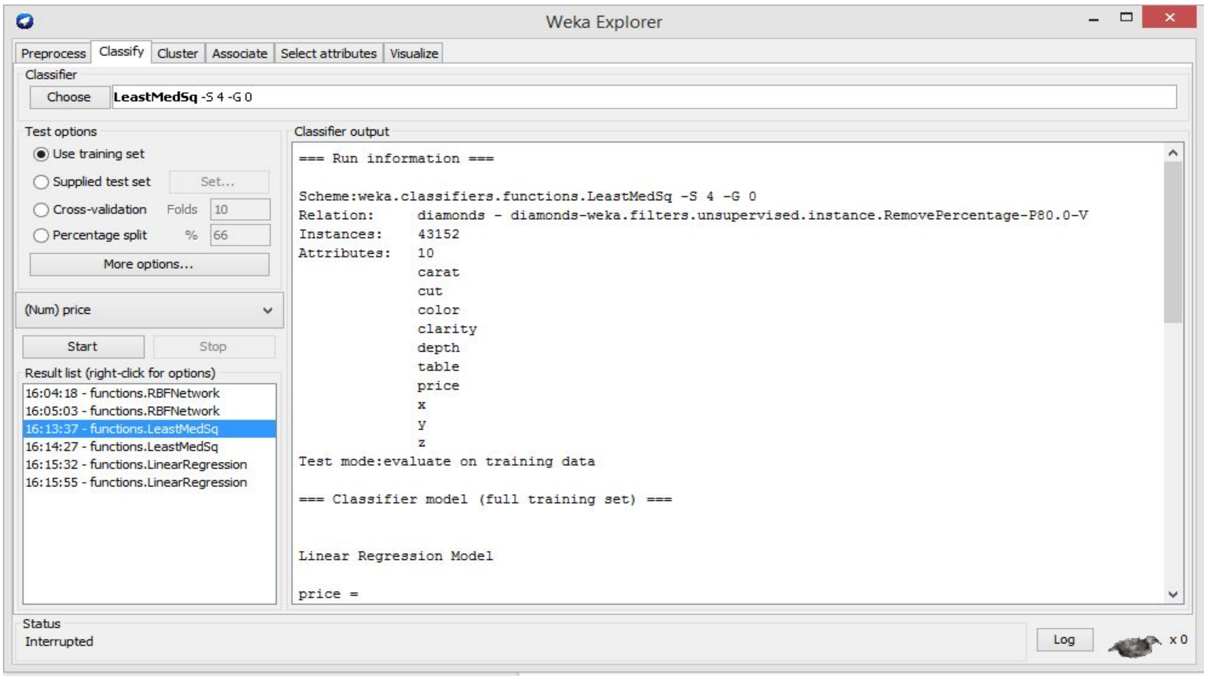
Solution Task 4

I used both R and Weka for this task. R was used for the pre processing while weka was used for the calculations and graphs. For the first model the data was extracted from R into a csv format where I went ahead to read it in to Weka. In weka I split the data into test and training set and performed the task. The screenshot and output data are as follows

N/B - In order to keep the data meaningful I will only give a summary of the important output data from Weka including the RMSE score. All train/test files attached in their various models in zip file

Model A (Train)

I used the Least Median Square in Weka to build my model because it implements a better style of linear regression. The summary of the obtainable results for the model are as follows



Classifier: **LeastMedSq** -S 4 -G 0

Test options

● Use training set

○ Supplied test set

○ Cross-validation Folds 10

○ Percentage split % 66

(Num) price

Result list (right-click for options)

16:04:18 - functions.RBFNetwork
16:05:03 - functions.RBFNetwork
16:13:37 - functions.LeastMedSq
16:14:27 - functions.LeastMedSq
16:15:32 - functions.LinearRegression
16:15:55 - functions.LinearRegression

Classifier output

```
=== Run information ===

Scheme:weka.classifiers.functions.LeastMedSq -S 4 -G 0
Relation:     diamonds - diamonds-weka.filters.unsupervised.instance.RemovePercentage-P80.0-V
Instances:    43152
Attributes:   10
              carat
              cut
              color
              clarity
              depth
              table
              price
              x
              y
              z
Test mode:evaluate on training data

=== Classifier model (full training set) ===


Linear Regression Model

price =
```

Status
Interrupted

Test mode:evaluate on training data

=== Classifier model (full training set) ===

=== Evaluation on training set ===

=== Summary ===

Correlation coefficient 0.946

Mean absolute error 1046.0265

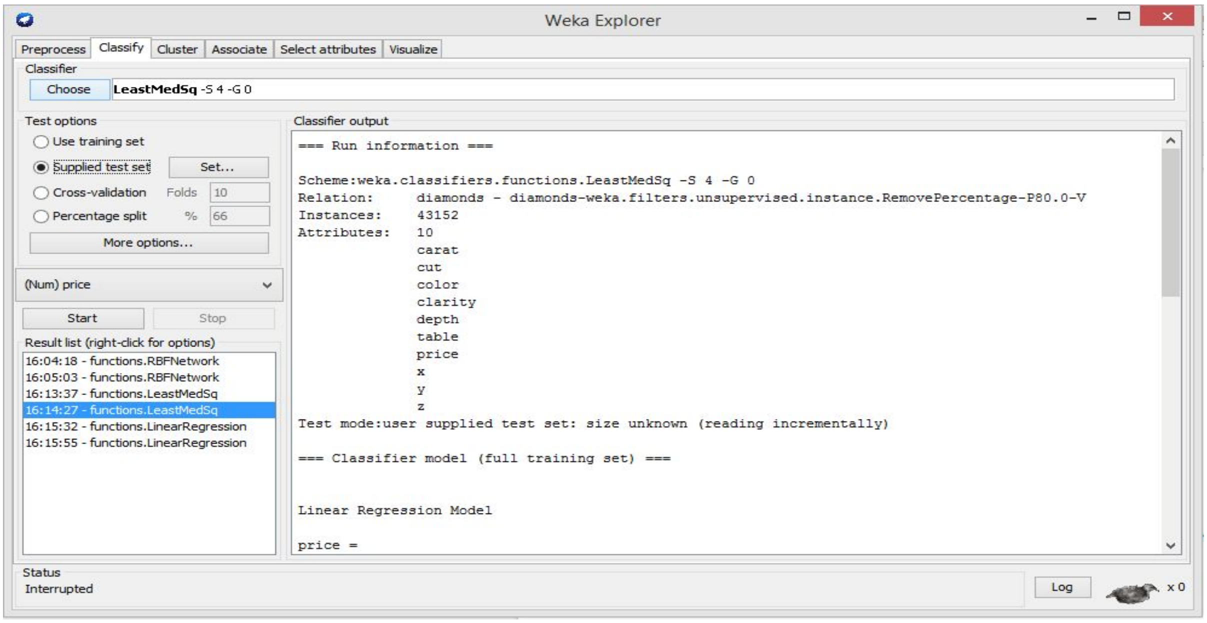
Root mean squared error 2020.2386

Relative absolute error 31.2203 %

Root relative squared error 46.9762 %

Total Number of Instances 43152

Model A (Test)



Classifier

Choose **LeastMedSq** -S 4 -G 0

Test options

○ Use training set
● Supplied test set
○ Cross-validation Folds 10
○ Percentage split % 66

(Num) price

Result list (right-click for options)

16:04:18 - functions.RBFNetwork
16:05:03 - functions.RBFNetwork
16:13:37 - functions.LeastMedSq
16:14:27 - functions.LeastMedSq
16:15:32 - functions.LinearRegression
16:15:55 - functions.LinearRegression

Classifier output

```
=== Run information ===

Scheme:weka.classifiers.functions.LeastMedSq -S 4 -G 0
Relation:     diamonds - diamonds-weka.filters.unsupervised.instance.RemovePercentage-P80.0-V
Instances:    43152
Attributes:   10
              carat
              cut
              color
              clarity
              depth
              table
              price
              x
              y
              z
Test mode:user supplied test set: size unknown (reading incrementally)

=== Classifier model (full training set) ===


Linear Regression Model

price =
```

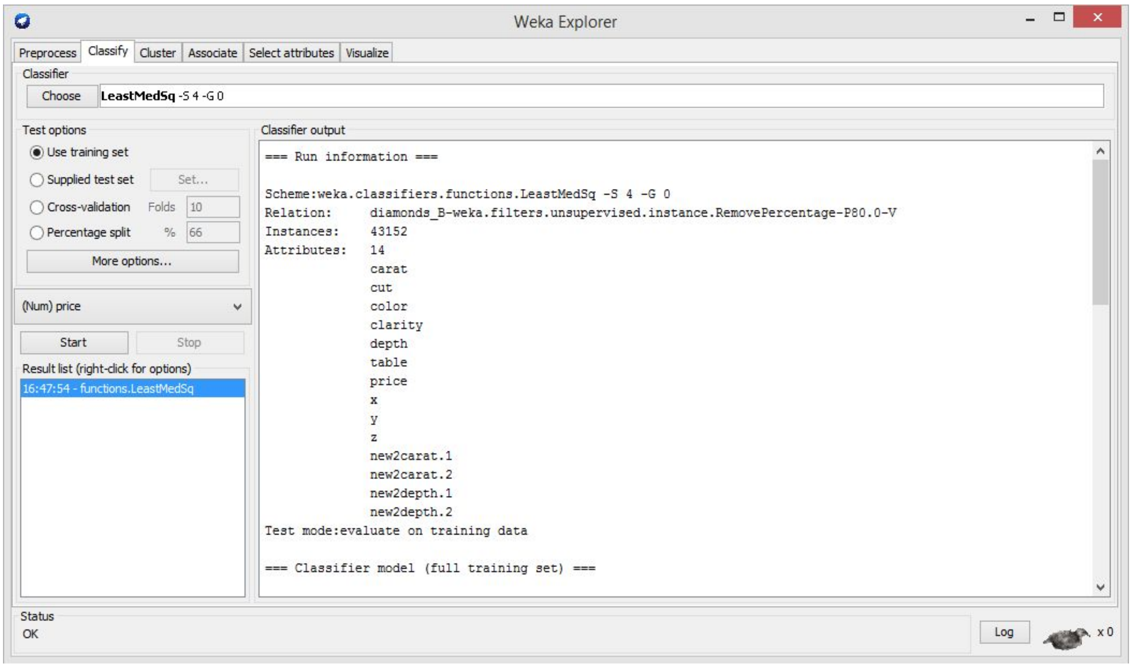
Status

Interrupted

Test mode:user supplied test set: size unknown (reading incrementally)
=== Classifier model (full training set) ===
=== Evaluation on test set ===
=== Summary ===

Correlation coefficient	**0.9246**
Mean absolute error	**349.9862**
Root mean squared error	**424.0939**
Relative absolute error	**13.6533 %**
Root relative squared error	**16.1314 %**
Total Number of Instances	**10788**

Model B (Train)



Classifier

Choose **LeastMedSq** -S 4 -G 0

Test options

● Use training set
○ Supplied test set
○ Cross-validation Folds 10
○ Percentage split % 66

(Num) price

Result list (right-click for options)

16:47:54 - functions.LeastMedSq

Classifier output

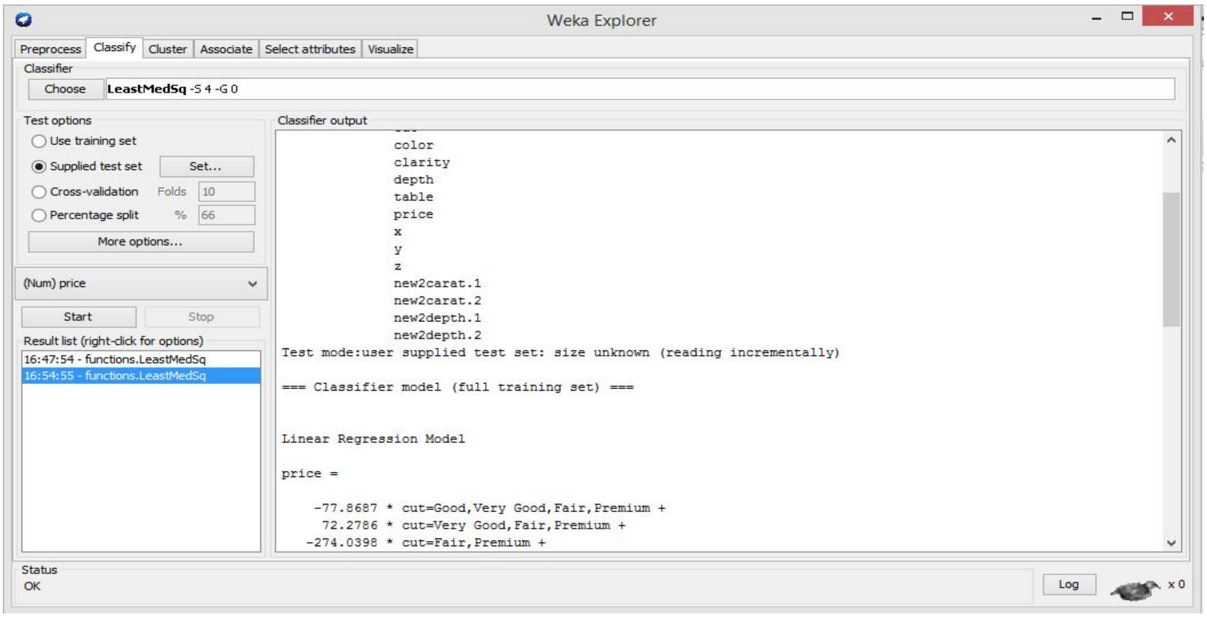
```
=== Run information ===

Scheme:weka.classifiers.functions.LeastMedSq -S 4 -G 0
Relation:     diamonds_B-weka.filters.unsupervised.instance.RemovePercentage-P80.0-V
Instances:    43152
Attributes:   14
              carat
              cut
              color
              clarity
              depth
              table
              price
              x
              y
              z
              new2carat.1
              new2carat.2
              new2depth.1
              new2depth.2
Test mode:evaluate on training data

=== Classifier model (full training set) ===
```

Status

OK

Test mode:evaluate on training data
=== Classifier model (full training set) ===
=== Evaluation on training set ===
=== Summary ===
Correlation coefficient 0.9337
Mean absolute error 1141.464
Root mean squared error 2143.0573
Relative absolute error 34.0687 %
Root relative squared error 49.8321 %
Total Number of Instances 43152

Model B (Test)



Classifier
LeastMedSq -S 4 -G 0

Test options
○ Use training set
● Supplied test set
○ Cross-validation Folds 10
○ Percentage split % 66

(Num) price

Result list (right-click for options)
16:47:54 - functions.LeastMedSq
16:54:55 - functions.LeastMedSq

Classifier output

```
              color
              clarity
              depth
              table
              price
              x
              y
              z
              new2carat.1
              new2carat.2
              new2depth.1
              new2depth.2
Test mode:user supplied test set: size unknown (reading incrementally)

=== Classifier model (full training set) ===


Linear Regression Model

price =

    -77.8687 * cut=Good,Very Good,Fair,Premium +
     72.2786 * cut=Very Good,Fair,Premium +
   -274.0398 * cut=Fair,Premium +
```

Status
OK

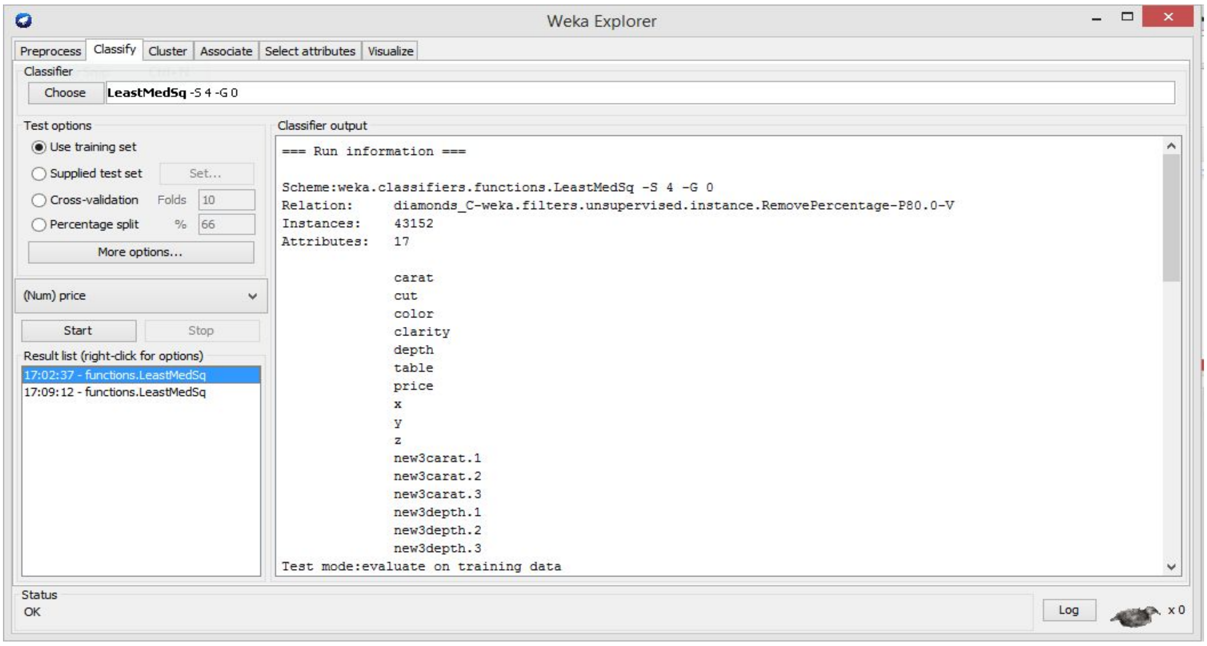
Test mode:user supplied test set: size unknown (reading incrementally)
=== Evaluation on test set ===
=== Summary ===

Correlation coefficient 0.8145
Mean absolute error 510.987
Root mean squared error 709.1636
Relative absolute error 19.9341 %
Root relative squared error 26.9746 %
Total Number of Instances 10788

Summary

For this model I was faced with the task of converting to third degree polynomial which obviously was a challenge to do in Weka so I switched to R, used the formula provided and went ahead to write the data into a csv format and used same on weka for the rest of the Models.

Model C (Train)



Classifier

Choose LeastMedSq -S 4 -G 0

Test options

● Use training set
○ Supplied test set Set...
○ Cross-validation Folds 10
○ Percentage split % 66

(Num) price

Result list (right-click for options)

17:02:37 - functions.LeastMedSq
17:09:12 - functions.LeastMedSq

Classifier output

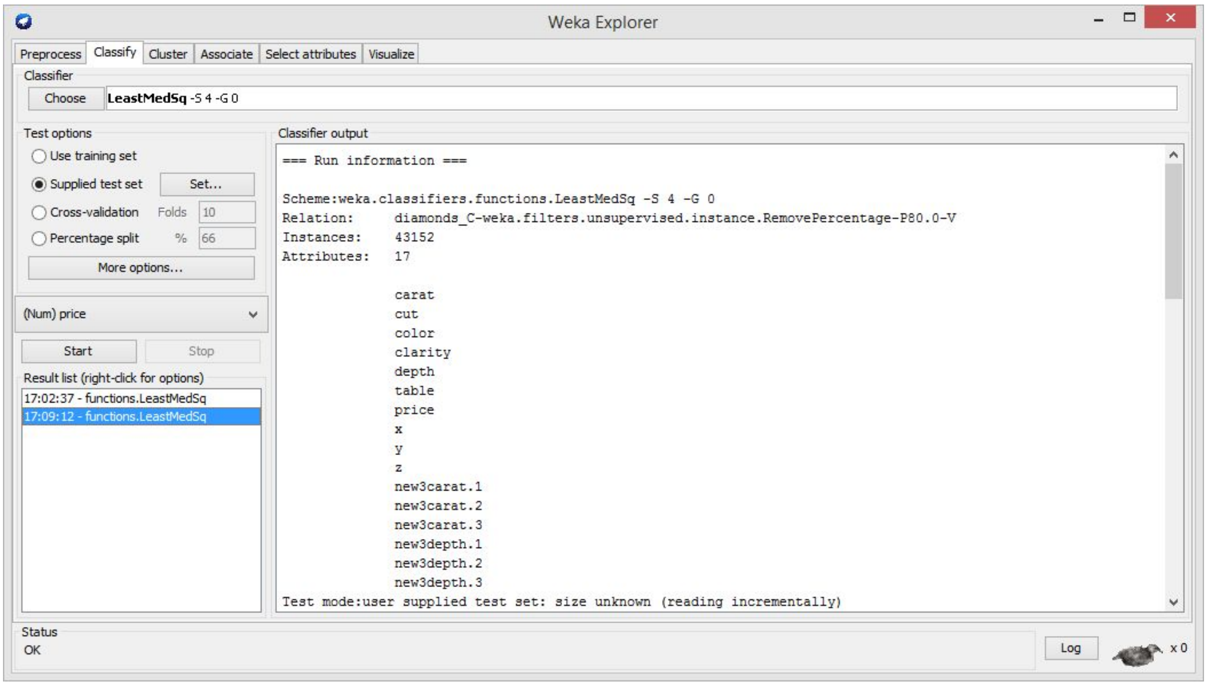
```
=== Run information ===

Scheme:weka.classifiers.functions.LeastMedSq -S 4 -G 0
Relation:     diamonds_C-weka.filters.unsupervised.instance.RemovePercentage-P80.0-V
Instances:    43152
Attributes:   17

              carat
              cut
              color
              clarity
              depth
              table
              price
              x
              y
              z
              new3carat.1
              new3carat.2
              new3carat.3
              new3depth.1
              new3depth.2
              new3depth.3
Test mode:evaluate on training data
```

Status

OK

```
Test mode:evaluate on training data
=== Evaluation on training set ===
=== Summary ===
Correlation coefficient                  0.9096
Mean absolute error                    800.9004
Root mean squared error                  1810.6231
Relative absolute error                 23.9041 %
Root relative squared error               42.1021 %
Total Number of Instances              43152
```

Model C (Test)



Classifier

Choose LeastMedSq -S 4 -G 0

Test options

○ Use training set
● Supplied test set Set...
○ Cross-validation Folds 10
○ Percentage split % 66

(Num) price

Result list (right-click for options)

17:02:37 - functions.LeastMedSq
17:09:12 - functions.LeastMedSq

Classifier output

```
=== Run information ===

Scheme:weka.classifiers.functions.LeastMedSq -S 4 -G 0
Relation:     diamonds_C-weka.filters.unsupervised.instance.RemovePercentage-P80.0-V
Instances:    43152
Attributes:   17

              carat
              cut
              color
              clarity
              depth
              table
              price
              x
              y
              z
              new3carat.1
              new3carat.2
              new3carat.3
              new3depth.1
              new3depth.2
              new3depth.3
Test mode:user supplied test set: size unknown (reading incrementally)
```

Status

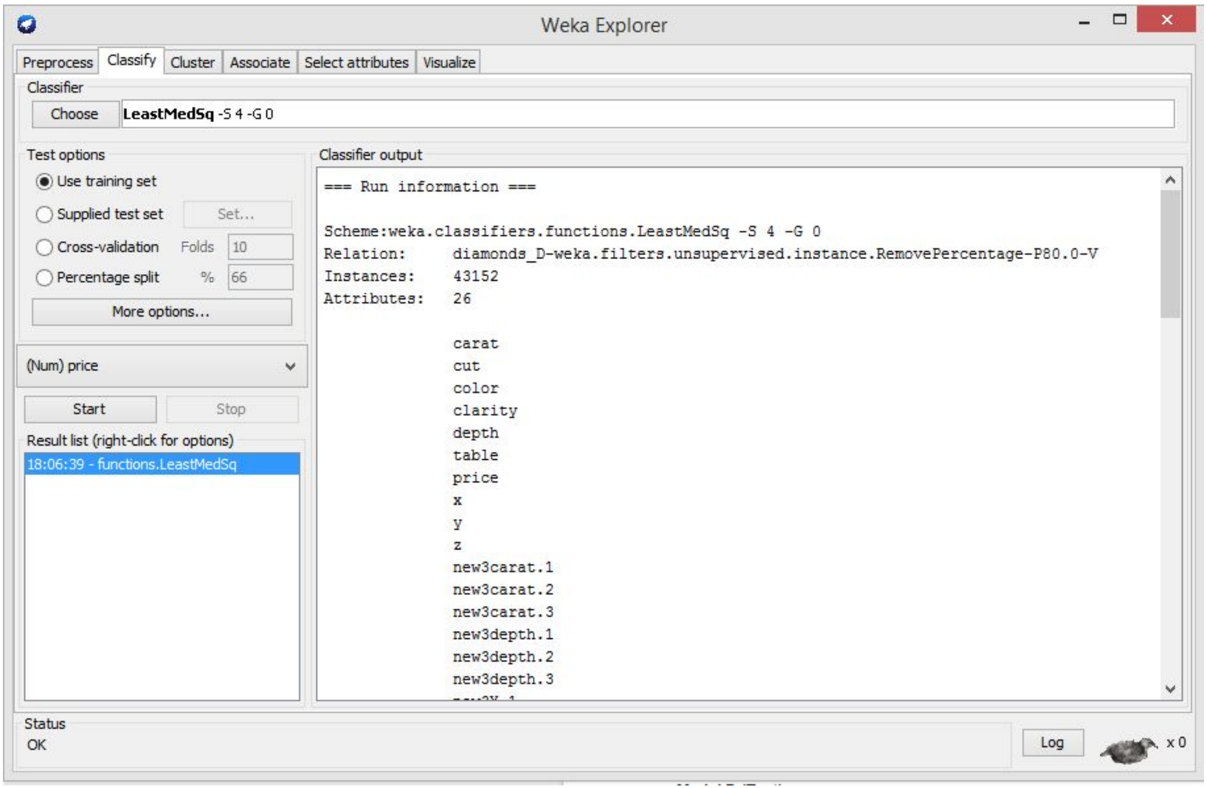
OK

Test mode:user supplied test set: size unknown (reading incrementally)
=== Evaluation on test set ===
=== Summary ===

Correlation coefficient	**0.7739**
Mean absolute error	**277.5207**
Root mean squared error	**535.7505**
Relative absolute error	**10.8264 %**
Root relative squared error	**20.3785 %**
Total Number of Instances	**10788**

Model D (Train)

Like for the previous models I did all the polynomial conversion in R and prooceeded to write the data file into csv format which enabled me to preprocess the data with Weka and build a model. The entire flow of calculated attributes ca be found in the appendix also.



LeastMedSq -S 4 -G 0

● Use training set
○ Supplied test set
○ Cross-validation Folds 10
○ Percentage split % 66

(Num) price

18:06:39 - functions.LeastMedSq

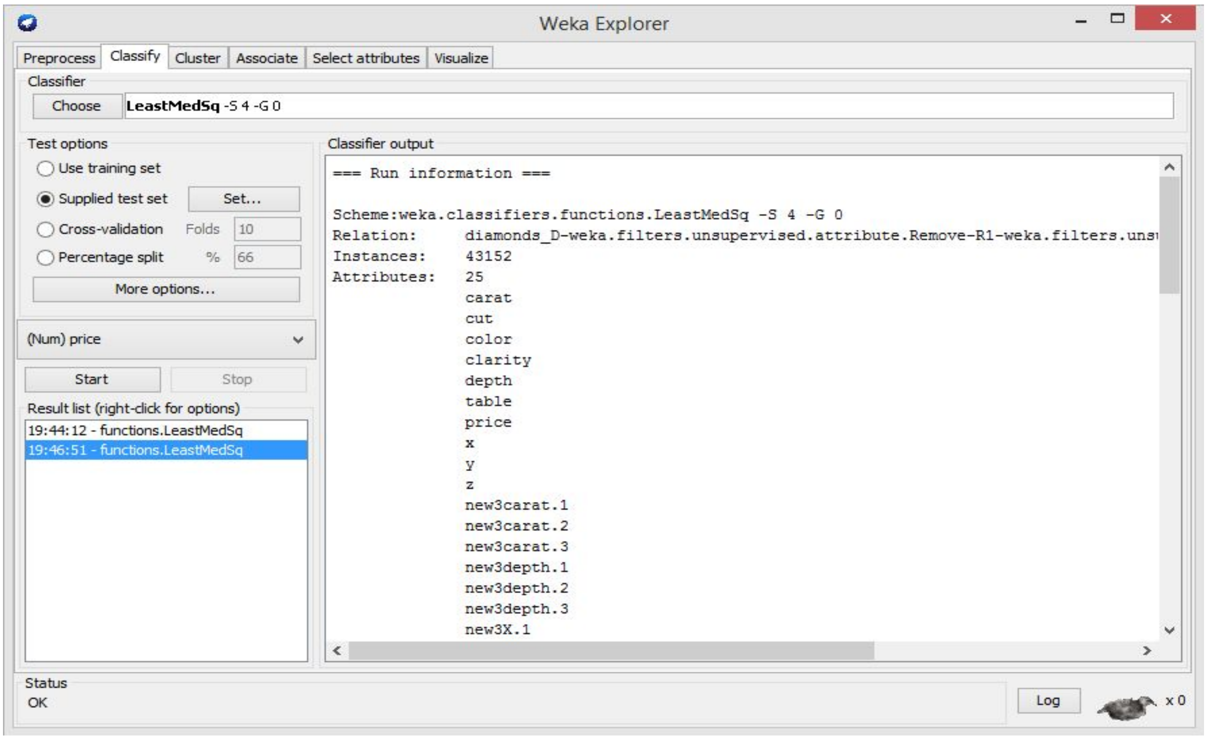
```
=== Run information ===

Scheme:weka.classifiers.functions.LeastMedSq -S 4 -G 0
Relation:     diamonds_D-weka.filters.unsupervised.instance.RemovePercentage-P80.0-V
Instances:    43152
Attributes:   26

              carat
              cut
              color
              clarity
              depth
              table
              price
              x
              y
              z
              new3carat.1
              new3carat.2
              new3carat.3
              new3depth.1
              new3depth.2
              new3depth.3
```

Status: OK

Test mode:evaluate on training data
=== Evaluation on training set ===
=== Summary ===

Correlation coefficient	**0.8987**
Mean absolute error	**1205.845**
Root mean squared error	**2547.3644**
Relative absolute error	**35.9903 %**
Root relative squared error	**59.2333 %**
Total Number of Instances	**43152**

Model D (Test)



LeastMedSq -S 4 -G 0

○ Use training set
● Supplied test set
○ Cross-validation Folds 10
○ Percentage split % 66

(Num) price

19:44:12 - functions.LeastMedSq
19:46:51 - functions.LeastMedSq

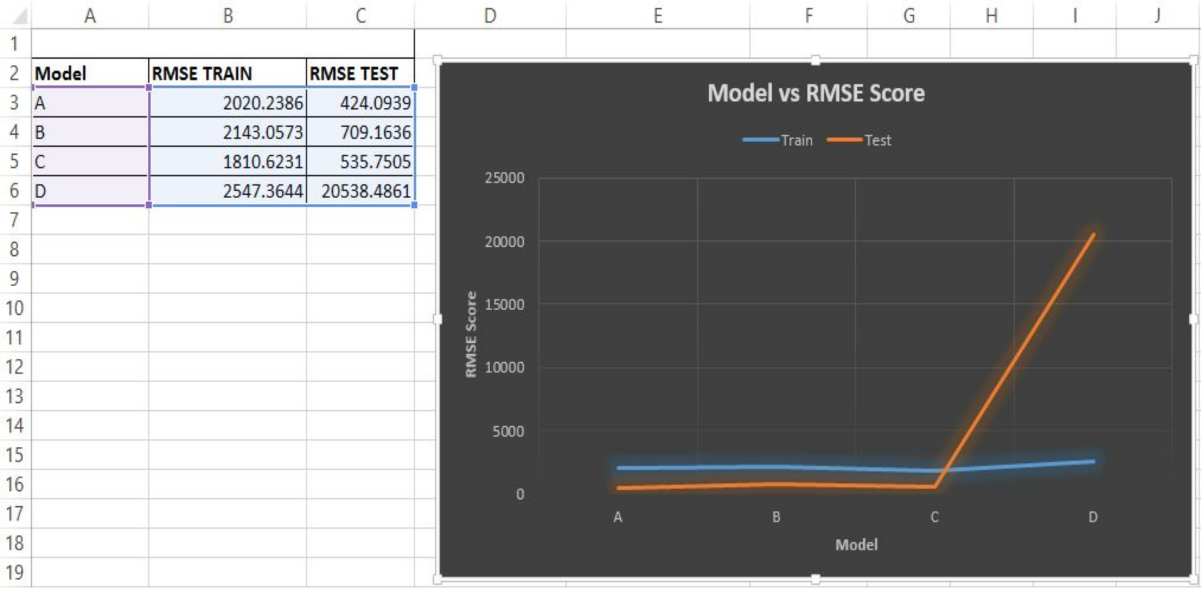
Status: OK

```
=== Run information ===

Scheme:weka.classifiers.functions.LeastMedSq -S 4 -G 0
Relation:     diamonds_D-weka.filters.unsupervised.attribute.Remove-R1-weka.filters.uns
Instances:    43152
Attributes:   25
              carat
              cut
              color
              clarity
              depth
              table
              price
              x
              y
              z
              new3carat.1
              new3carat.2
              new3carat.3
              new3depth.1
              new3depth.2
              new3depth.3
              new3X.1
```

Test mode:user supplied test set: size unknown (reading incrementally)
=== Evaluation on test set ===
=== Summary ===

```
Correlation coefficient              0.0348
Mean absolute error                491.6804
Root mean squared error              20538.4861
Relative absolute error              19.1809 %
Root relative squared error            781.227  %
Total Number of Instances             10788
```

Graph of Models vs RMSE Scores



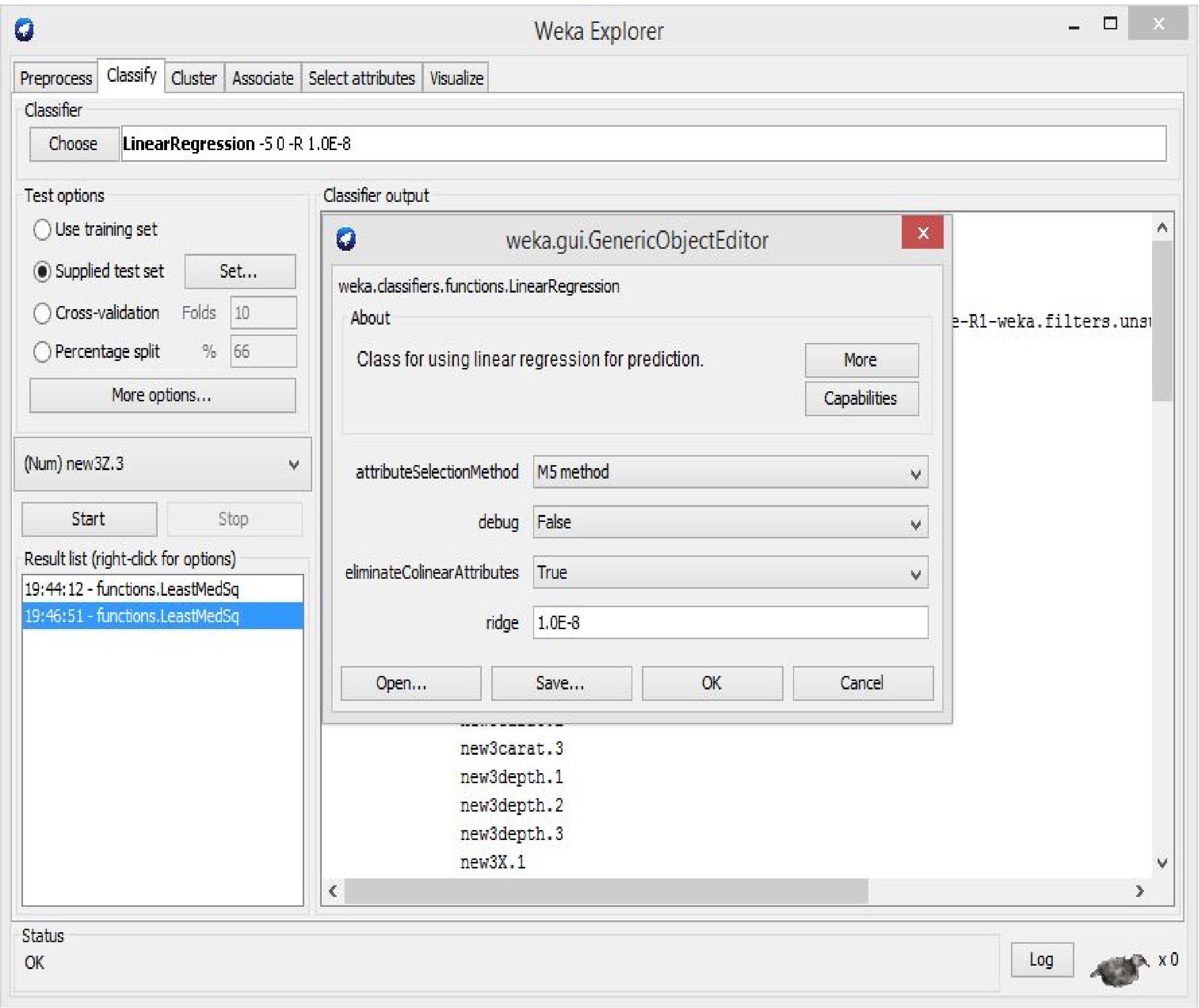
Model	RMSE TRAIN	RMSE TEST
A	2020.2386	424.0939
B	2143.0573	709.1636
C	1810.6231	535.7505
D	2547.3644	20538.4861

Observation from Graph

From the Graph above my observation is that an overfitting problem occurs. While both the test and train RMSE for all models between A to C are relatively proportional, there is a disjoint and disproportionate observation between the test and train RMSE score for Model D. This is obviously an overfitting in the model which I believe might be due to some random error or noise instead of the underlying relationship.

7. (optional bonus, 1p). Try ridge and lasso regressions for model 4 from task 4 and add the resulting RMSE of training and test set on the plot generated in task 4. Did it help?

Model D-Ridge



Classifier: LinearRegression -S 0 -R 1.0E-8

Test options
○ Use training set
● Supplied test set
○ Cross-validation Folds 10
○ Percentage split % 66

(Num) new3Z.3

Result list (right-click for options)
19:44:12 - functions.LeastMedSq
19:46:51 - functions.LeastMedSq

Classifier output

weka.gui.GenericObjectEditor

weka.classifiers.functions.LinearRegression

About

Class for using linear regression for prediction.

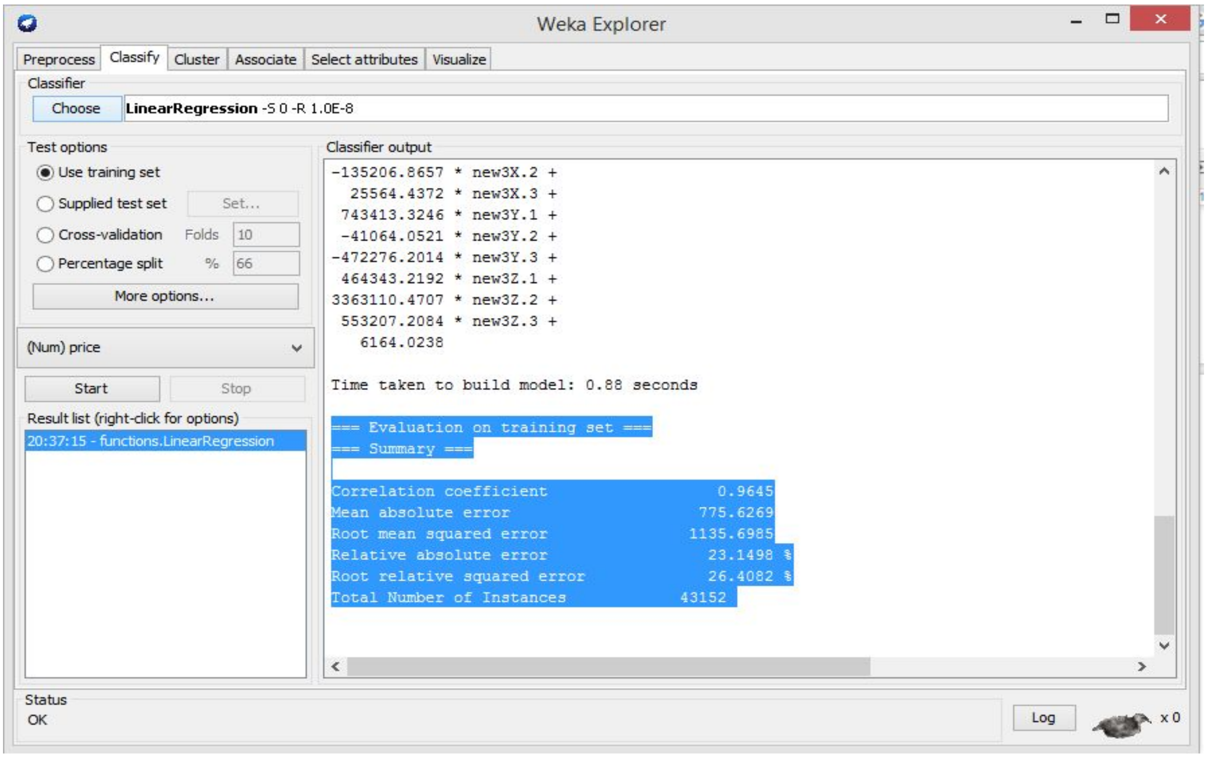
attributeSelectionMethod: M5 method
debug: False
eliminateColinearAttributes: True
ridge: 1.0E-8

```
e-R1-weka.filters.uns
new3carat.3
new3depth.1
new3depth.2
new3depth.3
new3X.1
```

Status
OK

To perform this Task I made use of the Linear regression function in Weka which implements ridge regression with a default value of 1.0E-8. As can be seen in the screenshot above, this was used to implement ridge regression on Model D. The results for the output of applying Ridge Regression to both Training and Test data on Model D are as follows.

Model D-Ridge (Train)



LinearRegression -S 0 -R 1.0E-8

● Use training set
○ Supplied test set
○ Cross-validation Folds 10
○ Percentage split % 66

(Num) price

20:37:15 - functions.LinearRegression

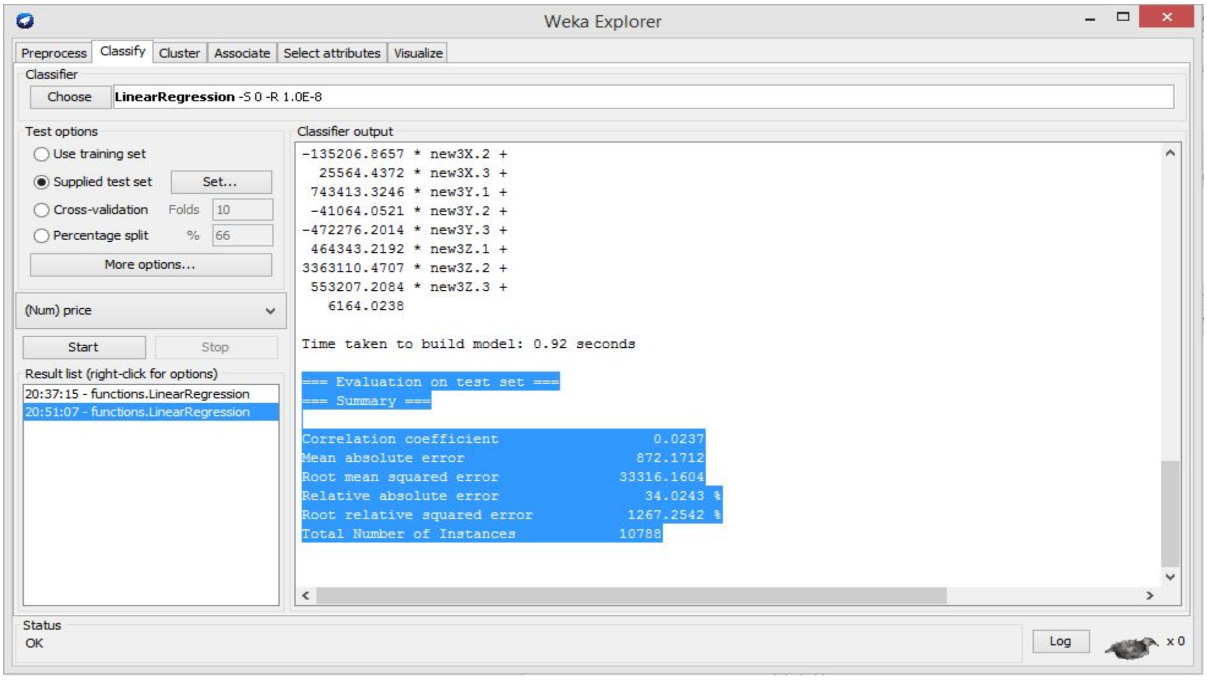
```
-135206.8657 * new3X.2 +
  25564.4372 * new3X.3 +
 743413.3246 * new3Y.1 +
 -41064.0521 * new3Y.2 +
-472276.2014 * new3Y.3 +
 464343.2192 * new3Z.1 +
3363110.4707 * new3Z.2 +
 553207.2084 * new3Z.3 +
   6164.0238

Time taken to build model: 0.88 seconds

=== Evaluation on training set ===
=== Summary ===

Correlation coefficient                  0.9645
Mean absolute error                    775.6269
Root mean squared error               1135.6985
Relative absolute error                 23.1498 %
Root relative squared error             26.4082 %
Total Number of Instances            43152
```

Status
OK

=== Evaluation on training set ===
=== Summary ===

Correlation coefficient	0.9645
Mean absolute error	775.6269
Root mean squared error	1135.6985
Relative absolute error	23.1498 %
Root relative squared error	26.4082 %
Total Number of Instances	43152

Model D-Ridge (Test)



LinearRegression -S 0 -R 1.0E-8

○ Use training set
● Supplied test set
○ Cross-validation Folds 10
○ Percentage split % 66

(Num) price

20:37:15 - functions.LinearRegression
20:51:07 - functions.LinearRegression

```
-135206.8657 * new3X.2 +
  25564.4372 * new3X.3 +
 743413.3246 * new3Y.1 +
 -41064.0521 * new3Y.2 +
-472276.2014 * new3Y.3 +
 464343.2192 * new3Z.1 +
3363110.4707 * new3Z.2 +
 553207.2084 * new3Z.3 +
   6164.0238

Time taken to build model: 0.92 seconds

=== Evaluation on test set ===
=== Summary ===

Correlation coefficient                  0.0237
Mean absolute error                    872.1712
Root mean squared error              33316.1604
Relative absolute error                 34.0243 %
Root relative squared error           1267.2542 %
Total Number of Instances            10788
```

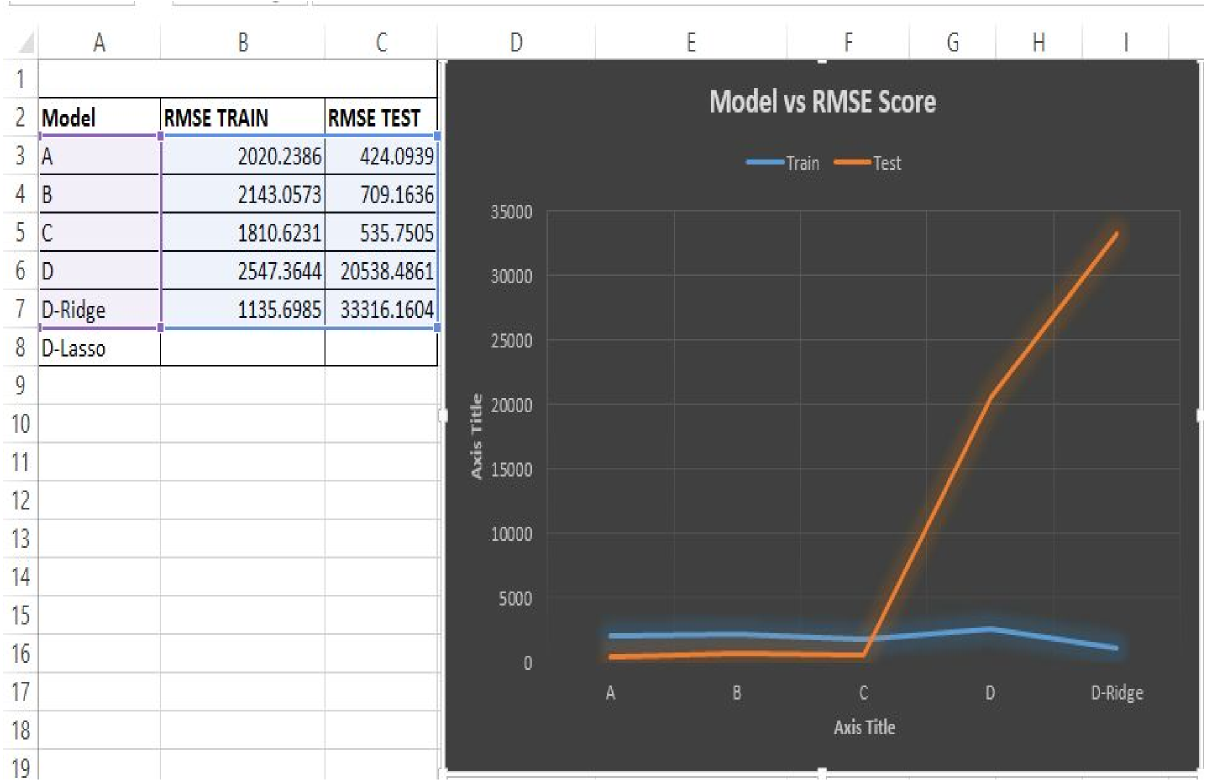
Status
OK

```
=== Evaluation on test set ===
=== Summary ===
Correlation coefficient               0.0237
Mean absolute error                 872.1712
Root mean squared error               33316.1604
Relative absolute error              34.0243 %
Root relative squared error           1267.2542 %
Total Number of Instances             10788
```

Model D - Lasso

Using Weka I noticed the absence of any function that implemented Lasso regression hence I did a little research and went through some previous questions asked about lasso regression on Weka. Indeed there is no available in built package for this however there exists an Rplugin that can perform Lasso regression as outlined by the University of Waikato here http://weka.8497.n7.nabble.com/Logistic-regression-with-Lasso-regularization-td32836.html
At the point of this submission I had not successfully implemented the plugin in Weka however I will do that and submit immediately I have been able to implement it.

Graph of RMSE scores vs Models



Model	RMSE TRAIN	RMSE TEST
A	2020.2386	424.0939
B	2143.0573	709.1636
C	1810.6231	535.7505
D	2547.3644	20538.4861
D-Ridge	1135.6985	33316.1604
D-Lasso		

Observation and conclusion

From the screenshot above it is safe to conclude on the basis of applying Ridge regression to Model D that this does not in anyway help us in reducing overfitting to at all as it can been seen that the curve moves upwards as opposed to downwards. There is a slight change in the train set however I personally believe this to be due to the difference in algorithm parameters used in Weka. Unfortunately I cannot compare yet with Lasso regression but I hope to do so soonest

APPENDIX

MODEL A TRAIN

```
Linear Regression Model
price =
   5166.6562 * carat +
   -138.4332 * cut=Good,Very Good,Fair,Premium +
     89.8772 * cut=Very Good,Fair,Premium +
   -382.884  * cut=Fair,Premium +
    396.1977 * cut=Premium +
    113.5176 * color=D,F,G,H,I,J +
   -186.174  * color=F,G,H,I,J +
    -82.5072 * color=G,H,I,J +
    -66.7112 * color=H,I,J +
   -262.6146 * color=I,J +
   -359.4282 * color=J +
     75.9191 * clarity=IF,VVS2,VS1,I1,VS2,SI1,SI2 +
   -173.2083 * clarity=VVS2,VS1,I1,VS2,SI1,SI2 +
    -97.1214 * clarity=VS1,I1,VS2,SI1,SI2 +
  -2055.9692 * clarity=I1,VS2,SI1,SI2 +
   1981.4252 * clarity=VS2,SI1,SI2 +
   -207.7204 * clarity=SI1,SI2 +
   -449.222  * clarity=SI2 +
     21.236  * depth +
     -0.1629 * table +
    -89.2691 * x +
    415.9614 * y +
     15.03   * z +
  -3416.4567 (Time taken to build model: 32.46 seconds)
```

MODEL A TEST

```
Linear Regression Model
price =
   5166.6562 * carat +
   -138.4332 * cut=Good,Very Good,Fair,Premium +
     89.8772 * cut=Very Good,Fair,Premium +
   -382.884  * cut=Fair,Premium +
    396.1977 * cut=Premium +
    113.5176 * color=D,F,G,H,I,J +
   -186.174  * color=F,G,H,I,J +
    -82.5072 * color=G,H,I,J +
    -66.7112 * color=H,I,J +
   -262.6146 * color=I,J +
   -359.4282 * color=J +
     75.9191 * clarity=IF,VVS2,VS1,I1,VS2,SI1,SI2 +
   -173.2083 * clarity=VVS2,VS1,I1,VS2,SI1,SI2 +
    -97.1214 * clarity=VS1,I1,VS2,SI1,SI2 +
  -2055.9692 * clarity=I1,VS2,SI1,SI2 +
   1981.4252 * clarity=VS2,SI1,SI2 +
   -207.7204 * clarity=SI1,SI2 +
   -449.222  * clarity=SI2 +
     21.236  * depth +
     -0.1629 * table +
    -89.2691 * x +
    415.9614 * y +
     15.03   * z +
  -3416.4567
Time taken to build model: 32.46 seconds
```

MODEL B TRAIN

```
Linear Regression Model
price =
    -77.8687 * cut=Good,Very Good,Fair,Premium +
     72.2786 * cut=Very Good,Fair,Premium +
   -274.0398 * cut=Fair,Premium +
    252.9161 * cut=Premium +
    107.5961 * color=D,F,G,H,I,J +
   -180.8728 * color=F,G,H,I,J +
    -65.9988 * color=G,H,I,J +
    -53.3338 * color=H,I,J +
   -246.579  * color=I,J +
   -338.7139 * color=J +
     81.2    * clarity=IF,VVS2,VS1,I1,VS2,SI1,SI2 +
   -175.5747 * clarity=VVS2,VS1,I1,VS2,SI1,SI2 +
   -138.0642 * clarity=VS1,I1,VS2,SI1,SI2 +
  -1999.4805 * clarity=I1,VS2,SI1,SI2 +
   1954.6267 * clarity=VS2,SI1,SI2 +
   -236.1905 * clarity=SI1,SI2 +
   -407.9434 * clarity=SI2 +
   -203.1043 * depth +
     19.8103 * table +
   1109.5151 * x +
    676.0609 * y +
   1169.1071 * z +
 109773.8603 * new2carat.2 +
  85712.2505 * new2depth.1 +
 -36251.3557 * new2depth.2 +
    922.8422
Time taken to build model: 36.31 seconds
```

Model C Train

```
Linear Regression Model
price =
    -93.0853 * cut=Good,Very Good,Fair,Premium +
     30.4907 * cut=Very Good,Fair,Premium +
   -261.7426 * cut=Fair,Premium +
    276.78   * cut=Premium +
    122.427  * color=D,F,G,H,I,J +
   -184.9651 * color=F,G,H,I,J +
    -36.6564 * color=G,H,I,J +
   -274.7847 * color=H,I,J +
   -328.0656 * color=I,J +
   -176.9376 * color=J +
     43.3987 * clarity=IF,VVS2,VS1,I1,VS2,SI1,SI2 +
   -106.1673 * clarity=VVS2,VS1,I1,VS2,SI1,SI2 +
      4.5306 * clarity=VS1,I1,VS2,SI1,SI2 +
  -1198.9112 * clarity=I1,VS2,SI1,SI2 +
   1109.1384 * clarity=VS2,SI1,SI2 +
   -504.0242 * clarity=SI1,SI2 +
   -596.9354 * clarity=SI2 +
      0.4427 * table +
    637.9665 * x +
   1267.1169 * y +
    342.0056 * z +
 275136.5102 * new3carat.1 +
  -2588.3872 * new3carat.2 +
-295302.4621 * new3carat.3 +
  18934.3875 * new3depth.1 +
 -21235.1859 * new3depth.2 +
   5702.6643 * new3depth.3 +
  -7749.3445
Time taken to build model: 39.85 seconds
```

Model C Test

```
Linear Regression Model
price =
    -93.0853 * cut=Good,Very Good,Fair,Premium +
     30.4907 * cut=Very Good,Fair,Premium +
   -261.7426 * cut=Fair,Premium +
    276.78   * cut=Premium +
    122.427  * color=D,F,G,H,I,J +
   -184.9651 * color=F,G,H,I,J +
    -36.6564 * color=G,H,I,J +
   -274.7847 * color=H,I,J +
   -328.0656 * color=I,J +
   -176.9376 * color=J +
     43.3987 * clarity=IF,VVS2,VS1,I1,VS2,SI1,SI2 +
   -106.1673 * clarity=VVS2,VS1,I1,VS2,SI1,SI2 +
      4.5306 * clarity=VS1,I1,VS2,SI1,SI2 +
  -1198.9112 * clarity=I1,VS2,SI1,SI2 +
   1109.1384 * clarity=VS2,SI1,SI2 +
   -504.0242 * clarity=SI1,SI2 +
   -596.9354 * clarity=SI2 +
      0.4427 * table +
    637.9665 * x +
   1267.1169 * y +
    342.0056 * z +
 275136.5102 * new3carat.1 +
  -2588.3872 * new3carat.2 +
-295302.4621 * new3carat.3 +
  18934.3875 * new3depth.1 +
 -21235.1859 * new3depth.2 +
   5702.6643 * new3depth.3 +
  -7749.3445
Time taken to build model: 41.55 seconds
```

Model D Train

```
Linear Regression Model
price =
     -0.0001 *  +
    -99.9338 * cut=Good,Very Good,Fair,Premium +
     34.9755 * cut=Very Good,Fair,Premium +
   -191.6308 * cut=Fair,Premium +
    221.6263 * cut=Premium +
    120.4651 * color=D,F,G,H,I,J +
   -183.0859 * color=F,G,H,I,J +
    -75.8913 * color=G,H,I,J +
    -78.2264 * color=H,I,J +
   -249.5117 * color=I,J +
   -341.3437 * color=J +
     86.7717 * clarity=IF,VVS2,VS1,I1,VS2,SI1,SI2 +
   -192.1514 * clarity=VVS2,VS1,I1,VS2,SI1,SI2 +
    -59.3148 * clarity=VS1,I1,VS2,SI1,SI2 +
  -1625.2994 * clarity=I1,VS2,SI1,SI2 +
   1498.2563 * clarity=VS2,SI1,SI2 +
   -191.1586 * clarity=SI1,SI2 +
   -454.8332 * clarity=SI2 +
    711.322  * depth +
     -6.2793 * table +
 294422.8361 * new3carat.1 +
-159765.937  * new3carat.2 +
  -5156.6533 * new3carat.3 +
-243407.892  * new3depth.1 +
  -7133.9858 * new3depth.2 +
  -2296.6572 * new3depth.3 +
-140196.7508 * new3X.1 +
-270820.3095 * new3X.2 +
 -69450.8059 * new3X.3 +
 248620.0389 * new3Y.1 +
-470036.477  * new3Y.3 +
 374383.4801 * new3Z.1 +
1344389.2257 * new3Z.2 +
 179250.2749 * new3Z.3 +
 -39817.1002
Time taken to build model: 46.28 seconds
```

Model D Test

```
Linear Regression Model
price =
     -0.0001 *  +
    -99.9338 * cut=Good,Very Good,Fair,Premium +
     34.9755 * cut=Very Good,Fair,Premium +
   -191.6308 * cut=Fair,Premium +
    221.6263 * cut=Premium +
    120.4651 * color=D,F,G,H,I,J +
   -183.0859 * color=F,G,H,I,J +
    -75.8913 * color=G,H,I,J +
    -78.2264 * color=H,I,J +
   -249.5117 * color=I,J +
   -341.3437 * color=J +
     86.7717 * clarity=IF,VVS2,VS1,I1,VS2,SI1,SI2 +
   -192.1514 * clarity=VVS2,VS1,I1,VS2,SI1,SI2 +
    -59.3148 * clarity=VS1,I1,VS2,SI1,SI2 +
  -1625.2994 * clarity=I1,VS2,SI1,SI2 +
   1498.2563 * clarity=VS2,SI1,SI2 +
   -191.1586 * clarity=SI1,SI2 +
   -454.8332 * clarity=SI2 +
    711.322  * depth +
     -6.2793 * table +
 294422.8361 * new3carat.1 +
-159765.937  * new3carat.2 +
  -5156.6533 * new3carat.3 +
-243407.892  * new3depth.1 +
  -7133.9858 * new3depth.2 +
  -2296.6572 * new3depth.3 +
-140196.7508 * new3X.1 +
-270820.3095 * new3X.2 +
 -69450.8059 * new3X.3 +
 248620.0389 * new3Y.1 +
-470036.477  * new3Y.3 +
 374383.4801 * new3Z.1 +
1344389.2257 * new3Z.2 +
 179250.2749 * new3Z.3 +
 -39817.1002
Time taken to build model: 48.94 seconds
```