Electronic Systems Engineering
University of Regina

ENEL 890AK Project Report

Selecting Best Model for Given Data Set using Multiple Regression Techniques

Submitted by: Umar Ahsan (SID: 200353718)

Submitted to: Dr. Abdul Bais

Date: 10 August, 2016

Problem Statement:

In machine learning, regression is a fundamental technique that is used to analyze and predict the behavior of given data set. Since every data set has its own properties depending on its variables, dimensions and other related characteristics, therefore selection of best regression technique that can model the data accurately is an important part of machine learning. This report discusses usage of ridge, lasso and elasticnet regression techniques on housing data [1] in order to generate model that can accurately predict house price for future sales.

Background:

Regression is a modelling technique that investigates the link between dependent and independent variables of data in order to predict its nature. There are multiple known regression techniques like logistic, polynomial, Ridge, Lasso and ElasticNet. The best regression technique for a given data set can be selected through first modeling it using these techniques and then analyzing different parameters of fitted data like R-Square, Bayesian Information Criterion (BIC), and Residual Sum of Square (RSS) etc [2]. These values play a decisive role in selecting the best fitted model for that data. The designed model is also dependent on the number of data entries and features. The data set with large number of entries and multiple features can be modeled more accurately as compared to one with less features.

Goals:

In this project following regression techniques are implemented:

- Ridge Regression
- Lasso Regression
- ElasticNet Regression

The best technique is selected for modeling the given data set, Housing Data [1], through calculating RSS from each model and then selecting the model that has least RSS value. In addition to that, more features are created from existing features due to limited number of features present in the data set. The RSS from the new data set is calculated and compared with the previous models.

Methodology:

The GraphLab Create is the foundational framework that is used through Jupyter Notebook. The data set i.e. Housing data is selected for this project which is available through Coursera course i.e. Machine Learning Specialization [1]. In order to achieve above mentioned goals, following steps are taken:

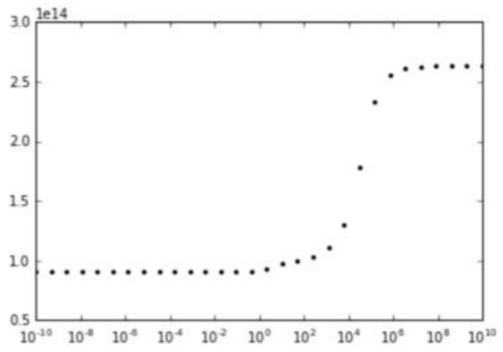
1. Useful libraries like Graphlab, Numpy are loaded along with data set.

2. The data set is divided into two subsets. 90 % is selected as training and validation data while 10 % is selected as testing data. Random split is implemented in order to get even distribution.
3. The training and validation subset in not further divided into separate subsets as k-fold cross validation technique is used throughout the regression techniques.
4. Housing data set is analyzed to understand all useful features that can be used for modeling.
5. Ridge regression technique is implemented using graphlab function, linear_regression.create (). This function takes following inputs: data set, L1/L2 penalty value, features for modeling and target output feature, in addition to multiple other optional inputs. In case of ridge regression, L1 penalty value should be set to '0' while L2 penalty can be inputted to model it.
6. K_fold_cross_validation_l2 function is defined that takes training and validation subset, value for k-fold cross validation, L2 penalty and features for modeling as inputs. This function generates ridge regression model and return RSS of that model after cross validation.
7. The L2 penalty value is iterated from $1x10^{-10}$ to $1x10^{10}$ and k_fold_cross_validation_l2 function is called for each value. The best regression model is selected that returns lowest RSS.
8. Lasso regression technique is implemented using graphlab function, linear_regression.create (). In case of lasso regression, L2 penalty value should be set to '0' while L1 penalty can be inputted to model it.
9. K_fold_cross_validation_l1 function is defined that takes training and validation subset, value for k-fold cross validation, L1 penalty and features for modeling as inputs. This function generates lasso regression model and return RSS of that model after cross validation.
10. The L1 penalty value is iterated from $1x10^{-10}$ to $1x10^{10}$ and k_fold_cross_validation_l1 function is called for each value. The best lasso model is selected that returns lowest RSS.
11. Elasticnet regression technique is implemented using graphlab function, linear_regression.create (). In case of elasticnet regression, both L1 and L2 penalty value should inputted to model it.
12. K_fold_cross_validation_l12 function is defined that takes training and validation subset, value for k-fold cross validation, L1 and L2 penalty and features for modeling as inputs. This function generates elasticnet regression model and return RSS of that model after cross validation.
13. The L1 penalty value is iterated from $1x10^{-10}$ to $1x10^{10}$, while for each L1 value, L2 value is also iterated from $1x10^{-10}$ to $1x10^{10}$. The k_fold_cross_validation_l12 function is called for each value. The best elasticnet model is selected that returns lowest RSS.
14. The linear_regression.create () function is also implemented with '0' L1 and L2 value in order to get RSS for model with no penalty.
15. The best ridge, lasso, elasticnet and no penalty models are tested over testing subset in order to select the model, using RSS, which best fits the data set.
16. New features are created using existing features and added to the existing data set.
17. Ridge, lasso, elasticnet regression techniques are again implemented as described above on new data set. The best models are selected for each respective technique.
18. The best ridge, lasso, elasticnet and no penalty models are tested over new testing subset in order to select the model, using RSS, which best fits the data set.
19. The results of both best models are compared in order to elaborate the importance of having extensive features for data modeling.

<u>Results:</u>

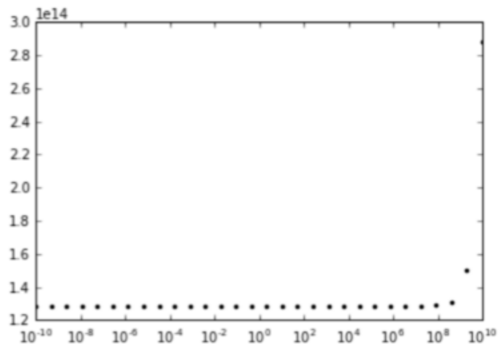
Here are the snapshots from the project along with their description:

```
The best L2 penalty is:  0.0923670857187
The lowest RSS corresponding to best L2 penalty is:  9.0830629956e+13
```



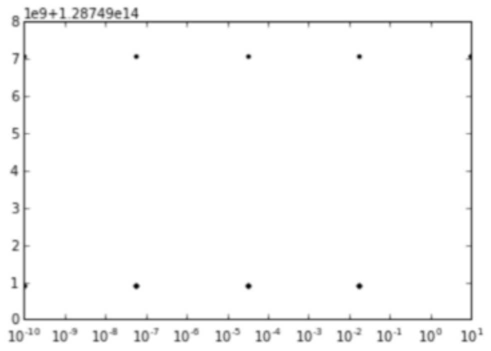
Ridge regression model. The best L2 penalty value is 0.09.

```
The best L1 penalty is:  1e-10
The lowest RSS corresponding to best L1 penalty is:  1.28749907777e+14
```



Lasso regression model. The best L1 penalty value is $1x10^{-10}$.

```
The best L1 penalty is:  1e-10
The best L2 penalty is:  1e-10
The lowest RSS corresponding to best L1/L2 penalty is:  1.28749907777e+14
```



Elasticnet regression model. The best L1 and L2 values are $1x10^{-10}$.

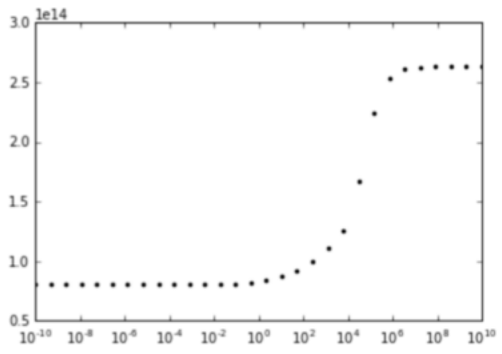
```
Best L2 Penalty:  0.0923670857187
Error corresponding to best L2 penalty:  9.0830629956e+13
Best L1 Penalty:  1e-10
Error corresponding to best L1 penalty:  1.28749907777e+14
Best L1 Penalty and L2 penalty respectively:  1e-10 1e-10
Error corresponding to best L1 and L2 penalty:  1.28749907777e+14
Value of No penalty:  0
Error corresponding to no penalty is:  9.08373912816e+13
```

Best models values along with their corresponding RSS value.

```
In [16]: # Ridge Regression
         # Here we set value of L2 penalty corresponding to best value that we calculated above
         model_trained_test_l2 = graphlab.linear_regression.create(training_and_validation_data, features= features_for_modeling ,
                                                      target='price', l2_penalty= best_l2_penalty,l1_penalty=0,feature_rescaling= True,
                                                  validation_set=None,verbose=False) # Training of model
         model_trained_l2 = model_trained_test_l2.predict(testing_data) # validating the model
         RSS_trained_l2 = RSS_data(model_trained_l2, testing_data['price']) # Comparing the values
         print "RSS value for testing data corresponding to best L2 Penalty: ", RSS_trained_l2

         # Lasso Regression
         # Here we set value of L1 penalty corresponding to best value that we calculated above
         model_trained_test_l1 = graphlab.linear_regression.create(training_and_validation_data, features= features_for_modeling ,
                                                      target='price', l2_penalty=0,l1_penalty=best_l2_penalty,max_iterations=100,
                                                  validation_set=None,verbose=False) # Training of model
         model_trained_l1 = model_trained_test_l1.predict(testing_data) # validating the model
         RSS_trained_l1 = RSS_data(model_trained_l1, testing_data['price'])  # Comparing the values
         print "RSS value for testing data corresponding to best L1 Penalty: ", RSS_trained_l1

         # ElasticNet Regression
         # Here we set value of L1 penalty corresponding to best value that we calculated above
         model_trained_test_l12 = graphlab.linear_regression.create(training_and_validation_data, features= features_for_modeling ,
                                                      target='price', l2_penalty=best_l12_penalty,l1_penalty=best_l11_penalty,
                                                  validation_set=None,verbose=False) # Training of model
         model_trained_l12 = model_trained_test_l12.predict(testing_data) # validating the model
         RSS_trained_l12 = RSS_data(model_trained_l12, testing_data['price']) # Comparing the values
         print "RSS value for testing data corresponding to best L1,L2 Penalty: ", RSS_trained_l12

         # No Regularization
         # Here we set value of L1 and L2 penalty to be zero
         model_trained_test_no = graphlab.linear_regression.create(training_and_validation_data, features= features_for_modeling ,
                                                      target='price', l2_penalty=0,l1_penalty=0,max_iterations=100,feature_rescaling= True,
                                                  validation_set=None,verbose=False) # Training of model
         model_trained_no = model_trained_test_no.predict(testing_data) # validating the model
         RSS_trained_no = RSS_data(model_trained_no, testing_data['price'])  # Comparing the values
         print "RSS value for testing data corresponding to no Penalty: ", RSS_trained_l12

         RSS value for testing data corresponding to best L2 Penalty:  1.03138268068e+14
         RSS value for testing data corresponding to best L1 Penalty:  1.12975857254e+14
         RSS value for testing data corresponding to best L1,L2 Penalty:  1.48515271857e+14
         RSS value for testing data corresponding to no Penalty:  1.48515271857e+14
```

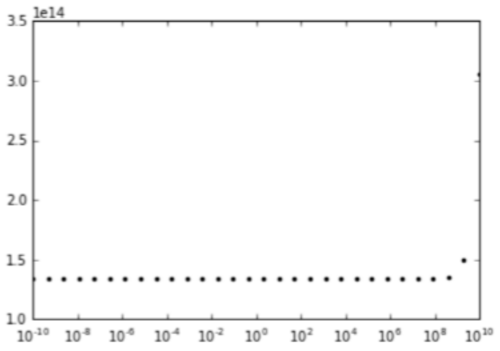
Testing data set RSS value for each best model.

```
The best L2 penalty with added features is:  0.0188739182214
The lowest RSS with added features corresponding to best L2 penalty is:  8.09215852421e+13
```



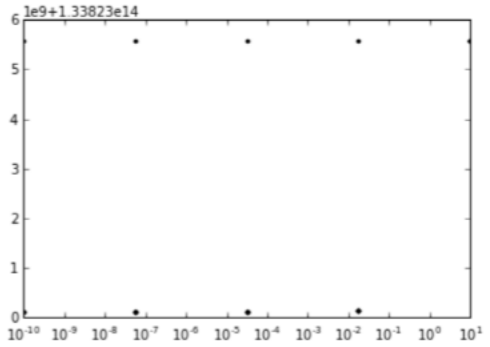
Ridge regression model with added features. The best L2 penalty value is 0.02.

```
The best L1 penalty with added features is:  1e-10
The lowest RSS with added features corresponding to best L1 penalty is:  1.33823131288e+14
```



Lasso regression model with added features. The best L1 penalty value is 1x10^{-10}.

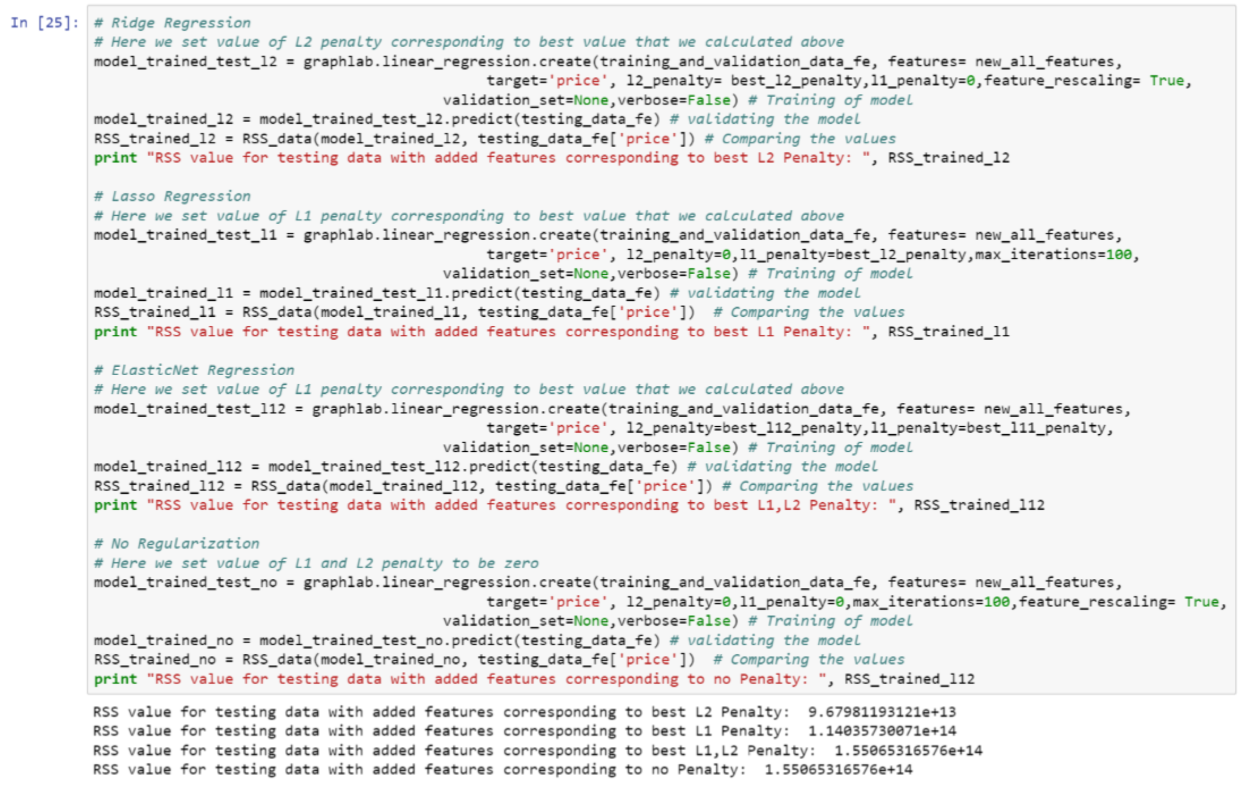
```
The best L1 penalty with Added Features is:  1e-10
The best L2 penalty with Added Features is:  1e-10
The lowest RSS with Added Features corresponding to best L1/L2 penalty is:  1.33823131288e+14
```



Elasticnet regression model with added features. The best L1 and L2 values are 1x10^{-10}.

```
Best L2 Penalty with added features:  0.0188739182214
Error with added features corresponding to best L2 penalty:  8.09215852421e+13
Best L1 Penalty with added features:  1e-10
Error with added features corresponding to best L1 penalty:  1.33823131288e+14
Best L1 Penalty and L2 penalty respectively with added features:  1e-10 1e-10
Error with added features corresponding to best L1 and L2 penalty:  1.33823131288e+14
Value of No penaltywith added features:  0
Error with added features corresponding to no penalty is:  8.09288926736e+13
```

Best models with added features values along with their corresponding RSS value.



```
In [25]: # Ridge Regression
         # Here we set value of L2 penalty corresponding to best value that we calculated above
         model_trained_test_l2 = graphlab.linear_regression.create(training_and_validation_data_fe, features= new_all_features,
                                                  target='price', l2_penalty= best_l2_penalty,l1_penalty=0,feature_rescaling= True,
                                             validation_set=None,verbose=False) # Training of model
         model_trained_l2 = model_trained_test_l2.predict(testing_data_fe) # validating the model
         RSS_trained_l2 = RSS_data(model_trained_l2, testing_data_fe['price']) # Comparing the values
         print "RSS value for testing data with added features corresponding to best L2 Penalty: ", RSS_trained_l2

         # Lasso Regression
         # Here we set value of L1 penalty corresponding to best value that we calculated above
         model_trained_test_l1 = graphlab.linear_regression.create(training_and_validation_data_fe, features= new_all_features,
                                                  target='price', l2_penalty=0,l1_penalty=best_l2_penalty,max_iterations=100,
                                             validation_set=None,verbose=False) # Training of model
         model_trained_l1 = model_trained_test_l1.predict(testing_data_fe) # validating the model
         RSS_trained_l1 = RSS_data(model_trained_l1, testing_data_fe['price'])  # Comparing the values
         print "RSS value for testing data with added features corresponding to best L1 Penalty: ", RSS_trained_l1

         # ElasticNet Regression
         # Here we set value of L1 penalty corresponding to best value that we calculated above
         model_trained_test_l12 = graphlab.linear_regression.create(training_and_validation_data_fe, features= new_all_features,
                                                  target='price', l2_penalty=best_l12_penalty,l1_penalty=best_l11_penalty,
                                             validation_set=None,verbose=False) # Training of model
         model_trained_l12 = model_trained_test_l12.predict(testing_data_fe) # validating the model
         RSS_trained_l12 = RSS_data(model_trained_l12, testing_data_fe['price']) # Comparing the values
         print "RSS value for testing data with added features corresponding to best L1,L2 Penalty: ", RSS_trained_l12

         # No Regularization
         # Here we set value of L1 and L2 penalty to be zero
         model_trained_test_no = graphlab.linear_regression.create(training_and_validation_data_fe, features= new_all_features,
                                                  target='price', l2_penalty=0,l1_penalty=0,max_iterations=100,feature_rescaling= True,
                                             validation_set=None,verbose=False) # Training of model
         model_trained_no = model_trained_test_no.predict(testing_data_fe) # validating the model
         RSS_trained_no = RSS_data(model_trained_no, testing_data_fe['price'])  # Comparing the values
         print "RSS value for testing data with added features corresponding to no Penalty: ", RSS_trained_l12

         RSS value for testing data with added features corresponding to best L2 Penalty:  9.67981193121e+13
         RSS value for testing data with added features corresponding to best L1 Penalty:  1.14035730071e+14
         RSS value for testing data with added features corresponding to best L1,L2 Penalty:  1.55065316576e+14
         RSS value for testing data with added features corresponding to no Penalty:  1.55065316576e+14
```

Testing data set, with added features, RSS value for each best model.

<u>Analysis:</u>

From the above results following points can be inferred:

1. Ridge regression model has the lowest RSS value for training and validation subset i.e. 9.0830629956e+13.
2. Ridge regression model has also lowest RSS value for testing subset i.e. 1.03138268068e+14. The lasso has 1.12975857254e+14, elasticnet has 1.48515271857e+14 and without any penalty has 1.48515271857e+14 RSS values which are larger than RSS value for ridge regression model.
3. After adding features in the data set the ridge regression has still lowest RSS value for training and validation subset i.e. 8.09215852421e+13.
4. After adding features, ridge regression model has still lowest RSS value for testing subset i.e. 9.67981193121e+13. The lasso has 1.14035730071e+14, elasticnet has 1.55065316576e+14 and without any penalty has 1.55065316576e+14 RSS values which are larger than RSS value for ridge regression model.
5. The ridge regression model has lowest RSS for testing data subset. The RSS value is 1.03138268068e+14 for model without any added features and 9.67981193121e+13 for model after adding additional features. This shows better modeling of data set due to larger available features.

Conclusion:

Ridge regression technique is best for modeling the given housing data set as it shows the least RSS value. In addition to that, the ridge regression shows even better modeling of data set after it is trained over additional features that are created from existing ones.

References:

[1] https://www.coursera.org/specializations/machine-learning

[2] https://www.analyticsvidhya.com/blog/2015/08/comprehensive-guide-regression/