Ames, IA Housing Prize Prediction Project Report

Final project report for OIS 6850

STUDENT GROUP MEMBERS –
SAGAR VASEKAR
SHEKHAR BHOSALE
SHRIKANT TAMBE

UNDER GUIDANCE OF –
DR. JEFF WEBB
PROFESSOR, UNIVERSITY OF UTAH

***Summary*—This document is a summary report of prize prediction of houses located in Ames, IA using various regression techniques we learned doing class OIS 6850 – Statistical and predictive analysis. The various components of this reports are introduction, data model, and modeling, model performance and conclusion.**

INTRODUCTION

I. WHAT ARE WE TRYING TO ACHIEVE?

We are given data set of house prices from Ames, Iowa and our task is to find out which are the most influential factors those affect final house prize and then apply same model to predict housing prices given set of new observations. There are approximately 80 explanatory variables describing (almost) every aspect of residential homes. Given data is split in 50% for train and 50% for test sets. Test set contains all the predictor variables in train set excluding target variable, SalePrice. We would be referring training data as in sample data to build, train and evaluate model and testing data as out of sample data for evaluating model performance for new data.

We have 20 continuous variables, 14 discrete variables, 24 nominal variables and 23 ordinal variables whose value is one from its levels. We have outcome variable SalePrice which shows known house prize in train data set and we have to analyze train data set, train statistical regression model for prediction using train set, evaluate it and finally apply it to test data set and predict SalePrice for each instance.

Ultimately, we are to analyze and explore input data of house prices to predict estimated sale price of each home given different dependent factors. We will talk about data, regression models features selected, discussions on generalizability of model with low variance while remaining cautious to the possibilities of overfitting.

II. MORE ABOUT DATA

So, we have around 80 variables remained that were directly related to property sales. Although too many to describe here individually, one may refer the documentation[1] for more details. These 80 variables focus on the quantity and quality of many physical attributes of the property. Most of the variables are exactly the type of information that a typical home buyer would want to know about a potential property (e.g. What is the size of property, when was it built? How many bedrooms? How many bathrooms are there? etc.)

There are many categorical variables (23 nominal, 23 ordinal) associated with this data set. They range from 2 to 28 classes with the smallest being STREET (gravel or paved) and the largest being NEIGHBORHOOD. The nominal variables typically identify various types of dwellings, garages, materials, and environmental conditions while the ordinal variables typically rate various items within the property. There are two variables (PID and

NEIGHBORHOOD) that may be of special interest to users of the data set. PID is the Parcel Identification Number assigned to each property within the Ames Assessor's system. The typical record will indicate the values for characteristics commonly quoted on most home flyers and will include a picture of the property. The NEIGHBORHOOD variable, typically of little interest other than to model the location effect, may be of more relevance when used with the map included in the supplementary materials.

DATA MODELING AND CLEANING

I DATA CLEANING PART ONE

Before we begin to build the regression model, it is imperative to handle missing or bad data. So that it does not impact model negatively and model is able to perform better. Handling Missing Data:

We gave descent time to explore and cleanse data properly so that we work on clean base. While exploring dataset, we observed that a few features (variables) like MasVnrType, Electrical, LotFrontage, MasVnrArea and GarageYrBlt and so many others had missing values denoted by 'NA'. Below figure 1 shows presence of NAs variable wise.

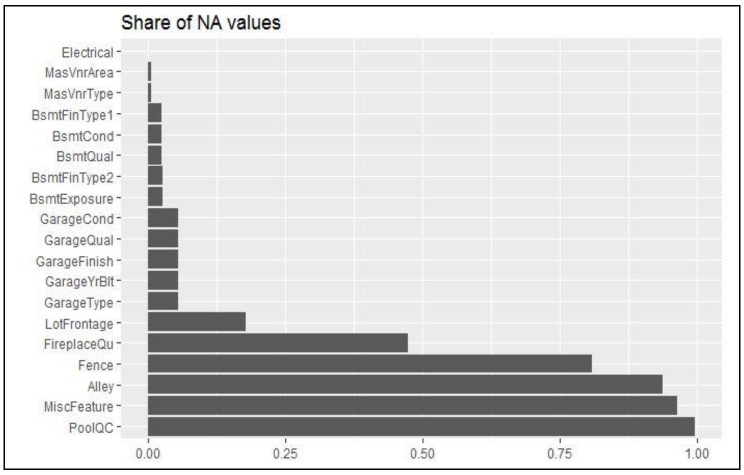


Figure 1: Missing values share information. We can see that PoolQC has highest missing values while Electrical has least numbers.

There are two ways how this data is handled:

- *Categorical Variables:* We have tried imputing missing data using rpart[2] wherever relevant. Since many variables, which are not applicable to an observation, have been assigned NA which can be misleading, it has been substituted by a new category 'None'.
- *Continuous variables:* Since we have continuous variables having missing values as NAs as well, we are using rpart[2] for such instances and predicting those values given other thousands of observations present. This is the one of the better way of handing NAs rather than just putting zero or mean values!

II DATA CLEANING PART TWO

There are five observations that an instructor may wish to remove from the data set. Figure 2 will quickly indicate these points. Three of them are true outliers (Partial Sales that likely don't represent actual market values) and two of them are simply unusual sales (very large houses priced relatively appropriately). We removed any houses with more than 4000 square feet from the data set (which eliminates these five unusual observations).

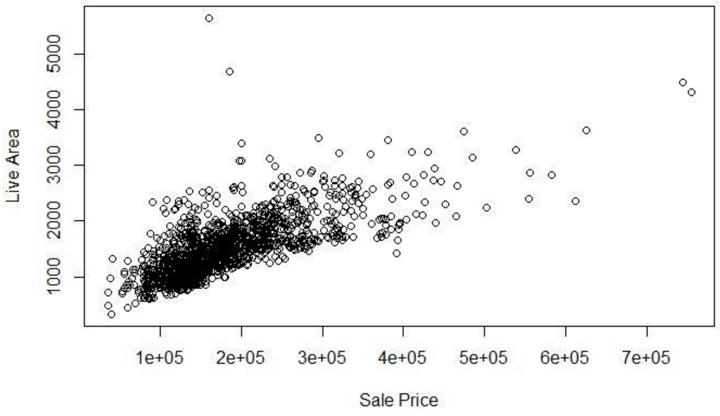


Figure 2: A plot of SALE PRICE versus GR LIV AREA will quickly indicate presence of outliers here.

We repeated this process for the other 20 most predictive explanatory variables (as identified in model 3) such as OverallQual, KitchenQual and YearBuilt. other outliers were identified. we then built a linear regression model using this trimmed training set and assessed its performance in the usual way.

III DATA MODELING

In this section, we will briefly overview the building and training regression various models. We tried different models with specific set of influential predictors. Although, not all of them were best or up to the mark, they helped us identify defined set of predictors which influence the target SalesPrice the most. We started with all predictors and then came down to the list of specific selected predictors.

After a while having reached a plateau in beta model development we came across various kernels to identify any areas of improvement. One kernel[3] mentioned that removing outliers from the training set would improve model performance. Therefore, we went ahead and tried that. In data cleaning part II.

MODEL AND MODEL DEVELOPMENT

After performing more cleaning steps, we built few set of models again and evaluated those.

- In model 1, we used cleansed data and with *liner regression method* we trained model. We are going to use 5-fold cross validation throughout this project for each model. There is no specific reason for selecting 5 folds but considering dataset size and model training time, we are keeping it to five. You may read more about cross validation on Model Performance Section of this document. Performance wise, this liner model performed fine but not so great. RMSE of this model is produced as 0.2023 and R-squared is 0.79. Ideally, we would want R-squared more than 0.90. You may read more about RMSE and R-Squared in Model Performance section of this document.
- In model 2, we tried different regression method *Random Forest*. Regression is a process that maps input variables to specific output variable and helps us predict values which might not be present. Regression methods helps us perform this task on given data. Random forests or random decision forests are an ensemble learning method for classification, regression, and other tasks, that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of overfitting to their training set. We used e1071[4] package to train model using Random forest method. This method used all 81 predictors and performed well than previous model. We have 0.1431 RMSE and 0.88 R-squared.
- In Model 3, we used further improved random forest method by changing it’s one model parameter ‘tuneLength’. We set this model for *two mtry values*. This is nothing but Number of variables randomly sampled as candidates at each split. By doing this we are further tuning the model to consider more sampling of predictors. Sometimes it produces desired effect. Performance wise this model did not perform any better than previous model 2 but it was relatively same. Model 3 has 0.1410 RMSE and 0.87 R-squared.
- In Model 4, we used same regression method as Model 2 and 3 but only with limited set of predictors. Model 2 and 3 used all predictors to predict value of SalePrice then we decided to try with top 20 predictors. We used ranger to analyze performance of all predictors and selected top 20. As a result we ended up selecting OverallQual, GrLivArea,TotalBsmtSF, GarageArea, GarageCars,X1stFlrSF, YearBuilt, ExterQual, BsmtFinSF1, FullBath, KitchenQual, LotArea, Fireplaces, FireplaceQu,YearRemodAdd,GarageYrBlt, X2ndFlrSF,TotRmsAbvGrd,MasVnrArea and LotFrontage. We kept mtry and other model input parameters to default as model 2. This model performed good as previous models but not much better. They were slightly similar. RMSE of Model 4 is 0.1419 and R-squared is 0.88

- Looking at no significant improvements in prior models we decided to try Support *vector machine method* in model 5. Having heard the hype surrounding support vector machines we thought we would try one in this project. In machine learning, support vector machines are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier. An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall. However, model 5 did not perform better than any model in fact performed poorly. It's RMSE is 10.04 and R-Squared is 0.04. Apparently, this one was worst of all. It is likely that further parameter tuning and data processing would be required for the support vector machine to perform better. Because we were only trying this algorithm out of curiosity we did not experiment with it any further.
- For Model 6, again, given the hype surrounding it and our own curiosity, we tried building *artificial neural network.* Artificial neural networks (ANNs) or connectionist systems are a computational model which is based on a large collection of simple neural units However, once again, it did not perform better than the linear model. It has very poor RMSE (11.02) and R-squared (0.09). Like in the case of the support vector machine this is likely due to further parameter tuning and data processing being required to maximize its performance (different machine learning algorithms have different requirements for performance maximization). Because we were only trying this algorithm out of curiosity and it has slow and resources consuming method we did not experiment with it any further.
- In model 7, we tried using KNN method using all 81 predictors. *k-nearest neighbor (KNN)* classification for test set from training set. For each row of the test set, the k nearest (in Euclidean distance) training set vectors are found, and the classification is decided by majority vote, with ties broken at random. If there are ties for the kth nearest vector, all candidates are included in the vote. With K as 9 it gave bad kind of performance. It's RMSE was 0.2621 and R-squared was 0.56. This model performed poorer than our very first model 1, linear regression.
- In model 8, we used *gradient boosting machines method*. While doing some research on internet we found about boosting methods. Basically, Boosting is a sequential technique which works on the principle of ensemble. It combines a set of weak learners and delivers improved prediction accuracy. At any instant t, the model outcomes are weighed based on the outcomes of previous instant t-1. The outcomes predicted correctly are given a lower weight and the ones miss-classified are weighted higher. This technique is followed for a classification problem while a similar technique is used for regression. Good news was this model worked better than all previous models. We customized its inputs to consider 50, 100, 150 and 200 trees and changes to various other tuning parameters. 50 trees gave best performance of 0.89 R-squared and 0.1324 RMSE.
- Considering positive results of model 8, we decided to groom it better. Model 9 used extreme gradient boosting methods to train the model. The *extreme gradient boosting algorithm* works much the same as the stochastic gradient boosting machine, but more aggressively guards against overfitting. It's far easier to use extreme gradient boosting in the caret package rather than the xgboost package because the latter requires

the training and test sets to be converted to sparse matrices using the concept of one-hot encoding. We used all available explanatory variables and three mtry values. The final values used for the model were nrounds = 100, lambda =0.4, alpha = 0.2 and eta = 0.3.100. It gave us some good numbers. We had 0.1300 RMSE and 0.89 R-Squared. However, we could not improve this model any better.

- For Model 10, we applied *regularized linear regression*. We used glmnet[] method to achieve this task. glmnet uses a combination of lasso regression (penalizing the number of non-zero coefficients) and ridge regression (penalizing the absolute magnitude of each coefficient) to prevent overfitting when building a generalized linear model. Performance wise this method helped to good extent and gave us better performance than other models. With final alpha as 0.1 and lambda 0.02 we had 0.1131 RMSE and 0.92 R-squared! This model was the highest performing model of all. Finally hunt for good was satisfied and we had some fruitful results.

This way we explored various models with data tuning and other adjustments and we decided to stay with our best-performing model of all at Model 10.

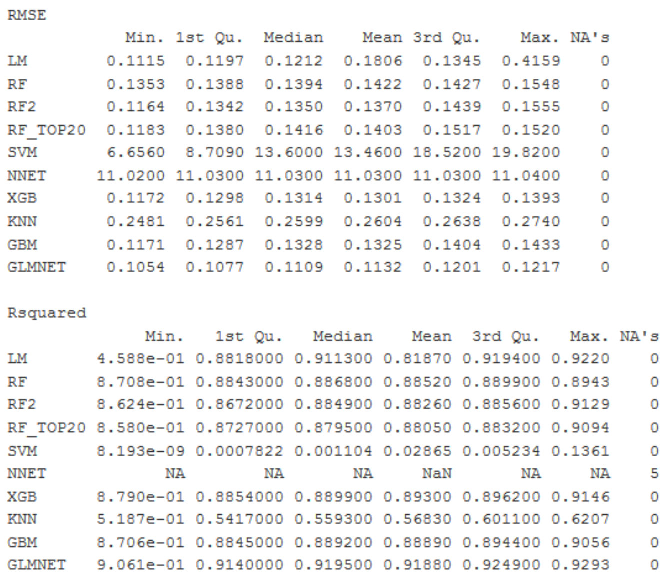
MODEL PERFORMANCE

I MODELS COMPARISON

To effectively gauge the performance of any model, we ought to check two major performances. In-sample performance and out-of-sample performance. To evaluate models in-sample performance we use various matrices to indicate its value and ability to predict values correctly. First RMSE - It is root mean squared error. Then we have R-Squared; it represents the variation in the outcome variable explained by the model as a proportion of the total variation. It is the squared correlation between the outcome and predictor variables. Better the R-Squared and less the RMSE, better the model is! We also have many other evaluations like AIC, LRT etc. but we are going to rely on most wildly matrices RMSE and R-Squared. For Out-of-sample, we test model for new data which was not present during model training. This is new data and includes new observations with same input variables.

SR.	Model	RMSE	R2*
1	Linear Regression	0.2023	0.79
2	Random Forest	0.1431	0.88
3	Random Forest with 2 mtry	0.1410	0.87
4	RF with specified predictors	0.1419	0.88
5	SVM	10.04	0.04
6	ANN	11.02	0.09
7	KNN	0.2621	0.56
8	GBM	0.1324	0.89
9	XGBOOST	0.1300	0.89
10	GLMNET	0.1131	0.92

Table 1: Model Performances. If we compare performances of all models, we could see that model 10 with gives best performance. (R2= R-Squared)*



```
RMSE
            Min. 1st Qu.  Median    Mean 3rd Qu.    Max. NA's
LM        0.1115  0.1197  0.1212  0.1806  0.1345  0.4159    0
RF        0.1353  0.1388  0.1394  0.1422  0.1427  0.1548    0
RF2       0.1164  0.1342  0.1350  0.1370  0.1439  0.1555    0
RF_TOP20  0.1183  0.1380  0.1416  0.1403  0.1517  0.1520    0
SVM       6.6560  8.7090 13.6000 13.4600 18.5200 19.8200    0
NNET     11.0200 11.0300 11.0300 11.0300 11.0300 11.0400    0
XGB       0.1172  0.1298  0.1314  0.1301  0.1324  0.1393    0
KNN       0.2481  0.2561  0.2599  0.2604  0.2638  0.2740    0
GBM       0.1171  0.1287  0.1328  0.1325  0.1404  0.1433    0
GLMNET    0.1054  0.1077  0.1109  0.1132  0.1201  0.1217    0

Rsquared
              Min.   1st Qu.   Median    Mean  3rd Qu.   Max. NA's
LM       4.588e-01 0.8818000 0.911300 0.81870 0.919400 0.9220    0
RF       8.708e-01 0.8843000 0.886800 0.88520 0.889900 0.8943    0
RF2      8.624e-01 0.8672000 0.884900 0.88260 0.885600 0.9129    0
RF_TOP20 8.580e-01 0.8727000 0.879500 0.88050 0.883200 0.9094    0
SVM      8.193e-09 0.0007822 0.001104 0.02865 0.005234 0.1361    0
NNET            NA        NA       NA     NaN       NA     NA    5
XGB      8.790e-01 0.8854000 0.889900 0.89300 0.896200 0.9146    0
KNN      5.187e-01 0.5417000 0.559300 0.56830 0.601100 0.6207    0
GBM      8.706e-01 0.8845000 0.889200 0.88890 0.894400 0.9056    0
GLMNET   9.061e-01 0.9140000 0.919500 0.91880 0.924900 0.9293    0
```

Figure 3 Models comparison table generated in R-code

We may treat portion of data as in-sample and out-of-sample data but in our case, we have given two clear slices of data. We used first one to train the data and we used second one to just test the model. This slicing could be done in n-folds as well. This way, we are dividing data into n-part and train model using n-1 sets and test using one set. So that when data is scattered and not evenly distributed we are testing every corner of data keeping one mutually exclusive part aside. At the end, we take the mean of these 'n' folds testing and this entire process is called as "Cross Validation".

Talking about our model performances, table in the Table 1 lists them all together. Figure 3 shows comparison generated using R-code and Figure 4 shows RMSE comparison view of all models.

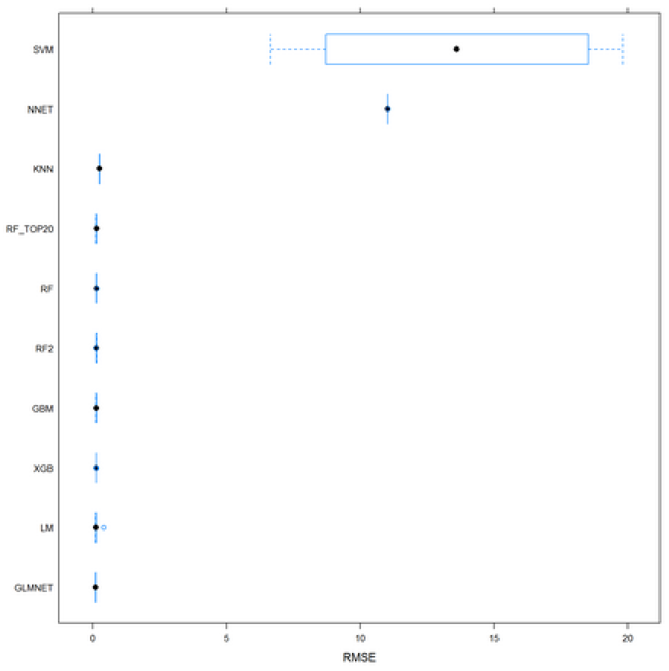


Figure 4 RMSE comparison chart of all models

We can observe results in comparative view and seems we could develop some good models with Random Forest, BGM, CGBOOST and GLMNET methods. high 50465 RMSE and low 0.68 R2. This is bad model. However, clear winner is GLMNET model which is described in model 10.

II MORE ABOUT FINAL MODEL

As indicated in model number 10 we used regularized regression method in train this model. Let us see what exactly this method does in brief.

Regularized regression techniques for linear regression have been created the last few ten years to reduce the flaws of ordinary least squares regression about predictions. We have five major methods that use penalty-based (regularization) shrinkage to handle Oxazolones and Oxazole's derivatives descriptor dataset with far more

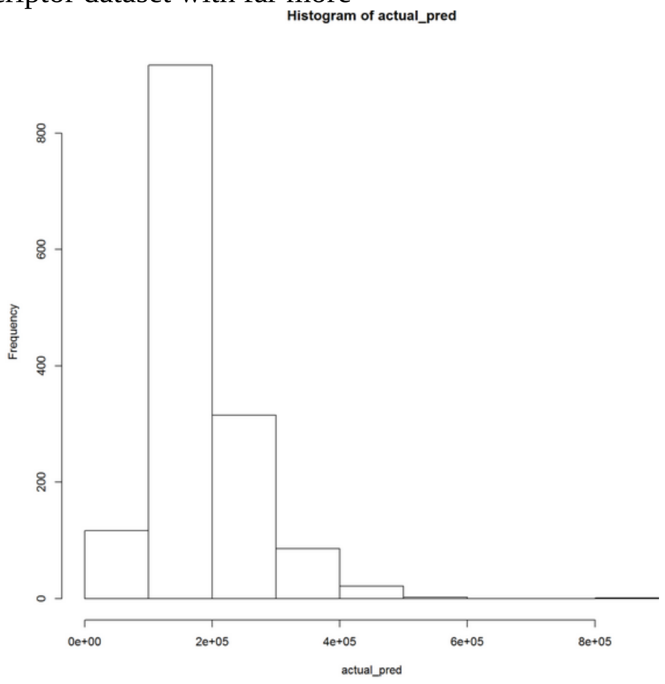


Figure 5: Histogram of SalePrice prediction. We can see that it is like actual SalePrice!

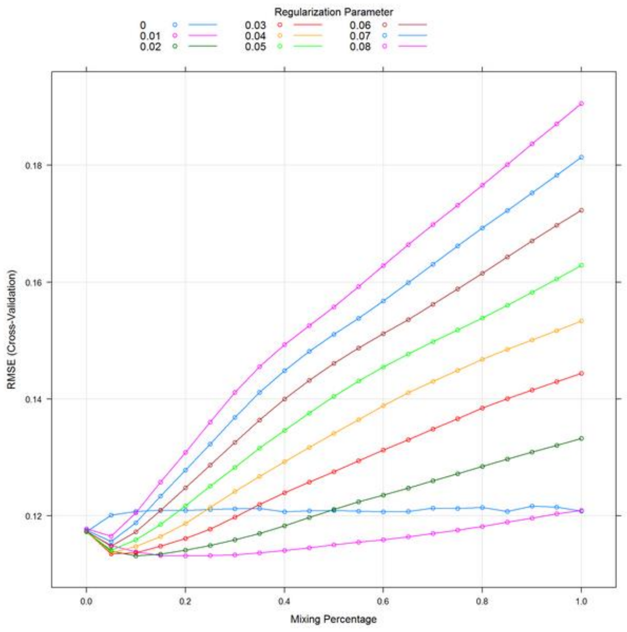


Figure 6 Regularized liner model plot

predictors than observations. The lasso, ridge, elasticnet, lars and relaxed lasso further possess the desirable property that they simultaneously select relevant predictive descriptors and optimally estimate their effects. Here, we comparatively evaluate the performance of five regularized regression methods. Predictive accuracy was evaluated using the RMSE and square of usual correlation between redactors and observed mean inhibitory concentration of antitubercular activity (R square). All five regularized regression models were able to produce feasible models and efficient capturing the linearity in the data. The elastic net and lars had similar accuracies as well as lasso and relaxed lasso had similar accuracies but outperformed ridge regression in terms of the RMSE and R square metrics

In our case, we used lasso and ridge methods to train the model and results are exceptional! Figure 5 shows predicted value histogram. It is produced very like actual SalePrice histogram. Also, in Figure 6, we plotted regularized liner model and its tuning values and its impact on RMSE. Finally we retained the parameters which gave us best RMSE and R-squared.

III CROSS VALIDATION

We have trained all models using 5-fold cross validation and all RMSE and R-squared reported are mean values of 5 folds.

IV KAGGLE LEADERBOARD STATISTICS

Since we have used cross validation for training model, to measure the test performance we have used cross evaluation. This way we have test RMSE as 0.1059 and R-Squared as 0.92

Our Kaggle score is 0.1211 and Kaggle team rank is 687!

CONCLUSION

This way we've achieved our goal of doing hour price prediction given Ames, IA dataset. We submitted our results to online competition portal and we could secure fair rand in there.

REFERENCES

All web references to more details:

[1] https://www.kaggle.com/c/house-prices-advanced-regression-techniques/data

[2] https://cran.r-project.org/web/packages/rpart/rpart.pdf

[3] https://www.kaggle.com/sidraina89/house-prices-advanced-regression-techniques/regularized-regression-housing-pricing

[4] https://cran.r-project.org/web/packages/e1071/e1071.pdf

[5] https://www.rdocumentation.org/packages/glmnet/versions/2.0-5/topics/glmnet