Implementing an efficient approach to studying and analyzing American colleges admissions
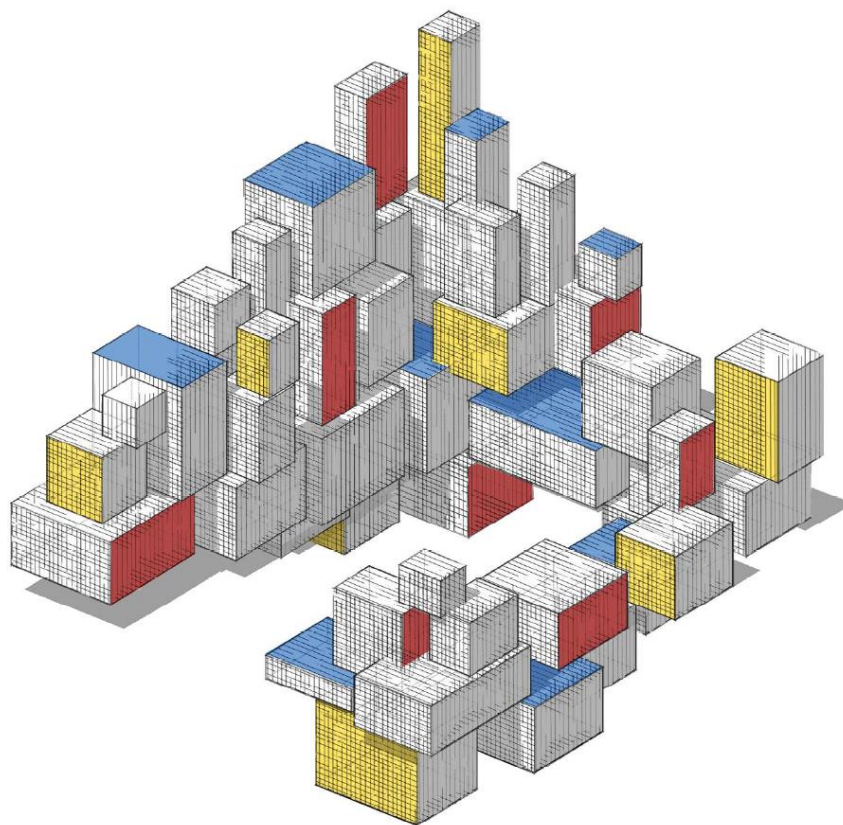
Mohamad Parichehreh[1] , Dr. Ali Ghanbari[2]

15th May 2022

[1] Mohamad Paricherheh Teroujeni, Graduate student of Mazandaran University of Science and Technology, mhmdparicherheh@gmail.com

2 Dr. Ali Ghanbari, Faculty of Mazandaran University of Science and Technology, ali.ghanbari@mazust.ac.ir

i

contents

ث

Chapter 1

1.1. Project definition

In this project, we intend to study, analyze and review data related to students in American colleges in 1995 using efficient methods and building optimal models. in addition to finding the parameters that are most effective in predicting the purpose variable which is the variable Apps in this project, predicting the total number of applications that students send for admission to the university also matters.

1.2. Existing assumptions, structures and challenges

As we have seen, the college.csv dataset is a ready-made database and we only use it, So we have had no role in observing and collecting it. This is the cause of a challenge not to have a great first understanding and a good perspective of data; As a result, we need to first process the data.

Also, using a prepared dataset and collecting data by someone else has made us unaware of the accuracy and correctness of the data and it is possible and common to occur a slight error the data is under the supervision of measuring and collecting.

To solve this problem, we have no choice except assume that the impeccability of data collection has been observed, or if an error has occurred, it has been so small to create a significant problem on the results.

But our main challenge is not data impeccability! Data impeccability can be important, but on top of that, the models we are going to build and implement on data matter much more. These models are mostly made of linear regression. Obviously, we will certainly have acute problems in data that are nonlinear and have a very low correlation.

Chapter 2

2.1. History

Data science is an interdisciplinary knowledge about extracting knowledge and awareness from data sets and information. Data science has emerged from a combination of different topics which is based on existing principles and methods in various scientific fields. Some of these areas are: mathematics, statistics, computer science, data engineering, pattern recognition and etc. the purpose of this science is to extract the concept of data and produce a data-driven product.

In the 2012 article "Data Science: The Most Attractive Job of the 21st Century," Thomas Dunport and DJ Patel define data scientists as: Those who know how to answer business questions from a large, unstructured body of information. Stanton defines data science in 2013 as follows: Data science is an emerging discipline that collects, prepares, analyzes, visualizes, manages, and maintains large volumes of information. Driscoll defines data science in 2014 as follows: Data science is data civil engineering. The data science expert has a practical knowledge of data and tools, as well as a theoretical understanding that determines what is scientifically possible.

The term data science has been around for more than a decade. William Cleveland was the first to coin the term data science in 2001. In his article "Data Science: A Plan to Expand the Technical Aspects of Statistics," he suggested that data science has to be recognized as an independent field. Cleveland linked the new field to computer science and data mining. He believed that the benefits of using a data analyst were limited. Because computer engineers have little knowledge of how to work with data and the computational knowledge of statisticians is also limited; therefore, combining these two groups can lead us to many innovations. Data science departments should have professors who can combine data knowledge with computational knowledge.

Although the term data science is a new term which has been around for many years. Napoleon Bonaparte used mathematical models to make decisions on the battlefield. These models were developed by mathematicians.

The past decades have seen a revolution in information technology. Routine collection of systematically generated data is now commonplace. Databases with hundreds of fields (variables), and billions of records (observations) are no longer considered unusual. This problem is commonly used for classical data analysis methods due to computer memory constraints and computational costs (eg time).

In this paper, we propose a different method of analyzing different regression distributions that is suitable for modeling large data sets. A good theoretical justification for the proposed method is presented and the experimental performance is discussed based on an extensive simulation study.

The importance of machine learning (ML) continues for many organizations in almost every field. Some examples of practical machine learning applications are:

- Predict the possibility of the patient returning to the hospital (readmission) within 30 days after discharge.

- Customer segmentation based on common features or buying behavior for targeted marketing.

- Predict coupon redemption rates for a given marketing campaign.

- Predicting customer decline so that the organization can take preventive intervention.

And many more!

Basically, these tasks are all about learning from the data. To deal with each scenario, we can use a specific set of features to teach an algorithm and extract insights. These algorithms or learners can be classified according to the amount and type of supervision required during the training. The two main groups that this book focuses on are: supervised learners who make predictive models and unsupervised learners who make descriptive models. Which type you use should depend on the educational work you hope to do.

2.2. Supervised learning

A forecasting model is used for tasks that involve predicting a specific output (or goal) using other variables (or attributes) in the data set. Or, as Cohen and Johnson (2013: 26: 2) put it, forecasting modeling is the process of developing a mathematical tool or model that produces an accurate forecast." In a predictive model, the learning algorithm tries to discover and model the relationships between the target variable (the predicted variable) and other attributes (known as the predictor variables). Examples of forecasting modeling are:

- Use customer features to predict the likelihood of a customer falling in the next 6 weeks;

- Use home features to forecast sales prices;

- Use employee characteristics to predict the likelihood of force adjustment of each employee;

- Use of patient characteristics and symptoms to predict the risk of readmission;

- Use production features to predict market entry time;

Each of these examples has a defined learning task. Each intends to use the (X) attribute to predict an outcome measure (Y).

Throughout this text we will use different terms instead of each other:

(X) : "predictor variable", "independent variable", "feature", "predictor", "attribute"

(Y) : "target variable", "dependent variable", "response", "outcome measurement"

In particular, given a set of data, the learning algorithm tries to optimize a function (algorithmic steps) to find a combination of attribute values that results in a predicted value that is as close as possible to the actual target output.

In supervised learning, the training data you feed into the algorithm includes the target values. As a result, solutions can be used to help monitor the training process to find the optimal algorithm parameters.

Most supervised learning issues can be placed in one of two categories of regression or classification, which we will discuss below.

2.3. Regression problems

When the goal of our supervised learning is to predict a numerical outcome, we refer to it as a regression problem (do not confuse it with linear regression modeling). Regression problems revolve around the predicting output that falls on a continuum. In the examples above, predicting home sales prices and time to market reflect a regression problem because the output is numeric and continuous. This means, given the combination of predictor values, the response value could fall anywhere along some continuous spectrum (e.g., the predicted sales price of a particular home could be between \$80,000 and \$755,000).
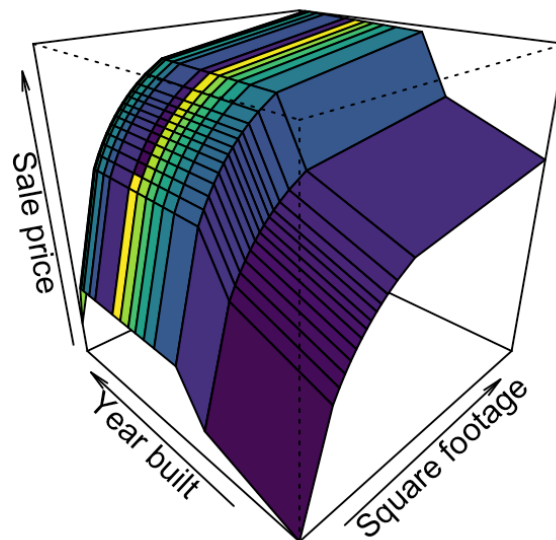

Figure 2.1

Illustrates average home sales prices as a function of two home features: year built and total square footage. Depending on the combination of these two features, the expected home sales price could fall anywhere along a plane.

2.4. Data sharing

The main goal of the machine learning process is to find an algorithm that more accurately predicts future values based on a set of features. In other words, we want an algorithm that not only fits in well with our past data, but more importantly, an algorithm that accurately predicts a future outcome. This is called the generalizability of our algorithm. How we "spend" our data will help us understand how well our algorithm generalizes to unseen data.

To provide an accurate understanding of the generalizability of our final optimal model, we can split our data into training and test data sets:

- **Training set:** these data are used to develop feature sets, train our algorithms, tune hyperparameters, compare models, and all of the other activities required to choose a final model (e.g., the model we want to put into production).
- **Test set:** having chosen a final model, these data are used to estimate an unbiased assessment of the model's performance, which we refer to as the generalization error.

It is critical that the test set not be used prior to selecting your final model. Assessing results on the test set prior to final model selection biases the model selection process since the testing data will have become part of the model development process.

Given a fixed amount of data, typical recommendations for splitting your data into training-test splits include 60% (training)–40% (testing), 70%–30%, or 80%–20%. Generally speaking, these are appropriate guidelines to follow; however, it is good to keep the following points in mind:

- Spending too much in training (e.g., >80%) won't allow us to get a good assessment of predictive performance. We may find a model that fits the training data very well, but is not generalizable (overfitting).
- Sometimes too much spent in testing (>40%) won't allow us to get a good assessment of model parameters.

2.5. Linear regression

Linear regression, a staple of classical statistical modeling, is one of the simplest algorithms for doing supervised learning. Though it may seem somewhat dull compared to some of the more modern statistical learning approaches described in later chapters, linear regression is still a useful and widely applied statistical learning method. Moreover, it serves as a good starting point for more advanced approaches

2.6. Simple linear regression

Pearson's correlation coefficient is often used to quantify the strength of the linear association between two continuous variables. *Simple linear regression* (SLR) assumes that the statistical relationship between two continuous variables is (at least approximately) linear:

$$Y_i = \beta_0 + \beta_1 X_i + \epsilon_i, \quad \text{for } i = 1, 2, \ldots, n,$$

At this point, we assume that the errors are usually distributed with a mean of zero and a constant variance.

2.7. Estimation

Ideally, we want to estimate a line that gives us the "best fitting" line. This form of linear regression is often referred to as Ordinary Least Squares (OLS) regression. There are multiple ways to measure "best fitting" line by minimizing the residual sum of squares (RSS).

One of the drawbacks of the LS method in linear regression is that it provides only estimates of coefficients and does not provide estimates of error variance. LS also has no assumptions about random errors.

2.8. Assessing model accuracy

But the question remains, which model is the "best"? To answer this question, we must define what we mean by "best". In this case, we will use the RMSE metric and cross-validation to determine the "best" model. We can use the caret::train () function to teach a linear model (ie the method = "lm") using

cross-validation (or a variety of other validation methods). In practice, a number of factors must be considered in determining a "best" model (e.g., time constraints, model production cost, forecast accuracy, etc.).

2.9. Model Concerns

As previously stated, linear regression has been a popular modeling tool due to the ease of interpreting the coefficients. However, linear regression makes several strong assumptions that are often violated as we include more predictors in our model. Violation of these assumptions can lead to flawed interpretation of the coefficients and prediction results.

1. Linear relationship: Linear regression assumes a linear relationship between the predictor and the response variable. However, nonlinear relationships can be made linear (or near-linear) by applying transformations to the response and/or predictors; However, there is still some non-linearity for older homes.
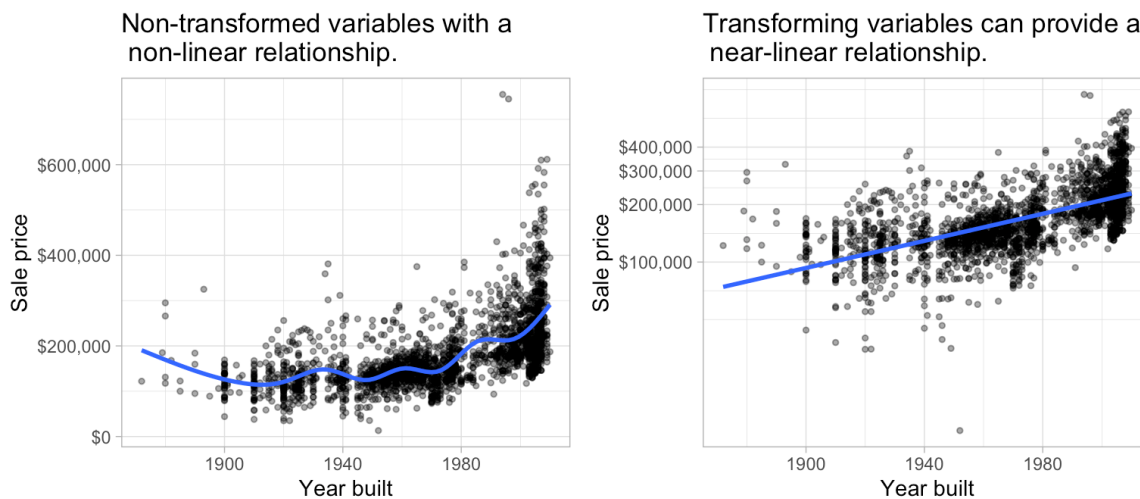

Figure 2.2 Nonlinear relationships can often be altered to near-linear by applying a transformation to the variable(s).

2. Constant variance among residuals: Linear regression assumes the variance among error terms are constant (this assumption is referred to as homoscedasticity). If the error variance is not constant, the p-values and confidence intervals for the coefficients will be invalid. Similar to the linear relationship assumption, non-constant variance can often be resolved with variable transformations or by including additional predictors.
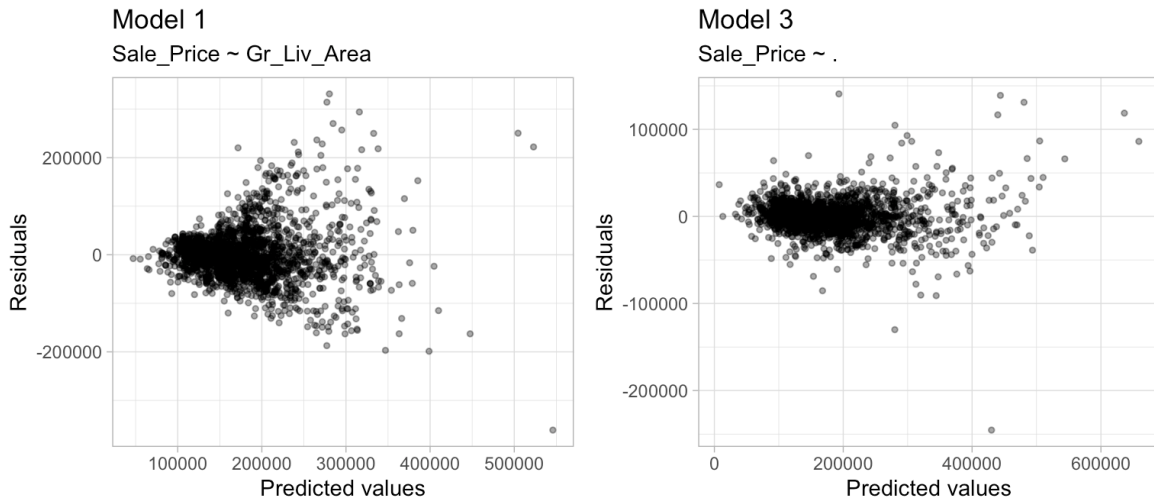
Figure 2.3 Linear regression assumes constant variance among the residuals. model1 (left) shows definitive signs of heteroskedasticity whereas model3 (right) appears to have constant variance.

3. No autocorrelation: Linear regression assumes the errors are independent and uncorrelated. If in fact, there is correlation among the errors, then the estimated standard errors of the coefficients will be biased leading to prediction intervals being narrower than they should be.
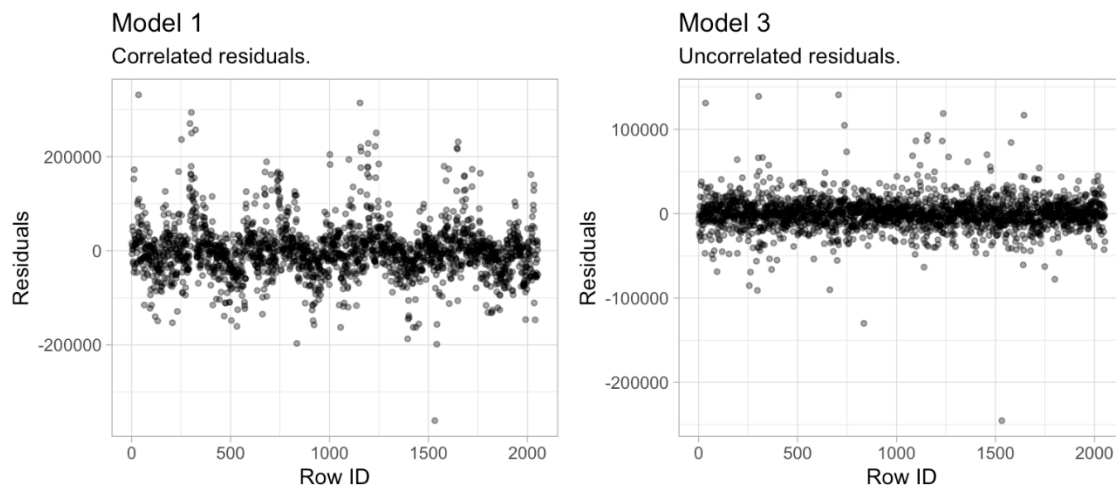

Figure 2.4 Linear regression assumes uncorrelated errors. The residuals in Model 1 (left) have a distinct pattern; whereas the Model 3 shows no signs of autocorrelation.

4. More Observations than Predictors: The OLS estimates are not obtainable when the number of features exceeds the number of observations (p>n). To resolve this issue an analyst can remove variables one-at-a-time until (p<n). Although pre-processing tools can be used to guide this manual approach (Kuhn and Johnson 2013, 26:43–47), it can be cumbersome and prone to errors. In the following, we'll introduce regularized regression which provides an alternative to OLS that can be used when (p>n).

5. No or little multicollinearity: Collinearity refers to the situation in which two or more predictor variables are closely related to one another. The presence of collinearity can pose problems in the OLS, since it can be difficult to separate out the individual effects of collinear variables on the response. In fact, collinearity

can cause predictor variables to appear as statistically insignificant when in fact they are significant. This obviously leads to an inaccurate interpretation of coefficients and makes it difficult to identify influential predictors.

One option is to manually remove the offending predictors (one-at-a-time) until all pairwise correlations are below some pre-determined threshold. However, when the number of predictors is large such as in our case, this becomes tedious. Moreover, multicollinearity can arise when one feature is linearly related to two or more features (which is more difficult to detect20). In these cases, manual removal of specific predictors may not be possible. Consequently, we offer some alternative methods where dimension reduction is applied prior to performing linear regression which are less affected by multicollinearity.

2.10. Feature interpretation

Once we've found the model that maximizes the predictive accuracy, our next goal is to interpret the model structure. Linear regression models provide a very intuitive model structure as they assume a monotonic linear relationship between the predictor variables and the response. The linear relationship part of that statement just means, for a given predictor variable, it assumes for every one unit change in a given predictor variable there is a constant change in the response. As discussed earlier, this constant rate of change is provided by the coefficient for a predictor. The monotonic relationship means that a given predictor variable will always have a positive or negative relationship. But how do we determine the most influential variables?

Variable importance seeks to identify those variables that are most influential in our model. For linear regression models, this is most often measured by the absolute value of the t-statistic for each model parameter used; though simple, the results can be hard to interpret when the model includes interaction effects and complex transformations. For a PLS model, variable importance can be computed using the weighted sums of the absolute regression coefficients. The weights are a function of the reduction of the RSS across the number of PLS components and are computed separately for each outcome. Therefore, the contribution of the coefficients are weighted proportionally to the reduction in the RSS.

Linear regression provides a solid fundamental understanding of the supervised learning task; however, as we've discussed there are several concerns that result from the assumptions required. Although extensions of linear regression that integrate dimension reduction steps into the algorithm can help address some of the problems with linear regression, more advanced supervised algorithms typically provide greater flexibility and improved accuracy. Nonetheless, understanding linear regression provides a foundation that will serve you well in learning these more advanced methods.

2.11. Regularized Regression

Linear models (LMs) provide a simple, yet effective, approach to predictive modeling. Moreover, when certain assumptions required by LMs are met (e.g., constant variance), the estimated coefficients are unbiased and, of all linear unbiased estimates, have the lowest variance. However, in today's world, data sets being analyzed typically contain a large number of features. As the number of features grow, certain assumptions typically break down and these models tend to overfit the training data, causing our

out of sample error to increase. Regularization methods provide a means to constrain or regularize the estimated coefficients, which can reduce the variance and decrease out of sample error.

The easiest way to understand regularized regression is to explain how and why it is applied to ordinary least squares (OLS). The objective in OLS regression is to find the hyperplane (e.g., a straight line in two dimensions) that minimizes the sum of squared errors (SSE) between the observed and predicted response values. This means identifying the hyperplane that minimizes the grey lines, which measure the vertical distance between the observed and predicted response values.

For classical statistical inference procedures (e.g., confidence intervals based on the classic t-statistic) to be valid, we also need to make stronger assumptions regarding normality (of the errors) and homoscedasticity (i.e., constant error variance).

Many real-life data sets, like those common to text mining and genomic studies are wide, meaning they contain a larger number of features (p>n). As p increases, we're more likely to violate some of the OLS assumptions and alternative approaches should be considered. This was briefly illustrated where the presence of multicollinearity was diminishing the interpretability of our estimated coefficients due to inflated variance. By reducing multicollinearity, we were able to increase our model's accuracy. Of course, multicollinearity can also occur when (n>p).

Having a large number of features invites additional issues in using classic regression models. For one, having a large number of features makes the model much less interpretable. Additionally, when (p>n), there are many (in fact infinite) solutions to the OLS problem! In such cases, it is useful (and practical) to assume that a smaller subset of the features exhibit the strongest effects (something called the bet on sparsity principle (see Hastie, Tibshirani, and Wainwright 2015, 2).). For this reason, we sometimes prefer estimation techniques that incorporate feature selection. One approach to this is called hard thresholding feature selection, which includes many of the traditional linear model selection approaches like forward selection and backward elimination. These procedures, however, can be computationally inefficient, do not scale well, and treat a feature as either in or out of the model (hence the name hard thresholding). In contrast, a more modern approach, called soft thresholding, slowly pushes the effects of irrelevant features toward zero, and in some cases, will zero out entire coefficients. As will be demonstrated, this can result in more accurate models that are also easier to interpret.

With wide data (or data that exhibits multicollinearity), one alternative to OLS regression is to use regularized regression (also commonly referred to as penalized models or shrinkage methods) to constrain the total size of all the coefficient estimates. This constraint helps to reduce the magnitude and fluctuations of the coefficients and will reduce the variance of our model (at the expense of no longer being unbiased—a reasonable compromise).

There are three common penalty parameters we can implement:

- Ridge;

- LASSO;

- Elastic Net; which is a combination of ridge and lasso.

Regularized regression provides many great benefits over traditional GLMs when applied to large data sets with lots of features. It provides a great option for handling the (n>p) problem, helps minimize the impact of multicollinearity, and can perform automated feature selection.

However, regularized regression does require some feature preprocessing. Notably, all inputs must be numeric; they cannot automatically handle missing data, which requires you to remove or impute them prior to modeling. Similar to GLMs, they are also not robust to outliers in both the feature and target. Lastly, regularized regression models still assume a monotonic linear relationship (always increasing or decreasing in a linear fashion). It is also up to the analyst whether or not to include specific interaction effects.

2.12. Decision trees

Tree-based models are a class of nonparametric algorithms that work by partitioning the feature space into a number of smaller (non-overlapping) regions with similar response values using a set of splitting rules. Predictions are obtained by fitting a simpler model (e.g., a constant like the average response value) in each region. Such divide-and-conquer methods can produce simple rules that are easy to interpret and visualize with tree diagrams. As we'll see, decision trees offer many benefits; however, they typically lack in predictive performance compared to more complex algorithms like neural networks and MARS.

There are many methodologies for constructing decision trees but the most well-known is the classification and regression tree (CART) algorithm proposed in Breiman (1984).26 A basic decision tree partitions the training data into homogeneous subgroups (i.e., groups with similar response values) and then fits a simple constant in each subgroup (e.g., the mean of the within group response values for regression). The subgroups (also called nodes) are formed recursively using binary partitions formed by asking simple yes-or-no questions about each feature (e.g., is age < 18?). This is done a number of times until a suitable stopping criteria is satisfied (e.g., a maximum depth of the tree is reached). After all the partitioning has been done, the model predicts the output based on (1) the average response values for all observations that fall in that subgroup (regression problem), or (2) the class that has majority representation (classification problem). For classification, predicted probabilities can be obtained using the proportion of each class within the subgroup.

In essence, our tree is a set of rules that allows us to make predictions by asking simple yes-or-no questions about each feature. For example, if the customer is loyal, has household income greater than $150,000, and is shopping in a store, the exemplar tree diagram below would predict that the customer will redeem a coupon.
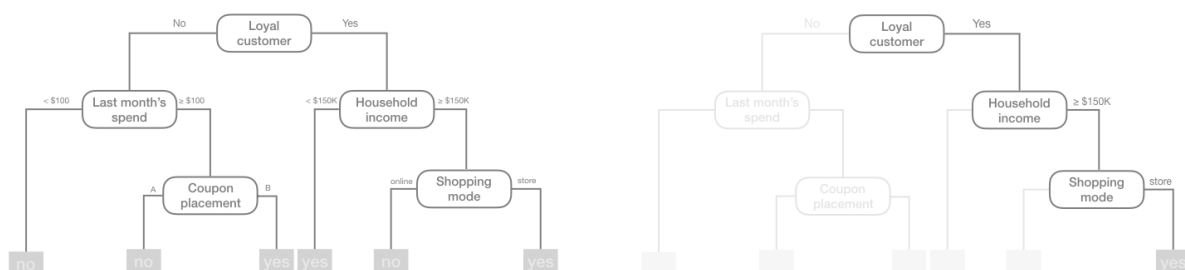
Figure 2.5 Exemplar decision tree predicting whether or not a customer will redeem a coupon (yes or no) based on the customer's loyalty, household income, last month's spend, coupon placement, and shopping mode.

We refer to the first subgroup at the top of the tree as the root node (this node contains all of the training data). The final subgroups at the bottom of the tree are called the terminal nodes or leaves. Every subgroup in between is referred to as an internal node. The connections between nodes are called branches.
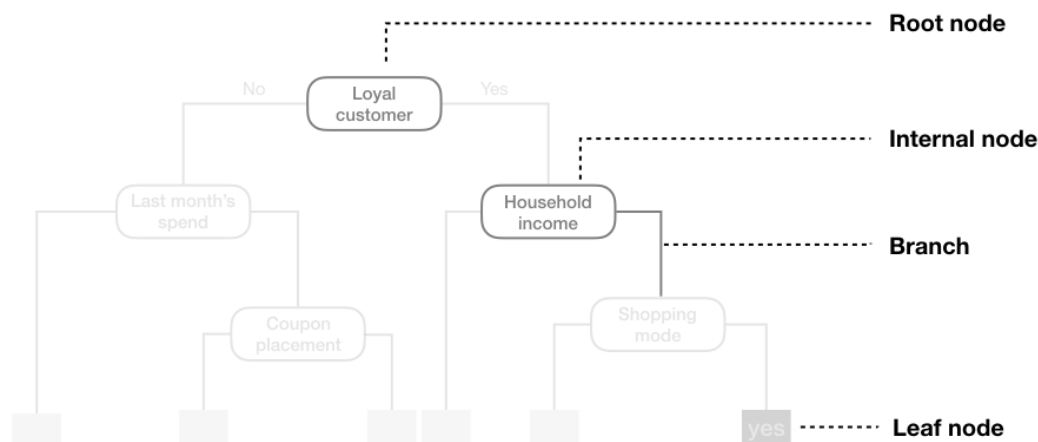
Figure 2.6 Terminology of a decision tree

Decision trees have a number of advantages. Trees require very little pre-processing. This is not to say feature engineering may not improve upon a decision tree, but rather, that there are no pre-processing requirements. Outliers typically do not bias the results as much since the binary partitioning simply looks for a single location to make a split within the distribution of each feature.

Decision trees can easily handle categorical features without preprocessing. For unordered categorical features with more than two levels, the classes are ordered based on the outcome (for regression problems, the mean of the response is used and for classification problems, the proportion of the positive outcome class is used).

Missing values often cause problems with statistical models and analyses. Most procedures deal with them by refusing to deal with them—incomplete observations are tossed out. However, most decision tree implementations can easily handle missing values in the features and do not require imputation. This is handled in various ways but most commonly by creating a new "missing" class for categorical variables or using surrogate splits.

However, individual decision trees generally do not often achieve state-of-the-art predictive accuracy. We saw that the best pruned decision tree, although it performed better than linear regression, had a very poor RMSE compared to some of the other models. This is driven by the fact that decision trees are composed of simple yes-or-no rules that create rigid non-smooth decision boundaries. Furthermore, we saw that deep trees tend to have high variance (and low bias) and shallow trees tend to be overly bias (but low variance).

2.13. Bagging

In the previous sections, we learned about bootstrapping as a resampling procedure, which creates b new bootstrap samples by drawing samples with replacement of the original training data. This chapter illustrates how we can use bootstrapping to create an ensemble of predictions. Bootstrap aggregating, also called bagging, is one of the first ensemble algorithms28 machine learning practitioners learn and is designed to improve the stability and accuracy of regression and classification algorithms. By model averaging, bagging helps to reduce variance and minimize overfitting. Although it is usually applied to decision tree methods, it can be used with any type of method.

Bootstrap aggregating (bagging) prediction models is a general method for fitting multiple versions of a prediction model and then combining (or ensembling) them into an aggregated prediction. Bagging is a fairly straight forward algorithm in which b bootstrap copies of the original training data are created, the regression or classification algorithm (commonly referred to as the base learner) is applied to each bootstrap sample and, in the regression context, new predictions are made by averaging the predictions together from the individual base learners. When dealing with a classification problem, the base learner predictions are combined using plurality vote or by averaging the estimated class probabilities together.

Bagging improves the prediction accuracy for high variance (and low bias) models at the expense of interpretability and computational speed. However, using various interpretability algorithms such as VIPs and PDPs, we can still make inferences about how our bagged model leverages feature information. Also, since bagging consists of independent processes, the algorithm is easily parallelizable.

However, when bagging trees, a problem still exists. Although the model building steps are independent, the trees in bagging are not completely independent of each other since all the original features are considered at every split of every tree. Rather, trees from different bootstrap samples typically have similar structure to each other (especially at the top of the tree) due to any underlying strong relationships.

2.14. Random Forrest

Random forests are a modification of bagged decision trees that build a large collection of de-correlated trees to further improve predictive performance. They have become a very popular "out-of-the-box" or "off-the-shelf" learning algorithm that enjoys good predictive performance with relatively little hyperparameter tuning. Many modern implementations of random forests exist; however, Leo Breiman's algorithm (Breiman 2001) has largely become the authoritative procedure.

Random forests are built using the same fundamental principles as decision trees and bagging. Bagging trees introduces a random component into the tree building process by building many trees on bootstrapped copies of the training data. Bagging then aggregates the predictions across all the trees; this aggregation reduces the variance of the overall procedure and results in improved predictive performance. However, simply bagging trees results in tree correlation that limits the effect of variance reduction.

Random forests help to reduce tree correlation by injecting more randomness into the tree-growing process.

Chapter 3

3.1. Dataset

Dataset download link:

https://github.com/nguyen-toan/ISLR/blob/master/dataset/College.csv

The data we used in this project is related to the statistics of a large number of US colleges in 1995, published by the US News Agency and the World Report.

This database has 777 universities (rows) and 18 predictor (columns), the description of lecturers is as follows:

• Private: Public / private indicator

• Apps: Number of applications received

• Accept: Number of applicants accepted

• Enroll: Number of new students enrolled

• Top10perc: New students from top 10% of high school class

• Top25perc: New students from top 25% of high school class

• F.Undergrad: Number of full-time undergraduates

• P.Undergrad: Number of part-time undergraduates

• Outstate: Out-of-state tuition

• Room.Board: Room and board costs

• Books: Estimated book costs

• Personal: Estimated personal spending

• PhD: Percent of faculty with Ph.D.'s

• Terminal: Percent of faculty with terminal degree

• S.F.Ratio: Student / faculty ratio

• perc.alumni: Percent of alumni who donate

• Expend: Instructional expenditure per student

• Grad.Rate: Graduation rate

And college data at a glance! :

```
Loading required package: Matrix
> str(data1)
'data.frame':    777 obs. of  19 variables:
 $ X          : chr  "Abilene Christian University" "Adelphi University" "Adrian College" "Agnes Scott College" ...
 $ Private    : Factor w/ 1 level "Yes": 1 1 1 1 1 1 1 1 1 1 ...
 $ Apps       : int  1660 2186 1428 417 193 587 353 1899 1038 582 ...
 $ Accept     : int  1232 1924 1097 349 146 479 340 1720 839 498 ...
 $ Enroll     : int  721 512 336 137 55 158 103 489 227 172 ...
 $ Top10perc  : int  23 16 22 60 16 38 17 37 30 21 ...
 $ Top25perc  : int  52 29 50 89 44 62 45 68 63 44 ...
 $ F.Undergrad: int  2885 2683 1036 510 249 678 416 1594 973 799 ...
 $ P.Undergrad: int  537 1227 99 63 869 41 230 32 306 78 ...
 $ Outstate   : int  7440 12280 11250 12960 7560 13500 13290 13868 15595 10468 ...
 $ Room.Board : int  3300 6450 3750 5450 4120 3335 5720 4826 4400 3380 ...
 $ Books      : int  450 750 400 450 800 500 500 450 300 660 ...
 $ Personal   : int  2200 1500 1165 875 1500 675 1500 850 500 1800 ...
 $ PhD        : int  70 29 53 92 76 67 90 89 79 40 ...
 $ Terminal   : int  78 30 66 97 72 73 93 100 84 41 ...
 $ S.F.Ratio  : num  18.1 12.2 12.9 7.7 11.9 9.4 11.5 13.7 11.3 11.5 ...
 $ perc.alumni: int  12 16 30 37 2 11 26 37 23 15 ...
 $ Expend     : int  7041 10527 8735 19016 10922 9727 8861 11487 11644 8991 ...
 $ Grad.Rate  : int  60 56 54 59 15 55 63 73 80 52 ...
> |
```

Figure 3.1

```r
college <- college[, -1]
head(college)
```

```
##                            Private Apps Accept Enroll Top10perc Top25perc
## Abilene Christian University   Yes 1660   1232    721        23        52
## Adelphi University             Yes 2186   1924    512        16        29
## Adrian College                 Yes 1428   1097    336        22        50
## Agnes Scott College            Yes  417    349    137        60        89
## Alaska Pacific University      Yes  193    146     55        16        44
## Albertson College              Yes  587    479    158        38        62
##                            F.Undergrad P.Undergrad Outstate Room.Board Books
## Abilene Christian University        2885         537     7440       3300   450
## Adelphi University                  2683        1227    12280       6450   750
## Adrian College                      1036          99    11250       3750   400
## Agnes Scott College                  510          63    12960       5450   450
## Alaska Pacific University            249         869     7560       4120   800
## Albertson College                    678          41    13500       3335   500
##                            Personal PhD Terminal S.F.Ratio perc.alumni Expend
## Abilene Christian University   2200  70       78      18.1          12   7041
## Adelphi University             1500  29       30      12.2          16  10527
## Adrian College                 1165  53       66      12.9          30   8735
## Agnes Scott College             875  92       97       7.7          37  19016
## Alaska Pacific University      1500  76       72      11.9           2  10922
## Albertson College               675  67       73       9.4          11   9727
##                            Grad.Rate
## Abilene Christian University        60
## Adelphi University                  56
## Adrian College                      54
## Agnes Scott College                 59
## Alaska Pacific University           15
## Albertson College                   55
```

Figure 3.2
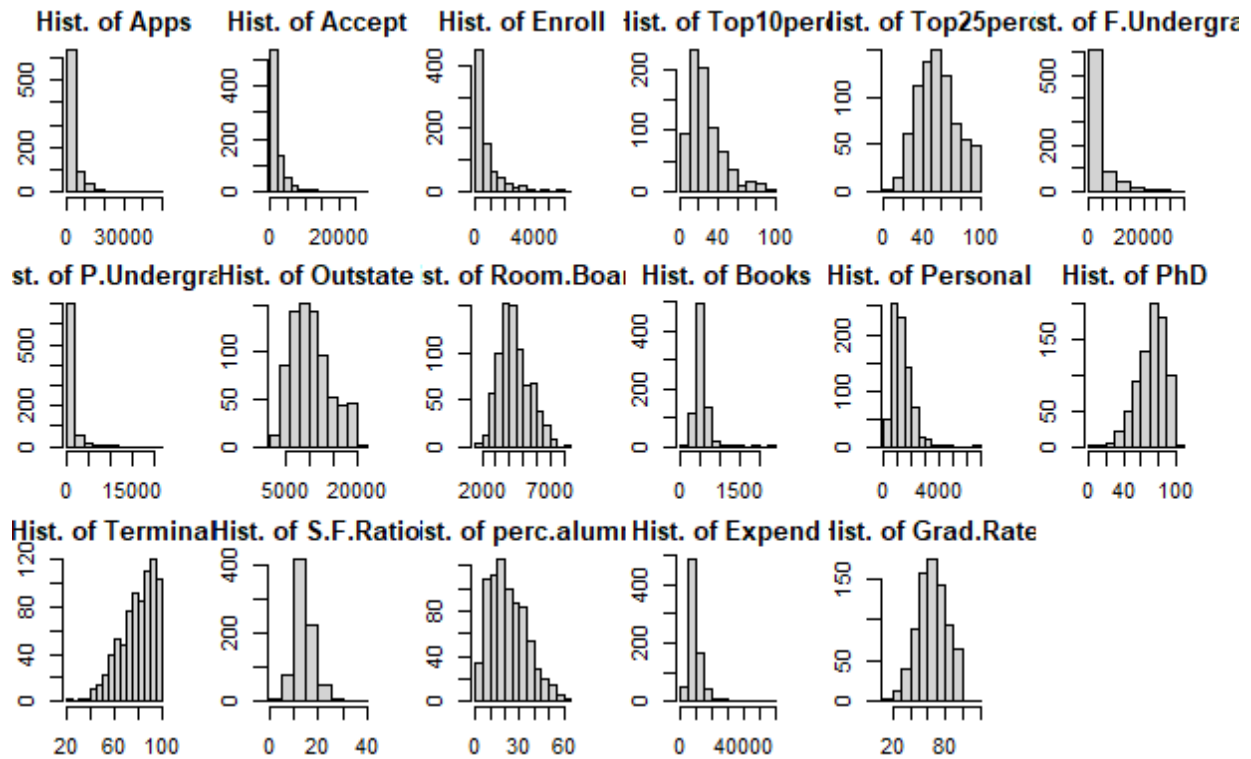
3.2. Data inspection

Figure 3.3

Here, we show the distribution of continuous variables by histogram, which is usually used to get a better view.

Note: All variables are continuous!

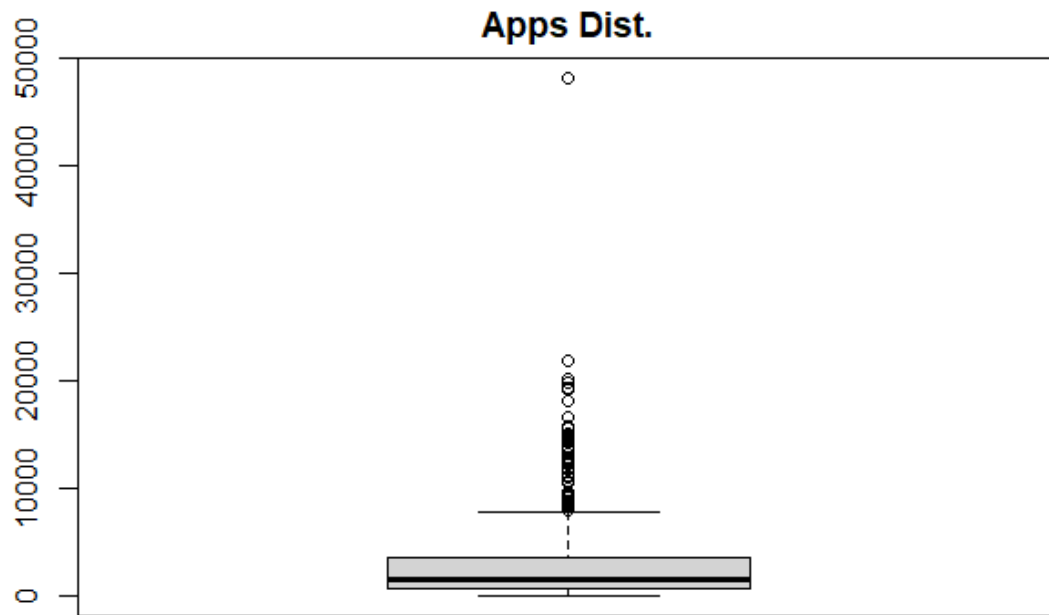As it turns out, normally distributed predictors can be more helpful in predicting the target variable.

Figure 3.4

Now, we can see the variable values of the Apps variable in a box plot calculated by the Tukey method. It is clear that we are going to face problems in the number of requests over ten thousand.
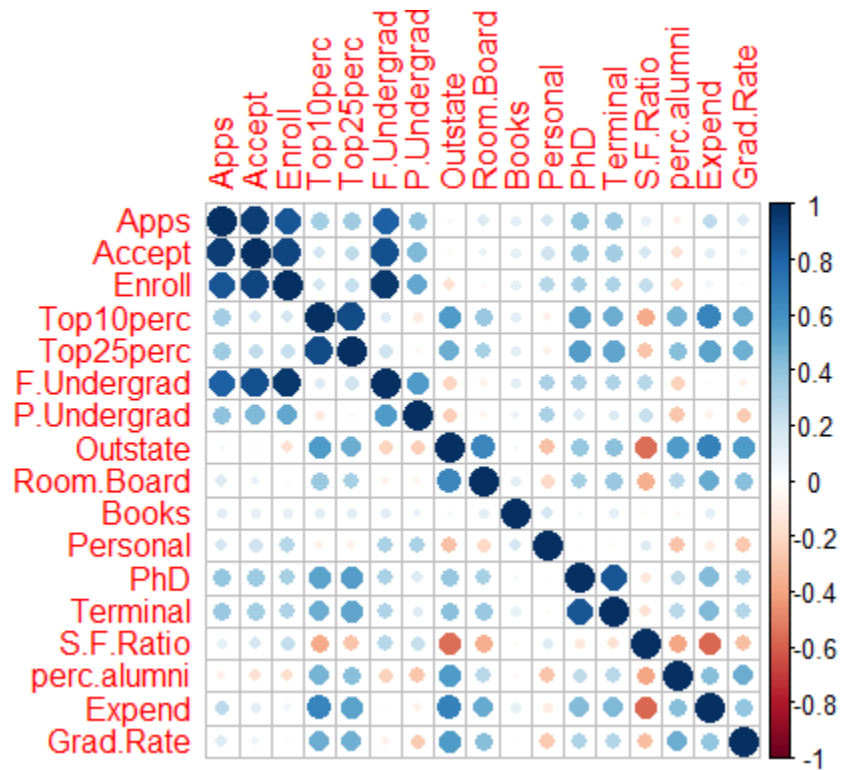

Figure 3.5

In this section, we draw the correlation plot of predictors to gain a good understanding of the correlation between predictors. It is clear that the greater correlation between the predictors, makes a better performance of the regression models.

The values that the correlation plot takes are between -1 and +1, the closer the value to -1, the lower the correlation, and the closer to +1, the greater the correlation.

As we can see, the variables "Accept", "Enroll" and "F.Undergrad" obviously have a good correlation with the requested variable.
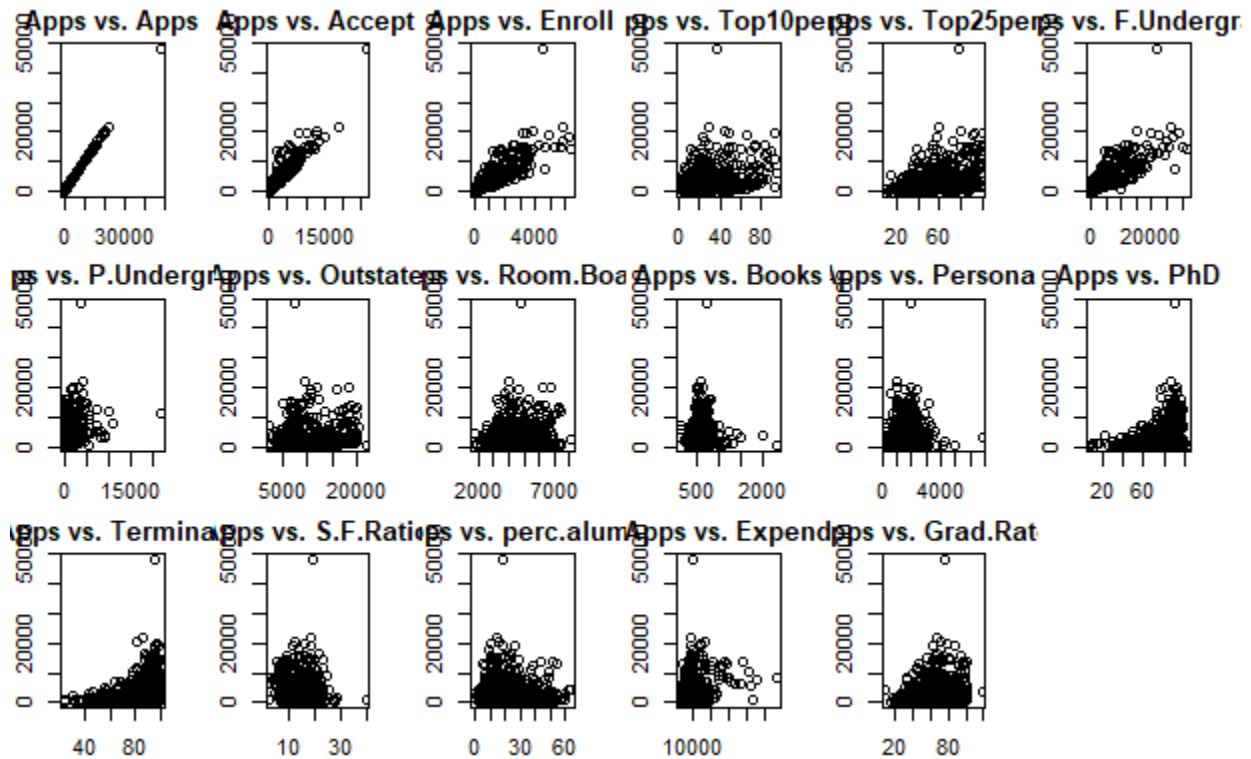
Figure 3.6

Scatter Plot tries to show the linear relationship between all the different continuous variables, or on the other hand, evaluates the Apps variable relative to all those different continuous variables. Also, we can find linear relationships and scatter between variables here and get an overview of it.

So far, with the tools provided by the R language, we have been able to obtain an interview or overview of the data that can continue to have a significant impact on our decisions.

3.3. Proposed models and their evaluation

3.3.1. Traditional Regression Model

First we run the traditional linear model with the 'lm' function on all variables, the result of which is as follows:

```
Console  Terminal ×  Jobs ×
E:/Bachelor Project/college/
> summary(lm_1)

Call:
lm(formula = Apps ~ ., data = train)

Residuals:
    Min      1Q  Median      3Q     Max
-5154.9  -446.2    -2.8   319.5  7447.6

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) -758.81100  453.60783  -1.673  0.09488 .
Accept         1.61552    0.04563  35.407  < 2e-16 ***
Enroll        -1.00625    0.20748  -4.850 1.57e-06 ***
Top10perc     50.90603    6.33832   8.031 5.06e-15 ***
Top25perc    -12.84654    5.11961  -2.509  0.01236 *
F.Undergrad    0.08415    0.03504   2.402  0.01661 *
P.Undergrad    0.05976    0.03695   1.617  0.10631
Outstate      -0.10441    0.02045  -5.106 4.41e-07 ***
Room.Board     0.12321    0.05448   2.261  0.02409 *
Books         -0.06868    0.26316  -0.261  0.79421
Personal       0.03163    0.07284   0.434  0.66426
PhD           -6.65547    5.24070  -1.270  0.20459
Terminal      -1.70459    5.74955  -0.296  0.76697
S.F.Ratio     14.77381   14.88983   0.992  0.32149
perc.alumni   -2.99864    4.79858  -0.625  0.53227
Expend         0.06779    0.01391   4.875 1.40e-06 ***
Grad.Rate      9.19843    3.49223   2.634  0.00866 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1075 on 604 degrees of freedom
Multiple R-squared:  0.9287,    Adjusted R-squared:  0.9268
F-statistic: 491.9 on 16 and 604 DF,  p-value: < 2.2e-16

>
```

Figure 3.7

Here, the t-test identifies the predicted predictors for us which is indicating that these 3-star predicted predictors can perform better in predicting the target variable which is Apps.

So after using the traditional linear model on the Apps variable relative to all variables, we are now going to run the traditional regression on the variables that were significant in the previous step (model lm_1) and make the regression assumptions normal and check them with histogram and qq-plot below:

```
> summary(lm_2)

Call:
lm(formula = Apps ~ Enroll + Top10perc + Top25perc + Outstate +
    Expend + Grad.Rate, data = train)

Residuals:
    Min      1Q  Median      3Q     Max
-8476.7  -649.5   -74.6   381.0 31400.2

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) -2.615e+03  3.966e+02  -6.593 9.28e-11 ***
Enroll       3.581e+00  9.043e-02  39.604  < 2e-16 ***
Top10perc    1.013e+01  1.110e+01   0.912 0.361971
Top25perc    3.694e-01  8.938e+00   0.041 0.967048
Outstate     3.406e-02  3.060e-02   1.113 0.266052
Expend       7.925e-02  2.313e-02   3.426 0.000653 ***
Grad.Rate    2.135e+01  5.865e+00   3.641 0.000294 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1938 on 614 degrees of freedom
Multiple R-squared:  0.7643,    Adjusted R-squared:  0.762
F-statistic: 331.9 on 6 and 614 DF,  p-value: < 2.2e-16

>
```

Figure 3.8

As can be seen, the value of R-squared and Adjusted R-squared in the lm_2 model has been reduced and improved.
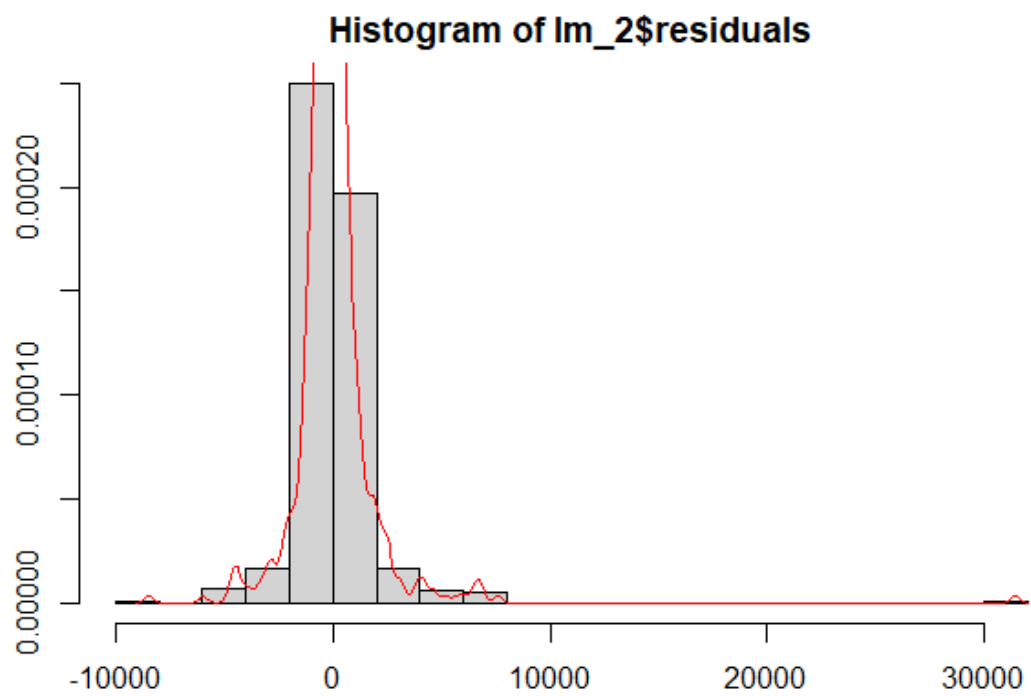
Figure 3.9

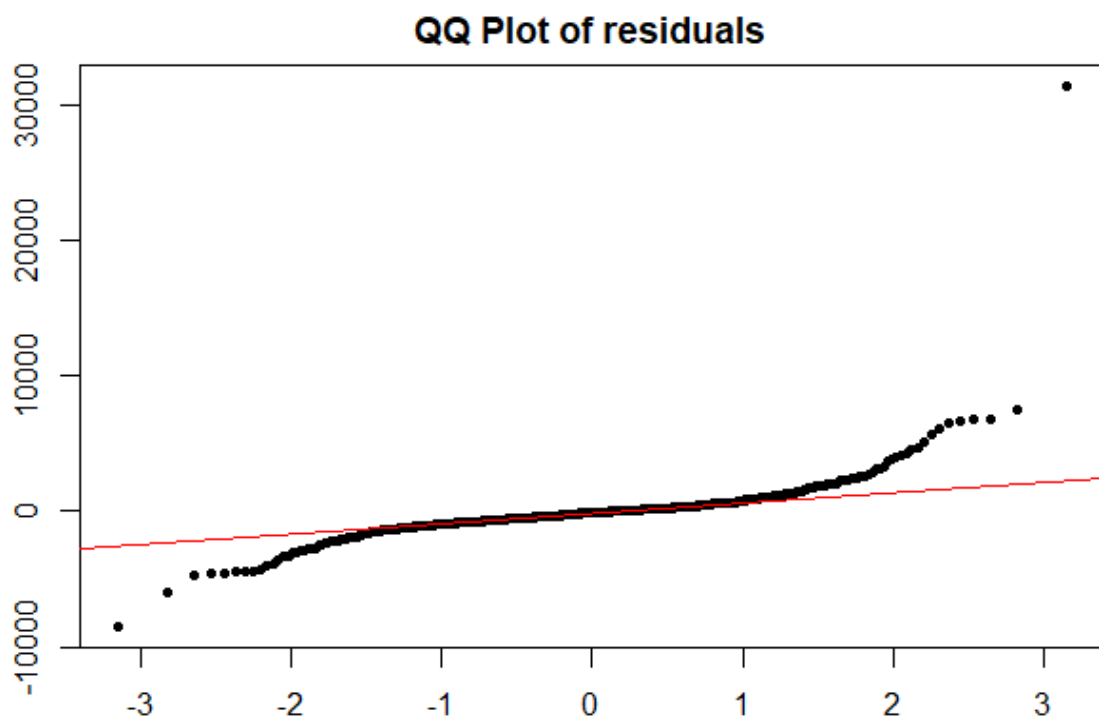Obviously the data is skewed to the right!

Figure 3.10

The right tail as well as the left tail can be a reason that indicates it is not normal.

To get just out of curiosity, we can check for skewness and kurtosis by performing the Jarque-Bera and Anscombe-Glynn tests. With p value of less than 0.05, we can consider the assumption of normality rejected.
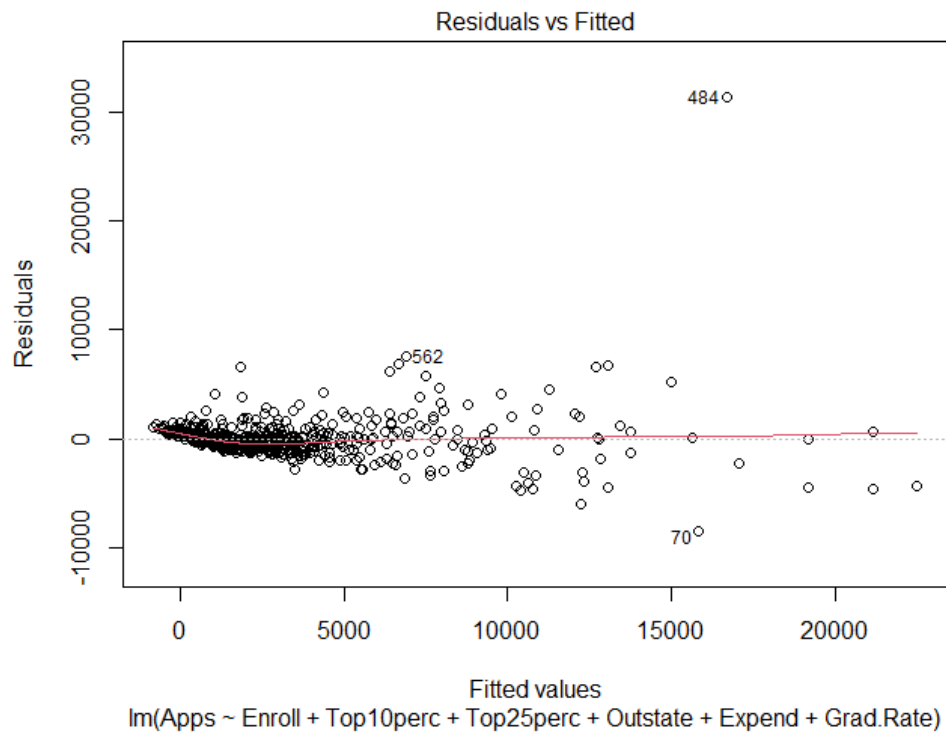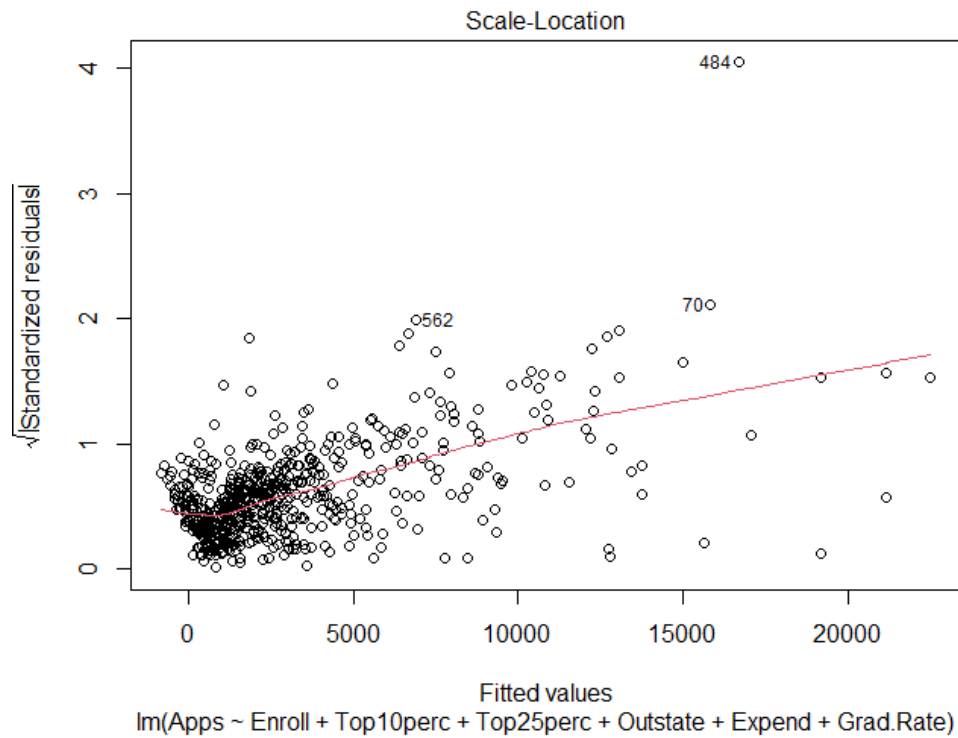

Figure 3.11

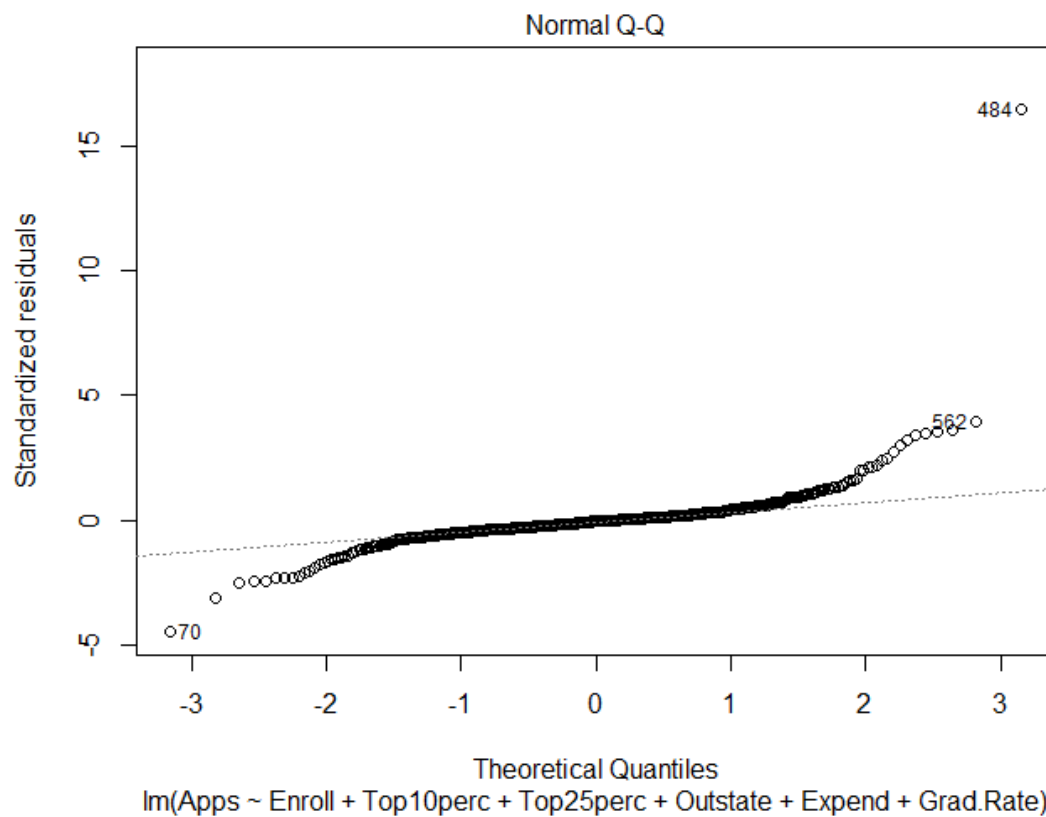Figure 3.12 both of these graphs show the variance of the residuals.

Figure 3.13 we already had this diagram on page 3, the assumption of normality was rejected!
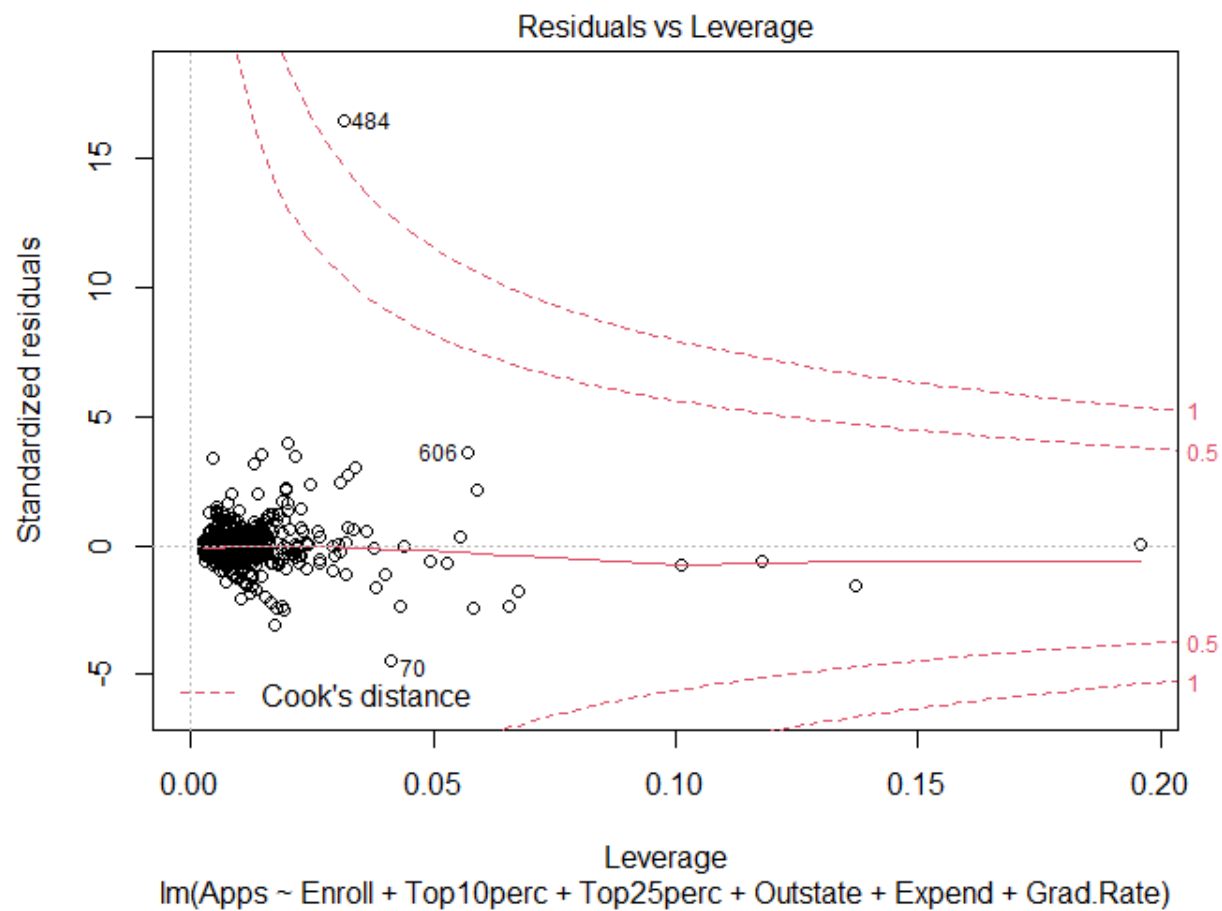
Figure 3.14 this diagram shows the outlier values from the perspective of Cook's Distance method, which concludes that they are not in a serious condition.
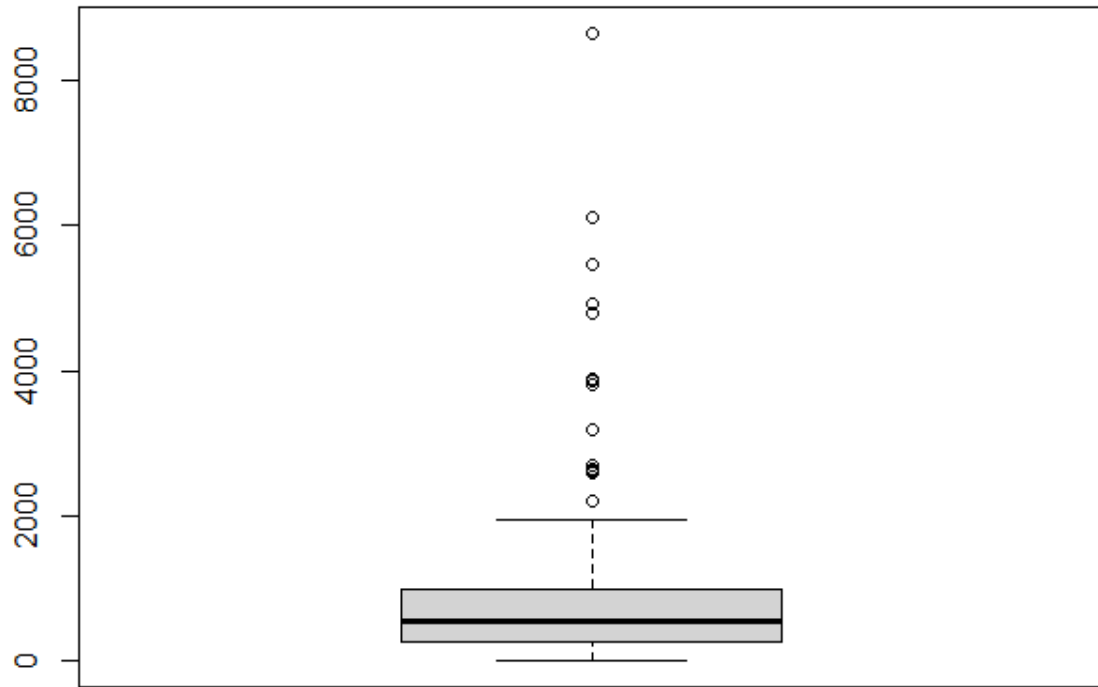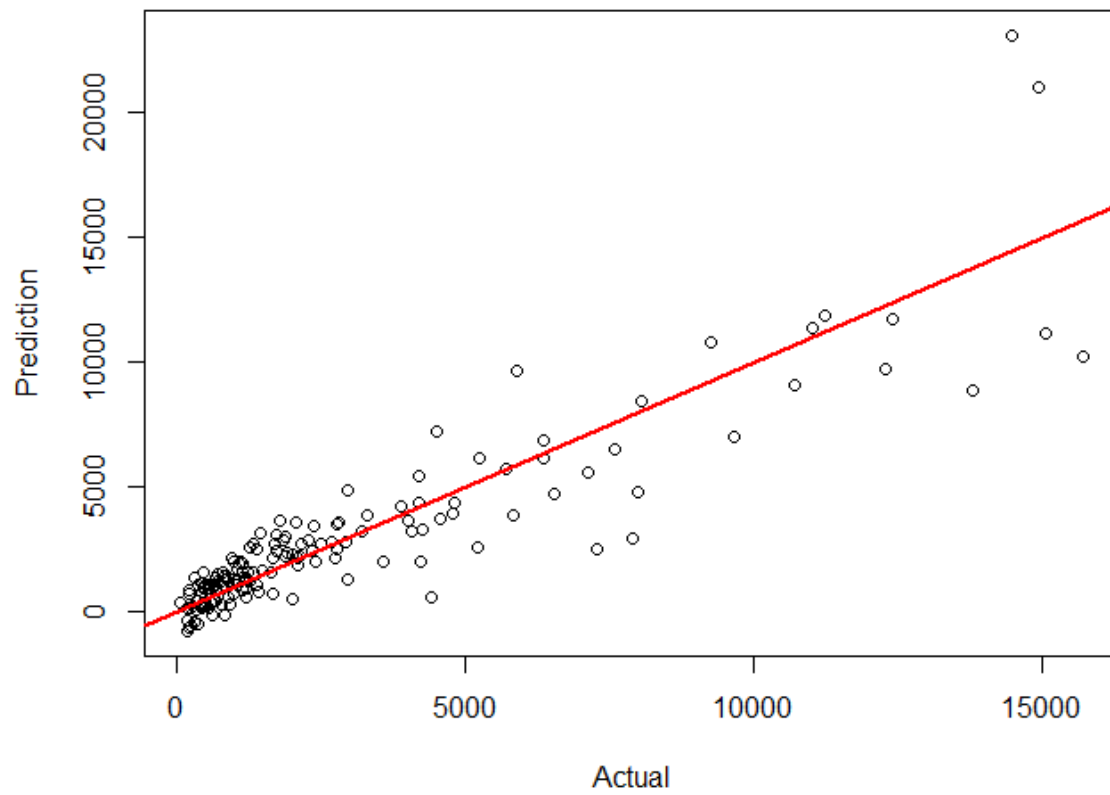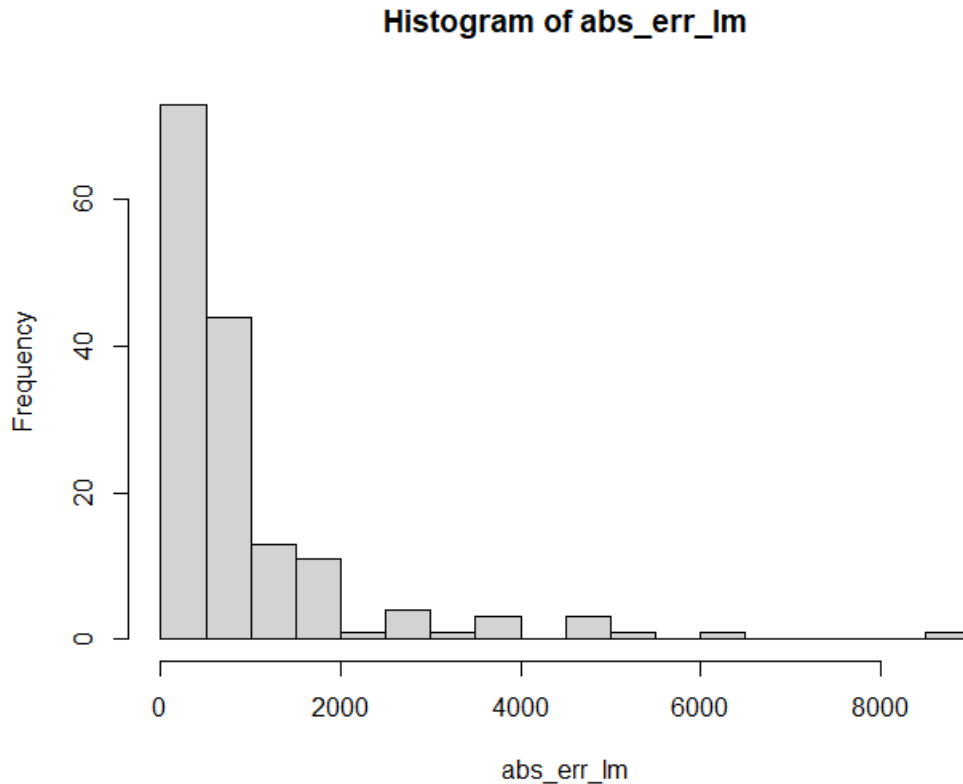
Figure 3.15

Figure 3.16

Histogram of abs_err_lm

Figure 3.17 the model has high errors in some cases that are problematic and must be improved in the future.

3.3.1.1. Box-Cox Transformation

As we have seen, correlation is one of our most important requirements in the traditional linear regression model. Therefore, we logarithmize all the data and scale them to a lower scale using Box-Cox Transformation.

By taking the logarithm of all the data and taking them to a lower scale, it is hoped to add some value to the data correlation so that the traditional linear regression model can perform better in predicting the data.

This approach cannot be described as a new model, but it can be a step towards improvement.

3.3.2. The Best Subset Selection Methods

This approach says let's start by examining all possible regressions with these variables that can be run, and then with a mechanism see which of these variables performs better.

So far, we have been using R-squared, and the more variables there are in R-squared, the more it tells us that this model is better, so we needed indicators that are not r-squared base and can purely measure predictability of the model.

#Algorithm:

1- Let M0 denote the null model, which contains no predictors.

2- For k = 1, 2, ... p:

(a) Fit all Cr (p, k) models that contain exactly k predictors.

(b) Pick the best among these models, and call it Mk.

the best is defined as having the largest R-squared.

3- Select a single best model from among M0, ..., Mp

using cross-validated prediction error, Cp, BIC, or adjusted R-squared

Now consider all p = 16 predictors; for each k, a combination of k is performed from p regression to the number of possible cases, and we select the best model for each k variable using the R-squared index. Here, because the number of variables is the same, we have no problem using the R-squared index to compare different models. Consider the case of one variable, for example; for each of those predictors, it builds a univariate model for us. We can make a combination of 1 out of 16, i.e. 16 univariate regressions, and then we will choose the best regression from these 16 different models based on the R-squared index. Then it becomes k = 2 and we choose all the regressions that can be made with bivariate structure, which can be a combination of 2 out of 16, and we come to R-squared, we calculate all of them and calculate the best of them. And then it becomes k = 3 and the combination of 3 out of 16 regressions is executed, and again we choose the best of them according to the R-squared index, and we continue in the same way until k = 16.

Finally, if we want to see which model was the best itself in k variable out of these 16 models, we will no longer use the R-squared index, we no longer use the R-squared index because R-squared is bias, that is, the more variables, the better R-squared gives. Here we have to use other indicators that more variables do not affect the result and give a penalty to more variables.

If we want to use the R-squared index here as well, the more variables there are, the more it is possible to overfit on train data, and it has problems in data that has not been seen yet, or in other words, performance gets down in test data. So we saw that R-squared does not give a penalty when the number of variables increase.

Therefore, we use indicators that give more penalties to the variables with more frequencies in the following:

Adjusted R-squared:

AdjR2 = 1 - [(1 - R2) (1 - n) / (n - d - 1)]

n: the number of samples
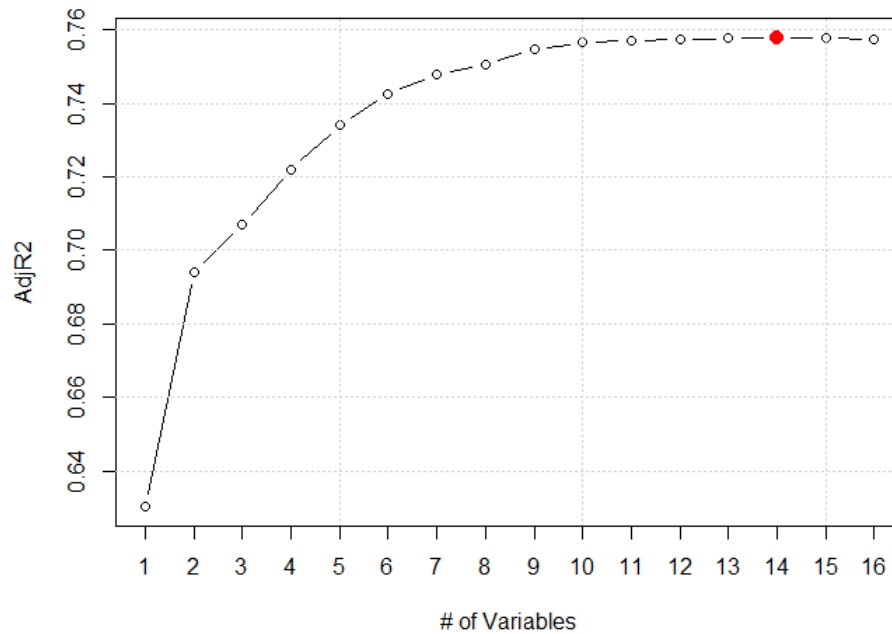
d: the number of predictors

Figure 3.18

```
> bestsub_2 <- lm(Log_Apps ~ Accept + Enroll + Top25perc + P.Undergrad + Outstate + Room.Board + Books + Personal + PhD + Terminal + S.F.Ratio + perc.alumni
  + Expend + Grad.Rate, data = train)
> summary(bestsub_2)

Call:
lm(formula = Log_Apps ~ Accept + Enroll + Top25perc + P.Undergrad +
    Outstate + Room.Board + Books + Personal + PhD + Terminal +
    S.F.Ratio + perc.alumni + Expend + Grad.Rate, data = train)

Residuals:
     Min       1Q   Median       3Q      Max
-2.12878 -0.29224  0.04857  0.32629  1.33028

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  3.944e+00  2.093e-01  18.843  < 2e-16 ***
Accept       1.347e-04  2.180e-05   6.179 1.18e-09 ***
Enroll       3.705e-04  6.316e-05   5.865 7.38e-09 ***
Top25perc    4.473e-03  1.514e-03   2.954 0.003262 **
P.Undergrad  2.403e-05  1.750e-05   1.373 0.170165
Outstate     1.426e-05  9.798e-06   1.455 0.146073
Room.Board   5.473e-05  2.636e-05   2.077 0.038253 *
Books        2.517e-04  1.272e-04   1.979 0.048242 *
Personal     4.657e-05  3.513e-05   1.326 0.185467
PhD          7.476e-03  2.514e-03   2.974 0.003059 **
Terminal     3.659e-03  2.757e-03   1.327 0.185067
S.F.Ratio    4.968e-02  7.177e-03   6.921 1.14e-11 ***
perc.alumni -7.889e-03  2.317e-03  -3.405 0.000705 ***
Expend       2.897e-05  6.261e-06   4.628 4.53e-06 ***
Grad.Rate    6.834e-03  1.686e-03   4.054 5.69e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.52 on 606 degrees of freedom
Multiple R-squared:  0.7634,    Adjusted R-squared:  0.7579
F-statistic: 139.6 on 14 and 606 DF,  p-value: < 2.2e-16
```

Figure 3.19

#Cp:

#Cp = 1 / n * (RSS + 2 * d * sigma_hat ^ 2)

n: the number of samples

RSS: Residual Sum of Squares

d: the number of predictors

sigma_hat: estimate of the variance of the error (estimated on a model containing all predictors)

```
> bestsub_2 <- lm(Log_Apps ~ Accept + Enroll + Top25perc + P.Undergrad + Outstate + Room.Board + Books + Personal + PhD + Terminal + S.F.Ratio + perc.alumni
 + Expend + Grad.Rate, data = train)
> summary(bestsub_2)

Call:
lm(formula = Log_Apps ~ Accept + Enroll + Top25perc + P.Undergrad +
    Outstate + Room.Board + Books + Personal + PhD + Terminal +
    S.F.Ratio + perc.alumni + Expend + Grad.Rate, data = train)

Residuals:
     Min       1Q   Median       3Q      Max
-2.12878 -0.29224  0.04857  0.32629  1.33028

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  3.944e+00  2.093e-01  18.843  < 2e-16 ***
Accept       1.347e-04  2.180e-05   6.179 1.18e-09 ***
Enroll       3.705e-04  6.316e-05   5.865 7.38e-09 ***
Top25perc    4.473e-03  1.514e-03   2.954 0.003262 **
P.Undergrad  2.403e-05  1.750e-05   1.373 0.170165
Outstate     1.426e-05  9.798e-06   1.455 0.146073
Room.Board   5.473e-03  2.636e-03   2.077 0.038253 *
Books        2.517e-04  1.272e-04   1.979 0.048242 *
Personal     4.657e-05  3.513e-05   1.326 0.185467
PhD          7.476e-03  2.514e-03   2.974 0.003059 **
Terminal     3.659e-03  2.757e-03   1.327 0.185067
S.F.Ratio    4.968e-02  7.177e-03   6.921 1.14e-11 ***
perc.alumni -7.889e-03  2.317e-03  -3.405 0.000705 ***
Expend       2.897e-05  6.261e-06   4.628 4.53e-06 ***
Grad.Rate    6.834e-03  1.686e-03   4.054 5.69e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.52 on 606 degrees of freedom
Multiple R-squared:  0.7634,     Adjusted R-squared:  0.7579
F-statistic: 139.6 on 14 and 606 DF,  p-value: < 2.2e-16
```

Figure 3.20

```
Console  Terminal ×  Jobs ×                                                                                                          _□
E:/Bachelor Project/college/
> bestsub_2 <- lm(Log_Apps ~ Accept + Enroll + Top25perc + P.Undergrad + Room.Board + Books + PhD + S.F.Ratio + perc.alumni + Expend + Grad.Rate, data = tra
in)
> summary(bestsub_2)

Call:
lm(formula = Log_Apps ~ Accept + Enroll + Top25perc + P.Undergrad +
    Room.Board + Books + PhD + S.F.Ratio + perc.alumni + Expend +
    Grad.Rate, data = train)

Residuals:
     Min       1Q   Median       3Q      Max
-2.21532 -0.29459  0.04367  0.33666  1.31526

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  4.105e+00  1.936e-01  21.201  < 2e-16 ***
Accept       1.388e-04  2.147e-05   6.464 2.09e-10 ***
Enroll       3.619e-04  6.159e-05   5.876 6.91e-09 ***
Top25perc    4.858e-03  1.505e-03   3.228  0.00131 **
P.Undergrad  2.586e-05  1.731e-05   1.494  0.13565
Room.Board   7.031e-05  2.404e-05   2.925  0.00358 **
Books        2.959e-04  1.248e-04   2.371  0.01806 *
PhD          1.026e-02  1.680e-03   6.108 1.79e-09 ***
S.F.Ratio    4.745e-02  7.068e-03   6.713 4.37e-11 ***
perc.alumni -7.094e-03  2.206e-03  -3.215  0.00137 **
Expend       3.210e-05  5.960e-06   5.385 1.04e-07 ***
Grad.Rate    6.937e-03  1.662e-03   4.174 3.43e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.521 on 609 degrees of freedom
Multiple R-squared:  0.7613,     Adjusted R-squared:  0.757
F-statistic: 176.5 on 11 and 609 DF,  p-value: < 2.2e-16
> |
```

Figure 3.21

#BIC:

#BIC (Bayesian Information Criterion) = -2 * LogLikelihood + log (n) * d

n: the number of samples

RSS: Residual Sum of Squares

d: the number of predictors

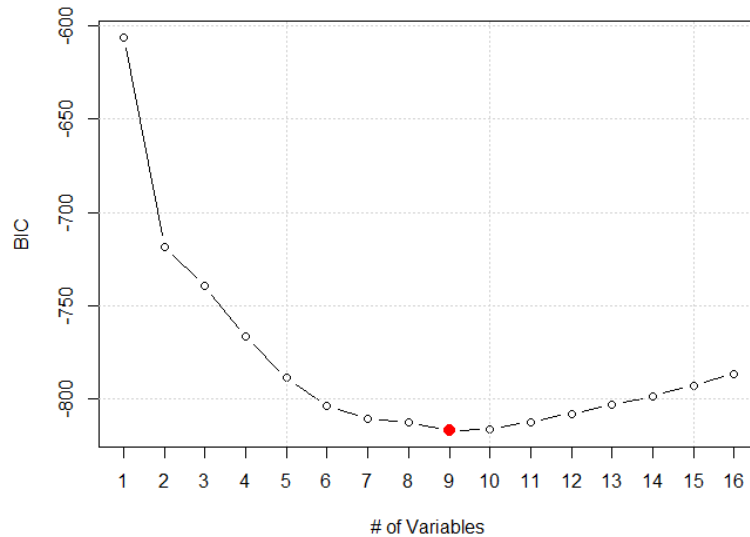# sigma_hat: estimate of the variance of the error

Figure 3.22 As you can see, the BIC index is stricter and tougher than the Cp index and gives more penalty to the variables.

Figure 3.23

Note: The step-wise regression method does not solve the problem of multypolinearity, meaning that we are only looking for the best combination that works best.

As can be seen from the review of residuals and adjusted R-squared and Multiple R-squared, the above three statistical indicators did not make much difference in performance, and we consider the selected model from the BIC index with a very small difference.

3.3.3. Forward and Backward Stepwise Selection

But if we take a quick look on Model 2 which was bestsub selection, the problem was it checked all the different combinations and cases, and it could be very difficult and costly operationally; So we introduced another approach which is very close to the logic of the previous model, but designed to reduce our computational volume called forward and backward step-wise regression. we try to check a smaller number of cases using a logic instead of looking at all the possible cases and we can have a good improvement in cases we have a large number of variables. We can also guess how they work from their names, Forward means that the number of variables gradually increases from the state where there are no variables, but Backward means that you subtract from the state where you have all the variables until you reach to zero variable or one variable.

#Forward Selection:

#Algorithm:

1- Let M0 denote the null model, which contains no predictors.

2- For k = 0, 2, ... p - 1:

(a) Consider all p ??? k models that augment the predictors in Mk

with one additional predictor.

(b) Pick the best among these models, and call it Mk + 1.

The best is defined as having the largest R-squared.

3- Select a single best model from among M0, ..., Mp

using cross-validated prediction error, Cp, BIC, or adjusted R-Squared

In first step, we have to look for the best variable to make a univariate regression with. We bring this to the next step and then we will see which of the other 15 variables makes the best combination with variable we brought up from previous step for our regression. In the next step, we keep these two variables and get prepared to see which of the remaining 14 variables makes the best model for us and so we go forward to reach all the variables. Again with the same logic, if we want to make the best three-variable model, we can handle the problem with R-squared because we are comparing all three-variable models together; But whenever we want to compare all the 16 selected variables together, we use adjusted R-squared or BIC or Cp to decide which one is the best.

#backward Selection:

#Algorithm:

1- Let Mp denote the full model, which contains all predictors.

2- For k = p, p - 1, ..., 1:

(a) Consider all k models that contain all but one of the predictors

in Mk, for a total of k ??? 1 predictors.

(b) Pick the best among these models, and call it Mk-1.

The best is defined as having the largest R-squared.

3- Select a single best model from among M0, ..., Mp

using cross-validated prediction error, Cp, BIC, or adjusted R

Backward is the opposite of Forward. In the first step we have all the variables (16 variable mode). In the next step, if we delete which variable, the best model makes 15 variables according to R-squared, and in the same way, in each step, we remove the variable that makes the best regression model to reach the one-variable model.

3.3.3.1. Using K-fold Cross-Validation approach

So far, in the approaches we have had, whether forward or backward or bestSubset, we have used a series of statistical indicators such as Adjust R-squared and BIC and Cp to state which model has better predictive power over the data which has not been seen yet; And we can do the same thing using the k-fold cross validation method, instead of using statistical indicators to guess the model better, so in this situation, we are supposed to train the data on a part of the train dataset and hold a part of the same train for validation and then measure the performance of this models.

So here we want to use the k-fold cross validation approach instead of using statistical indicators:

In short, for example, if we set k = 10, we create 10 folds, each time the model tests with 9 train folds and the remaining 1 fold. This cycle repeats k times. Here we set the criterion to MSE (Mean Squared Error) at each cycle, calculate the Mean Squared Error of each fold labeled i (ie the errors of each fold left out for the test phase) and put them in matrix design we made earlier. (The design matrix is a k*16 matrix that has k rows (fold) and 16 columns (predictor).)

And in the end, we calculate the average error of 10 folds.

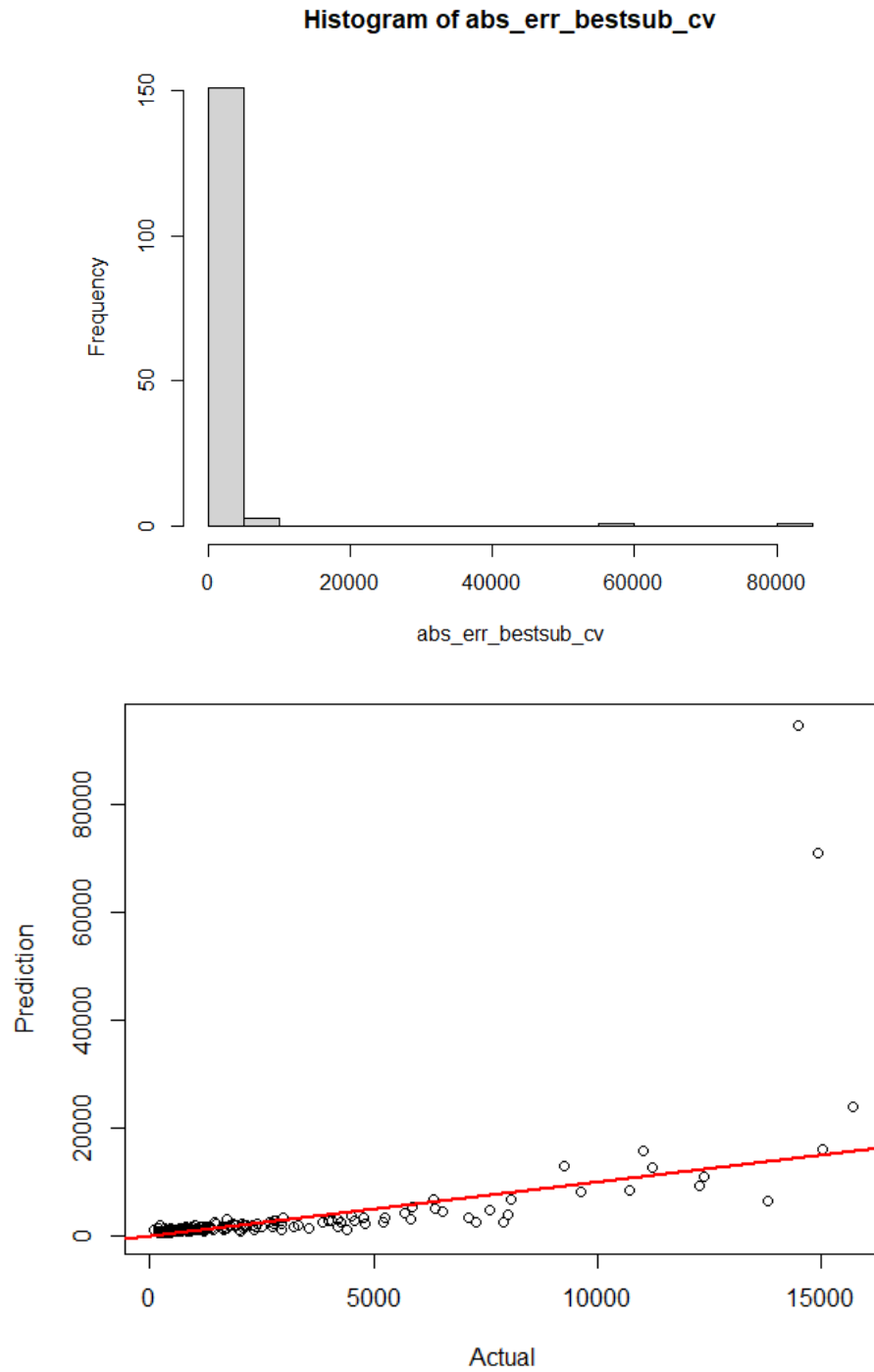**Histogram of abs_err_bestsub_cv**
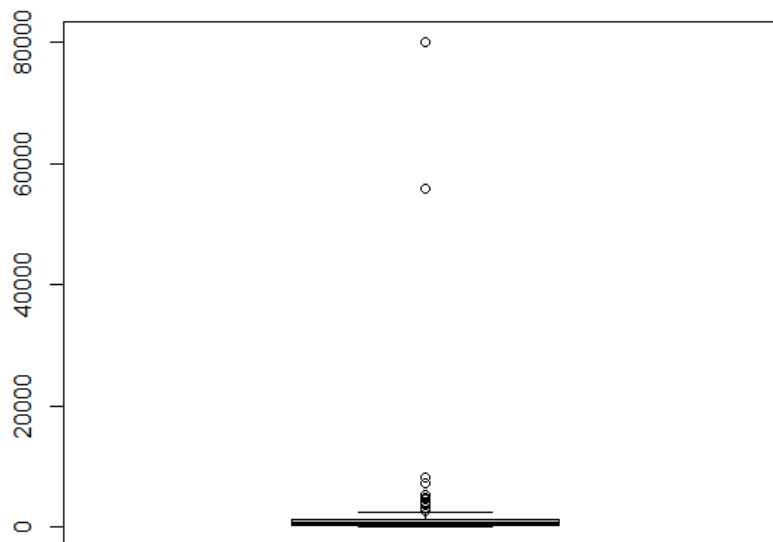
Figure 3.24,25,26 As we can see, we still have a number of outliers that are problematic.

3.3.4. Best Sub Selection Using Trimmed Train and CV

In the following, we will do the same for the case where we have removed outliers to see if we have an improvement or not, which we certainly do not expect a huge progress due to the high percentage existence of outliers.

Note that we did not delete the outlier data in the test because they are the data that we finally have to predict, so we accept that my errors in predicting the outlier data are a bit higher, but it definitely gets a better performance in data which is more normal and closer to the number of applications submitted to the most of universities.
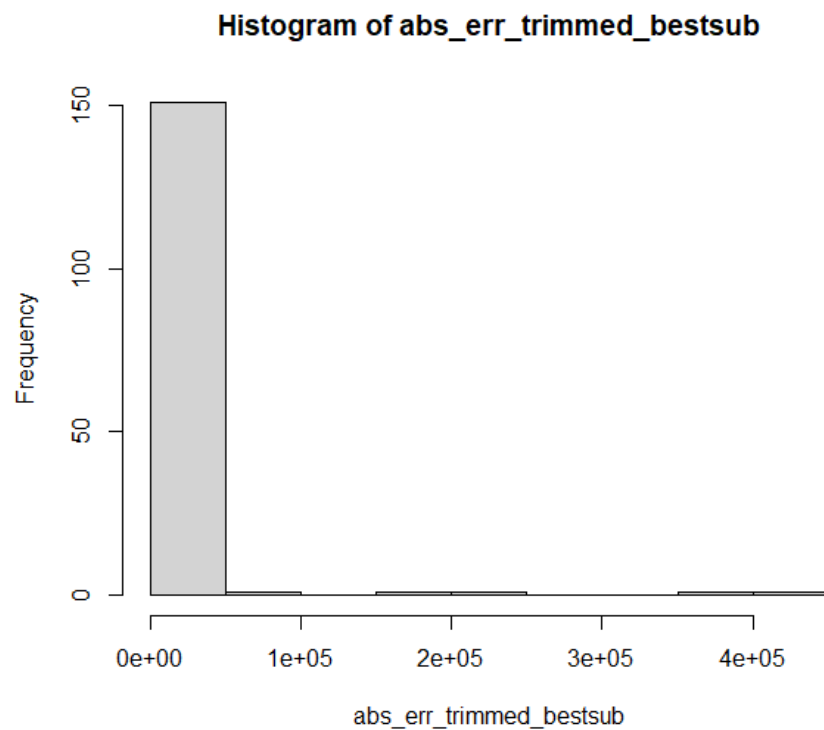
Histogram of abs_err_trimmed_bestsub
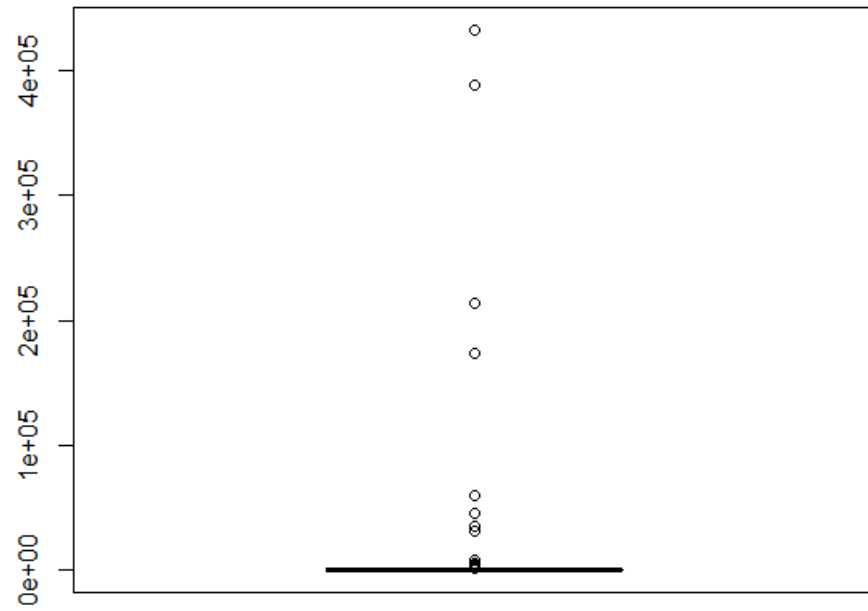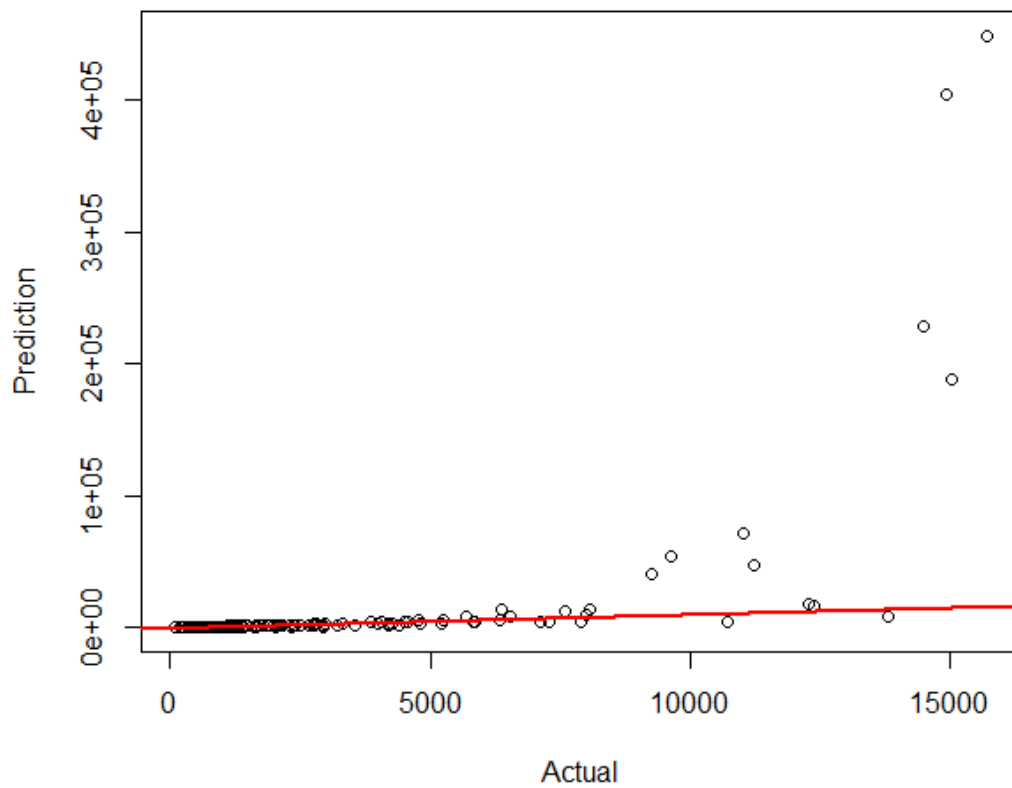

Figure 3.27

Figure 3.28

Figure 3.29

As you can see, the number of applications for more ordinary universities is better predicted, but the errors in forecasting the number of applications for specific universities are greater.

Figure 3.30

In terms of Mean of Errors, Traditional Linear Model has the best performance.

In terms of Median of Errors, Best Subset Using Trimmed train and CV has the best performance.

In terms of Standard Deviation of Errors, Traditional Linear Model has the best performance.

In terms of Interquartile Range of Errors, Best Subset Using Trimmed train and CV has the best performance.

In terms of Minimum of Errors, almost all models perform well.

In terms of Maximum of Errors, the Traditional Linear Model has the best performance.

As a result, in general, it could be concluded that the Best Subset Model with K-Fold Cross Validation approach has had the best performance so far compared to other models.

3.3.4.1. Regularization

The OLS method tries to minimize the RSS function.

$$\text{RSS} = \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2$$

In regularization approach, we minimize this function plus a penalty but the question is what the penalty is supposed to be. In the following, we will examine and check these penalties with three approaches of Ridge Regression, LASSO and Elastic-Net.

3.3.4.2. Ridge regression approach

As with the OLS approach, the goal is to minimize RSS + Penalty.

$$\sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 = \text{RSS} + \lambda \sum_{j=1}^{p} \beta_j^2$$

3.3.4.3. Intuitive understanding of the ridge regression approach

Suppose we are looking for a relation between x and y and we have no more than 2 points in total. As we can see, the line that connects these two points is the best and most ideal line in train data which

in most cases is not satisfactory. But the question is how does this line work in predicting the points it has not seen? Which is often in such cases where the number of data is less than the number of predictors; But what is the solution?
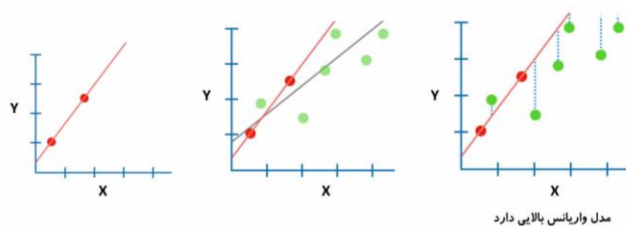

Figure 3.31

Regularization offers a solution to this problem in which we intentionally add some bias to the train data of model and let some error be trained in our data in the hope that the resulting line is a good predictor of unseen data. In other words, reduce its variance in test data; So in general, the basis of regularization work is that we deliberately add bias to the train model to improve the predictive power on unseen data, which is also used in various fields.
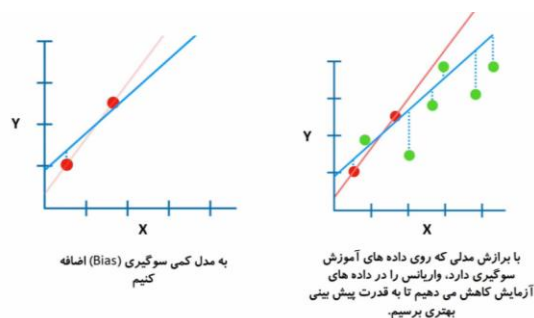

Figure 3.32

But when the number of our variables exceeds the number of our data, a series of problems appear that can be seen in some specialized fields such as genetic data or ... that, as we have seen, ridge regression in cases where the number of variables is more than Number of data can be a good approach.

3.3.4.4. Landa selection mechanism

But the question that arises is how much to add bias to our model? We build the model for different landa using the methods we have learned so far. Each time we add bias to the train data to our model, and each time we analyze the performance of the model on the data that it has not seen, and finally we come to the conclusion which one is the best Landa.

Tip: The bigger Landa gets, the more coefficients tend to be zero but never zero.

As a result, the difference between ridge regression and step-wise regression is that in step-wise regression we used k-fold cross validation to find out which variables are present in the model and which are not, but in ridge regression no variables do not eliminate completely, but its coefficients become smaller and smaller and smaller, and all variables are used in it.

3.3.4.5. K-Fold Cross Validation Reminder

We divide the data into K folds, measure the performance of the model on the test section, and conclude based on the sum of the functions.

The ridge regression method should not be used to infer data or interpret regression coefficients; Because we intentionally added to the bias model and these coefficients are no longer explanatory coefficients and we change the slope of the lines and this method is used only in prediction.

But the reasons why we are interested in using regularization in regression are as follows:

- Excessive fitting on training data: especially when the number of our variables is too large.

- Alignment: When we are interested in using all variables in some way.

- Instability: If alignment occurs, it causes the coefficients to be unstable.

- Selection of predictor variables by step-wise method: If the number of variables increases, the computational volume will be much higher and it will be slow and time consuming.

3.3.4.6. LASSO approach

It is almost similar to the ridge regression approach, except that the penalty is an absolute value instead of the power of the two coefficients; As a result, what happens is that because of the type of penalty that is awarded, your odds can be zero.

$$\sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^{p} |\beta_j| \ = \text{RSS} + \lambda \sum_{j=1}^{p} |\beta_j|$$

As a result, the LASSO approach can perform better than ridge regression for cases where only some variables are useful for prediction.

3.3.4.7. Elastic-Net approach

This method is almost similar to previous approaches, except that the penalty is equal to the sum of LASSO and Ridge Regression penalties.

3.3.5. Ridge Regression

The goal is to optimize:

#RSS + lambda * Sum (beta_i ^ 2)

#lambda => 0, a tuning parameter

To do this, we need to create a matrix model. Here the matrix model is the same as the regression syntax. In the next step, we find the appropriate Landa using the k-fold cross validation approach.
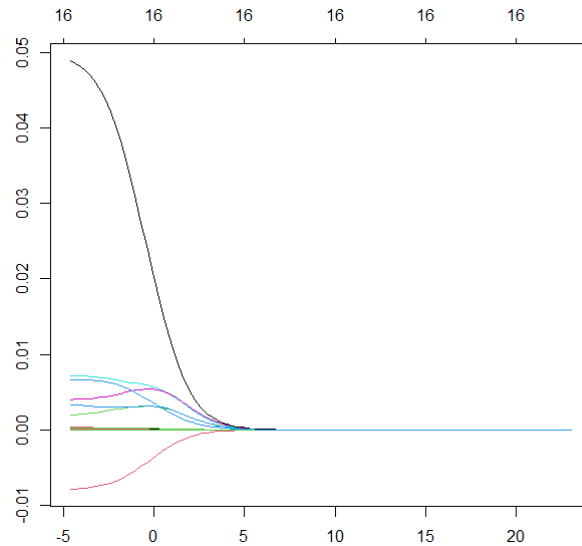
Figure 3.33

As it turns out, a series of variables have probably been very influential, and a series much less so, and then as Landa gets older, the variables start to tend to zero which never become zero. (Landa logarithm is drawn in the figure above)

The resulting Landa gives us the minimum error of the function. In the final step, we perform the train data using the 'glmnet' library and the test data using the same 'predict' function, the results of which are as follows.

	Mean.of.AbsErrors	Median.of.AbsErrors	SD.of.AbsErrors	IQR.of.AbsErrors	Min.of.AbsErrors	Max.of.AbsErrors
LM	946.6732	555.6521	1251.981	739.1078	2.72140412	8650.938
BestSub	1865.4376	422.4048	8243.686	938.4780	3.00590749	77414.299
BestSubCV	1703.3551	423.3370	7093.398	811.3109	0.02791144	67045.207
TrimmedBestSub	9552.1233	275.5476	51331.082	509.5759	5.37114423	433022.299
RidgeReg	1532.6778	456.3536	5665.915	836.5366	26.73521513	57221.365

Figure 3.34

We see that ridge regression on test data caused us to perform better than previous models, and especially to reduce Mean, Median, SD and INQ errors.
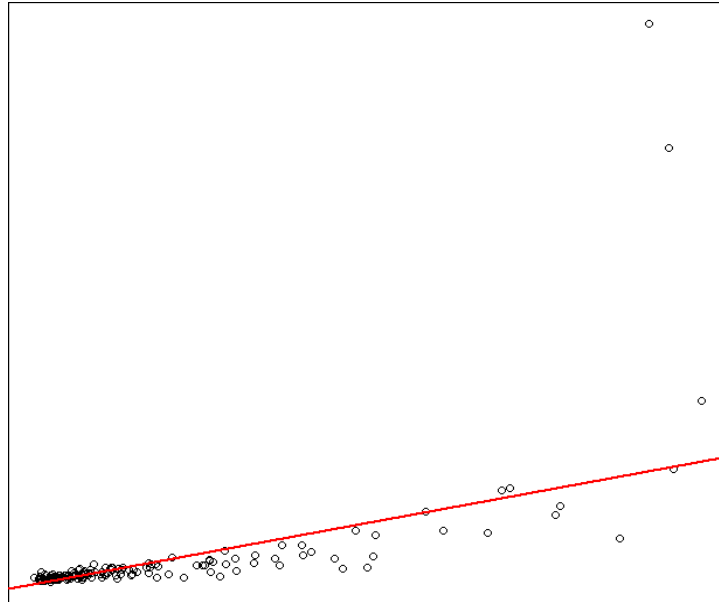
Figure 3.35 As expected, the model could suffer more errors in the high number of requests. (In the model comparison table, max errors also confirmed the same scenario.)

3.3.6. LASSO Regression

 #The goal is to optimize:

#RSS + lambda * Sum (abs (beta_i))

#lambda => 0, a tuning parameter

   In the LASSO approach similar to the Ridge approach, we use the same matrix model and grid ridge to find the appropriate Landa using the k-fold cross validation approach.
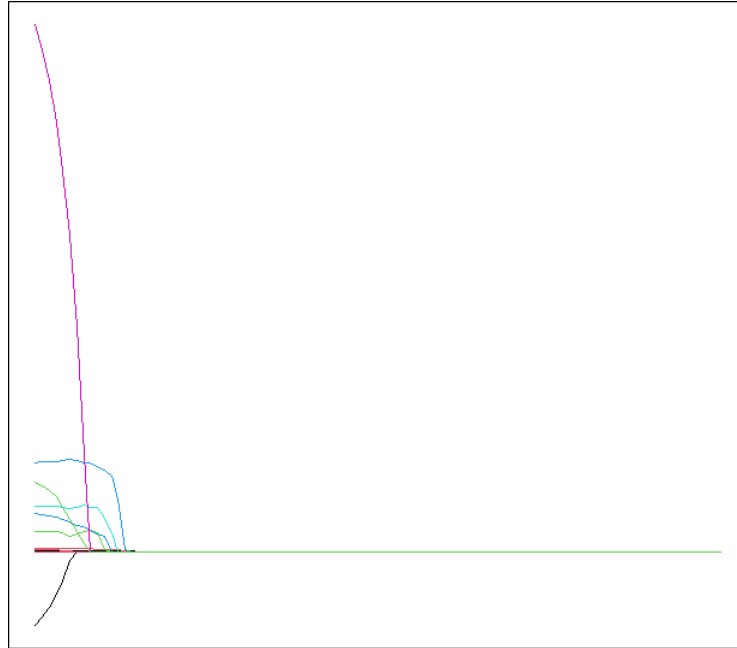
Figure 3.36 As can be seen, in LASSO, contrary to the Ridge approach, the effect of predictors in different Landas can be zero.

```
> lasso_cv$lambda.min
[1] 0.003465393
>
```

The resulting Landa gives us the minimum error of the function. In the final step, we perform the train using the glmnet library and the test using the same predict function, the results of which are as follows.

	Mean.of.AbsErrors	Median.of.AbsErrors	SD.of.AbsErrors	IQR.of.AbsErrors	Min.of.AbsErrors	Max.of.AbsErrors
LM	946.6732	555.6521	1251.981	739.1078	2.72140412	8650.938
BestSub	1865.4376	422.4048	8243.686	938.4780	3.00590749	77414.299
BestSubCV	1703.3551	423.3370	7093.398	811.3109	0.02791144	67045.207
TrimmedBestSub	9552.1233	275.5476	51331.082	509.5759	5.37114423	433022.299
RidgeReg	1532.6778	456.3536	5665.915	836.5366	26.73521513	57221.365
LassoReg	1685.6235	435.9577	6882.569	892.8117	0.01594953	66036.598

Figure 3.37

We see that the result did not make much difference and the errors did not change much from the previous approach, the ridge regression approach and did not differ much.
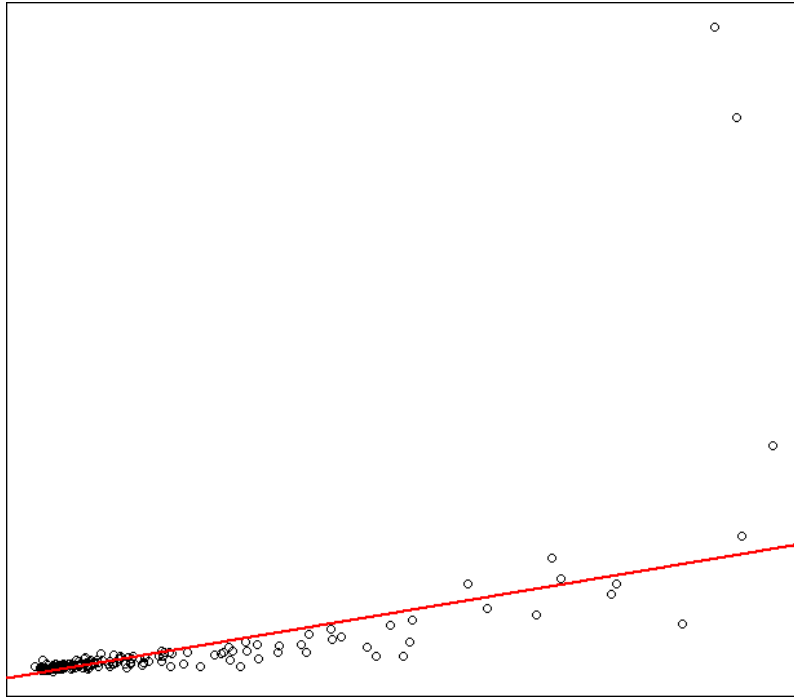
Figure 3.38 In this section, similar to the previous model, there are significant errors in high number of requests.

3.3.7. Decision Tree

We now turn to nonlinear approaches to see how these work for us; We start with the decision tree and implement this model using the powerful 'rpart' library.

The most important point in making a decision tree is the complexity or Cp of the tree. The more complex the tree, the more detailed the tree is. In the tree pruning section, by finding the best number for cp, putting it in function, and pruning the tree, we draw the minimum tree complexity.
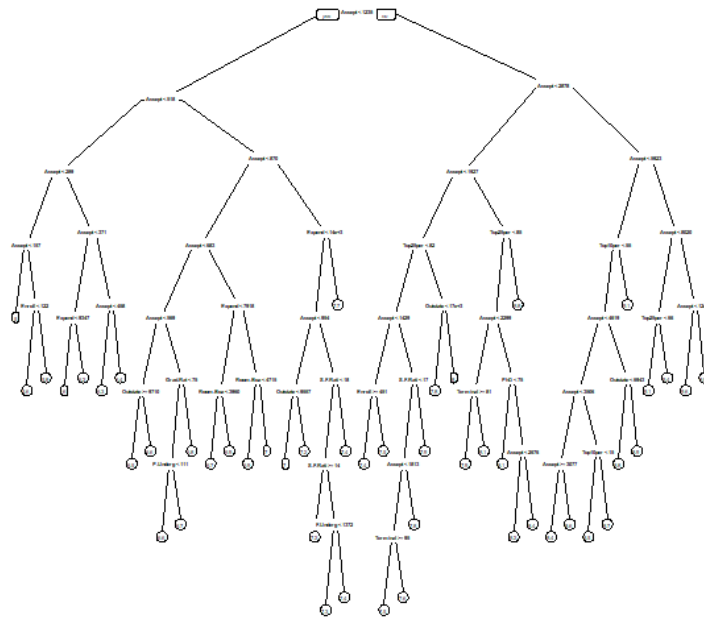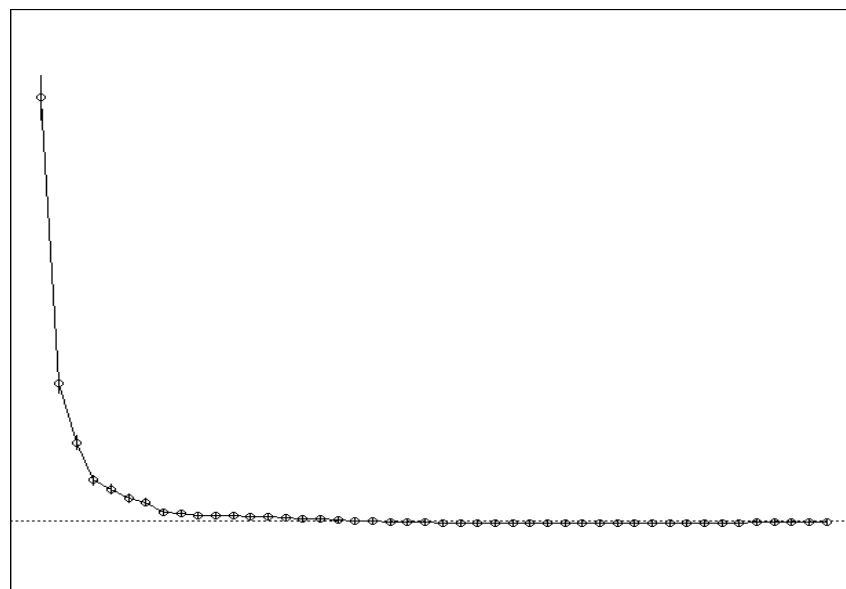
Figure 3.39


```
> tree_4$cptable[which.min(tree_4$cptable[,"xerror"])]
[1] 0.0003520591
>
```

Figure 3.40 The appropriate complexity value is found above and then we prune the tree using the 'prune.rpart' function.
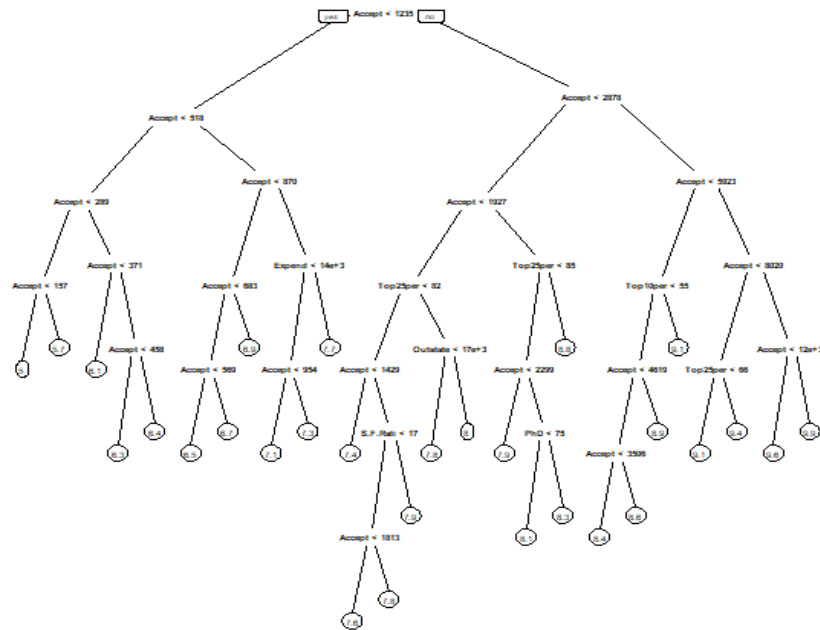
Figure 3.41

Figure 3.42

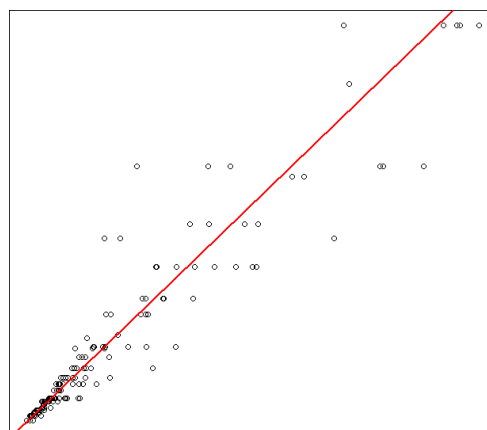|	Mean.of.AbsErrors	Median.of.AbsErrors	SD.of.AbsErrors	IQR.of.AbsErrors	Min.of.AbsErrors	Max.of.AbsErrors
LM	946.6732	555.6521	1251.9808	739.1078	2.72140412	8650.938
BestSub	1865.4376	422.4048	8243.6863	938.4780	3.00590749	77414.299
BestSubCV	1703.3551	423.3370	7093.3980	811.3109	0.02791144	67045.207
TrimmedBestSub	9552.1233	275.5476	51331.0821	509.5759	5.37114423	433022.299
RidgeReg	1532.6778	456.3536	5665.9154	836.5366	26.73521513	57221.365
LassoReg	1685.6235	435.9577	6882.5689	892.8117	0.01594953	66036.598
Tree	571.8707	214.0000	947.3851	452.2674	0.52382390	5344.907

Figure 3.43

It is as if the decision tree was able to make a significant improvement over its regression models and distributions and reduce absolute errors with a high slope.

In the following, we try to make more improvements by using bootstrap approaches or in other words, alternative sampling.

3.3.8. Bagging

|	Mean.of.AbsErrors	Median.of.AbsErrors	SD.of.AbsErrors	IQR.of.AbsErrors	Min.of.AbsErrors	Max.of.AbsErrors
LM	946.6732	555.6521	1251.9808	739.1078	2.72140412	8650.938
BestSub	1865.4376	422.4048	8243.6863	938.4780	3.00590749	77414.299
BestSubCV	1703.3551	423.3370	7093.3980	811.3109	0.02791144	67045.207
TrimmedBestSub	9552.1233	275.5476	51331.0821	509.5759	5.37114423	433022.299
RidgeReg	1532.6778	456.3536	5665.9154	836.5366	26.73521513	57221.365
LassoReg	1685.6235	435.9577	6882.5689	892.8117	0.01594953	66036.598
Tree	571.8707	214.0000	947.3851	452.2674	0.52382390	5344.907
Bagging	462.8395	142.6351	769.2172	440.3666	1.02175118	5373.069

Figure 3.44
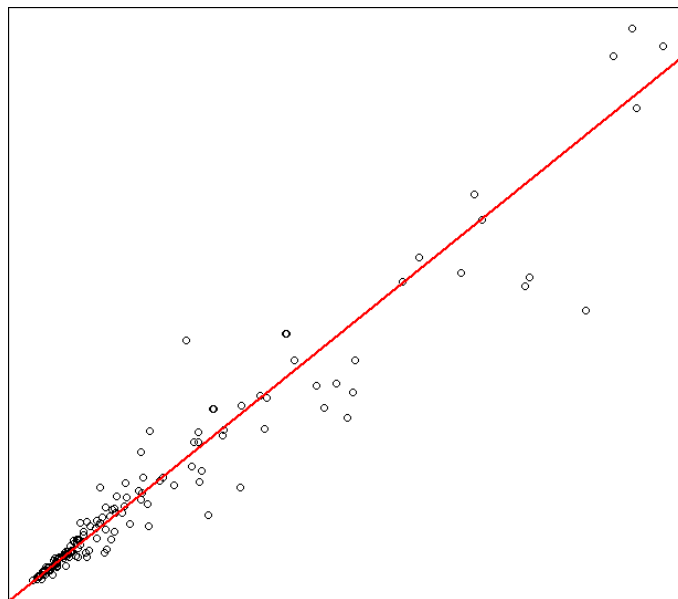
Figure 3.45

There is a good amount of improvement in reducing absolute errors, and so far we have been able to make relatively good predictions.
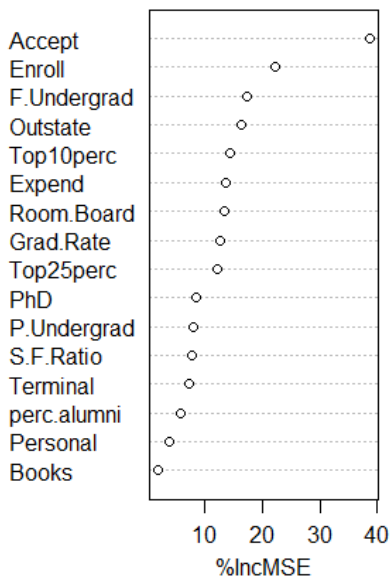
3.3.9. Random Forest

rt-1

43

Figure 3.46

#% IncMSE: is based upon the mean decrease of accuracy in predictions on the out of bag samples when a given variable is excluded from the model.

Algorithm:

K-fold Cross-Validation for feature selection

Do not forget to remove "Apps" [1] & "Log_Apps" [18]

#step = 0.75,

#step 1:16, step 2: example: round (0.75 * 16) = 12

#mtry: a function of number of remaining predictor variables to use

#as the mtry parameter in the randomForest call

#example: default: floor (sqrt (p)), floor (p / 3)

#recursive: whether variable importance is (re-) assessed at each step of variable reduction

Figure 3.47

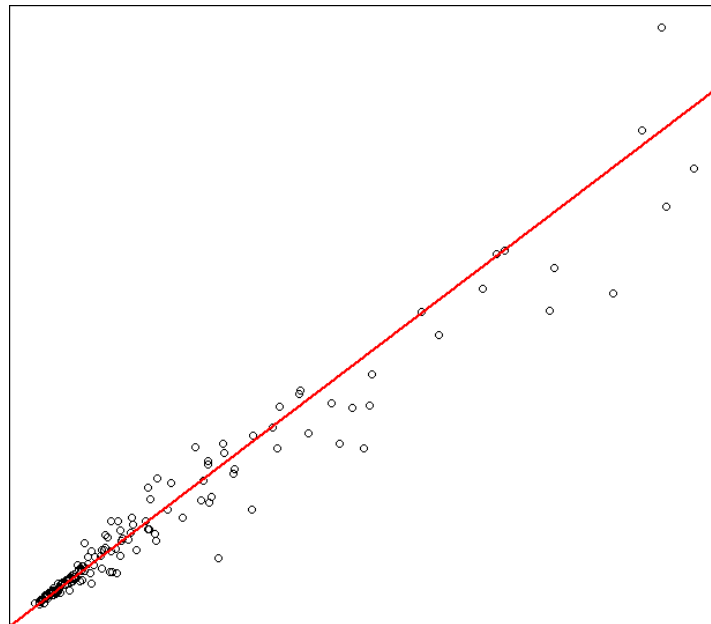|	Mean.of.AbsErrors	Median.of.AbsErrors	SD.of.AbsErrors	IQR.of.AbsErrors	Min.of.AbsErrors	Max.of.AbsErrors
LM	946.6732	555.6521	1251.9808	739.1078	2.72140412	8650.938
BestSub	1865.4376	422.4048	8243.6863	938.4780	3.00590749	77414.299
BestSubCV	1703.3551	423.3370	7093.3980	811.3109	0.02791144	67045.207
TrimmedBestSub	9552.1233	275.5476	51331.0821	509.5759	5.37114423	433022.299
RidgeReg	1532.6778	456.3536	5665.9154	836.5366	26.73521513	57221.365
LassoReg	1685.6235	435.9577	6882.5689	892.8117	0.01594953	66036.598
Tree	571.8707	214.0000	947.3851	452.2674	0.52382390	5344.907
Bagging	462.8395	142.6351	769.2172	440.3666	1.02175118	5373.069
RandomForrest	425.6269	133.7863	671.0528	467.9809	0.65580016	3924.897

Figure 3.48

The random forest model was also able to reduce our prediction errors and improve our model.

But the question is, can a model still be developed that further improves absolute errors in predicting unseen data?

3.3.10. Bagging w/ Trimmed Train

	Mean.of.AbsErrors	Median.of.AbsErrors	SD.of.AbsErrors	IQR.of.AbsErrors	Min.of.AbsErrors	Max.of.AbsErrors
LM	946.6732	555.6521	1251.9808	739.1078	2.72140412	8650.938
BestSub	1865.4376	422.4048	8243.6863	938.4780	3.00590749	77414.299
BestSubCV	1703.3551	423.3370	7093.3980	811.3109	0.02791144	67045.207
TrimmedBestSub	9552.1233	275.5476	51331.0821	509.5759	5.37114423	433022.299
RidgeReg	1532.6778	456.3536	5665.9154	836.5366	26.73521513	57221.365
LassoReg	1685.6235	435.9577	6882.5689	892.8117	0.01594953	66036.598
Tree	571.8707	214.0000	947.3851	452.2674	0.52382390	5344.907
Bagging	462.8395	142.6351	769.2172	440.3666	1.02175118	5373.069
RandomForrest	425.6269	133.7863	671.0528	467.9809	0.65580016	3924.897
TrimmedBagging	753.1846	158.4603	1601.2534	489.7588	0.77969593	8465.520

Figure 3.49

Unfortunately, by removing outlier data from the train data, No improvement in model performance occurred in data not seen or even got worsened.
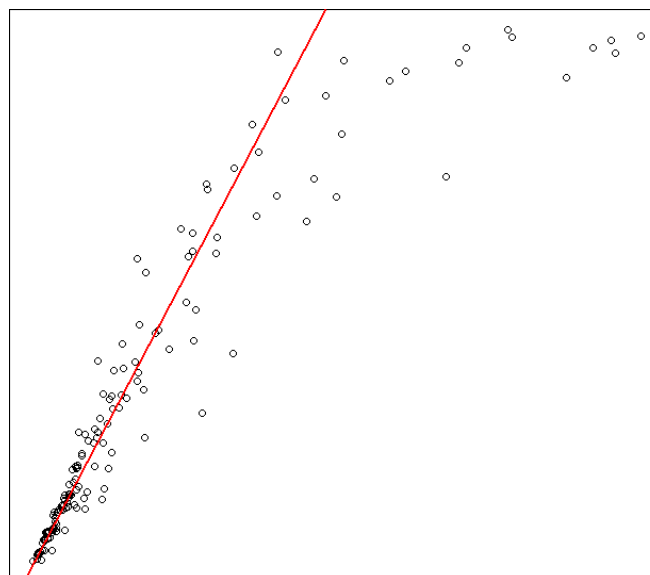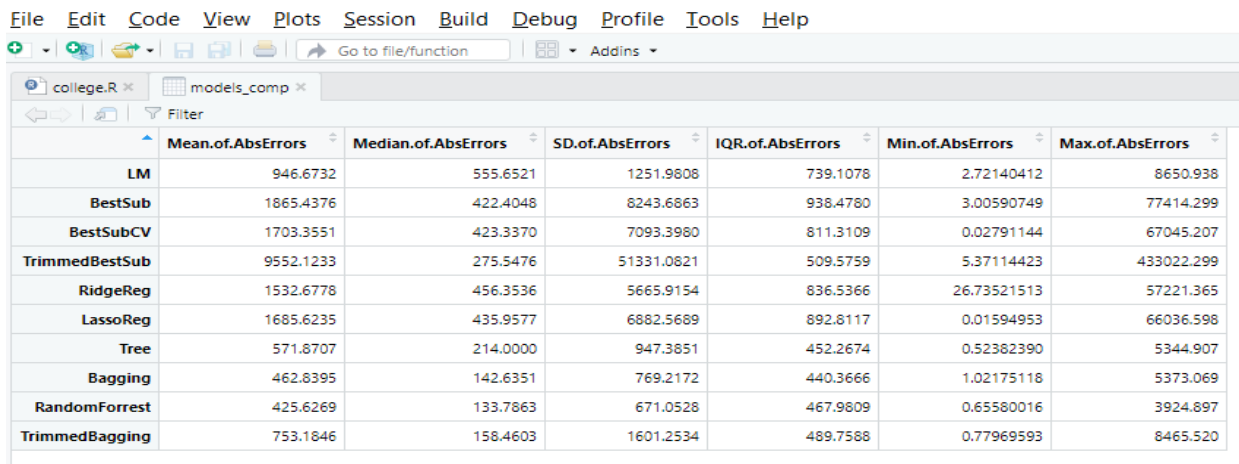

Figure 3.50

It seems that trimmed has caused us to not have a good forecast in the high number of requests and it has not performed well.

Chapter 4

4.1. Conclusion

	Mean.of.AbsErrors	Median.of.AbsErrors	SD.of.AbsErrors	IQR.of.AbsErrors	Min.of.AbsErrors	Max.of.AbsErrors
LM	946.6732	555.6521	1251.9808	739.1078	2.72140412	8650.938
BestSub	1865.4376	422.4048	8243.6863	938.4780	3.00590749	77414.299
BestSubCV	1703.3551	423.3370	7093.3980	811.3109	0.02791144	67045.207
TrimmedBestSub	9552.1233	275.5476	51331.0821	509.5759	5.37114423	433022.299
RidgeReg	1532.6778	456.3536	5665.9154	836.5366	26.73521513	57221.365
LassoReg	1685.6235	435.9577	6882.5689	892.8117	0.01594953	66036.598
Tree	571.8707	214.0000	947.3851	452.2674	0.52382390	5344.907
Bagging	462.8395	142.6351	769.2172	440.3666	1.02175118	5373.069
RandomForrest	425.6269	133.7863	671.0528	467.9809	0.65580016	3924.897
TrimmedBagging	753.1846	158.4603	1601.2534	489.7588	0.77969593	8465.520

Figure 4.1

Comparing the performance results of the models, we see that regression and its distributions could not have significant performance due to non-linearity, non-correlation and multicolinearity problems between variables, and it is obvious that decision tree models and Boosting and Bagging approaches have a significant improvement in results.

Finally, the Random Forest approach with less error mean, less median error, less error standard deviation, IQR error with less difference in the third category, less error max and an appropriate min of error is generally known the best model for detecting and predicting the number of annual university applications of the United States.

So we managed to build a model that can predict, with minimal error, the total number of applications that students send to each university each year for admission.

4.2. Upcoming works

So far we have seen clearly how we constructed our proposed model and observed the predicted values along with the percentage of prediction error. However, due to the nature of modeling and predictability, it can be used in almost all fields and projects that perform analysis work using modeling on numerical (non-textual) data; Especially in academic, military, medical, educational and ... projects.

The challenge that education faces every year is the inability and incapability to estimate the number of enrollees of different grades in each school. As an example, we can estimate the number of students applying to a school for tenth grade experimental or math. But the features and predictors that we can use and have in this issue can be decisive factors in reducing the percentage error of prediction values. Sample questionnaires such as the academic guidance questionnaire that is filled out by students in the previous grade can also be used to collect data.

References

1 - https://www.kaggle.com/faressayah/college-data

2 - https://online.stat.psu.edu/stat500/

3 - http://www.r-tutor.com/elementary-statistics

4 - https://mml-book.github.io/book/mml-book.pdf

5 - https://analica.ir/correlation/

6 - https://analica.ir/statistical-thinking/

7 - https://leanpub.com/rprogramming

8 - R Programming for Data Science, By Roger D. Peng https://leanpub.com/rprogramming

9 - R for Data Science, By Garrett Grolemund, and Hadley Wickham https://r4ds.had.co.nz/

10 - Hands-On Machine Learning with R, By Bradley Boehmke & Brandon Greenwell: https://bradleyboehmke.github.io/HOML/

11 - Time Series Forecasting: Principles & Practice in R, By Rob Hyndman, and George Athanasopoulos: https://otexts.com/fpp2/

12 - Advanced R, By Hadley Wickham: https://adv-r.hadley.nz/

13 - The Elements of Statistical Learning, By Jerome H. Friedman, Robert Tibshirani, and Trevor Hastie: https://web.stanford.edu/~hastie/Papers/ESLII.pdf

14 - https://analica.ir/statistical-significance/

15 - Data Preprocessing in Data Mining

16 - Multivariate Data Analysis

17 - Chapters one, two, three, four, six, ten, eleven and twelve of Hands-On Machine Learning with R

https://bradleyboehmke.github.io/HOML/

18 - Chapters one, two, six and eight of An Introduction to Statistical Learning

http://faculty.marshall.usc.edu/gareth-james/ISL/ISLR%20Seventh%20Printing.pdf

19-https://www.linkedin.com/pulse/data-scientists-how-talk-your-subject-matter-experts-alastair/?trackingId=7Dg%2BN1B2e2xVd%2BsIb5uLOg%3D%3D

20-https://spectrum.ieee.org/tech-talk/artificial-intelligence/machine-learning/understanding-causality-is-the-next-challenge-for-machine-learning

21 - The first, third, ninth and tenth chapters of The Elements of Statistical Learning

https://web.stanford.edu/~hastie/ElemStatLearn/

22 - https://venturebeat.com/2020/11/14/you-cant-eliminate-bias-from-machine-learning-but-you-can-pick-your-bias/