Final Report

Jisu Kim

University of Minnesota - Twin cities
STAT4052 (001)

Sara Algeri

May 12, 2021

Introduction

In this final project, I wanted to find out which factors contribute most to the failure of affordable housing projects. To be a little more specific, our ultimate goal was to predict and analyze the probability of failure for future projects offering affordable housing, which was the major of San Francisco's goal, and to build a helpful predictive model. Before I started the analysis, I looked at the data for this project. The dataset contained information about several projects aimed to provide affordable housing in San Francisco. There was a total of 28 variables in the data; from the 28 variables, the response variable was 'Failing' variable. This was because the variable 'Failing' is the most important factor in determining whether a reasonably priced housing project will fail or not. I was required to select at least six variables in this project; out of the remaining 27, I decided to consider and analyze 7 variables (ProjectType, HousingTenure, StudioUnits, OneBd, TwoBd, ThreeBd, and FamilyUnit) that would have a significant impact on the failure factor. Although it was good to consider all variables, I believed that the analysis and prediction through seven variables would be an appropriate choice. For when I apply all variables to statistical models, an overfitting problem can occur, and a strong correlation between variables interferes with accurate analysis. In addition, I processed 'missing values' in one data cleaning process in two ways to make this data predictable. The first was to exclude categorical variables for 'missing values' and replace continuous variables with the median. The second way was to replace categorical variables and continuous variables by the means of iterative regression for 'missing values'. I wanted to apply the seven variables through the two cleaned data sets mentioned previously. I chose four different statistical models along with the response variable 'Failing' to see which model would produce the most accurate predictions through the ROC curve and the prediction accuracy rate. The statistical methods I considered for the prediction were the Logistic regression, KNN, Decision tree (Classification tree), and Random forest. In this project, I tried to determine which model would most accurately analyze the objectives through using two different datasets and determine which statistical method and data should be applied to this project. Through our final consideration of the data and methods, I expect to contribute to providing affordable housing by identifying and reducing the probability of failure and increasing the probability of success for future projects.

Methods

As I mentioned briefly in the introduction, I would like to discuss the logistic regression first. The logistic regression is the model which can be applied to categorical variable as a response variable, and mixed variable first as an independent variable. The logistic regression can be specifically divided by three logistic regression models: binomial, ordinal, and multinomial logistic regression. Among the three models, I chose the binomial logistic model to analyze the project's data, because I considered the 'Failing' variable, which had "0" (not failing) or "1" (failing), to be the response variable. The binomial logistic regression can use the three link functions: logit, probit, and complementary log-log. I decided to use logit link which has the formula $\log\left(\frac{\pi}{1-\pi}\right) = X\underline{\beta} = \theta$, and $\pi = P(y=1|x)$, which means the probability of "Failing". In addition, there are several advantages in using the logistic regression model. First, it is "easier to implement, interpret, and very efficient to train"; second, it "makes no assumptions about distributions of classes in feature space"; third, it has "good accuracy for many simple data sets, and it performs well when the dataset is linearly separable" (Saraswat, 2020).

The second model I used in this project was KNN (K-Nearest Neighbors) classifier. For KNN, both response variable and independent variable can be applied categorical and continuous variable. This was the point I believed that through KNN statistical methods, our data can be analyzed and predicted. KNN can predict the probability $\hat{P}(Y=q|X=x_0) = \frac{1}{K}\sum_{i\in N0} I(y_i = q)$,where I(•) is the indicator function and N_0 is the set of the points, by identifying K points in the training data which is closest to the x_0 through using Euclidean distance under the condition given an observation with $X=x_0$. Finally, it applies Bayes rule and classifies the test observation with $X=x_0$ to the class q which has the largest probability

($\hat{P}(Y=q|X=x_0)$). What I needed to be careful about using KNN was that K of neighboring points affects the classifier's performance, hence it is necessary to choose carefully for K. Moreover, KNN has a variety of advantages. First, it is easier to implement and interpret. Second, it has no assumption which different with linear regression. Third, it has great performance which contains the prediction accuracy if training data set is sufficient.

The decision tree is divided into a regression tree and a classification tree; the regression tree is used when the response variable is a continuous variable, and the classification tree is used when the response variable is a categorical variable. I decided to analyze through the classification tree because the response variable was categorical. Classification trees are very similar to regression trees, however there are significant differences in the process of growing the trees. The classification tree focuses on minimizing the Gini index or the Cross-Entropy index when dividing data using cut points of each variable at each node. In addition, the classification tree predict observation (x* and y*) with the class whose estimated posterior probability is the largest for the region to which (x* and y*) belongs. Lastly, the classification tree evaluates the prediction performance by the means of the test error rate. Nevertheless, regression tree and classification tree both stop growing the tree when a specific rule is satisfied. The two trees also prune the tree by choosing the subtree for minimizing the cost complexity pruning. Most people prefer to use the decision tree because of the following reasons: it is easy to explain and interpret and it can be easy to "handle qualitative predictors without the need to create dummy variable".

The last method we decided to use was the random forest. Through bootstrap sampling from the training data, the random forest gathers various trees and aggregates the most common prediction results into the final prediction result. Random forest is also a model that can improve the disadvantages of decision trees, reducing overfitting problems and strong correlation between variables. These features lead to an improvement in the prediction rate. Unlike decision trees, random forests grow trees by selecting only a few variables from all predictors; it randomly selects $\sqrt{m}$ or $\frac{m}{3}$ variables from all predictors which is called 'm'. This procedure helps to improve the drawback of the decision tree that I mentioned previously. In conclusion, the random forest is similar but has more advantages than the decision tree.

Results

As mentioned in the introduction, I found 'missing values' in the process of cleaning the data used in this project and handled it in two ways. From the two ways, I decided to analyze 'missing values' through the method of excluding categorical variables and replacing continuous variables with the median. I analyzed and predicted through the following four methods: Logistic regression, KNN, Decision tree, Random forest. Prior to the analysis, I divided the data into 80% of training set and 20% of validation set to obtain the correct prediction accuracy rate. The results for the analysis were as follows. First, we analyzed the data through logistic regression. To do an accurate analysis, I found out which model was the most appropriate model fit; the model selection was done through AIC (Appendix A). The final model that we obtained through AIC included the Project Type, Family Unit, Housing Tenure, One Bd, and Two Bd as independent variable; it represented glm (Failing ~ ProjectType + FamilyUnit + HousingTenure + OneBd + TwoBd, family=binomial, data=Train) in R. Below the result was the final model summary.

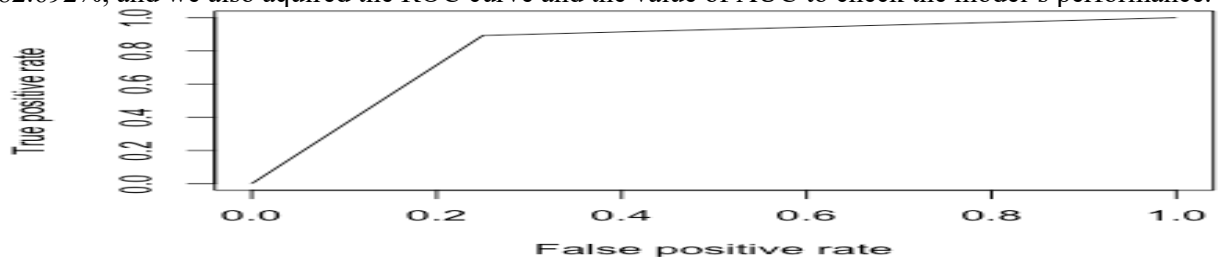
```
## glm(formula = Failing ~ ProjectType + FamilyUnit + HousingTenure +
##     OneBd + TwoBd, family = binomial, data = Train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.9106  -0.4684   0.1111   0.4214   2.9973
##
## Coefficients:
```

```
##                                  Estimate Std. Error z value Pr(>|z|)
## (Intercept)                     5.319e+00  1.382e+00   3.849 0.000118 ***
## ProjectTypeRehabilitation      -4.097e+00  7.254e-01  -5.648 1.62e-08 ***
## FamilyUnit                     -3.097e-02  6.144e-03  -5.041 4.63e-07 ***
## HousingTenureOwnership, Rental  1.351e+01  1.455e+03   0.009 0.992595
## HousingTenureRental            -2.943e+00  1.343e+00  -2.191 0.028487 *
## HousingTenureUnknown           -2.775e+00  1.424e+00  -1.949 0.051266 .
## OneBd                          -4.151e-02  1.651e-02  -2.514 0.011940 *
## TwoBd                           3.774e-02  1.540e-02   2.451 0.014238 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 275.09  on 208  degrees of freedom
## Residual deviance: 130.15  on 201  degrees of freedom
## AIC: 146.15
```

Furthermore, I acquired the odds ratio with the value obtained from the model summary and made a meaningful interpretation. For example, $e^{\beta \boldsymbol{ProjecttypeRehabilitation}} = 0.02 = \frac{1}{50}$. It is 50 times more likely to fail the future housing project if project type is new construction rather than rehabilitation while all other variables are fixed. This was also confirmed once again by obtaining direct probabilities using the positive function.

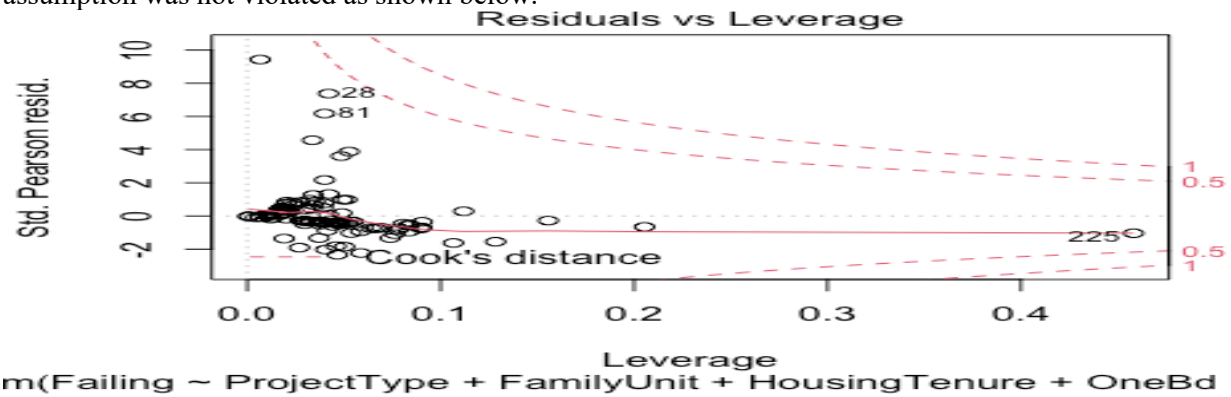
```
predict(Final.model,data.frame(ProjectType="New
Construction",FamilyUnit=38.5933,HousingTenure="Ownership", OneBd=10.82297,
TwoBd=10.94737),type="response")

##         1
## 0.9835011

predict(Final.model,data.frame(ProjectType="Rehabilitation",FamilyUnit=38.5933,HousingTenure="Ownership", OneBd=10.82297, TwoBd=10.94737),type="response")

##         1
## 0.4975972
```

The above results showed that when the project type was new construction, the probability of failure was 98.35%, and when it is rehabilitation, the probability of failure was 49.76%. Thus, it could be seen that the probability of failure of the new construction project type was much higher than that of rehabilitation. Confusion matrix was created by comparing the validation set and fitted model (Appendix B), and prediction error rate was obtained for the logistic regression model; the prediction error rate was 82.692%, and we also aquired the ROC curve and the value of AUC to check the model's performance.



The graph above represented the ROC curve of the final model that we decided, and the value of AUC was 0.821. Hence, our final model seemed to perform well for the data. Finally, I needed to check that the fitted model did not violate any assumptions; the assumptions I checked was deviance, Pearson- χ^2, and cook's distance. Through the deviance test the result was 0.99, this meant the model provided a good fit for the data. Nevertheless, through the Pearson- χ^2 test, the result was 1.44e-05, which indicates that the

model did not provide a good fit for the data because it was less than 0.05. Lastly, the cook's distance assumption was not violated as shown below.



m(Failing ~ ProjectType + FamilyUnit + HousingTenure + OneBd

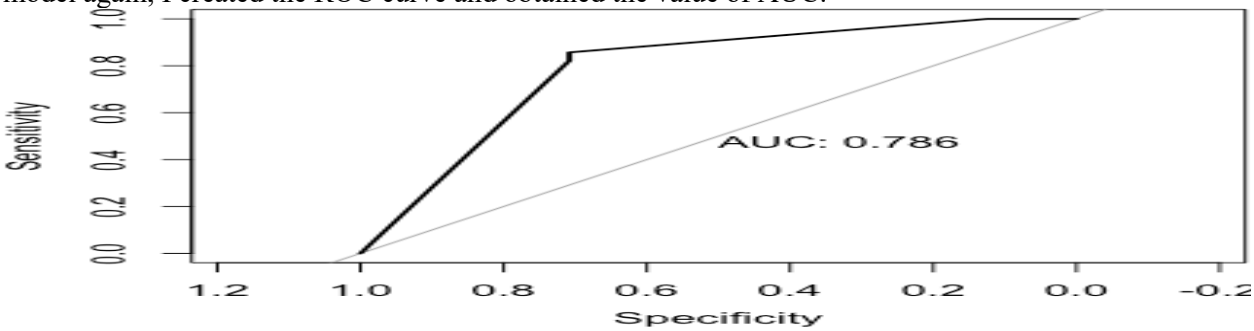
There were no points that exceeded the 0.5 dashed line; this shows it did not have any influential points. In conclusion, even though logistic regression model performed well with the data, it violated the Pearson- χ^2 assumption. Therefore, the model was not reliable.

I then analyzed the data through K-Nearest Neighbors. To fit the model for the KNN method, I changed the categorical variable (Project type and Housing Tenure) to a dummy variable, and I divided the data into 80% of training set and 20% of validation set. As noted earlier, since KNN tends to affect the model's performance depending on K, I decided to fit the models where K is 3,5, and 10 (Appendix C).

The confusion matrices were created by comparing the validation set and the fitted models (Appendix D), and the accuracy rate was obtained for the three models. I compared the error rates from the fitted models to find the best performance model. The chart below shows the error rates that I found.

K	Error Rate
3	0.1923077
5	0.1730769
10	0.25

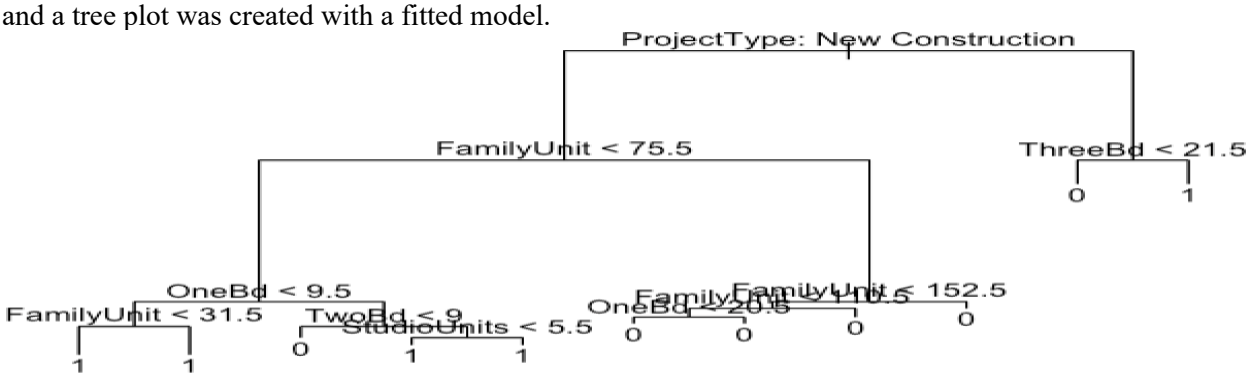
From the chart of the prediction error rates, I found that when K was 5, the fitted model would yield the lowest prediction error rate. Hence, I decided the fitted model with K= 5 to the final model for the KNN method, and the fitted model had an 82.69% of prediction accuracy rate. To check the performance of the model again, I created the ROC curve and obtained the value of AUC.



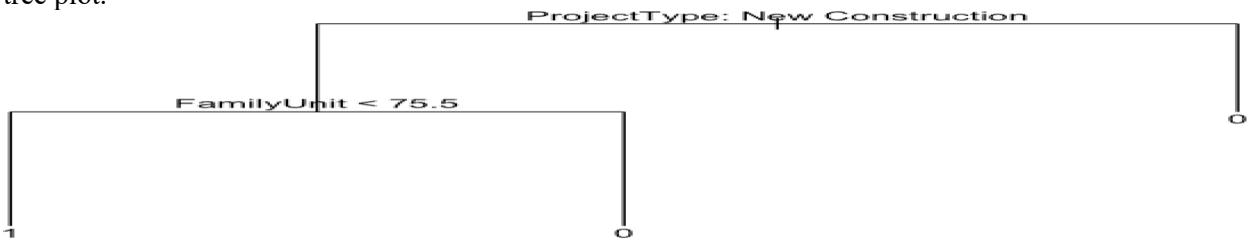
The value of AUC was 0.786, which meant the final model for KNN performed good for the data.

I then analyzed the data from the decision tree (classification tree). Unlike the KNN, there was no need to modify the data; I used the data that was created before fitting the logistic regression model. Both the categorical variable 'Failing' and the 7 predictors were fitted using a tree function with training data,

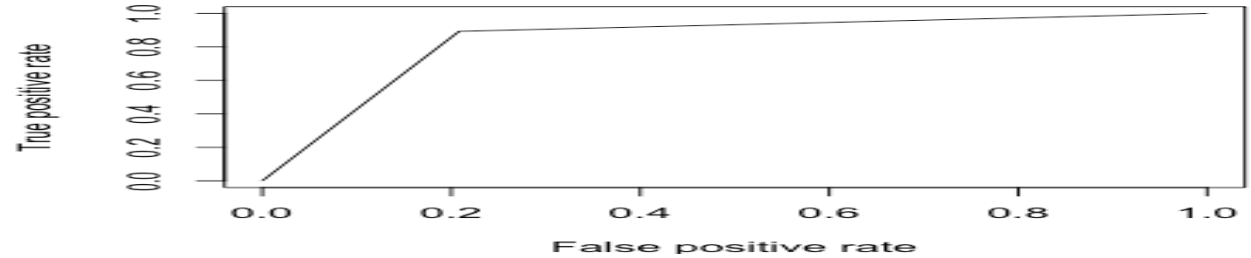
and a tree plot was created with a fitted model.



The plot above was created by using a fitted tree model, the plot could be understood through the information of the fitted model (Appendix E); the plot can be interpreted from the information. For example, if the project type is rehabilitation and three bed is less than 21.5, the probability of not failing the project was 100%; if the project type is new construction and family unit is less than 75.5 and one bed is less than 9.5 and family unit is greater than 31.5, the probability of failing the project was 62.5 %. The above tree was also significant enough, but if the tree is too large the overfitting problem would occur. I decided to prune the tree to reduce the probability of overfitting; the small tree had a small variance, which reduced the risk of overfitting. During the process of pruning, I identified the optimal number of leaf nodes through cv.tree function (Appendix F). The optimal number of leaf nodes we checked was 3, and I fitted a new pruned tree model with 3 leaf nodes and created a tree plot. The following is the pruned tree plot.

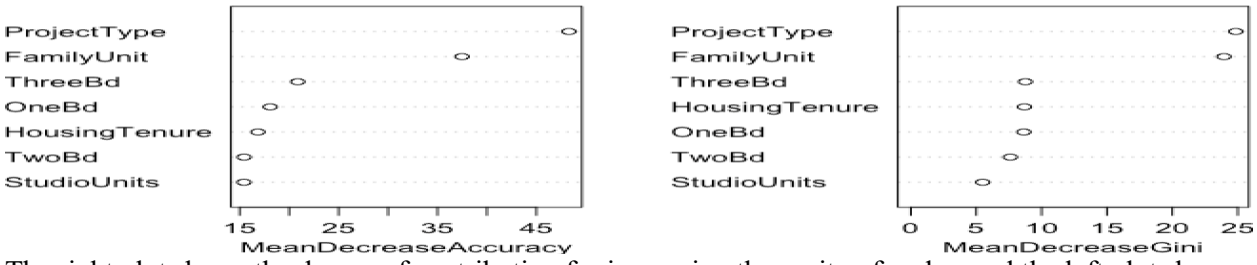


This plot can also be interpreted as follows through the information in the pruned tree model (Appendix G): if project type is new construction and family unit is less than 75.5, the probability of failing the project was 93.13%. Furthermore, the confusion matrices were created by comparing the validation set and the fitted models (Appendix H), and the accuracy rate was obtained for the decision tree. The prediction error rate was 84.615%; to check the model's performance, we also got the ROC curve and the AUC value.

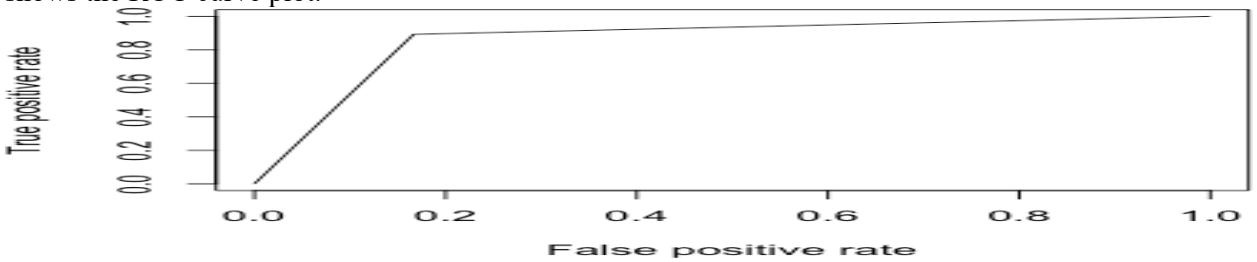


Above, the graph shows the ROC curve for the decision tree, and the AUC value was 0.842. Hence, the final model for the decision tree perfomed great with the data.

I analyzed the data from the random forest and used the same data (training and validation set), which was used for the logistic regression and the decision tree). Both the categorical variable 'Failing' and the 7 predictors were fitted using a random forest function with the training data. As I mentioned in the introduction, I needed to decide how many variables the random forest model selects from all predictors when fitting the model; I selected 2 for this, because $\frac{7}{3}$ is around 2. After fitting the random forest model, the variable importance plot was created like below.

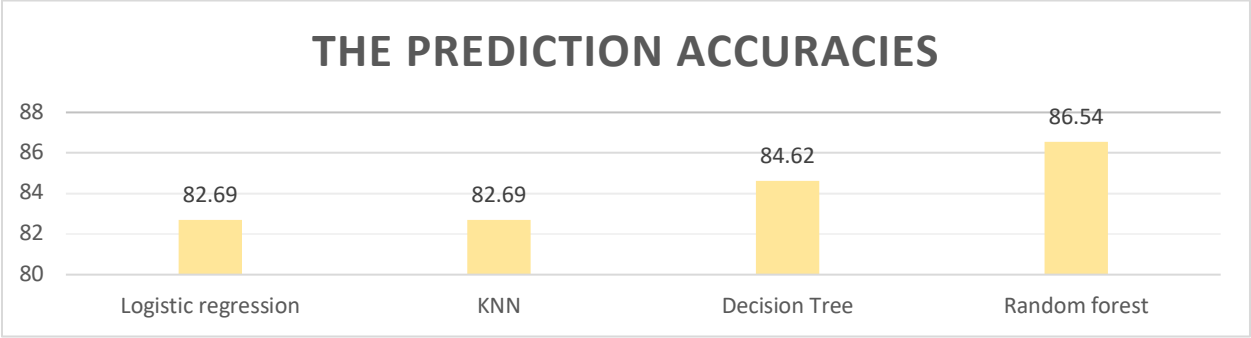


The right plot shows the degree of contribution for improving the purity of nodes, and the left plot shows the degree of contribution for the prediction accuracy. Based on the two plots, the Project Type and Family Unit seemed to be a very important variable to perform the model. It could also be interpreted that if the project type is not selected, the prediction accuracy decreased by 48.29 and the purity of nodes decreased by 24.85. The confusion matrices were created by comparing the validation set and the fitted models (Appendix I), and the accuracy rate was obtained for the random forest. The prediction error rate was 86.54%, and to check the model's performance I also got the ROC curve and the AUC value. Below shows the ROC curve plot.



The AUC value was 0.863 out of 1, indicating that the model performed well on the data.

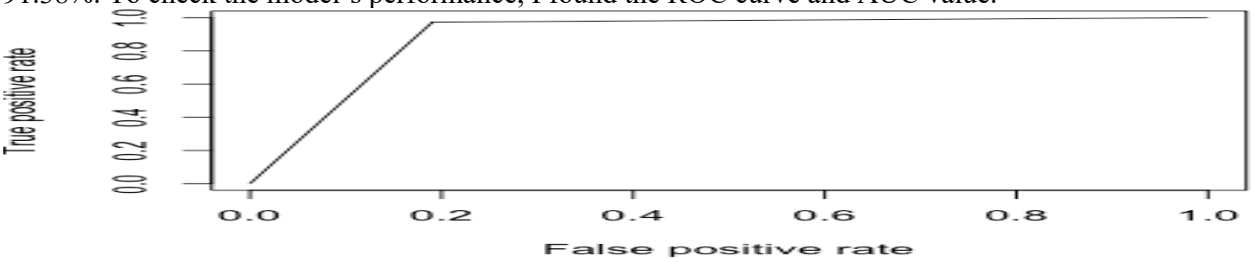
After the first data analysis using the four statistical methods, I was able to identify the following prediction rate.



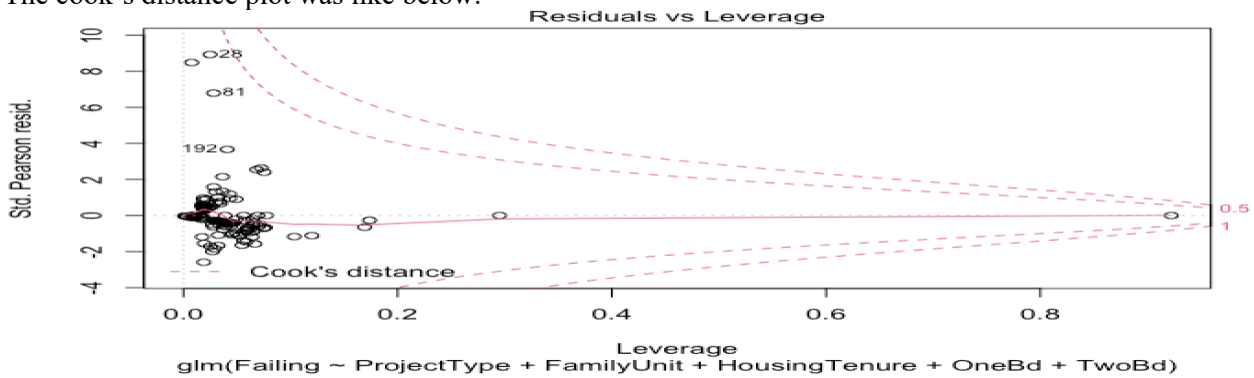
Based on the above graph, logistic regression and KNN had approximately the same prediction accuracy rates. Also, the decision tree had 84.62% accrual rates, an improvement of 2%. Finally, the random forest showed 86.54% accuracy rate, 2% higher than the decision tree, and it had the highest accuracy rate in the data which was the first method of imputing the 'missing values'. The AUC values were slightly different, but they had almost similar trends, and the random forest also had the highest AUC value.

Among the methods of imputing 'missing values', I analyzed the data using the second method of alternating categorical variable and continuous variable as an iterative regression method. In addition, I analyzed the data by using the same 4 analysis methods, and I split the data into training data and validation data in the same ratio that I did earlier (80:20). Through the AIC, I obtained same model with the logistic regression used from the first data (Appendix J). However, the summary output was different (Appendix K), demonstrating that the inference of the result for the model should be different with the first analysis of logistic regression. The confusion matrix (Appendix L) and prediction accuracy was made by comparing the fitted model validation data; the accuracy rate for the logistic regression was

91.38%. To check the model's performance, I found the ROC curve and AUC value.



The graph above is the ROC curve, with the AUC value of 0.891. Hence, I concluded that the logistic regression performed well with the data. Lastly, I checked the three assumptions: deviance, Pearson- χ^2, and cook's distance. The deviance test's result was 0.99 and the Pearson- χ^2 test's result was 1.44e-06. The cook's distance plot was like below.

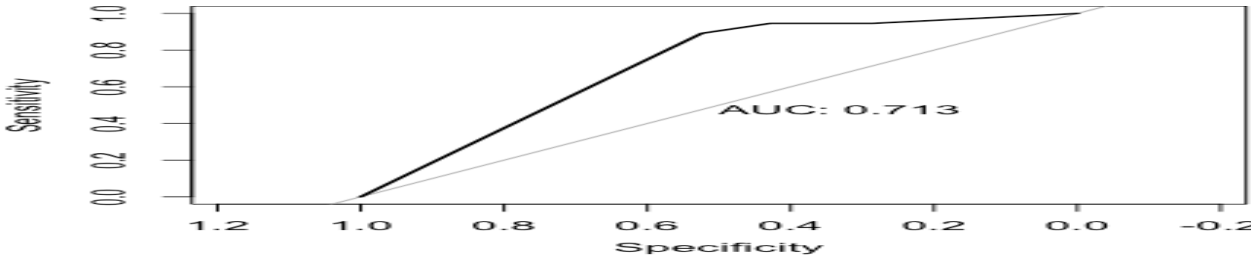


The logistic regression model violated the Pearson- χ^2 assumption, and the other two assumptions were not violated. Therefore, the logistic regression was not reliable like the first logistic regression model.

The data was then analyzed through the KNN method. I also converted the categorical variable to a dummy variable, and the data was divided into 80% of training data and 20% of validation data. I fitted three KNN models each with a K value of 3, 5, and 10 to find a most accurate model, and the confusion matrices and prediction error rates were created. Below shows the error rates for the three models.

K	Error Rate
3	0.1206897
5	0.137931
10	0.1206897

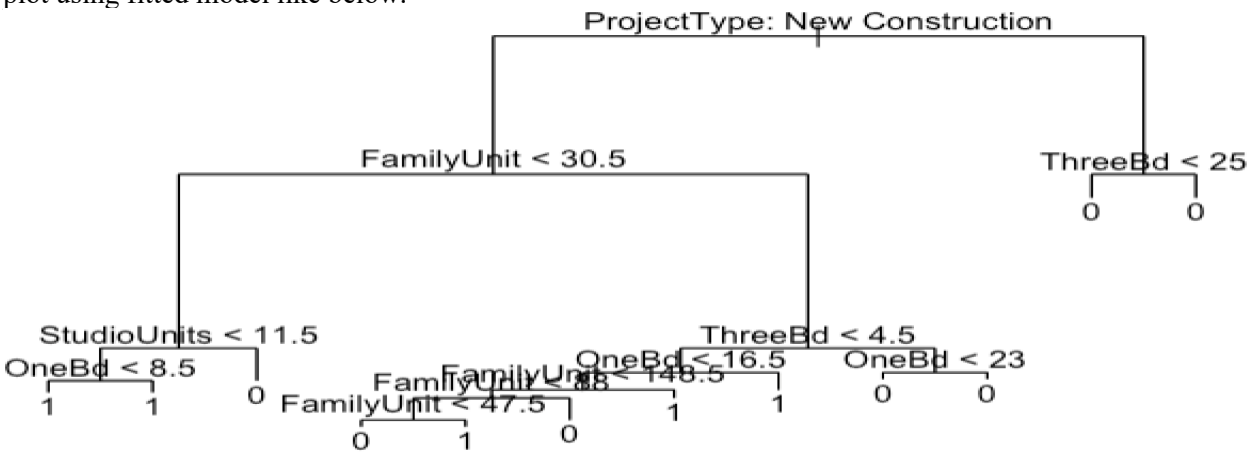
According to the above chart, I found that when k was 3 and 10, the accuracy rates are highest (87.83%). I decided to choose the final model with k=3. To check the final model's performance, I created the ROC curve and found the AUC value.



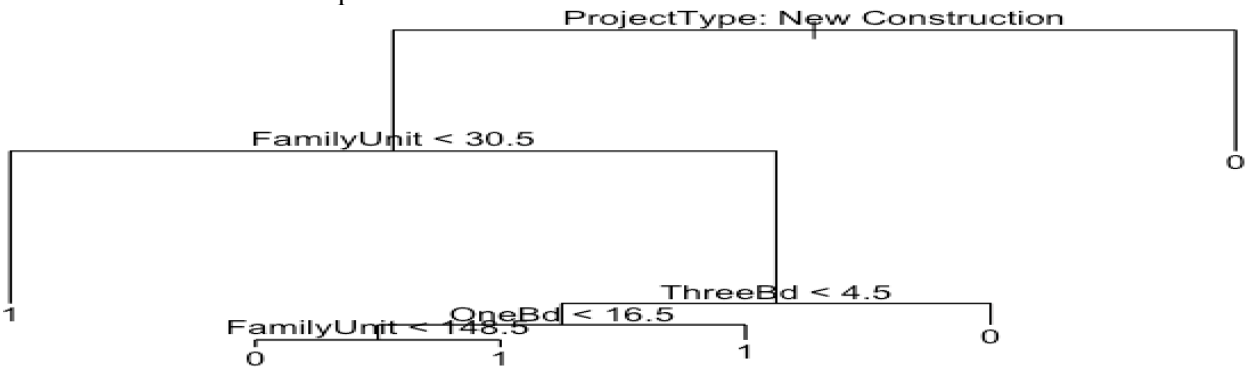
The graph above is the ROC curve with the AUC value of 0.713. This shows that the model performed not bad with the data.

Then, through the decision tree, I analyzed the data using the data that was created before fitting the logistic regression model. The categorical variable 'Failing' was fitted as the response variable and the 7 predictors were fitted as independent variable using a tree function with training data, and I made a tree

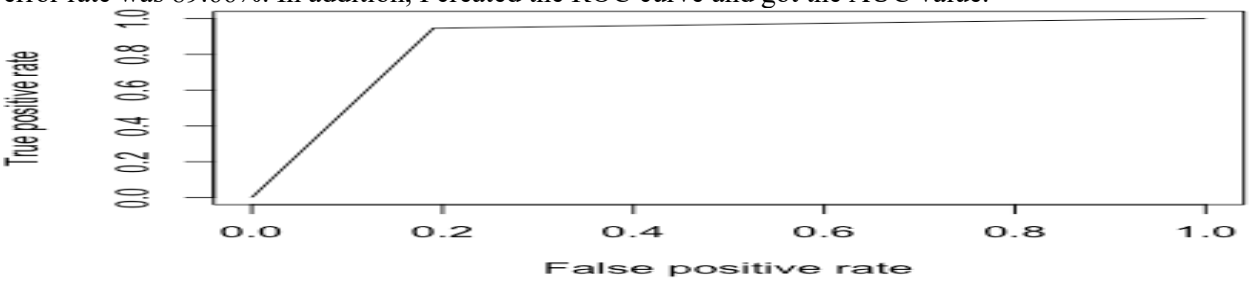
plot using fitted model like below.



Through the plot and with the information of the fitted model, I decided to prune it to reduce the overfitting problem. I identified the optimal number of leaf nodes when creating the plot by using cv.tree function (Appendix M), and found the optimal number of leaf nodes to be 6. The pruned tree model with 6 leaf nodes created the tree plot below.

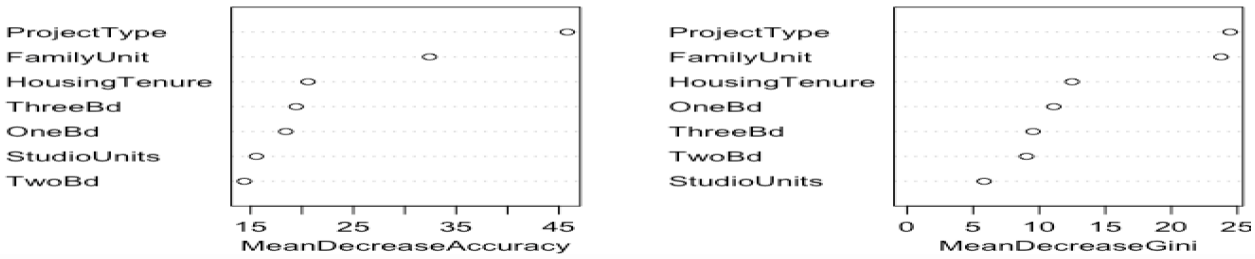


This plot could be interpreted with the information of the pruned tree model (Appendix N). The error rate and confusion matrix (Appendix) were obtained by comparing the fitted model and validation data; the error rate was 89.66%. In addition, I created the ROC curve and got the AUC value.

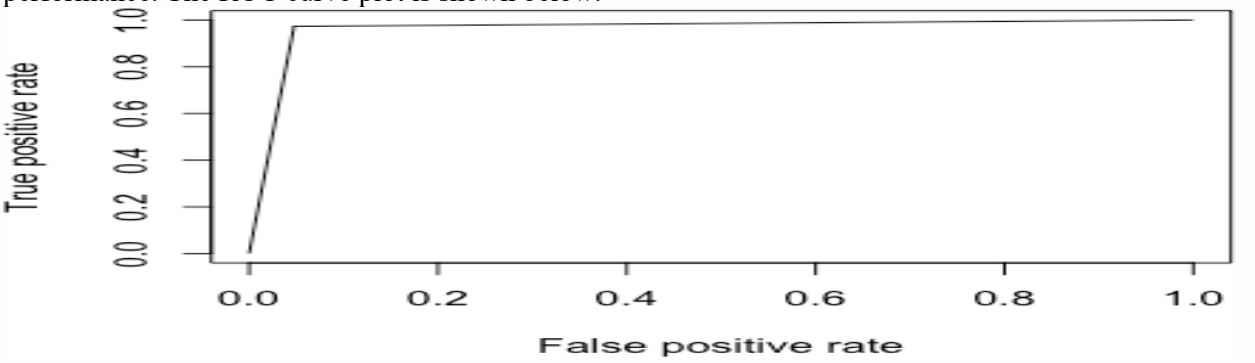


The plot above is the ROC curve with an AUC value of 0.878. Hence, the decision tree model was great to perform on the data.

Through random forest, I then used the same training and validation data as the logistic regression and the decision tree. The categorical variable 'Failing' was fitted as the response variable and the 7 predictors were fitted as independent variable using a random forest function with the training data. In addition, before fitting the random forest model, I decided to choose the number of optimal selective variable from all the variables; I selected 2 for this because $\frac{7}{3}$ is around 2. Through the fitted random forest model, I could create the variable importance plot.

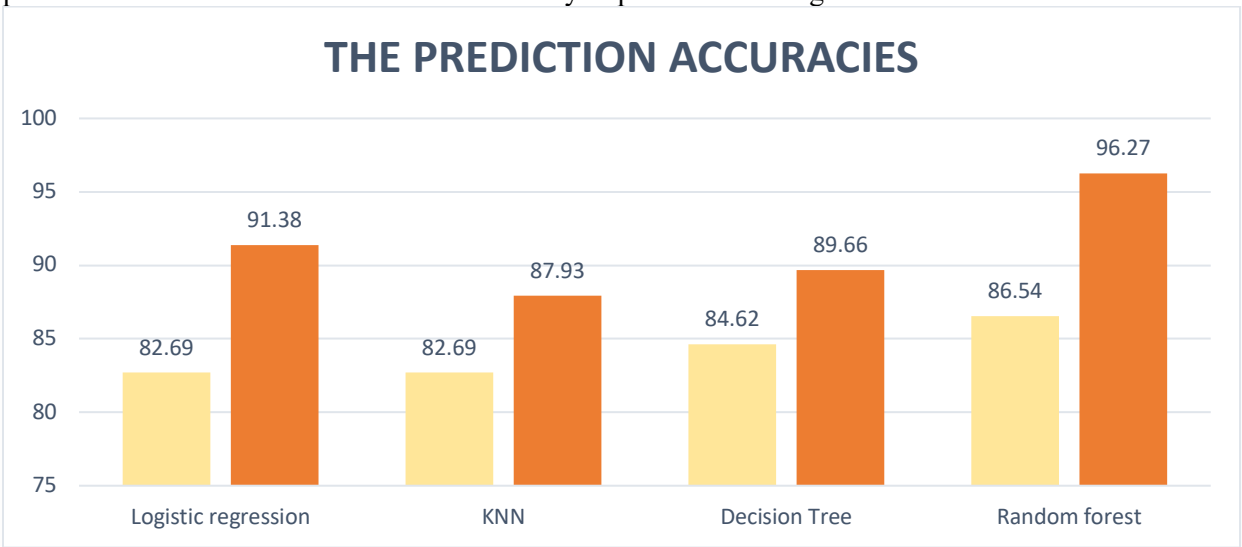


The graph above can be interpreted as previously described, and the Project Type and Family Unit variables are very important for this model based on the graph. Through comparing the fitted model and validation set, I created the confusion matrix (Appendix O) and the prediction error rate; the error rate was 96.55. Furthermore, I made the ROC curve and obtained the AUC value to check the model's performance. The ROC curve plot is shown below.



The value of AUC was 0.963, and I could conclude that this model performed very well on the data.

After the second data analysis using the four statistical methods, I was able to compare the prediction rates of the first and second data analysis performed through the four statistical methods.



Based on the above graph, the yellow bar represents the first data analysis, and the orange bar represents the second data analysis. Comparing the two bars, the analysis' accuracy rates using the second data were approximately 7% higher on average than those of the first. In conclusion, the second data that imputed missing values using an iterative regression showed better analytical performance. In particular, the random forest had the highest prediction rate in the first and second data; it obtained a 96.27% prediction accuracy rate from the second data. Therefore, the most accurate statistical analysis can be achieved by analyzing and predicting through the random forest with the data imputing 'missing value' as a second method for this project.

Discussion

At the end of the project, I created two data by imputing the missing value in two ways and analyzed and predicted the data through 4 statistical methods. Through this analysis and prediction process, the random forest model proved to be the most accurate model to make predictions; the model also had the highest accuracy rate, as well as the highest ROC curve plot and AUC value. In particular, the model's ideal performance was shown when analyzing data imputing 'missing values' through an iterative regression. Hence, through the random forest, I predicted and analyzed the probability of failure of future projects providing affordable housing, which was the main goal of the major of San Francisco. The variable importance plot also confirmed that Project Type and Family Unit variables have the greatest impact on the failure of the housing business. Even though the random forest model had the highest accuracy rate and performed great with the project's data, there could be more steps within this project to further improve the prediction model and accuracy. The random forest's disadvantage was that it is difficult to interpret the outcome. Hence, I need to figure this problem out. Furthermore, there are other statistical methods: bagging and boosting which has the potential to improve the prediction accuracy rate. Analyzing and predicting the data through those models are also necessary. It will increase the quality of the project to fulfill the goal. In conclusion, through this examination of statistical analysis and prediction methods, I expect to accurately understand the factors which can impact the probability of failure for future projects and contribute to providing affordable housing by increasing the probability of project success.

Reference

Saraswat, M. (2020, August 20). *Practical Guide to Logistic Regression Analysis in R*. HackerEarth Blog. https://www.hackerearth.com/blog/developers/practical-guide-logistic-regression-analysis-r/

GeeksforGeeks. (2020, September 2). *Advantages and Disadvantages of Logistic Regression*. https://www.geeksforgeeks.org/advantages-and-disadvantages-of-logistic-regression/#:%7E:text=Logistic%20regression%20is%20easier%20to,of%20classes%20in%20feature%20space.

Obtaining an AUC value from the KNN function. (2018, April 19). Stack Overflow. https://stackoverflow.com/questions/49915843/obtaining-an-auc-value-from-the-knn-function

Appendix

Appendix A:

```
step(model0,scope=list(upper=model1,lower=~1),direction="forward")
```

```
## Start:  AIC=277.09
## Failing ~ 1
##
##                 Df Deviance    AIC
## + ProjectType    1   208.98 212.98
## + HousingTenure  3   215.99 223.99
## + FamilyUnit     1   239.33 243.33
## + OneBd          1   247.14 251.14
## + StudioUnits    1   249.86 253.86
## + ThreeBd        1   259.27 263.27
## + TwoBd          1   266.71 270.71
## <none>               275.09 277.09
##
## Step:  AIC=212.98
## Failing ~ ProjectType
##
##                 Df Deviance    AIC
## + FamilyUnit     1   153.57 159.57
## + HousingTenure  3   180.95 190.95
## + OneBd          1   187.18 193.18
## + ThreeBd        1   194.87 200.87
## + StudioUnits    1   197.28 203.28
## + TwoBd          1   202.08 208.08
## <none>               208.98 212.98
##
## Step:  AIC=159.57
## Failing ~ ProjectType + FamilyUnit
##
##                 Df Deviance    AIC
## + HousingTenure  3   140.63 152.63
## + StudioUnits    1   149.07 157.07
## + TwoBd          1   149.34 157.34
## + OneBd          1   151.39 159.39
## <none>               153.57 159.57
## + ThreeBd        1   153.37 161.37
##
## Step:  AIC=152.63
## Failing ~ ProjectType + FamilyUnit + HousingTenure
##
##               Df Deviance    AIC
## + OneBd        1   137.72 151.72
## + TwoBd        1   137.74 151.74
## + StudioUnits  1   137.84 151.84
## <none>             140.63 152.63
```

```
## + ThreeBd      1   140.31 154.31
##
## Step:  AIC=151.72
## Failing ~ ProjectType + FamilyUnit + HousingTenure + OneBd
##
##               Df Deviance    AIC
## + TwoBd        1   130.15 146.15
## <none>             137.72 151.72
## + StudioUnits  1   136.25 152.25
## + ThreeBd      1   136.98 152.98
##
## Step:  AIC=146.15
## Failing ~ ProjectType + FamilyUnit + HousingTenure + OneBd +
##     TwoBd
##
##               Df Deviance    AIC
## <none>             130.15 146.15
## + ThreeBd      1   129.25 147.25
## + StudioUnits  1   129.49 147.49

##
## Call:  glm(formula = Failing ~ ProjectType + FamilyUnit + HousingTenure +
##     OneBd + TwoBd, family = binomial(link = "logit"), data = Train)
##
## Coefficients:
##                     (Intercept)       ProjectTypeRehabilitation
##                         5.31924                         -4.09744
##                      FamilyUnit  HousingTenureOwnership, Rental
##                        -0.03097                         13.50693
##             HousingTenureRental            HousingTenureUnknown
##                         -2.94263                         -2.77486
##                           OneBd                            TwoBd
##                         -0.04151                          0.03774
##
## Degrees of Freedom: 208 Total (i.e. Null);  201 Residual
## Null Deviance:       275.1
## Residual Deviance: 130.1     AIC: 146.1
```

Appendix B:

```
table(predict_logic.classes,Valid$Failing)

##
## predict_logic.classes  0  1
##                     0 18  3
##                     1  6 25
```

Appendix C:

```
data.frame(k=c(3,5,10),Val_error=c(val_err3,val_err5,val_err10))
```

```
##    k Val_error
## 1  3 0.1923077
## 2  5 0.1730769
## 3 10 0.2500000
```

Appendix D:

```
table(knn.Valid$Failing, knn.model3)

##    knn.model3
##      0  1
##   0 17  7
##   1  3 25

table(knn.Valid$Failing, knn.model5)

##    knn.model5
##      0  1
##   0 18  6
##   1  3 25

table(knn.Valid$Failing, knn.model10)

##    knn.model10
##      0  1
##   0 14 10
##   1  3 25
```

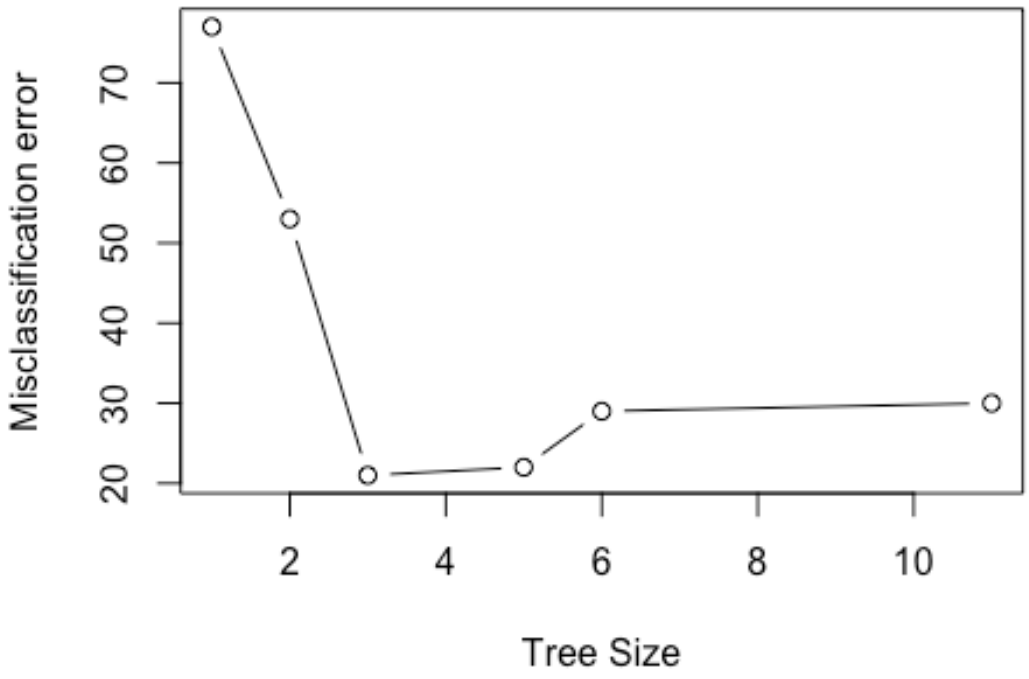
Appendix E:

```
model2

## node), split, n, deviance, yval, (yprob)
##       * denotes terminal node
##
##  1) root 209 275.100 1 ( 0.36842 0.63158 )
##    2) ProjectType: New Construction 170 187.800 1 ( 0.24118 0.75882 )
##      4) FamilyUnit < 75.5 131  65.570 1 ( 0.06870 0.93130 )
##        8) OneBd < 9.5 113  27.690 1 ( 0.02655 0.97345 )
##         16) FamilyUnit < 31.5 105   0.000 1 ( 0.00000 1.00000 ) *
##         17) FamilyUnit > 31.5 8  10.590 1 ( 0.37500 0.62500 ) *
##        9) OneBd > 9.5 18  22.910 1 ( 0.33333 0.66667 )
##         18) TwoBd < 9 5   5.004 0 ( 0.80000 0.20000 ) *
##         19) TwoBd > 9 13  11.160 1 ( 0.15385 0.84615 )
##           38) StudioUnits < 5.5 5   6.730 1 ( 0.40000 0.60000 ) *
##           39) StudioUnits > 5.5 8   0.000 1 ( 0.00000 1.00000 ) *
##      5) FamilyUnit > 75.5 39  36.710 0 ( 0.82051 0.17949 )
##       10) FamilyUnit < 152.5 33  24.380 0 ( 0.87879 0.12121 )
##         20) FamilyUnit < 110.5 18  19.070 0 ( 0.77778 0.22222 )
##           40) OneBd < 20.5 12   6.884 0 ( 0.91667 0.08333 ) *
##           41) OneBd > 20.5 6   8.318 0 ( 0.50000 0.50000 ) *
```

```
##         21) FamilyUnit > 110.5 15   0.000 0 ( 1.00000 0.00000 ) *
##       11) FamilyUnit > 152.5 6   8.318 0 ( 0.50000 0.50000 ) *
##    3) ProjectType: Rehabilitation 39  21.150 0 ( 0.92308 0.07692 )
##      6) ThreeBd < 21.5 34   0.000 0 ( 1.00000 0.00000 ) *
##      7) ThreeBd > 21.5 5   6.730 1 ( 0.40000 0.60000 ) *
```

Appendix F:

```
plot(model.2$size, model.2$dev, type="b",xlab="Tree
Size",ylab="Misclassification error")
```



Appendix G:

```
prune.model
```

```
## node), split, n, deviance, yval, (yprob)
##       * denotes terminal node
##
## 1) root 209 275.10 1 ( 0.36842 0.63158 )
##   2) ProjectType: New Construction 170 187.80 1 ( 0.24118 0.75882 )
##     4) FamilyUnit < 75.5 131  65.57 1 ( 0.06870 0.93130 ) *
##     5) FamilyUnit > 75.5 39  36.71 0 ( 0.82051 0.17949 ) *
##   3) ProjectType: Rehabilitation 39  21.15 0 ( 0.92308 0.07692 ) *
```

Appendix H:

```
table(predtion2,Valid$Failing)
```

```
##
## predtion2  0  1
```

```
##         0 19  3
##         1  5 25
```

Appendix I:

```
table(predict3,Valid$Failing)

##
## predict3  0  1
##        0 20  3
##        1  4 25
```

Appendix J:

```
step(model0,scope=list(upper=model1,lower=~1),direction="forward")

## Start:  AIC=309.01
## Failing ~ 1
##
##                 Df Deviance    AIC
## + HousingTenure  3   223.80 231.80
## + ProjectType    1   235.37 239.37
## + FamilyUnit     1   265.40 269.40
## + OneBd          1   274.65 278.65
## + ThreeBd        1   282.34 286.34
## + StudioUnits    1   283.06 287.06
## + TwoBd          1   291.23 295.23
## <none>               307.01 309.01
##
## Step:  AIC=231.8
## Failing ~ HousingTenure
##
##               Df Deviance    AIC
## + ProjectType  1   189.84 199.84
## + FamilyUnit   1   204.24 214.24
## + OneBd        1   212.18 222.18
## + ThreeBd      1   215.60 225.60
## + StudioUnits  1   217.46 227.46
## + TwoBd        1   218.46 228.46
## <none>             223.80 231.80
##
## Step:  AIC=199.84
## Failing ~ HousingTenure + ProjectType
##
##               Df Deviance    AIC
## + FamilyUnit   1   158.12 170.12
## + OneBd        1   173.73 185.73
## + ThreeBd      1   181.18 193.18
## + TwoBd        1   182.93 194.93
## + StudioUnits  1   185.34 197.34
```

```
## <none>             189.84 199.84
##
## Step:  AIC=170.12
## Failing ~ HousingTenure + ProjectType + FamilyUnit
##
##               Df Deviance    AIC
## + OneBd        1   151.13 165.13
## + StudioUnits  1   155.66 169.66
## <none>             158.12 170.12
## + TwoBd        1   157.66 171.66
## + ThreeBd      1   158.07 172.07
##
## Step:  AIC=165.13
## Failing ~ HousingTenure + ProjectType + FamilyUnit + OneBd
##
##               Df Deviance    AIC
## + TwoBd        1   148.79 164.79
## <none>             151.13 165.13
## + StudioUnits  1   149.87 165.87
## + ThreeBd      1   151.11 167.11
##
## Step:  AIC=164.79
## Failing ~ HousingTenure + ProjectType + FamilyUnit + OneBd +
##     TwoBd
##
##               Df Deviance    AIC
## <none>             148.79 164.79
## + ThreeBd      1   147.24 165.24
## + StudioUnits  1   147.79 165.79

##
## Call:  glm(formula = Failing ~ HousingTenure + ProjectType + FamilyUnit +
##     OneBd + TwoBd, family = binomial, data = Train)
##
## Coefficients:
##                     (Intercept)  HousingTenureOwnership, Rental
##                        20.19856                         1.33191
##            HousingTenureRental            HousingTenureUnknown
##                       -18.31433                       -17.95558
##      ProjectTypeRehabilitation                      FamilyUnit
##                        -3.67073                        -0.02331
##                           OneBd                           TwoBd
##                        -0.03607                         0.02144
##
## Degrees of Freedom: 230 Total (i.e. Null);  223 Residual
## Null Deviance:       307
## Residual Deviance: 148.8     AIC: 164.8
```

Appendix K:

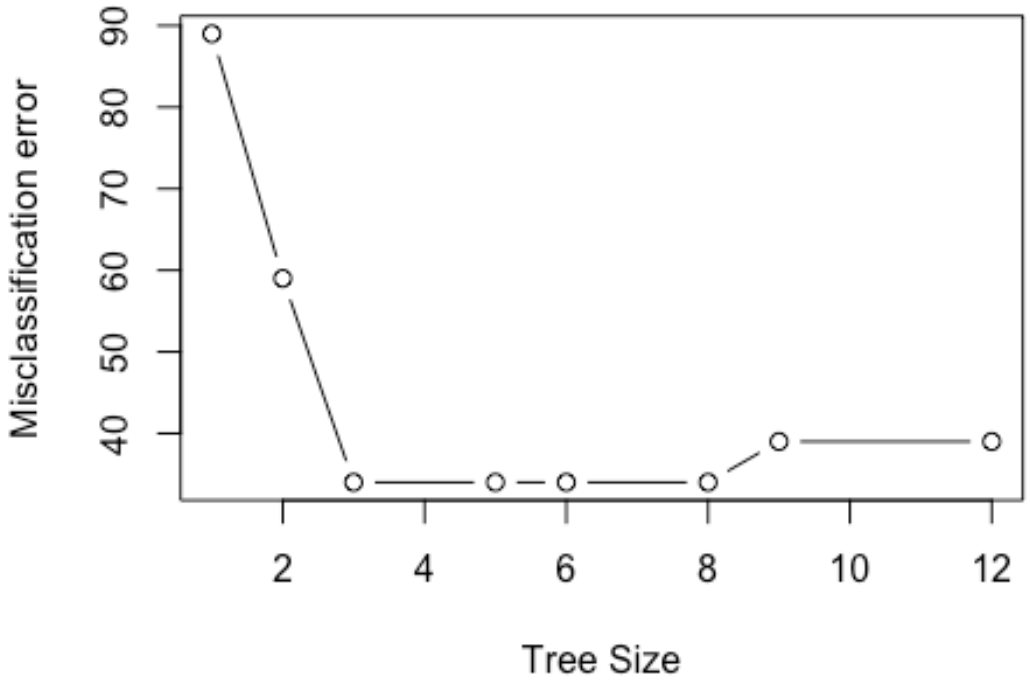
```
## glm(formula = Failing ~ ., family = binomial, data = Train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.8802  -0.4770   0.0001   0.4945   3.1662
##
## Coefficients:
##                                  Estimate Std. Error z value Pr(>|z|)
## (Intercept)                     2.007e+01  1.419e+03   0.014   0.9887
## ProjectTypeRehabilitation      -3.619e+00  7.012e-01  -5.160 2.46e-07 ***
## HousingTenureOwnership, Rental  1.363e+00  6.986e+03   0.000   0.9998
## HousingTenureRental            -1.811e+01  1.419e+03  -0.013   0.9898
## HousingTenureUnknown           -1.788e+01  1.419e+03  -0.013   0.9899
## StudioUnits                    -2.073e-02  1.835e-02  -1.130   0.2586
## OneBd                          -3.475e-02  1.488e-02  -2.336   0.0195 *
## TwoBd                           3.933e-02  1.915e-02   2.053   0.0400 *
## ThreeBd                        -4.344e-02  3.208e-02  -1.354   0.1756
## FamilyUnit                     -2.204e-02  5.248e-03  -4.200 2.67e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 307.01  on 230  degrees of freedom
## Residual deviance: 145.67  on 221  degrees of freedom
## AIC: 165.67
##
## Number of Fisher Scoring iterations: 18
```

Appendix L:

```
table(predict_logic.classes,Valid$Failing)

##
## predict_logic.classes  0  1
##                     0 17  1
##                     1  4 36
```

Appendix M:

```
plot(model.2$size, model.2$dev, type="b",xlab="Tree
Size",ylab="Misclassification error")
```



Appendix N:

```
prune.model

## node), split, n, deviance, yval, (yprob)
##       * denotes terminal node
##
##  1) root 231 307.000 1 ( 0.38095 0.61905 )
##    2) ProjectType: New Construction 188 213.600 1 ( 0.25532 0.74468 )
##      4) FamilyUnit < 30.5 120  35.070 1 ( 0.03333 0.96667 ) *
##      5) FamilyUnit > 30.5 68  88.300 0 ( 0.64706 0.35294 )
##       10) ThreeBd < 4.5 41  56.810 1 ( 0.48780 0.51220 )
##         20) OneBd < 16.5 31  41.380 0 ( 0.61290 0.38710 )
##           40) FamilyUnit < 148.5 26  32.100 0 ( 0.69231 0.30769 ) *
##           41) FamilyUnit > 148.5 5   5.004 1 ( 0.20000 0.80000 ) *
##         21) OneBd > 16.5 10   6.502 1 ( 0.10000 0.90000 ) *
##       11) ThreeBd > 4.5 27  18.840 0 ( 0.88889 0.11111 ) *
##    3) ProjectType: Rehabilitation 43  21.760 0 ( 0.93023 0.06977 ) *
```

Appendix O:

```
table(predtion2,Valid$Failing)

##
## predtion2  0  1
##         0 17  2
##         1  4 35

mean(predtion2==Valid$Failing)
```

Appendix P: All codes

#data 1 (imputing missing values by excluding NA in the categorical variable)

```
dat=read.delim("afford_housing30.txt",sep="")
str(dat)

#data construction
Project.dat=dat[,c(3,10,11,13:16,19)]

#change numeric to factor
Project.dat$Failing = as.factor(Project.dat$Failing)
Project.dat$ProjectType = as.factor(Project.dat$ProjectType)
Project.dat$HousingTenure = as.factor(Project.dat$HousingTenure)

#Check the variable type
str(Project.dat)

Project.data=na.omit(Project.dat)
table(is.na(Project.data))

n<-dim(Project.data)[1]
set.seed(1234)
table(Project.data$Failing)

Training_IDs<-sample(1:n,size=round(0.8*n))
Train<-Project.data[Training_IDs,]
Valid<-Project.data[-Training_IDs,]
```

#logistic regression

```
model0 = glm(Failing~1, family=binomial(link='logit'),data=Train)
model1 = glm(Failing ~.,family=binomial(link='logit'),data=Train)
model1 = glm(Failing ~.,family=binomial,data=Train)
summary(model1)

step(model0,scope=list(upper=model1,lower=~1),direction="forward")

Final.model=glm(Failing ~ ProjectType + FamilyUnit + HousingTenure + OneBd +
TwoBd,family=binomial,data=Train)
summary(Final.model)

Final.model=glm(Failing ~ ProjectType + FamilyUnit + HousingTenure + OneBd +
TwoBd,family=binomial,data=Train)

predict(Final.model,data.frame(ProjectType="New
Construction",FamilyUnit=38.5933,HousingTenure="Ownership", OneBd=10.82297,
TwoBd=10.94737),type="response")

predict(Final.model,data.frame(ProjectType="Rehabilitation",FamilyUnit=38.593
3,HousingTenure="Ownership", OneBd=10.82297, TwoBd=10.94737),type="response")
```

```
predict_logistic=predict(Final.model,Valid, type="response")
predict_logic.classes <- ifelse(predict_logistic > 0.5, "1", "0")
table(predict_logic.classes,Valid$Failing)

mean(predict_logic.classes == Valid$Failing)
```

#ROC curve

```
library(pROC)

library(Metrics)

library(ROCR)
pr <- prediction(as.integer(predict_logic.classes),Valid$Failing)
perf <- performance(pr,measure = "tpr",x.measure = "fpr")
plot(perf)

auc(Valid$Failing,predict_logic.classes)
```

#Assumption for logistic regression

```
#The deviance
pchisq(Final.model$deviance,201,lower.tail = FALSE)

#The pearson chi square test
Pearson =sum(residuals(Final.model,type = "pearson")^2)
pchisq(Pearson,201,lower.tail = FALSE)

#Cook's distance
plot(Final.model,which=5)

## Warning: not plotting observations with leverage one:
##   197
```

#KNN

#data constructing

```
library(fastDummies)
#change numeric to factor
Project.dat$Failing = as.factor(Project.dat$Failing)

#Creating dummy variable to change facor variables;
dat.dummy <- dummy_cols(Project.data, select_columns =
c('ProjectType','HousingTenure'))
dim(dat.dummy)

knn.data=dat.dummy[,c(1,4:14)]

n<-dim(knn.data)[1]
set.seed(1234)
table(knn.data$Failing)
```

```
Training_IDs<-sample(1:n,size=round(0.8*n))
knn.Train<-knn.data[Training_IDs,]
knn.Valid<-knn.data[-Training_IDs,]
```

#Run knn model

```
library(class)
knn.model3 = knn(knn.Train[,-1],knn.Valid[,-1],cl=knn.Train[,1],k=3)
knn.model5 = knn(knn.Train[,-1],knn.Valid[,-1],cl=knn.Train[,1],k=5)
knn.model10 = knn(knn.Train[,-1],knn.Valid[,-1],cl=knn.Train[,1],k=10)
table(knn.Valid$Failing, knn.model3)

table(knn.Valid$Failing, knn.model5)

table(knn.Valid$Failing, knn.model10)

val_err3=mean(knn.Valid$Failing!=knn.model3)
val_err5=mean(knn.Valid$Failing!=knn.model5)
val_err10=mean(knn.Valid$Failing!=knn.model10)
data.frame(k=c(3,5,10),Val_error=c(val_err3,val_err5,val_err10))

val_err5=mean(knn.Valid$Failing==knn.model5)
```

#ROCcurve

```
library(ROCR)
knn.auc = knn(knn.Train[,-1],knn.Valid[,-1],cl=knn.Train[,1],k=5, prob =
TRUE)
plot(roc(knn.Valid$Failing, attributes(knn.auc)$prob),print.auc = T)

## Setting levels: control = 0, case = 1

## Setting direction: controls < cases
```

#Decision tree

```
library(tree)
library(randomForest)

## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

model2 = tree(Failing~.,data=Train)
summary(model2)

plot(model2)
text(model2, pretty=0)

model2
```

#Pruning

```
model.2=cv.tree(model2,FUN=prune.misclass)
model.2

plot(model.2$size, model.2$dev, type="b",xlab="Tree
Size",ylab="Misclassification error")

prune.model=prune.misclass(model2,best=3)
prune.model

summary(prune.model)

plot(prune.model)
text(prune.model, pretty = 0)
```

#Prediction error rate

```
#pruned tree model
predtion2=predict(prune.model,Valid,type="class")
table(predtion2,Valid$Failing)

mean(predtion2==Valid$Failing)
```

#ROC Curve

```
Prediction.prune=prediction(as.numeric(predtion2),as.numeric(Valid$Failing))
performance.prune <- performance(Prediction.prune, "tpr","fpr")
plot(performance.prune)

performance(Prediction.prune,"auc")@y.values[[1]] #AUC value
```

#Random forest

```
library(randomForest)
model3 = randomForest(Failing~.,data=Train,mtry=2,importance=TRUE)
summary(model3)

importance(model3)

varImpPlot(model3)

predict3=predict(model3,Valid)
table(predict3,Valid$Failing)

mean(predict3==Valid$Failing)
```

#ROC Curve

```
Prediction.rf=prediction(as.numeric(predict3),as.numeric(Valid$Failing))
performance.rf <- performance(Prediction.rf, "tpr","fpr")
plot(performance.rf)

performance(Prediction.rf,"auc")@y.values[[1]] #AUC value
```

#data 2 (imputing missing values through using iterative regression)

```
dat = read.delim("afford_housing30.txt",sep="")

#data construction
Project.dat = dat[,c(3,10,11,13:16,19)]

#change numeric to factor
Project.dat$Failing = as.factor(Project.dat$Failing)
Project.dat$ProjectType = as.factor(Project.dat$ProjectType)
Project.dat$HousingTenure = as.factor(Project.dat$HousingTenure)

iter_reg_data = Project.dat
summary(iter_reg_data)

summary(iter_reg_data$Failing)

iter_reg_data$Failing[is.na(iter_reg_data$Failing)] = "1"
summary(iter_reg_data$Failing)

n_iter=10
for(i in 1:n_iter) {

logit_Failing=glm(Failing~.,iter_reg_data,subset=!is.na(Project.dat$Failing),
family=binomial)

pred_Failing=predict(logit_Failing,iter_reg_data[is.na(Project.dat$Failing),]
)
  predict_logic.classes <- ifelse(pred_Failing > 0.5, "1", "0")
  iter_reg_data$Failing[is.na(Project.dat$Failing)]=predict_logic.classes
}

table(is.na(iter_reg_data))

barplot(prop.table(table(iter_reg_data$Failing)))

barplot(prop.table(table(Project.dat$Failing)))

#Check N.A

table(is.na(iter_reg_data))
```

#split the data into 80% of training set and 20% of validation set

```
n<-dim(iter_reg_data)[1]
set.seed(1234)
table(iter_reg_data$Failing)

Training_IDs<-sample(1:n,size=round(0.8*n))
Train<-iter_reg_data[Training_IDs,]
Valid<-iter_reg_data[-Training_IDs,]
```

#Model Selection

```
model0 = glm(Failing~1, family=binomial ,data=Train)
model1 = glm(Failing ~.,family=binomial,data=Train)
summary(model1)

step(model0,scope=list(upper=model1,lower=~1),direction="forward")

Final.model=glm(Failing ~ ProjectType + FamilyUnit + HousingTenure + OneBd +
TwoBd,family=binomial,data=Train)
summary(Final.model)

predict(Final.model,data.frame(ProjectType="New
Construction",FamilyUnit=38.5933,HousingTenure="Ownership", OneBd=10.82297,
TwoBd=10.94737),type="response")

predict(Final.model,data.frame(ProjectType="Rehabilitation",FamilyUnit=38.593
3,HousingTenure="Ownership", OneBd=10.82297, TwoBd=10.94737),type="response")

predict_logistic=predict(Final.model,Valid, type="response")
predict_logic.classes <- ifelse(predict_logistic > 0.5, "1", "0")
table(predict_logic.classes,Valid$Failing)

mean(predict_logic.classes == Valid$Failing)
```

#ROC curve

```
library(pROC)

library(Metrics)

library(ROCR)

pr <- prediction(as.integer(predict_logic.classes),Valid$Failing)
perf <- performance(pr,measure = "tpr",x.measure = "fpr")
plot(perf)

auc(Valid$Failing,predict_logic.classes)
```

#Assumption for logistic regression

```
#The deviance
pchisq(Final.model$deviance,201,lower.tail = FALSE)

#The pearson chi square test
Pearson =sum(residuals(Final.model,type = "pearson")^2)
pchisq(Pearson,201,lower.tail = FALSE)

#Cook's distance
plot(Final.model,which=5)
```

#KNN

```
library(fastDummies)
str(iter_reg_data)

#Creating dummy variable to change facor variables;
dat.dummy <- dummy_cols(iter_reg_data, select_columns =
c('ProjectType','HousingTenure'))
dim(dat.dummy)
```

#KNN

```
library(fastDummies)
str(iter_reg_data)

#Creating dummy variable to change facor variables;
dat.dummy <- dummy_cols(iter_reg_data, select_columns =
c('ProjectType','HousingTenure'))
dim(dat.dummy)

knn.data=dat.dummy[,c(1,4:14)]

n<-dim(knn.data)[1]
set.seed(1234)
table(knn.data$Failing)

Training_IDs<-sample(1:n,size=round(0.8*n))
knn.Train<-knn.data[Training_IDs,]
knn.Valid<-knn.data[-Training_IDs,]

library(class)
knn.model3 = knn(knn.Train[,-1],knn.Valid[,-1],cl=knn.Train[,1],k=3)
knn.model5 = knn(knn.Train[,-1],knn.Valid[,-1],cl=knn.Train[,1],k=5)
knn.model10 = knn(knn.Train[,-1],knn.Valid[,-1],cl=knn.Train[,1],k=10)


table(knn.Valid$Failing, knn.model3)

table(knn.Valid$Failing, knn.model5)

table(knn.Valid$Failing, knn.model10)

val_err3=mean(knn.Valid$Failing!=knn.model3)
val_err5=mean(knn.Valid$Failing!=knn.model5)
val_err10=mean(knn.Valid$Failing!=knn.model10)
data.frame(k=c(3,5,10),Val_error=c(val_err3,val_err5,val_err10))

val_err3=mean(knn.Valid$Failing==knn.model3)
val_err10=mean(knn.Valid$Failing==knn.model10)
```

#ROCcurve

```
library(ROCR)
knn.auc = knn(knn.Train[,-1],knn.Valid[,-1],cl=knn.Train[,1],k=5, prob =
```

```
TRUE)
plot(roc(knn.Valid$Failing, attributes(knn.auc)$prob),print.auc = T)
```

#Decision tree

```
library(tree)
library(randomForest)

## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

model2 = tree(Failing~.,data=Train)
summary(model2)

plot(model2)
text(model2, pretty=0)

model.2=cv.tree(model2,FUN=prune.misclass)
model.2

plot(model.2$size, model.2$dev, type="b",xlab="Tree
Size",ylab="Misclassification error")

prune.model=prune.misclass(model2,best=6)
prune.model

summary(prune.model)

plot(prune.model)
text(prune.model, pretty = 0)

predtion2=predict(prune.model,Valid,type="class")
table(predtion2,Valid$Failing)

mean(predtion2==Valid$Failing)
```

#ROC Curve

```
Prediction.prune=prediction(as.numeric(predtion2),as.numeric(Valid$Failing))
performance.prune <- performance(Prediction.prune, "tpr","fpr")
plot(performance.prune)

performance(Prediction.prune,"auc")@y.values[[1]] #AUC value
```

#Random forest

```
library(randomForest)
model3 = randomForest(Failing~.,data=Train,mtry=2,importance=TRUE)
summary(model3)

plot(model3)

importance(model3)
```

```
varImpPlot(model3)

predict3=predict(model3,Valid)
table(predict3,Valid$Failing)

mean(predict3==Valid$Failing)
```

#ROC Curve

```
Prediction.rf=prediction(as.numeric(predict3),as.numeric(Valid$Failing))
performance.rf <- performance(Prediction.rf, "tpr","fpr")
plot(performance.rf)

performance(Prediction.rf,"auc")@y.values[[1]] #AUC value
```