Zirong Huang
STAT 4052
Final Project

Introduction

The main goals of this analysis are to determine the predictive model that works the best to predict the probability of failure of future projects aiming to provide affordable housing and variables that contribute the most to the failure of projects. 26 variables are being given in the dataset, and I decided to choose 6 out of those 26 variables to build my predictive models. The first step that I did is to eliminate those variables that contain a large number of 0. For instance, the variable *Two Bedroom* has 139 out of 289 observations that are 0, and the failing status of the building contains both 0 and 1 when OneBd=0. It is hard to find relationships between failing and the predictive variable when the variable contains a large number of 0. The next step is to eliminate variables that have a large number of NAs. In my dataset, the variable *Proceed* and *Days Proceed* both have 148 NAs out of 289 observations. The third step is to eliminate variables that have too many categories. *Street* has over 130 categories, and *Zipcode* has over 15 categories. Since we only have 289 observations, having too many categories means we only get a few observations per category; the lack of observations can jeopardize the performance of those predictive models. After this step, 8 variables are still left: *Status*, *Supervisor District*, *Housing Tenure*, *Program Area*, *Project Type*, *Market Rate*, *Family Units*, and *Project Units*. Since I decided to choose 6 variables, I eliminated *Status* and *Supervisor District*. Eventually, the variables that I decided to choose are *Housing Tenure*, *Program Area*, *Project Type*, *Market Rate*, *Family Units*, and *Project Units*.

Methods

First Iteration

The logistic model that I decided to use is the nested logistic model. All continuous variables have both their first power and second power in the model. I started with the logistic model that contains all variables, and then I used the step() function to make the model selection. Here is the model that I got after the model selection:

$$\log\left(\frac{\pi}{1-\pi}\right) = \beta_0 + \beta_1 Program\ Area + \beta_2 Family\ Units + \beta_3 Project\ Units$$

Housing Tenure, *Project Type*, and *Market Rate* are being eliminated throughout the model selection process. Since there are continuous variables in the model, I decided to test out the nested model below:

$$\log\left(\frac{\pi}{1-\pi}\right) = \beta_0 + \beta_1 Program\ Area + \beta_2 Family\ Units + \beta_3 Project\ Units + \beta_4 Family\ Units^2 + \beta_5 Project\ Unit^2$$

The ANOVA comparison of those 2 models shows that the nested model is the better model, So I decided my final model is the nested logistic model.

The model that I have chosen out of LDA, QDA, and KNN is the KNN model. This is because both LDA and QDA require all predictors to be continuous and follow normal distributions. First of all, I have selected 3 categorical predictors. Second, according to the histogram of those 3 continuous variables that I have chosen, all 3 variables have a right-skewed distribution trend, indicating LDA and QDA are not suitable for this situation. To determine the number of k, I have used two approaches: the validation set approach and the cross-validation approach. Both approaches show that k=10 is the best number of k. The final KNN model that I have chosen is the model with k=10.

For the decision tree, I also started with the model that contains all predictors. After pruning the tree using cv.tree(), the final tree that I got is the tree that has 3 nodes-Market Rate<4.5, ProjUnits<100, and ProjUnits<107.5.

When it comes to the random forest, after testing out several values of m, the final number of m that I have chosen is 2, which minimizes the test error rate.

Second Iteration

The logistic model that I decided to use for the second iteration is the quasi-likelihood model. I followed the same procedures that I did in the first iteration. But this time, the model selected became this:

$$\log\left(\frac{\pi}{1-\pi}\right) = \beta_0 + \beta_1 Program\ Area + \beta_2 Project\ Type + \beta_3 Market\ Rate + \beta_4 Family\ Units + \beta_5 Project\ Units$$

Also, the nested model did not outperform the selected model above. In the selected model diagnostic plot, there still exist 2 outliers, so I decided to go with the quasi-likelihood model with those chosen predictors in the selected model above.

Following the same procedures that I have mentioned above, I still get k=10 as the best number of k for my KNN model.

This time, I only get one node for the decision tree after pruning, which is Market Rate<7.

For the random forest, I used similar procedures in the first iteration. The number of m that I got that minimizes the test error rate is 1.

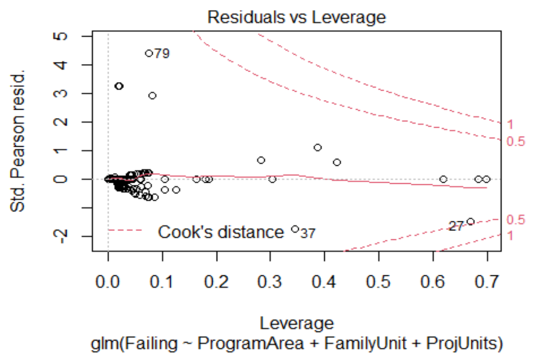
Results

First Iteration

Logistic

As I have mentioned above, for the logistic model, I first started with the model that contains all predictors and used step() function to select the best model.

There only exists 3 predictors: *Program Area*, *Family Unit*, and *Project Unit* after the selection. The p-value both equal to 1 for the χ^2significant test and the Pearson test. The reason that those extremely large p-values occurred might be the warnings that happened during the model fitting processes, which are "fitted probability numerically 0 or 1 occurred". But at least those large p-values indicate the model is a good fit.
The diagnostic plot of this selected model is below:



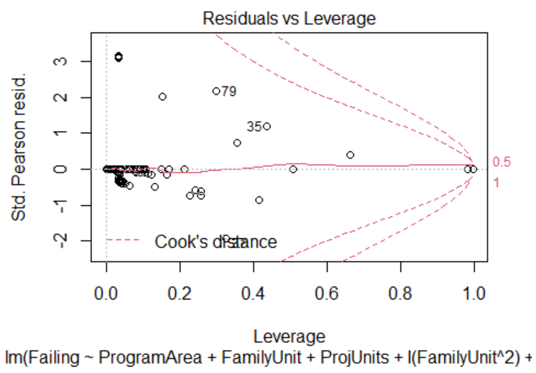
We can see there do not exist any outliers, so we do not need to consider those approaches for the overdispersion.

Next is comparing the selected model with the nested model. Below is the ANOVA comparison output for those two models:

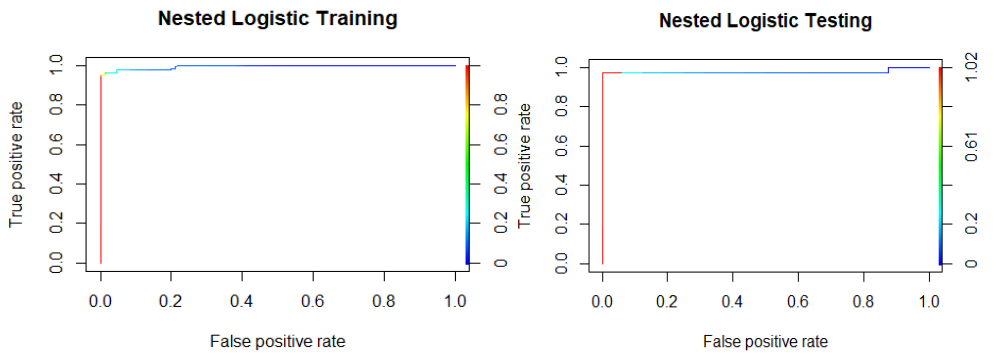
```
Analysis of Deviance Table

Model 1: Failing ~ ProgramArea + FamilyUnit + ProjUnits
Model 2: Failing ~ ProgramArea + FamilyUnit + ProjUnits + I(FamilyUnit^2) +
    I(ProjUnits^2)
```

```
  Resid. Df Resid. Dev Df Deviance Pr(>Chi)
1       204     41.319
2       202     33.799  2   7.5202  0.02328 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The p-value is 0.02, which is less than α=0.05, pointed out that the nested model is better than the selected model. So I decided to go with the nested model. The nested model increased the flexibility of the selected model, the bias decreases, and the variance increases. Due to the space limit, I am not able to provide the summary output for the model, but *Program Area* is the only statistically significant variable in the nested model. Here is the diagnostic plot for the nested model; we can see no outlier exists:



Using threshold = 0.5, the training error rate for the nested model is 0.0280, and the testing error rate for the nested model is 0.0185. We can see those two error rates are not too far away, which shows that the model is not overfitting the data. The nested model performs better in the testing set when threshold = 0.5. The ROC plots for the training and testing set below evaluated the performance of the nested model while changing the threshold:



The AUC for the training set is 0.9943 and 0.9769 for the testing set. We can see that the nested model performs better in the training set when changing the threshold.

KNN

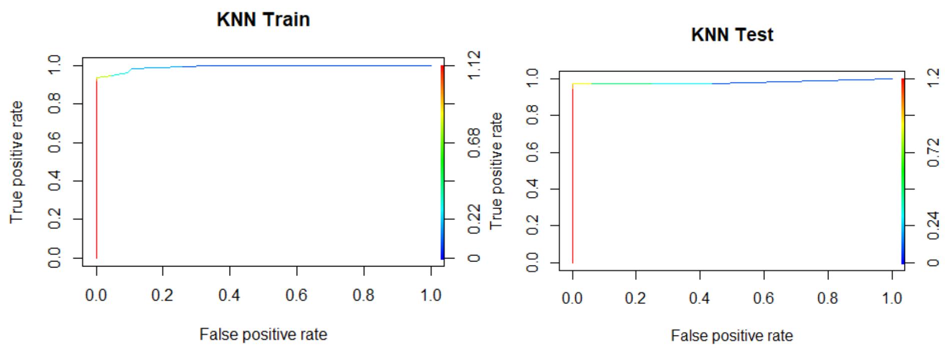
For KNN, the potential k values that are being tested are 3, 5, and 10. After transforming all categorical variables into dummy variables, I directly perfromed 3 KNN classifications to get the general idea of how those k values work for this dataset. The confusion matrices that I got for those 3 different k values using the validation set approach are below:

```
     knn3                                   knn5                                       knn10
     0   1                                   0   1                                      0   1
  0 14   2                                0 14   2                                   0 16   0
  1  1  37                                1  1  37                                   1  1  37
```

It seems k=10 performs the best. To further investigate, I did a 10 fold cross-validation. Here is the CV error rate that I got from the 10 fold cross-validation approach:

```
   k   CV_error
1  3 0.04524176
2  5 0.05294546
3 10 0.03465975
```

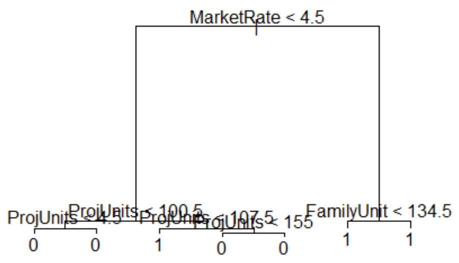
We can see that k =10 still performs the best out of the 3 k values. Thus the final KNN model will use k=10 as its k value. We can see that a less flexible KNN model works better than those flexible models (i.e. lower k values) for this dataset. The training error rate is 0.0374, and the testing error rate is 0.0185. Those two numbers are pretty close to each other, and there is no overfitting. Below are the ROC plots for the training and testing dataset using k=10:



The AUC for the training set is 0.9927 and 0.9811 for the testing set. KNN does perform better than the nested logistic model since we got higher AUC for both training and testing sets. The difference between the nested logistic and the KNN models is that KNN is a non-parametric classification method, whereas the logistic model is a parametric classification method.

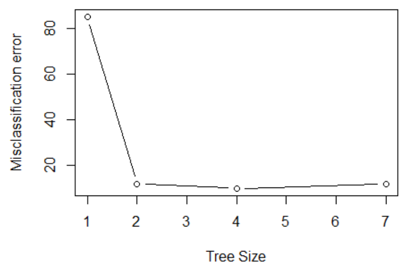
Decision Tree

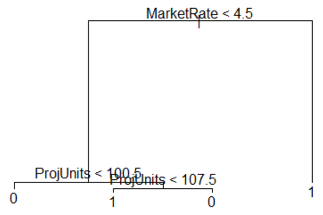
Decision Tree is also a non-parametric classification method. Here is the tree that I got before pruning:



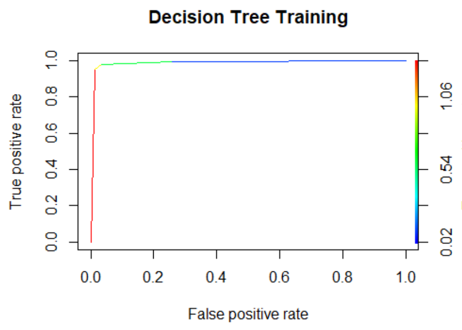
There are a total of 6 nodes before pruning. *Market Rate*, *Project Units*, and *Family Units* are used to construct the decision tree.

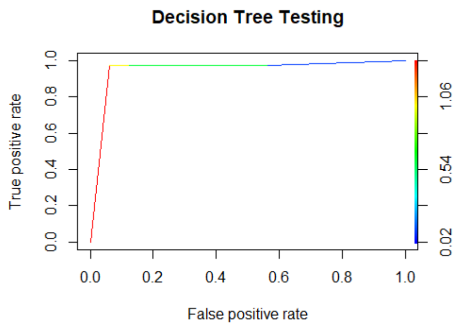
The plots of misclassification error while changing the number of trees and the tree after pruning are below:





When tree size equals 4, we get the lowest misclassification error, and there are only 3 nodes left after pruning. The variables that were left after pruning are *Market Rate* and *Project Units*. The training error rate is 0.0281, and the testing error rate is 0.0555. We are good. Let's check the ROC plots for the decision tree:

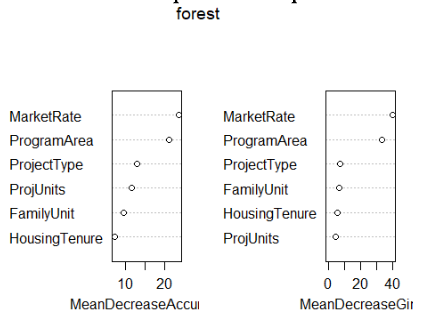




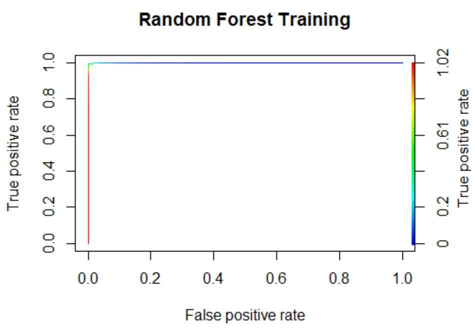
The AUC for the training set is 0.9867 and 0.9490 for the testing set. We can see that the decision tree performs worse than both logistic regression and KNN. According to Varghese (2019), decision trees are sensitive to outliers, and they lose valuable information when handling continuous variables. There do exist several outliers in the dataset, and 3 out of 6 predictors are continuous variables. Those might be the reason that caused the decision tree method performed worse than both logistic regression and KNN.

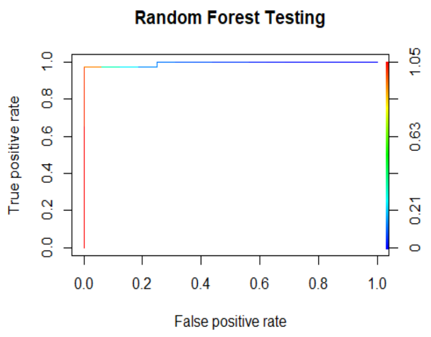
Random Forest

After testing out several values of m, the value m that I got that minimizes the test error rate is 2. This is consistent with what we got from the decision tree-two variables are used to construct the tree. The variables importance plot is below:



The variable *Market Rate* and *Program area* seem to be the most important variable from the plot above. In the decision tree, we can see that *Market Rate* is being used as the initial node, which is consistent with this plot. The training error rate is 0.009, and the testing error rate is 0.0185. There does not exist a considerable difference between those two rates. Here are the ROC plots for the random forest:





The AUC for the training set is 0.9998 and 0.9934 for the testing set. We can see that although KNN and random forest get the same test error rate when threshold=0.5, random forest has a higher testing AUC. It is feasible to conclude that random forest performs better than KNN in the first iteration.

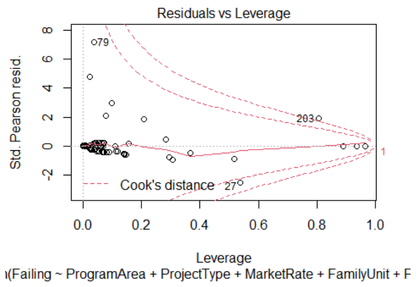
Second Iteration

Logistic

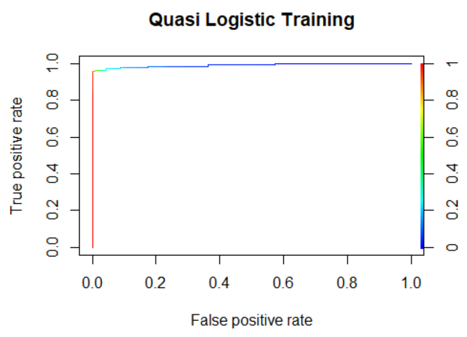
This time, after using the step() function, there were 5 variables left. Those variables are *Program Area*, *Project Type*, *Market Rate*, *Family Unit*, and *Project Units*. Similar to the procedures in the first iteration, I compared the selected model with the nested model. Below is the ANOVA comparison table:

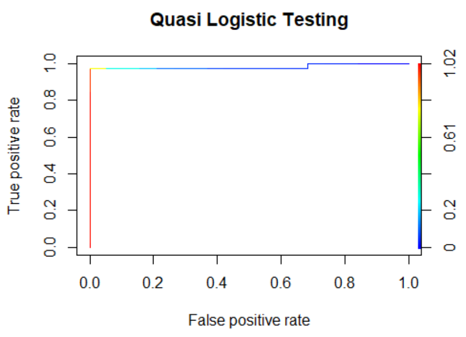
```
Analysis of Deviance Table

Model 1: Failing ~ ProgramArea + ProjectType + MarketRate + FamilyUnit +
    ProjUnits
Model 2: Failing ~ ProgramArea + ProjectType + MarketRate + FamilyUnit +
    ProjUnits + +I(MarketRate^2) + I(FamilyUnit^2) + I(ProjUnits^2)
  Resid. Df Resid. Dev Df Deviance Pr(>Chi)
1       219     40.320
2       216     36.758  3   3.5622   0.3128
```

Different from the first iteration, the selected model is now better than the nested model. So let's go with the selected model. The p-value still equal to 1 for the χ^2significant test and the Pearson test for the selected model. Here is the chosen model diagnostic plot:



There exist 2 outliers, so we need to consider approaches for the overdispersion. I decided to use the quasi-likelihood method for the logistic model. The summary output of the model shows that *Market Rate*, *Program Area*, *Family Units*, and *Project Units* are statistically significant. The training error for the quasi-likelihood model is 0.0259, and the testing error is 0.0172. We do not have a problem with overfitting. Here are the ROC plots for the quasi-likelihood model:





The AUC for the training set is 0.9911 and 0.9824 for the testing set. Compared to the first iteration, we can see that although the AUC for the training set decreased, AUC for the testing set increased. Also, the test error rate decreased to 0.0172 compared to the first iteration (0.0185).

KNN

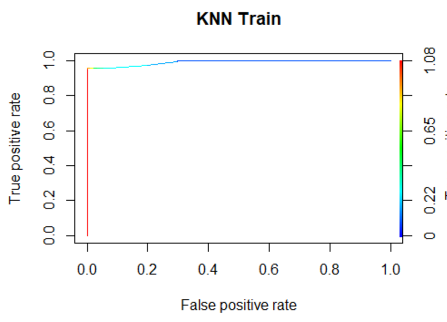
The potential k values for KNN are still 3, 5, and 10. This time after directly performing 3 classifications, the confusion matrices that I got for the 3 different k values using the validation set approach are down below:

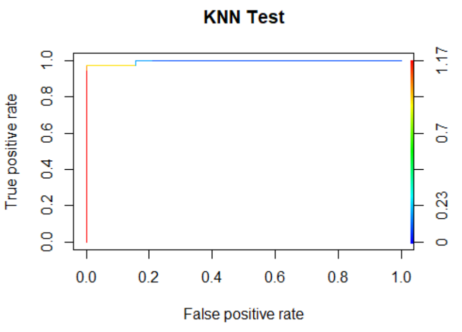
```
     knn3                            knn5                        knn10
      0  1                            0  1                        0  1
  0 17  2                         0 19  0                     0 19  0
  1  1 38                         1  1 38                     1  1 38
```

We can see the confusion matrix for k=5 is the same as the confusion matrix for k=10. This means using cross-validation would help to determine the appropriate k value. Here are the test error rates we got after performing a 10 fold cross-validation for those 3 k values:

```
   k   CV_error
1  3 0.03441549
2  5 0.03463796
3 10 0.03118968
```

The 10 fold cross-validation test error rates above showed that k=10 is the most appropriate k value out of the 3 different k values. The training error for the k=10 KNN model is 0.0303, and the testing error is 0.0172. The testing error is the same as the one we got from the quasi-likelihood model in the previous section. ROC plots and AUC can help to determine which model performs better.

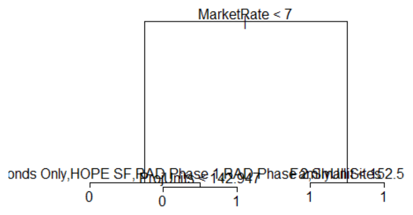




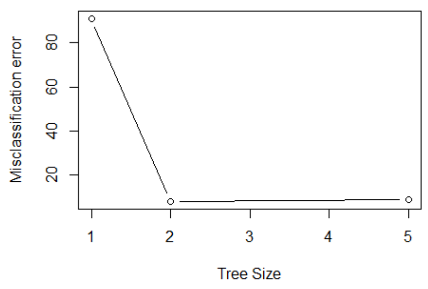
AUC for the training set 0.9907, and the AUC for the testing set is 0.9959. We can see that KNN's AUC for the testing set is larger than the one for the quasi-likelihood model. So it is practical to conclude that KNN does perform better than the quasi-likelihood model using AUC for the testing set.

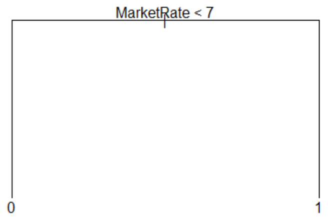
Decision Tree

Below is the tree that I got before pruning:

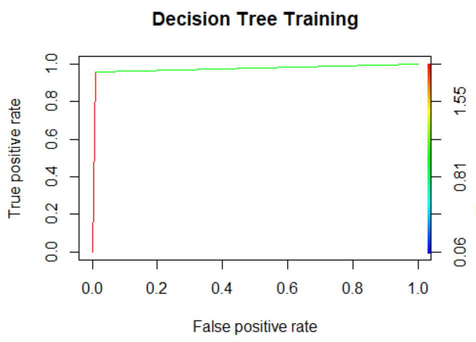


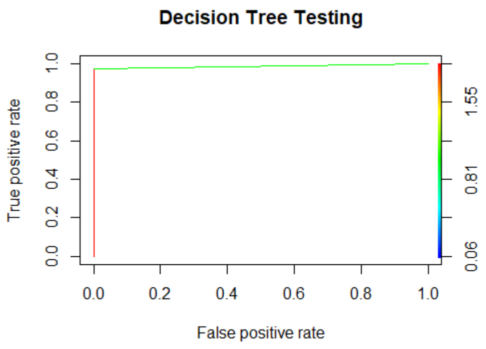
There are a total of 4 nodes in the tree. We can see that *Market Rate*, *Program Area*, *Family Unit*, and *Project Units* are being used to construct the tree before pruning. The number of nodes decreased, and the number of used variables increased compared to the first iteration. The plots of misclassification error while changing the number of trees and the tree after pruning are below:





We can see that tree size=2 gives the lowest misclassification error. There is only 1 node left after pruning. Surprisingly, this single-node tree actually did a relatively great job of predicting the outcome. The training error rate is 0.0303, and the testing error rate is 0.0172, which is exactly the same as the one we got from KNN! This time we can also use ROC plots and AUC to help us determine which method performs better.

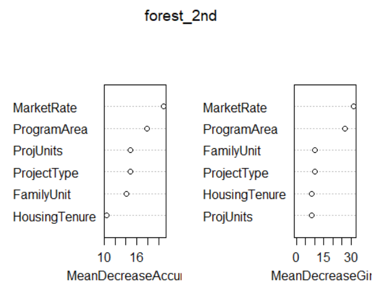




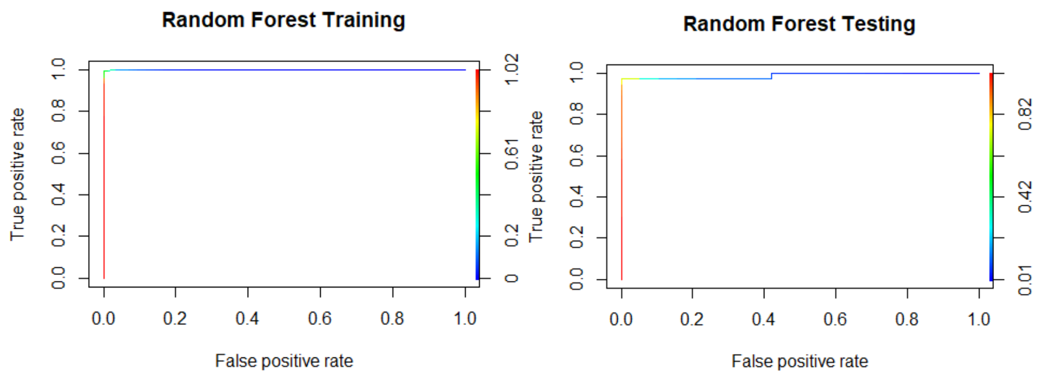
AUC for the training set 0.9731, and the AUC for the testing set is 0.9871. We can see that both AUCs are lower than those we got from KNN, sofar KNN outperforms the decision tree method and the quasi-likelihood model.

Random Forest

This time, the m that I got that minimizes the test error rate is 1. This is also consistent with what we got from the decision tree--only one variable is used to construct the tree. The variables importance plot is below:



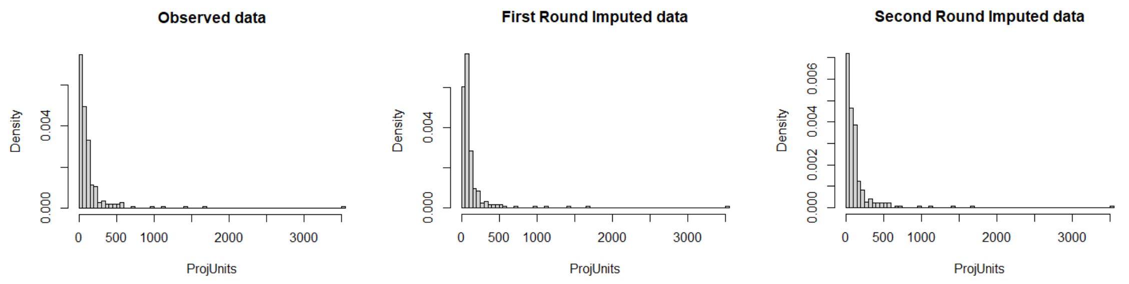
We can see that *Market Rate* is the most crucial variable, and *Program Area* is also an essential variable. In the decision tree, *Market Rate* is the only variable being used to construct the tree, which corresponds to what we observed from the variable importance plot. The training and testing error rates for the random forest are also equal to 0.0303 and 0.0172. Let's check ROC plots and AUCs:



AUC for the training set is 0.9995, and AUC for the testing set is 0.9892. We can see that although the random forest's AUC for the training set is higher than the one for KNN, its testing set AUC is still lower than the one for KNN, which is 0.9959. So KNN performs better than the random forest according to the testing set AUC.

Changes Observed When Imputing Missing Value

The first change is that iterative regression keeps the missing variables' original distribution. Below are the histograms for the variable Project Unit in the original missing dataset, the first round dataset (using its median for imputing), and the second round dataset (using iterative regression):



We can see that the majority of the number of project units fall in between the interval of 0-50 (i.e., the first bar in the histogram) in the original dataset. Nevertheless, after using the median for those missing data points, the majority of the number of project units fall in between the interval of 51-100 (i.e., the second bar). This distorted the distribution of the variable itself, creates bias, and decreases variance. When it comes to the second round, after using the iterative regression, we can see that histogram we got is similar to the one we have for the original dataset, showing that iterative regression did help to keep the variable's original distribution and prevents bias. Using iterative regression helps to make the variance of the variable similar to the one before imputing.

The change in the imputation method leads to the changes in the logistic model selection- the nested model is selected in the first round, and the quasi-likelihood model is selected in the second round. It also decreases the testing error rates. The testing error rates that we got for all models in the second round are lower than those in the first round, and AUCs in the second round are larger than those in the first round. The method that performs the best also being affected by the different imputation methods. In the first round, random forest performs the best; in the second round, KNN performs the best.

Discussion

In conclusion, when it comes to missing data imputation, using more educated imputation methods such as iterative regressions would help elevate predictive models' performance and reduce test error rates. Using simple imputation methods and discarding the row containing missing data introduce bias to the dataset and affecting the model construction. For predicting the probability of failure of future projects aiming to provide affordable housing, non-parametric models such as KNN and random forest work better than parametric models (e.g., logistic model). Using the dataset after performing iterative regression, KNN with k=10 became the most precise predictive model.

From the summary outputs of logistic models, variables that were left in the decision trees after pruning, and variable importance plots from the random forest, *Market Rate* and *Program Area* are the two variables that contribute the most to the failure of affordable housing projects.

The suggestion that I want to make is to try to avoid creating missing data as much as possible. Although iterative regression did a great job in imputing missing variables, the original data record still works the best for model construction. Using a complete dataset is able to help to construct a more precise model.

Reference

Varghese, D. (2019, May 10). Comparative study on classic machine learning algorithms. Retrieved May 12, 2021, from https://towardsdatascience.com/comparative-study-on-classic-machine-learning-algorithms-24f9ff6ab222#:~:text=Decision%20tree%20vs%20KNN%20%3A,KNN's%20expensive%20real%20time%20execution.

Appendix

Certification of academic integrity
(Final project, individual)

I (name)__Zirong Huang___ hereby certify that I have not discussed this project with anyone except the course instructors. All the work submitted is entirely my own and any source of material other than the textbooks of the course, labs and lecture notes has been properly cited.

I understand that discussing the content of this project with anyone other than the course instructor is considered academic misconduct, and it will result in failing this course and possible expulsion from the University.

I understand that the instructor is entitled to ask for further clarifications on our report and respective R code through an online discussion and the latter may affect the final grade of this exam.

I understand that the online exam discussion will be recorded and may be used as proof in support of a claim of academic dishonesty.

Signature: Zirong Huang   Date: 5/12/2021

Coding:

```
#Importing Data
housing <- read.table("afford_housing24.txt")
attach(housing)
## Select Predictors
selected.housing <- housing[,c(3,11,9,10,12,19,28)]
##Checking for NA
summary(selected.housing)
# First Round
## Impute missing ProjUnits with its median, adjusting data
#Original
original <- selected.housing
original[119,2] <- "Unknown"
original[151,2] <- "Unknown"
original$HousingTenure <- as.factor(original$HousingTenure)
original$ProgramArea <- as.factor(original$ProgramArea)
original$ProjectType <- as.factor(original$ProjectType)
original$Failing <- as.factor(original$Failing)

#First Round
first.round <- selected.housing
first.round$ProjUnits[is.na(selected.housing$ProjUnits)]=median(selected.
housing$ProjUnits,na.rm = TRUE)
first.round <- na.omit(first.round)
first.round$HousingTenure <- as.factor(first.round$HousingTenure)
first.round$ProgramArea <- as.factor(first.round$ProgramArea)
first.round$ProjectType <- as.factor(first.round$ProjectType)
first.round$Failing <- as.factor(first.round$Failing)
par(mfrow=c(1,2))
hist(original$ProjUnits,breaks=60,main="Observed
data",xlab="ProjUnits",freq=FALSE)
hist(first.round$ProjUnits,breaks=60,main="First Round Imputed
data",xlab="ProjUnits",freq=FALSE)
barplot(table(original$Failing),main="Observed data")
barplot(table(first.round$Failing),main="Imputed data")
## Split Data
set.seed(4052)
n <- length(first.round$Failing)
index <- sample(1:n,0.8*n)
first.round.train <- first.round[index,]
first.round.test <- first.round[-index,]
# Logistic Model
library(alr4)
log1 <- glm(Failing~., data=first.round.train, family="binomial")
## deviance test p-value
pchisq(log1$deviance,200,lower.tail = FALSE)
##Pearson chi-square test
Pearson <- sum(residuals(log1,type = "pearson")^2)
pchisq(Pearson,200,lower.tail = FALSE)
##Backward Selection
step(log1)
##Outlier
plot(log1,which = 5)
## Model After Selection
```

```
log2 <- glm(formula = Failing ~ ProgramArea + FamilyUnit + ProjUnits,
    family = "binomial", data = first.round.train)
plot(log2,which = 5)
## deviance test p-value
pchisq(log2$deviance,204,lower.tail = FALSE)
##Pearson chi-square test
Pearson <- sum(residuals(log2,type = "pearson")^2)
pchisq(Pearson,204,lower.tail = FALSE)
## Tested Out Nested Model
nest_log <- glm(formula = Failing ~ ProgramArea + FamilyUnit + ProjUnits
                +I(FamilyUnit^2)+I(ProjUnits^2), family = "binomial",data
=
                  first.round.train)
anova(log2,nest_log,test = "Chisq")
## deviance test p-value
pchisq(nest_log$deviance,202,lower.tail = FALSE)
##Pearson chi-square test
Pearson <- sum(residuals(nest_log,type = "pearson")^2)
pchisq(Pearson,202,lower.tail = FALSE)
##Diagnostic Plot
plot(nest_log,which = 5)
## Test Error Rate & ROC
#Nested Model
##Train Error Rate
y_train<-as.numeric(first.round.train[,1])-1
y_test<-as.numeric(first.round.test[,1])-1
pi_logit_train<-predict(nest_log, first.round.train,type="response")
y_logit_train<-ifelse(pi_logit_train>0.5,1,0)
ER_logit_train<-mean((y_train-y_logit_train)^2)
ER_logit_train
#Training ROC
library(ROCR)
pred_logit_train<-prediction(pi_logit_train, y_train)
perf_logit_train <- performance(pred_logit_train,"tpr","fpr")
plot(perf_logit_train,colorize=TRUE,main="Nested Logistic Training")
AUC_logit_train<-performance(pred_logit_train,"auc")@y.values[[1]]
AUC_logit_train

##Test Error Rate
pi_logit_test<-predict(nest_log, first.round.test,type="response")
y_logit_test<-ifelse(pi_logit_test>0.5,1,0)
ER_logit_test<-mean((y_test-y_logit_test)^2)
ER_logit_test
#Testing ROC
pred_logit_test<-prediction(pi_logit_test, y_test)
perf_logit_test <- performance(pred_logit_test,"tpr","fpr")
plot(perf_logit_test,colorize=TRUE,main="Nested Logistic Testing")
AUC_logit_test<-performance(pred_logit_test,"auc")@y.values[[1]]
AUC_logit_test
# KNN
## Transforming data into dummies
### Training set
#Project Type
first.round.train$ProjectType <- as.factor(first.round.train$ProjectType)
```

```
ff<-first.round.train$ProjectType
ll<-levels(ff)
X.pt<-rep(0,(dim(first.round.train)[1]))
for(i in 1:length(ll)){
  dummy<-ifelse(ff==ll[i],1,0)
  X.pt<-cbind(X.pt,dummy)
}
X.pt<-X.pt[,-1]
colnames(X.pt) <- c("New Construction","Rehabilitation")

#Program area
first.round.train$ProgramArea <- as.factor(first.round.train$ProgramArea)
ff<-first.round.train$ProgramArea
ll<-levels(ff)
X.pa<-rep(0,(dim(first.round.train)[1]))
for(i in 1:length(ll)){
  dummy<-ifelse(ff==ll[i],1,0)
  X.pa<-cbind(X.pa,dummy)
}
X.pa<-X.pa[,-1]
colnames(X.pa) <- c("Bonds Only","HOPE SF","Inclusionary","Inclusionary-
OCII","Multifamily"                     ,"RAD Phase 1","RAD Phase
2","Small Sites" )

#Housing Tenure
first.round.train$HousingTenure <-
as.factor(first.round.train$HousingTenure)
ff<-first.round.train$HousingTenure
ll<-levels(ff)
X.ht<-rep(0,(dim(first.round.train)[1]))
for(i in 1:length(ll)){
  dummy<-ifelse(ff==ll[i],1,0)
  X.ht<-cbind(X.ht,dummy)
}
X.ht<-X.ht[,-1]
colnames(X.ht) <- c("Ownership","Rental","Unknown")
X.train <-
cbind(X.ht,X.pa,X.pt,first.round.train$MarketRate,first.round.train$Famil
yUnit,
                 first.round.train$ProjUnits)
colnames(X.train)[c(14,15,16)] <-
c("MarketRate","FamilyUnit","ProjUnits")
Failing.train <- first.round.train$Failing
### Testing Set
#Project Type
first.round.test$ProjectType <- as.factor(first.round.test$ProjectType)
ff<-first.round.test$ProjectType
ll<-levels(ff)
Z.pt<-rep(0,(dim(first.round.test)[1]))
for(i in 1:length(ll)){
  dummy<-ifelse(ff==ll[i],1,0)
  Z.pt<-cbind(Z.pt,dummy)
}
Z.pt<-Z.pt[,-1]
```

```
colnames(Z.pt) <- c("New Construction","Rehabilitation")

#Program area
first.round.test$ProgramArea <- as.factor(first.round.test$ProgramArea)
ff<-first.round.test$ProgramArea
ll<-levels(ff)
Z.pa<-rep(0,(dim(first.round.test)[1]))
for(i in 1:length(ll)){
  dummy<-ifelse(ff==ll[i],1,0)
  Z.pa<-cbind(Z.pa,dummy)
}
Z.pa<-Z.pa[,-1]
colnames(Z.pa) <- c("Bonds Only","HOPE SF","Inclusionary","Inclusionary-
OCII","Multifamily"                        ,"RAD Phase 1","RAD Phase
2","Small Sites" )

#Housing Tenure
first.round.test$HousingTenure <-
as.factor(first.round.test$HousingTenure)
ff<-first.round.test$HousingTenure
ll<-levels(ff)
Z.ht<-rep(0,(dim(first.round.test)[1]))
for(i in 1:length(ll)){
  dummy<-ifelse(ff==ll[i],1,0)
  Z.ht<-cbind(Z.ht,dummy)
}
Z.ht<-Z.ht[,-1]
colnames(Z.ht) <- c("Ownership","Rental","Unknown")
Z.test <-
cbind(Z.ht,Z.pa,Z.pt,first.round.test$MarketRate,first.round.test$FamilyU
nit,
                 first.round.test$ProjUnits)
colnames(Z.test)[c(14,15,16)] <- c("MarketRate","FamilyUnit","ProjUnits")

## Classification
library(class)
#K=3
knn3 <- knn(X.train,Z.test,cl=first.round.train$Failing,k=3)
table(first.round.test$Failing, knn3)
##test error
mean(first.round.test$Failing!=knn3)

#K=5
knn5 <- knn(X.train,Z.test,cl=first.round.train$Failing,k=5)
table(first.round.test$Failing, knn5)
##test error
mean(first.round.test$Failing!=knn5)

#K=10
knn10 <- knn(X.train,Z.test,cl=first.round.train$Failing,k=10)
table(first.round.test$Failing, knn10)
##test error
mean(first.round.test$Failing!=knn10)
## Cross Validation
```

```
### Transform Variables into Dummies
#Project Type
first.round$ProjectType <- as.factor(first.round$ProjectType)
ff<-first.round$ProjectType
ll<-levels(ff)
CV.pt<-rep(0,(dim(first.round)[1]))
for(i in 1:length(ll)){
  dummy<-ifelse(ff==ll[i],1,0)
  CV.pt<-cbind(CV.pt,dummy)
}
CV.pt<-CV.pt[,-1]
colnames(CV.pt) <- c("New Construction","Rehabilitation")

#Program area
first.round$ProgramArea <- as.factor(first.round$ProgramArea)
ff<-first.round$ProgramArea
ll<-levels(ff)
CV.pa<-rep(0,(dim(first.round)[1]))
for(i in 1:length(ll)){
  dummy<-ifelse(ff==ll[i],1,0)
  CV.pa<-cbind(CV.pa,dummy)
}
CV.pa<-CV.pa[,-1]
colnames(CV.pa) <- c("Bonds Only","HOPE SF","Inclusionary","Inclusionary-
OCII","Multifamily"                        ,"RAD Phase 1","RAD Phase
2","Small Sites" )

#Housing Tenure
first.round$HousingTenure <- as.factor(first.round$HousingTenure)
ff<-first.round$HousingTenure
ll<-levels(ff)
CV.ht<-rep(0,(dim(first.round)[1]))
for(i in 1:length(ll)){
  dummy<-ifelse(ff==ll[i],1,0)
  CV.ht<-cbind(CV.ht,dummy)
}
CV.ht<-CV.ht[,-1]
colnames(CV.ht) <- c("Ownership","Rental","Unknown")
CV.KNN <-
cbind(CV.ht,CV.pa,CV.pt,first.round$MarketRate,first.round$FamilyUnit,fir
st.round$ProjUnits)
colnames(CV.KNN)[c(14,15,16)] <- c("MarketRate","FamilyUnit","ProjUnits")

### Validate
nfolds=10
library(caret)
fold=createFolds(1:nrow(first.round),k=nfolds,list=FALSE)
kCV_err=rep(0,3)
for(i in 1:nfolds)
{
  pre3.CV=knn(CV.KNN[fold !=i,],CV.KNN[fold
==i,],cl=first.round$Failing[fold!=i],k=3)
  pre5.CV=knn(CV.KNN[fold !=i,],CV.KNN[fold
==i,],cl=first.round$Failing[fold!=i],k=5)
```

```
  pre10.CV=knn(CV.KNN[fold !=i,],CV.KNN[fold
==i,],cl=first.round$Failing[fold!=i],k=10)

kCV_err[1]=kCV_err[1]+mean(pre3.CV!=first.round$Failing[fold==i])/nfolds

kCV_err[2]=kCV_err[2]+mean(pre5.CV!=first.round$Failing[fold==i])/nfolds

kCV_err[3]=kCV_err[3]+mean(pre10.CV!=first.round$Failing[fold==i])/nfolds
}
data.frame(k=c(3,5,10),CV_error=kCV_err)
## ROC & AUC
##Training
failing_pi_knn_train<-attr(knn(X.train,X.train,first.round.train$Failing,
                               k=10,prob=TRUE),"prob")
failing_class <- knn(X.train,X.train,first.round.train$Failing,k=10)
pi_knn_train<-ifelse(failing_class==1,failing_pi_knn_train,1-
failing_pi_knn_train)
y_knn_train<-ifelse(pi_knn_train>0.5,1,0)
ER_knn_train<-mean((y_train-y_knn_train)^2)
ER_knn_train
pred_knn_train <- prediction(pi_knn_train, y_train)
perf_knn_train <- performance(pred_knn_train,"tpr","fpr")
plot(perf_knn_train,colorize=TRUE,main="KNN Train")
AUC_knn_train<-performance(pred_knn_train,"auc")@y.values[[1]]
AUC_knn_train


##Testing
failing_pi_knn_test<-
as.numeric(attr(knn(X.train,Z.test,first.round.train$Failing,
                                         k=10,prob=TRUE),"prob"))

failing_class <- knn(X.train,Z.test,first.round.train$Failing,k=10)
pi_knn_test<-ifelse(failing_class==1,failing_pi_knn_test,1-
failing_pi_knn_test)
y_knn_test<- ifelse(pi_knn_test>0.5,1,0)
ER_knn_test<-mean((y_test-y_knn_test)^2)
ER_knn_test
pred_knn_test <- prediction(pi_knn_test, y_test)
perf_knn_test <- performance(pred_knn_test,"tpr","fpr")
plot(perf_knn_test,colorize=TRUE,main="KNN Test")
AUC_knn_test<-performance(pred_knn_test,"auc")@y.values[[1]]
AUC_knn_test
# Decision Tree
first.round.train$Failing <- as.factor(first.round.train$Failing)
first.round.test$Failing <- as.factor(first.round.test$Failing)
library(tree)
tree1 <- tree(Failing~., data = first.round.train)
plot(tree1)
text(tree1,pretty=0)
#Prune
##cv.tree
m2 <- cv.tree(tree1,FUN=prune.misclass)
```

```
plot(m2$size, m2$dev, type="b",xlab="Tree Size",ylab="Misclassification
error")
tree2 <- prune.misclass(tree1,best=4)
plot(tree2)
text(tree2,pretty=0)
## ROC and AUC
##Training Error Rate
pi_dt_train <- predict(tree2,first.round.train)
y_dt_train <- ifelse(pi_dt_train[,2]>0.5,1,0)
ER_dt_train<-mean((y_train-y_dt_train)^2)
ER_dt_train
#Training ROC
pred_dt_train<-prediction(pi_dt_train[,2], y_train)
perf_dt_train <- performance(pred_dt_train,"tpr","fpr")
plot(perf_dt_train,colorize=TRUE,main="Decision Tree Training")
AUC_dt_train<-performance(pred_dt_train,"auc")@y.values[[1]]
AUC_dt_train

##Testing Error Rate
pi_dt_test <- predict(tree2,first.round.test)
y_dt_test <- ifelse(pi_dt_test[,2]>0.5,1,0)
ER_dt_test<-mean((y_test-y_dt_test)^2)
ER_dt_test
#Testing ROC
pred_dt_test<-prediction(pi_dt_test[,2], y_test)
perf_dt_test <- performance(pred_dt_test,"tpr","fpr")
plot(perf_dt_test,colorize=TRUE,main="Decision Tree Testing")
AUC_dt_test<-performance(pred_dt_test,"auc")@y.values[[1]]
AUC_dt_test
# Random Forest
library(randomForest)
forest <- randomForest(Failing~., data = first.round.train,mtry =
2,importance=TRUE)
forest.pred <- predict(forest,first.round.test)
mean(forest.pred != first.round.test$Failing)
varImpPlot(forest)
## Error Rate, ROC & AUC
##Training Error Rate
pi_rf_train <- predict(forest,first.round.train, type = "prob")
y_rf_train <- ifelse(pi_rf_train[,2]>0.5,1,0)
ER_rf_train<-mean((y_train-y_rf_train)^2)
ER_rf_train
#Training ROC
pred_rf_train<-prediction(pi_rf_train[,2], y_train)
perf_rf_train <- performance(pred_rf_train,"tpr","fpr")
plot(perf_rf_train,colorize=TRUE,main="Random Forest Training")
AUC_rf_train<-performance(pred_rf_train,"auc")@y.values[[1]]
AUC_rf_train

##Testing Error Rate
pi_rf_test <- predict(forest,first.round.test, type = "prob")
y_rf_test <- ifelse(pi_rf_test[,2]>0.5,1,0)
ER_rf_test<-mean((y_test-y_rf_test)^2)
ER_rf_test
```

```
#Testing ROC
pred_rf_test<-prediction(pi_rf_test[,2], y_test)
perf_rf_test <- performance(pred_rf_test,"tpr","fpr")
plot(perf_rf_test,colorize=TRUE,main="Random Forest Testing")
AUC_rf_test<-performance(pred_rf_test,"auc")@y.values[[1]]
AUC_rf_test

# Second Round
## Second Round Dataset & The original Dataset
#Second Round
second.round <- selected.housing
second.round[119,2] <- "Unknown"
second.round[151,2] <- "Unknown"
second.round$HousingTenure <- as.factor(second.round$HousingTenure)
second.round$ProgramArea <- as.factor(second.round$ProgramArea)
second.round$ProjectType <- as.factor(second.round$ProjectType)
second.round$Failing <- as.factor(second.round$Failing)

## First Simply Impute
#ProjUnits with its mean
second.round$ProjUnits[is.na(second.round$ProjUnits)]=mean(second.round$P
rojUnits,na.rm = TRUE)
#Failing with its mode
second.round$Failing[is.na(second.round$Failing)]="1"

## Iterative Regression
n_iter = 20
for(i in 1:n_iter){
  #Impute ProjUnits given rest
  m_ProjUnit = lm(ProjUnits~., second.round, subset =!
is.na(original$ProjUnits))
  pred_ProjUnit =
predict(m_ProjUnit,second.round[is.na(original$ProjUnits),])
  second.round$ProjUnits[is.na(original$ProjUnits)] = pred_ProjUnit
  #Impute Failings given rest
  m_Failing = glm(Failing~., second.round, subset =!
is.na(original$Failing)
                 ,family="binomial")
  pred_Failing =
predict(m_Failing,second.round[is.na(original$Failing),],
                         type = "response")
  second.round$Failing[is.na(original$Failing)] =
ifelse(pred_Failing>0.5,1,0)
}
##ProjUnit Distribution
par(mfrow=c(1,2))
hist(original$ProjUnits,breaks=60,main="Observed
data",xlab="ProjUnits",freq=FALSE)
hist(second.round$ProjUnits,breaks=60,main="Second Round Imputed
data",xlab="ProjUnits",freq=FALSE)
barplot(table(original$Failing),main="Observed data")
barplot(table(second.round$Failing),main="Imputed data")
summary(second.round)
```

```
## Split the data
set.seed(4052)
n <- length(second.round$Failing)
index <- sample(1:n,0.8*n)
second.round.train <- second.round[index,]
second.round.test <- second.round[-index,]

# Logistic Regression
log1.2nd <- glm(Failing~., data=second.round.train, family="binomial")
## deviance test p-value
pchisq(log1.2nd$deviance,217,lower.tail = FALSE)
##Pearson chi-square test
Pearson <- sum(residuals(log1.2nd,type = "pearson")^2)
pchisq(Pearson,217,lower.tail = FALSE)
##Backward Selection
step(log1.2nd)
##Outlier
plot(log1.2nd,which = 5)
## Model After Selection
log2.2nd <- glm(formula = Failing ~ ProgramArea + ProjectType +
MarketRate +
    FamilyUnit + ProjUnits, family = "binomial", data =
second.round.train)
plot(log2.2nd,which = 5)
## deviance test p-value
pchisq(log2.2nd$deviance,219,lower.tail = FALSE)
##Pearson chi-square test
Pearson <- sum(residuals(log2.2nd,type = "pearson")^2)
pchisq(Pearson,219,lower.tail = FALSE)
## Tested Out Nested Model
nest_log.2nd <- glm(formula = Failing ~ ProgramArea + ProjectType+
MarketRate + FamilyUnit +                     ProjUnits++I(MarketRate^2)
+I(FamilyUnit^2)+I(ProjUnits^2), family =
                   "binomial",data = second.round.train)
anova(log2.2nd,nest_log.2nd,test = "Chisq")
#No nested Model
##Quasi-Likelihood
quasi_log.2nd <- glm(formula = Failing ~ ProgramArea + ProjectType +
MarketRate +
                   FamilyUnit + ProjUnits, family = quasibinomial, data =
                     second.round.train)
summary(quasi_log.2nd)
## Test Error Rate & ROC
#Quasi Model
##Train Error Rate
y_train_2nd<-as.numeric(second.round.train[,1])-1
y_test_2nd<-as.numeric(second.round.test[,1])-1
pi_logit_train<-predict(quasi_log.2nd,
second.round.train,type="response")
y_logit_train<-ifelse(pi_logit_train>0.5,1,0)
ER_logit_train<-mean((y_train_2nd-y_logit_train)^2)
ER_logit_train
#Training ROC
library(ROCR)
```

```
pred_logit_train<-prediction(pi_logit_train, y_train_2nd)
perf_logit_train <- performance(pred_logit_train,"tpr","fpr")
plot(perf_logit_train,colorize=TRUE,main="Quasi Logistic Training")
AUC_logit_train<-performance(pred_logit_train,"auc")@y.values[[1]]
AUC_logit_train

##Test Error Rate
pi_logit_test<-predict(quasi_log.2nd, second.round.test,type="response")
y_logit_test<-ifelse(pi_logit_test>0.5,1,0)
ER_logit_test<-mean((y_test_2nd-y_logit_test)^2)
ER_logit_test
#Testing ROC
pred_logit_test<-prediction(pi_logit_test, y_test_2nd)
perf_logit_test <- performance(pred_logit_test,"tpr","fpr")
plot(perf_logit_test,colorize=TRUE,main="Quasi Logistic Testing")
AUC_logit_test<-performance(pred_logit_test,"auc")@y.values[[1]]
AUC_logit_test
# KNN
## Transforming data into dummies
### Training set
#Project Type
second.round.train$ProjectType <-
as.factor(second.round.train$ProjectType)
ff<-second.round.train$ProjectType
ll<-levels(ff)
X.pt<-rep(0,(dim(second.round.train)[1]))
for(i in 1:length(ll)){
  dummy<-ifelse(ff==ll[i],1,0)
  X.pt<-cbind(X.pt,dummy)
}
X.pt<-X.pt[,-1]
colnames(X.pt) <- c("New Construction","Rehabilitation")

#Program area
second.round.train$ProgramArea <-
as.factor(second.round.train$ProgramArea)
ff<-second.round.train$ProgramArea
ll<-levels(ff)
X.pa<-rep(0,(dim(second.round.train)[1]))
for(i in 1:length(ll)){
  dummy<-ifelse(ff==ll[i],1,0)
  X.pa<-cbind(X.pa,dummy)
}
X.pa<-X.pa[,-1]
colnames(X.pa) <- c("Bonds Only","HOPE SF","Inclusionary","Inclusionary-
OCII","Multifamily"                      ,"RAD Phase 1","RAD Phase
2","Small Sites" )

#Housing Tenure
second.round.train$HousingTenure <-
as.factor(second.round.train$HousingTenure)
ff<-second.round.train$HousingTenure
ll<-levels(ff)
X.ht<-rep(0,(dim(second.round.train)[1]))
```

```
for(i in 1:length(ll)){
  dummy<-ifelse(ff==ll[i],1,0)
  X.ht<-cbind(X.ht,dummy)
}
X.ht<-X.ht[,-1]
colnames(X.ht) <- c("Ownership","Rental","Unknown")
X.train <-
cbind(X.ht,X.pa,X.pt,second.round.train$MarketRate,second.round.train$Fam
ilyUnit,
                 second.round.train$ProjUnits)
colnames(X.train)[c(14,15,16)] <-
c("MarketRate","FamilyUnit","ProjUnits")
Failing.train <- second.round.train$Failing

### Testing Set
#Project Type
second.round.test$ProjectType <- as.factor(second.round.test$ProjectType)
ff<-second.round.test$ProjectType
ll<-levels(ff)
Z.pt<-rep(0,(dim(second.round.test)[1]))
for(i in 1:length(ll)){
  dummy<-ifelse(ff==ll[i],1,0)
  Z.pt<-cbind(Z.pt,dummy)
}
Z.pt<-Z.pt[,-1]
colnames(Z.pt) <- c("New Construction","Rehabilitation")

#Program area
second.round.test$ProgramArea <- as.factor(second.round.test$ProgramArea)
ff<-second.round.test$ProgramArea
ll<-levels(ff)
Z.pa<-rep(0,(dim(second.round.test)[1]))
for(i in 1:length(ll)){
  dummy<-ifelse(ff==ll[i],1,0)
  Z.pa<-cbind(Z.pa,dummy)
}
Z.pa<-Z.pa[,-1]
colnames(Z.pa) <- c("Bonds Only","HOPE SF","Inclusionary","Inclusionary-
OCII","Multifamily"                        ,"RAD Phase 1","RAD Phase
2","Small Sites" )

#Housing Tenure
second.round.test$HousingTenure <-
as.factor(second.round.test$HousingTenure)
ff<-second.round.test$HousingTenure
ll<-levels(ff)
Z.ht<-rep(0,(dim(second.round.test)[1]))
for(i in 1:length(ll)){
  dummy<-ifelse(ff==ll[i],1,0)
  Z.ht<-cbind(Z.ht,dummy)
}
Z.ht<-Z.ht[,-1]
colnames(Z.ht) <- c("Ownership","Rental","Unknown")
```

```
Z.test <-
cbind(Z.ht,Z.pa,Z.pt,second.round.test$MarketRate,second.round.test$Famil
yUnit,
                 second.round.test$ProjUnits)
colnames(Z.test)[c(14,15,16)] <- c("MarketRate","FamilyUnit","ProjUnits")

## Classification
library(class)
#K=3
knn3 <- knn(X.train,Z.test,cl=second.round.train$Failing,k=3)
table(second.round.test$Failing, knn3)
##test error
mean(second.round.test$Failing!=knn3)

#K=5
knn5 <- knn(X.train,Z.test,cl=second.round.train$Failing,k=5)
table(second.round.test$Failing, knn5)
##test error
mean(second.round.test$Failing!=knn5)

#K=10
knn10 <- knn(X.train,Z.test,cl=second.round.train$Failing,k=10)
table(second.round.test$Failing, knn10)
##test error
mean(second.round.test$Failing!=knn10)
## Cross Validation
### Transform Variables into Dummies
#Project Type
second.round$ProjectType <- as.factor(second.round$ProjectType)
ff<-second.round$ProjectType
ll<-levels(ff)
CV.pt<-rep(0,(dim(second.round)[1]))
for(i in 1:length(ll)){
  dummy<-ifelse(ff==ll[i],1,0)
  CV.pt<-cbind(CV.pt,dummy)
}
CV.pt<-CV.pt[,-1]
colnames(CV.pt) <- c("New Construction","Rehabilitation")

#Program area
second.round$ProgramArea <- as.factor(second.round$ProgramArea)
ff<-second.round$ProgramArea
ll<-levels(ff)
CV.pa<-rep(0,(dim(second.round)[1]))
for(i in 1:length(ll)){
  dummy<-ifelse(ff==ll[i],1,0)
  CV.pa<-cbind(CV.pa,dummy)
}
CV.pa<-CV.pa[,-1]
colnames(CV.pa) <- c("Bonds Only","HOPE SF","Inclusionary","Inclusionary-
OCII","Multifamily"                      ,"RAD Phase 1","RAD Phase
2","Small Sites" )

#Housing Tenure
```

```
second.round$HousingTenure <- as.factor(second.round$HousingTenure)
ff<-second.round$HousingTenure
ll<-levels(ff)
CV.ht<-rep(0,(dim(second.round)[1]))
for(i in 1:length(ll)){
  dummy<-ifelse(ff==ll[i],1,0)
  CV.ht<-cbind(CV.ht,dummy)
}
CV.ht<-CV.ht[,-1]
colnames(CV.ht) <- c("Ownership","Rental","Unknown")
CV.KNN <-
cbind(CV.ht,CV.pa,CV.pt,second.round$MarketRate,second.round$FamilyUnit,
                second.round$ProjUnits)
colnames(CV.KNN)[c(14,15,16)] <- c("MarketRate","FamilyUnit","ProjUnits")

### Validate
set.seed(2021)
nfolds=10
library(caret)
fold=createFolds(1:nrow(second.round),k=nfolds,list=FALSE)
kCV_err=rep(0,3)
for(i in 1:nfolds)
{
  pre3.CV=knn(CV.KNN[fold !=i,],CV.KNN[fold
==i,],cl=second.round$Failing[fold!=i],k=3)
  pre5.CV=knn(CV.KNN[fold !=i,],CV.KNN[fold
==i,],cl=second.round$Failing[fold!=i],k=5)
  pre10.CV=knn(CV.KNN[fold !=i,],CV.KNN[fold
==i,],cl=second.round$Failing[fold!=i],k=10)

kCV_err[1]=kCV_err[1]+mean(pre3.CV!=second.round$Failing[fold==i])/nfolds

kCV_err[2]=kCV_err[2]+mean(pre5.CV!=second.round$Failing[fold==i])/nfolds

kCV_err[3]=kCV_err[3]+mean(pre10.CV!=second.round$Failing[fold==i])/nfold
s
}
data.frame(k=c(3,5,10),CV_error=kCV_err)
## ROC & AUC
##Training
failing_pi_knn_train<-
attr(knn(X.train,X.train,second.round.train$Failing,
                               k=10,prob=TRUE),"prob")
failing_class <- knn(X.train,X.train,second.round.train$Failing,k=10)
pi_knn_train<-ifelse(failing_class==1,failing_pi_knn_train,1-
failing_pi_knn_train)
y_knn_train<-ifelse(pi_knn_train>0.5,1,0)
ER_knn_train<-mean((y_train_2nd-y_knn_train)^2)
ER_knn_train
pred_knn_train <- prediction(pi_knn_train, y_train_2nd)
perf_knn_train <- performance(pred_knn_train,"tpr","fpr")
plot(perf_knn_train,colorize=TRUE,main="KNN Train")
AUC_knn_train<-performance(pred_knn_train,"auc")@y.values[[1]]
AUC_knn_train
```

```
##Testing
failing_pi_knn_test<-
as.numeric(attr(knn(X.train,Z.test,second.round.train$Failing,
                                        k=10,prob=TRUE),"prob"))

failing_class <- knn(X.train,Z.test,second.round.train$Failing,k=10)
pi_knn_test<-ifelse(failing_class==1,failing_pi_knn_test,1-
failing_pi_knn_test)
y_knn_test<- ifelse(pi_knn_test>0.5,1,0)
ER_knn_test<-mean((y_test_2nd-y_knn_test)^2)
ER_knn_test
pred_knn_test <- prediction(pi_knn_test, y_test_2nd)
perf_knn_test <- performance(pred_knn_test,"tpr","fpr")
plot(perf_knn_test,colorize=TRUE,main="KNN Test")
AUC_knn_test<-performance(pred_knn_test,"auc")@y.values[[1]]
AUC_knn_test

# Desision Tree
second.round.train$Failing <- as.factor(second.round.train$Failing)
second.round.test$Failing <- as.factor(second.round.test$Failing)
library(tree)
tree1.2nd <- tree(Failing~., data = second.round.train)
plot(tree1.2nd)
text(tree1.2nd,pretty=0)
#Prune
##cv.tree
m2 <- cv.tree(tree1.2nd,FUN=prune.misclass)
plot(m2$size, m2$dev, type="b",xlab="Tree Size",ylab="Misclassification
error")
tree2.2nd <- prune.misclass(tree1.2nd,best=2)
plot(tree2.2nd)
text(tree2.2nd,pretty=0)

## Error Rate, ROC and AUC
##Training Error Rate
pi_dt_train_2nd <- predict(tree2.2nd,second.round.train)
y_dt_train_2nd <- ifelse(pi_dt_train_2nd[,2]>0.5,1,0)
ER_dt_train_2nd<-mean((y_train_2nd-y_dt_train_2nd)^2)
ER_dt_train_2nd
#Training ROC
pred_dt_train<-prediction(pi_dt_train_2nd[,2], y_train_2nd)
perf_dt_train <- performance(pred_dt_train,"tpr","fpr")
plot(perf_dt_train,colorize=TRUE,main="Decision Tree Training")
AUC_dt_train<-performance(pred_dt_train,"auc")@y.values[[1]]
AUC_dt_train

##Testing Error Rate
pi_dt_test_2nd <- predict(tree2.2nd,second.round.test)
y_dt_test_2nd <- ifelse(pi_dt_test_2nd[,2]>0.5,1,0)
ER_dt_test_2nd<-mean((y_test_2nd-y_dt_test_2nd)^2)
ER_dt_test_2nd
#Testing ROC
```

```
pred_dt_test<-prediction(pi_dt_test_2nd[,2], y_test_2nd)
perf_dt_test <- performance(pred_dt_test,"tpr","fpr")
plot(perf_dt_test,colorize=TRUE,main="Decision Tree Testing")
AUC_dt_test<-performance(pred_dt_test,"auc")@y.values[[1]]
AUC_dt_test

# Random Forest
library(randomForest)
forest_2nd <- randomForest(Failing~., data = second.round.train,mtry =
1,importance=TRUE)
forest.pred <- predict(forest_2nd,second.round.test)
mean(forest.pred != second.round.test$Failing)
varImpPlot(forest_2nd)
## Error Rate, ROC & AUC
##Training Error Rate
pi_rf_train_2nd <- predict(forest_2nd,second.round.train, type = "prob")
y_rf_train_2nd <- ifelse(pi_rf_train_2nd[,2]>0.5,1,0)
ER_rf_train_2nd<-mean((y_train_2nd-y_rf_train_2nd)^22)
ER_rf_train_2nd
#Training ROC
pred_rf_train_2nd<-prediction(pi_rf_train_2nd[,2], y_train_2nd)
perf_rf_train_2nd <- performance(pred_rf_train_2nd,"tpr","fpr")
plot(perf_rf_train,colorize=TRUE,main="Random Forest Training")
AUC_rf_train<-performance(pred_rf_train_2nd,"auc")@y.values[[1]]
AUC_rf_train


##Testing Error Rate
pi_rf_test_2nd <- predict(forest_2nd,second.round.test, type = "prob")
y_rf_test_2nd <- ifelse(pi_rf_test_2nd[,2]>0.5,1,0)
ER_rf_test<-mean((y_test_2nd-y_rf_test_2nd)^2)
ER_rf_test
#Testing ROC
pred_rf_test_2nd<-prediction(pi_rf_test_2nd[,2], y_test_2nd)
perf_rf_test_2nd <- performance(pred_rf_test_2nd,"tpr","fpr")
plot(perf_rf_test_2nd,colorize=TRUE,main="Random Forest Testing")
AUC_rf_test<-performance(pred_rf_test_2nd,"auc")@y.values[[1]]
AUC_rf_test
```