# Analysis2

*Sara Bolf, Leslie An, Camille Little*

*12/14/2019*

```
#import data

data<-read.csv("/Users/Sara/Downloads/new.csv", header = TRUE,stringsAsFactors = FALSE, fileEncoding="la
```

## Dropped irrelvant columns and transformed certain columns

```
#Remove irrelevant columns and transform certain variables to factors or integers

# drop unnecessary columns from data

drop <- c("url","id","floor","tradeTime","Cid","DOM","totalPrice","constructionTime")

data <- data[,!(names(data) %in% drop)]

# transform columns to integers

data$livingRoom = as.integer(data$livingRoom)
```

```
## Warning: NAs introduced by coercion
```

```
data$drawingRoom = as.integer(data$drawingRoom)
```

```
## Warning: NAs introduced by coercion
```

```
data$bathRoom = as.integer(data$bathRoom)
```

```
## Warning: NAs introduced by coercion
```

```
# omit NA's

data = na.omit(data)
```

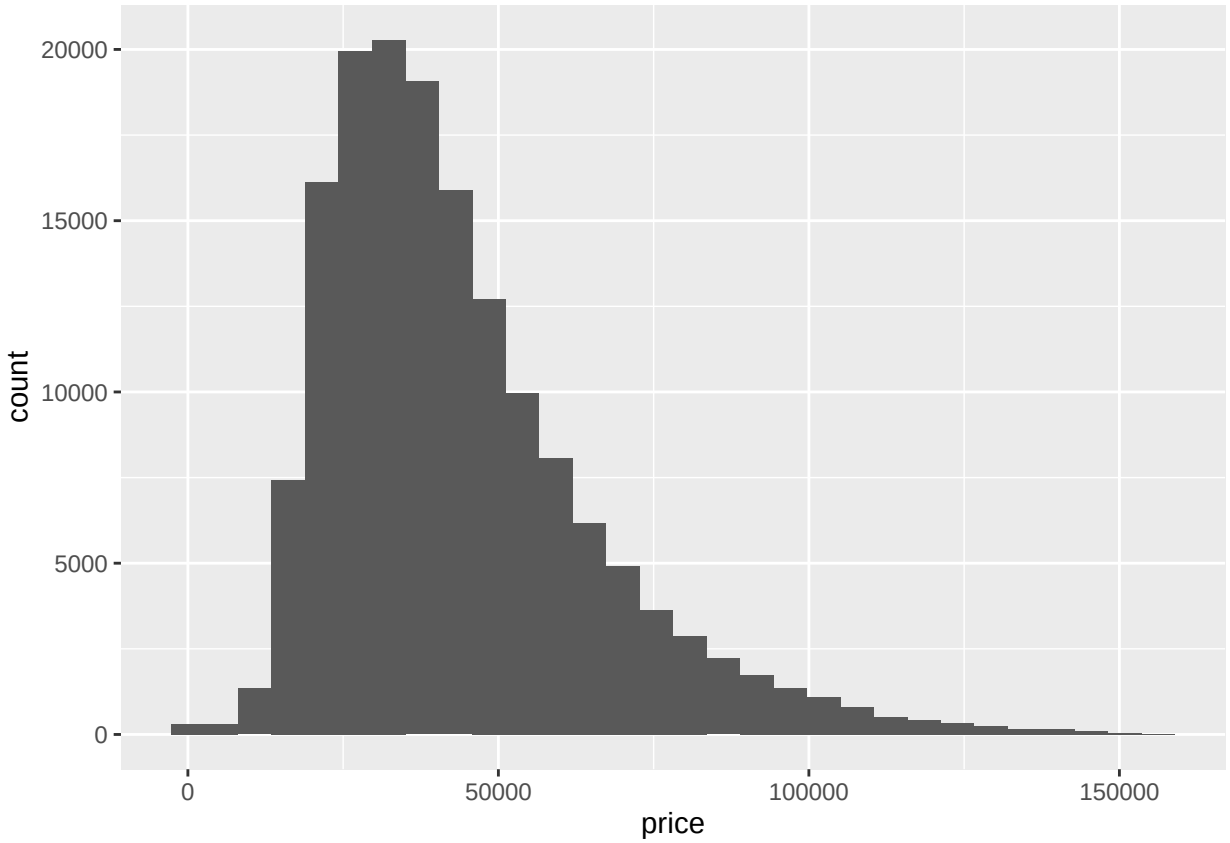
## Data Split

```
# Split Data into two data sets: data1 and data2.  data1 is the data set with 50% of data, randomly cho

splitsize = floor(.50*(nrow(data)))

set.seed(123)

split1 = sample(1:nrow(data),size=splitsize)

data1 = data[split1,]

data2 = data[-split1,]
```

## Exploratory Data Analysis

```
# Histogram of prices

ggplot(data1, aes(x=price)) + geom_histogram()
```

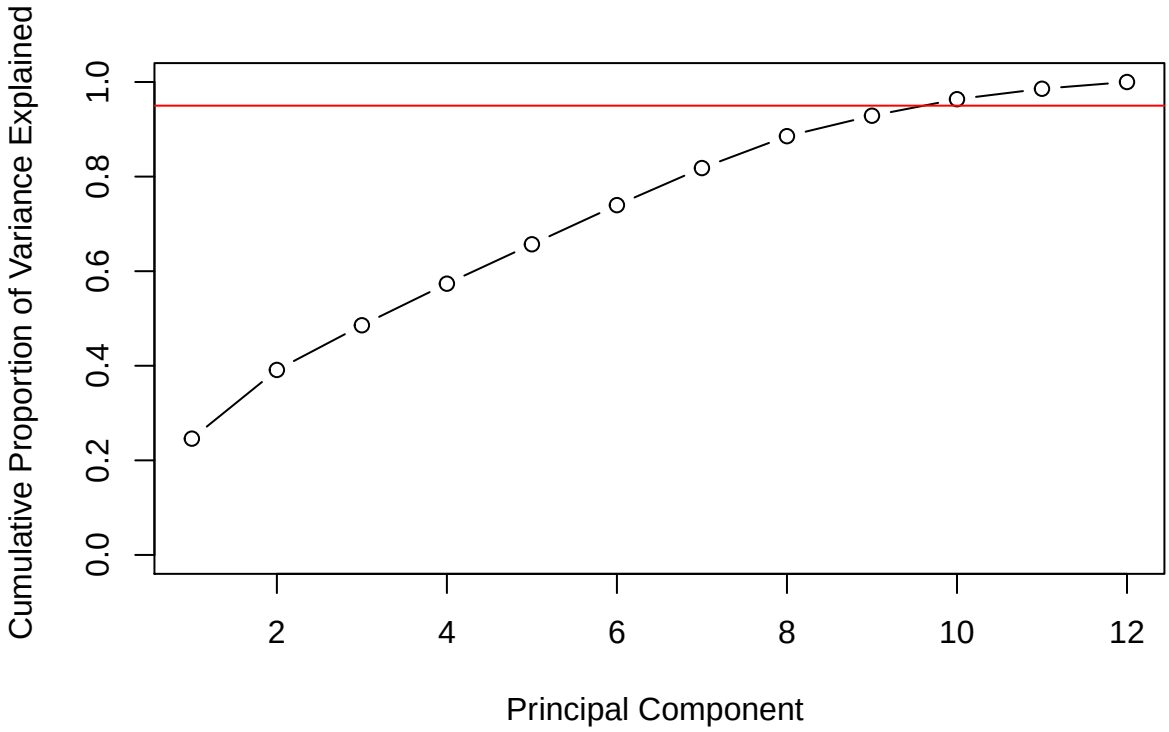
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



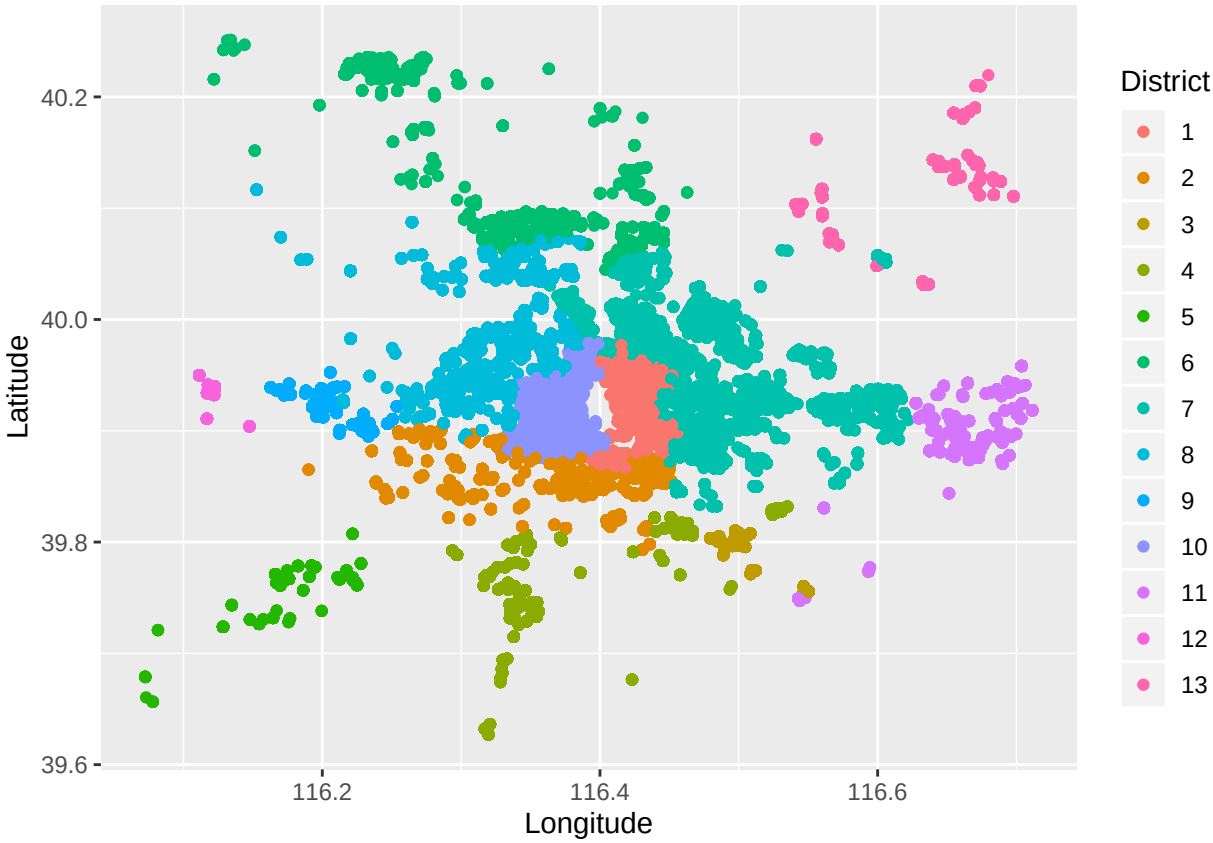

```
# predictors without categorical variables

X = data1[,c(-10,-11,-12,-15,-16,-17)]
```

```
# Principal Component Analysis

pc = princomp(X,cor=TRUE)
```

```
# Plot CPVE

prinvar = pc$sdev^2 # variance

pveprin = prinvar/sum(prinvar) # PVE

cpveprin = cumsum(pveprin) # cumulative PVE = CPVE

plot(cpveprin, xlab="Principal Component ", ylab="Cumulative Proportion of Variance Explained ", ylim=c

abline(h=0.95,col="red")
```




```
# Plot longitude vs. latitude, colored by district

qplot(X$Lng, X$Lat,color=as.factor(data1$district),xlab="Longitude",ylab="Latitude")+labs(colour="Distri
```



# Analysis

# Regression Analysis

```
# subset data

subset_data<- sample(1:nrow(data1), size = 5000)

subset_data<-data1[subset_data,]
```

```
#Remove certain columns

subset_data<-subset(subset_data, select=-c(buildingType, buildingStructure, Lng, Lat))
```

```
#Standardize Data

x = subset_data
x$district = as.character(x$district)
x$fiveYearsProperty = as.character(x$fiveYearsProperty)
x$subway = as.character(x$subway)
x$renovationCondition = as.character(x$renovationCondition)
x$livingRoom = as.integer(x$livingRoom)
x$drawingRoom = as.integer(x$drawingRoom)
x$bathRoom = as.integer(x$bathRoom)
for (j in 1:length(x)){
  if ( typeof(x[1,j])!= "character") {
    x[,j] = (x[,j] - mean(x[,j])) / sd(x[,j])

    }
}
```

```
# Run OLS

ols = lm(price ~ ., data = x)

summary(ols)
```

```
##
## Call:
## lm(formula = price ~ ., data = x)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.2759 -0.3854 -0.0841  0.3090  3.4979
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)          -0.332255   0.049056  -6.773 1.41e-11 ***
## followers             0.203128   0.009096  22.331  < 2e-16 ***
## square               -0.095225   0.017630  -5.401 6.92e-08 ***
## livingRoom            0.025308   0.013514   1.873 0.061159 .
## drawingRoom          -0.005418   0.011259  -0.481 0.630361
## kitchen               0.008109   0.009205   0.881 0.378381
## bathRoom              0.010105   0.012814   0.789 0.430365
```

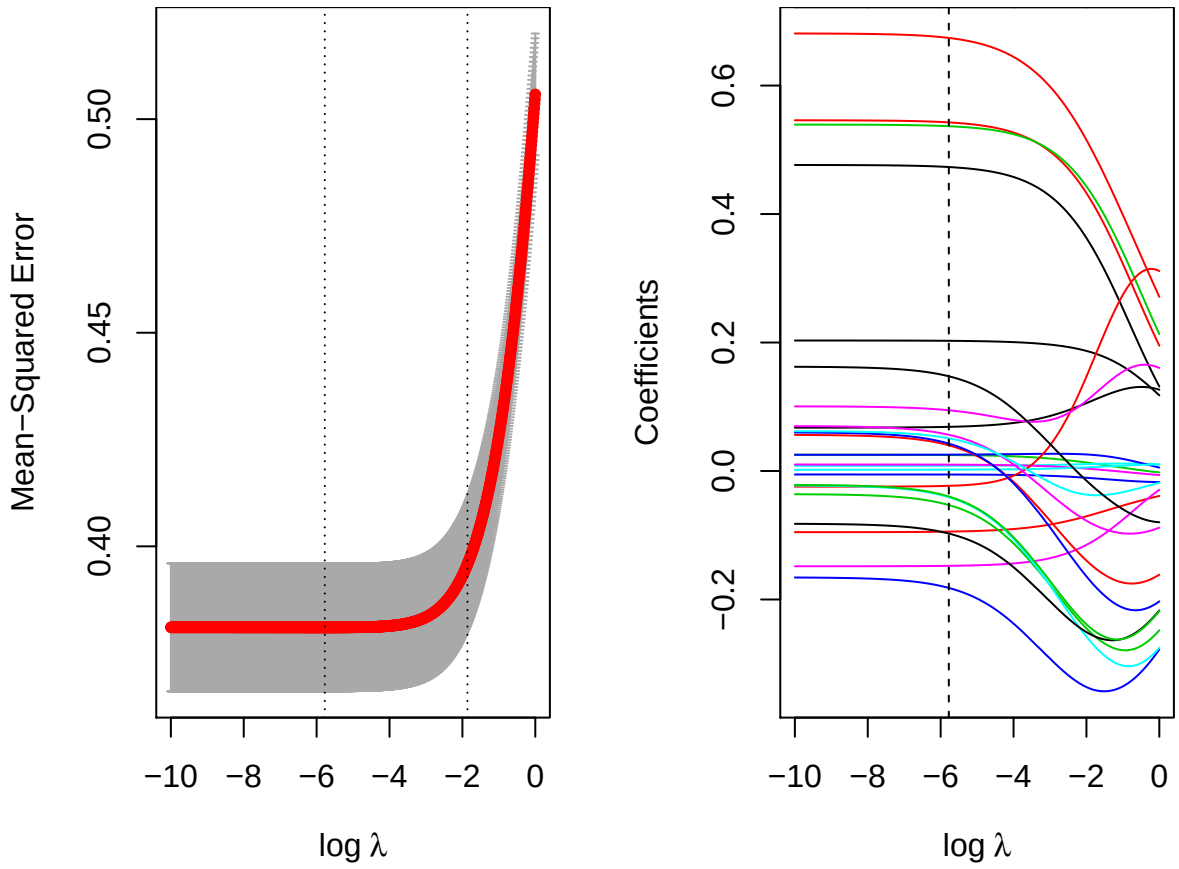
```
## renovationCondition2  0.476415   0.063472   7.506 7.19e-14 ***
## renovationCondition3  0.546000   0.023757  22.983  < 2e-16 ***
## renovationCondition4  0.539269   0.021165  25.479  < 2e-16 ***
## ladderRatio           0.025254   0.009877   2.557 0.010587 *
## elevator              0.001957   0.010121   0.193 0.846659
## fiveYearsProperty1   -0.148296   0.018971  -7.817 6.55e-15 ***
## subway1               0.067477   0.019416   3.475 0.000515 ***
## district10           -0.024345   0.049512  -0.492 0.622952
## district11           -0.035892   0.066332  -0.541 0.588466
## district12           -0.165539   0.142791  -1.159 0.246385
## district13           -0.022399   0.075093  -0.298 0.765498
## district2             0.070073   0.055087   1.272 0.203419
## district3            -0.081995   0.104840  -0.782 0.434192
## district4             0.056447   0.065447   0.862 0.388460
## district5            -0.021414   0.109561  -0.195 0.845046
## district6             0.060237   0.058674   1.027 0.304642
## district7             0.062172   0.047036   1.322 0.186292
## district8             0.100689   0.047708   2.111 0.034864 *
## district9             0.162470   0.067721   2.399 0.016472 *
## communityAverage      0.681168   0.015580  43.721  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6151 on 4973 degrees of freedom
## Multiple R-squared:  0.6236, Adjusted R-squared:  0.6217
## F-statistic: 316.9 on 26 and 4973 DF,  p-value: < 2.2e-16
```

```
# Ridge Regression

set.seed(123)

x_ridge = model.matrix(price~.,x)[,-1]

y_ridge = x$price

lambda_vec = exp(seq(-10,0,len=1000))

cv.out = cv.glmnet(x_ridge, y_ridge, alpha = 0, lambda=lambda_vec)

ridge_lambda = cv.out$lambda.min

ridge_lambda
```

```
## [1] 0.00310179
```

```
# Plot Ridge Regression

ridge = glmnet(x_ridge, y_ridge, alpha = 0, lambda = lambda_vec)

par(mfrow=c(1,2),mar=c(4,4,0,2))

plot(cv.out,xlab=expression(log~lambda))

plot(ridge,xvar='lambda',xlab=expression(log~lambda))
```

```
abline(v=log(ridge_lambda),lty=2)
```

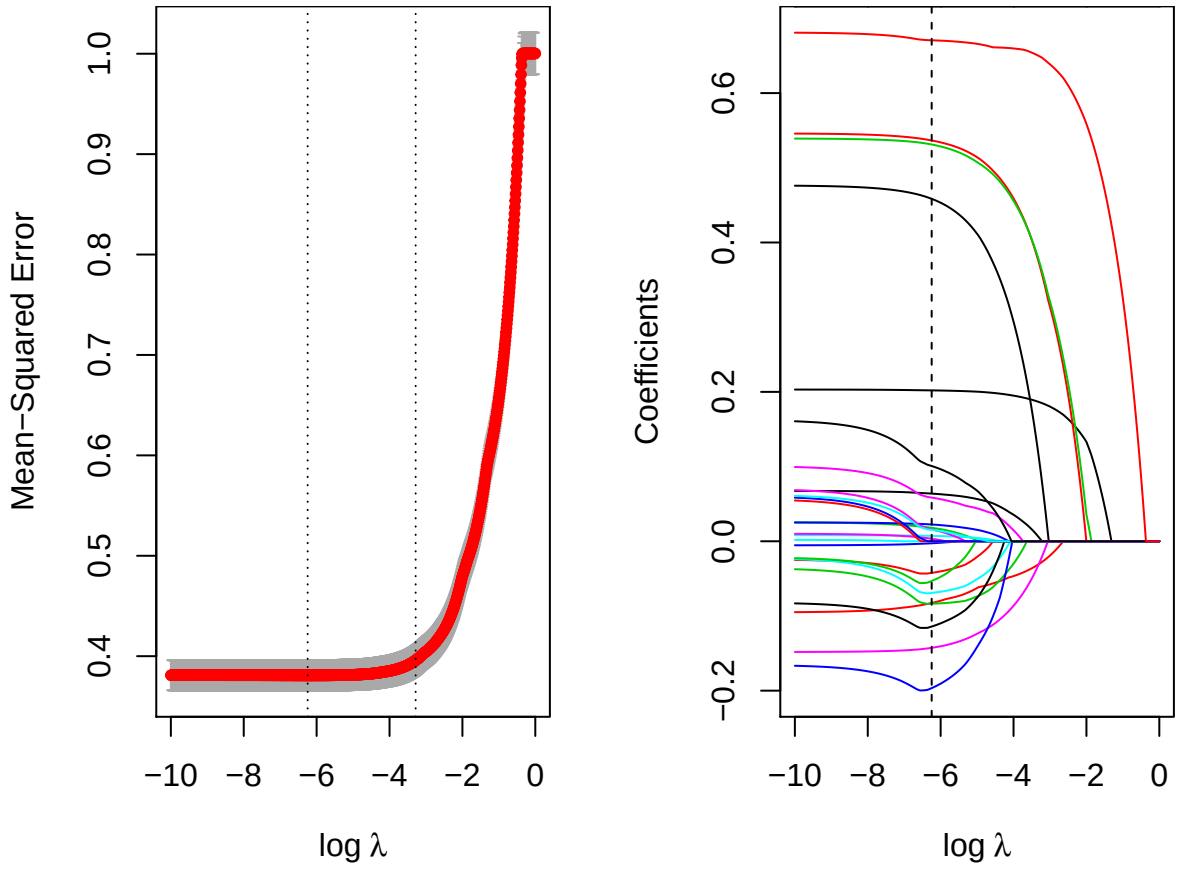



```
# Lasso Regression

set.seed(123)

x_lasso = model.matrix(price ~ ., x)[,-1]

y_lasso = x$price

lambda_vec = exp(seq(-10,0,len=1000))

cv.out = cv.glmnet(x_lasso, y_lasso, alpha = 1, lambda=lambda_vec)

lasso_lambda = cv.out$lambda.min

lasso_lambda
```

```
## [1] 0.001937714
```

```
# Plot Lasso

lasso = glmnet(x_lasso, y_lasso, alpha = 1, lambda = lambda_vec)

par(mfrow=c(1,2),mar=c(4,4,0,2))

plot(cv.out,xlab=expression(log~lambda))
```

```
plot(lasso,xvar='lambda',xlab=expression(log~lambda))

abline(v=log(lasso_lambda),lty=2)
```

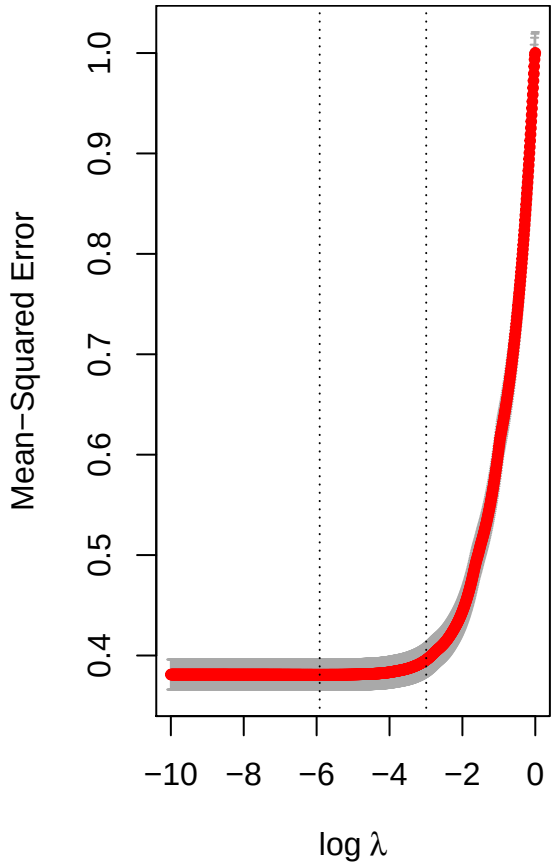


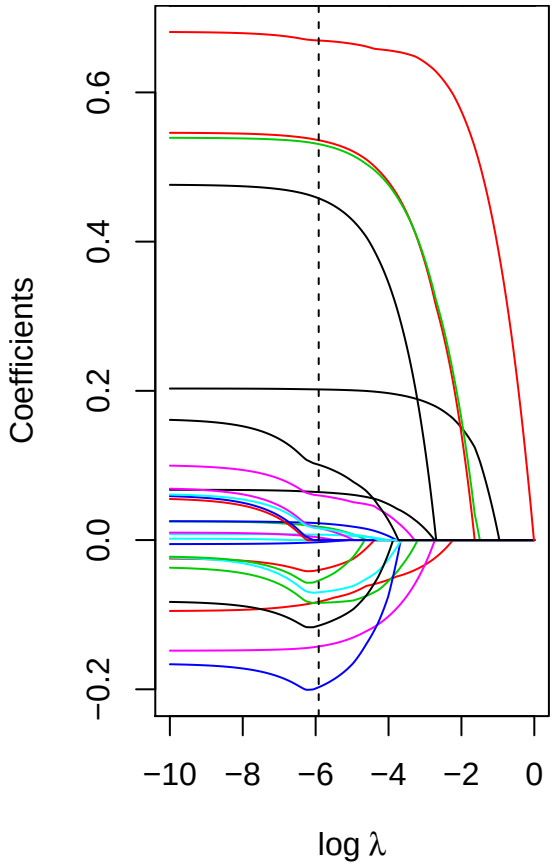

```
# Elastic Net

set.seed(123)

x_enet = model.matrix(price ~ ., x)[,-1]

y_enet = x$price

cv.out = cv.glmnet(x_enet, y_enet, alpha = 0.7, lambda=lambda_vec)

enet_lambda = cv.out$lambda.min

enet_lambda
```

```
## [1] 0.002696189
```

```
# Plot Elastic Net

enet = glmnet(x_enet, y_enet, alpha = 0.7, lambda = lambda_vec)

par(mfrow=c(1,2),mar=c(4,4,0,2))
```

```r
plot(cv.out,xlab=expression(log~lambda))

plot(enet,xvar='lambda',xlab=expression(log~lambda))

abline(v=log(enet_lambda),lty=2)
```




```r
# Sample from DATA2

set.seed(1234)
test_data<- sample(1:nrow(data2), size = 5000)
test_data<-data2[test_data,]
test_data<-subset(test_data, select=-c(buildingType, buildingStructure, Lng, Lat))
test_data<-na.omit(test_data)

x2 = test_data
x2$district = as.character(x2$district)
x2$fiveYearsProperty = as.character(x2$fiveYearsProperty)
x2$subway = as.character(x2$subway)
x2$renovationCondition = as.character(x2$renovationCondition)
x2$livingRoom = as.integer(x2$livingRoom)
x2$drawingRoom = as.integer(x2$drawingRoom)
x2$bathRoom = as.integer(x2$bathRoom)
for (j in 1:length(x2)){
  if ( typeof(x2[1,j])!= "character") {
    x2[,j] = (x2[,j] - mean(x2[,j])) / sd(x2[,j])
```

```
    }
    }
```

```
# MSE for OLS

y = (x2$price - mean(x2$price)) / sd(x2$price)

test_mat = model.matrix(price ~ ., data = x2)[,-1]

ols_pred = predict(ols, newdata=x2)

ols_error = (y - ols_pred)^2

ols_MSE = mean(ols_error)

ols_MSE
```

```
## [1] 0.3899533
```

```
# MSE for Ridge

ridge_pred = predict(ridge, s = ridge_lambda, newx = test_mat)

ridge_error = (y - ridge_pred) ^2

ridge_MSE = mean(ridge_error)

ridge_MSE
```

```
## [1] 0.3899128
```

```
# MSE for LASSO

lasso_pred = predict(lasso, s = lasso_lambda, newx = test_mat)

lasso_error = (y - lasso_pred)^2

lasso_MSE = mean(lasso_error)

lasso_MSE
```

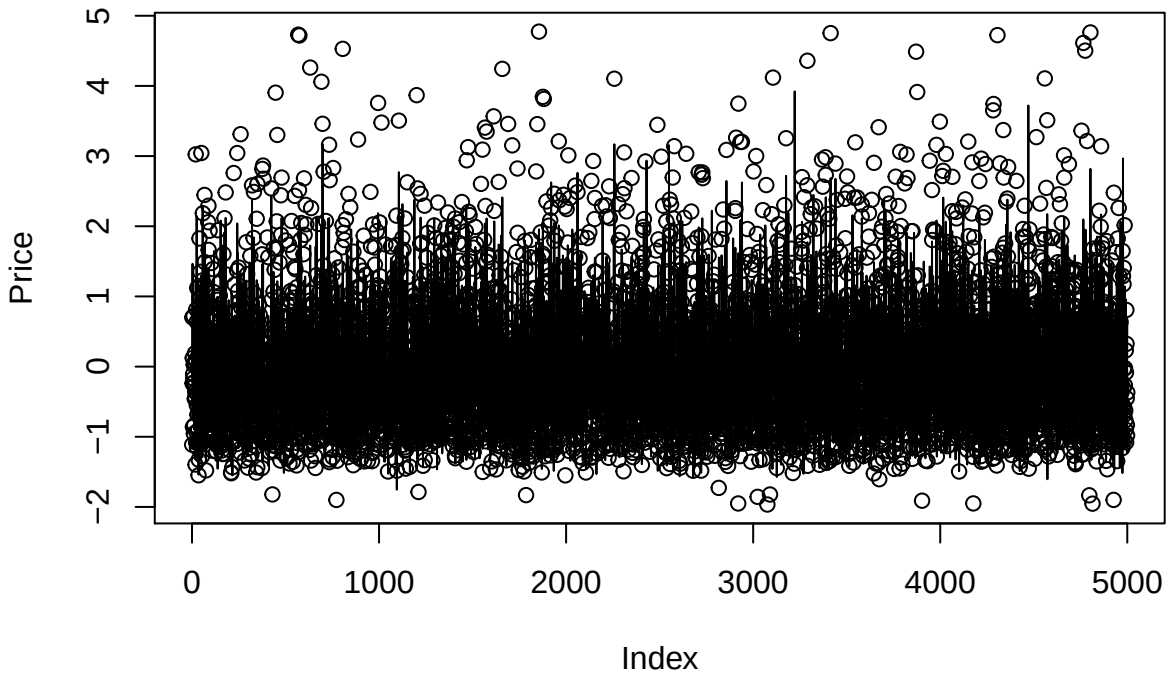
```
## [1] 0.389982
```

```
# MSE for Elastic Net

enet_pred = predict(enet, s = enet_lambda, newx = test_mat)

enet_error = (y - enet_pred)^2

enet_MSE = mean(enet_error)

enet_MSE
```

```
## [1] 0.3899816
```

```
# Plot Goodness of Fit for Elastic Net
```

```r
plot(x2$price, ylab = "Price")

lines(enet_pred)
```




```r
# Table of errors and lambdas

mean_error<- matrix(c(ols_MSE ,ridge_MSE, lasso_MSE, enet_MSE),nrow=4,byrow=TRUE)

lambda_vec <- matrix(c("NA",ridge_lambda, lasso_lambda, enet_lambda),nrow=4,byrow=TRUE)

finalvec = cbind(mean_error,lambda_vec)

rownames(finalvec) <- c("OLS","Ridge", "Lasso", "Elastic Net")

colnames(finalvec) <- c("Mean Squared Error","Lambda")

final_table <- as.table(finalvec)

final_table
```

```
##             Mean Squared Error Lambda
## OLS         0.389953329738332  NA
## Ridge       0.389912765660478  0.00310179043115763
## Lasso       0.389981990054226  0.00193771420341532
## Elastic Net 0.389981618579442  0.00269618918500105
```

## Classification Analysis

```r
# transform columns to factors

data1$fiveYearsProperty = as.factor(data1$fiveYearsProperty)
```

```r
data1$buildingType = as.factor(data1$buildingType)

data1$renovationCondition = as.factor(data1$renovationCondition)

data1$buildingStructure = as.factor(data1$buildingStructure)

data1$subway = as.factor(data1$subway)

data1$district = as.factor(data1$district)

data2$fiveYearsProperty = as.factor(data2$fiveYearsProperty)

data2$buildingType = as.factor(data2$buildingType)

data2$renovationCondition = as.factor(data2$renovationCondition)

data2$buildingStructure = as.factor(data2$buildingStructure)

data2$subway = as.factor(data2$subway)

data2$district = as.factor(data2$district)
```

```r
# Further split data1 and data2, randomly sampling 5% of data1 and data2, in order to simplify calculat

samplesize = floor(.05*nrow(data1))

sample1 = sample(1:nrow(data1),size=samplesize)

data1sub = data1[sample1,]

samplesize = floor(.05*nrow(data2))

sample1 = sample(1:nrow(data2),size=samplesize)

data2sub = data2[sample1,]
```

```r
# Remove latitude and longitude columns

data1sub2 = data1sub[,3:ncol(data1sub)]

data2sub2 = data2sub[,3:ncol(data2sub)]

# make sure factor levels are equal
```

```r
# multinomial model

fit_nom = multinom(district ~., data = data1sub2)
```

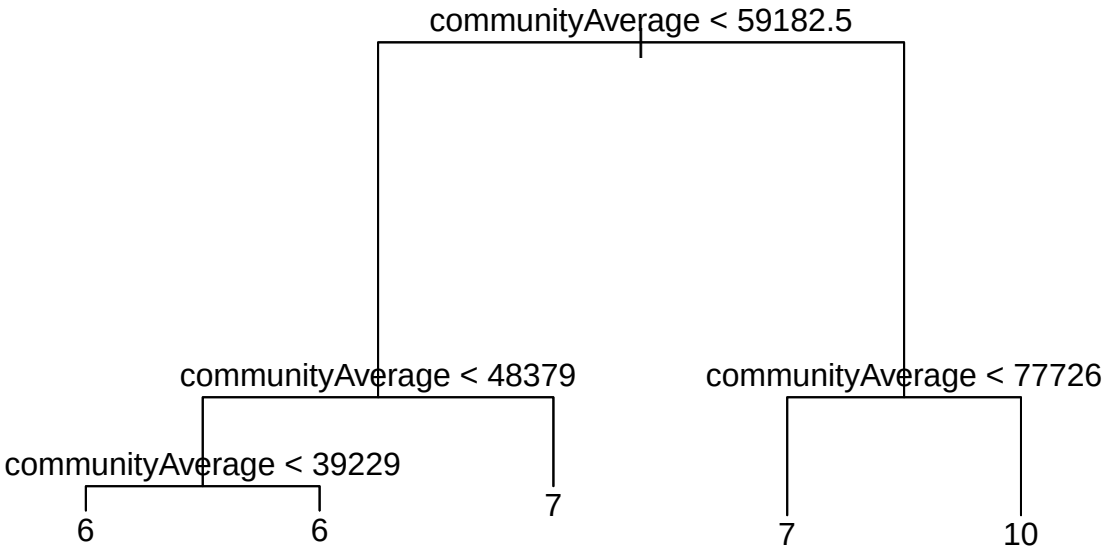
```
## # weights:  325 (288 variable)
## initial  value 20291.314367
## iter  10 value 16696.443542
## iter  20 value 16662.182796
## iter  30 value 14997.396108
## iter  40 value 14607.284367
```

```
## iter  50 value 13922.054448
## iter  60 value 12708.729146
## iter  70 value 11649.946161
## iter  80 value 11320.265667
## iter  90 value 11146.353151
## iter 100 value 10952.484207
## final  value 10952.484207
## stopped after 100 iterations
```

```r
# predictors without categorical variables

X = data1sub[,c(-10,-11,-12,-15,-16,-17)]
```

```r
# QDA model

X2 = data.frame(X[,3:ncol(X)])

X2 = X2[,-6]

X2$district = data1sub$district # create a data set with only continuous predictors and response
                                # district

mod2 = qda(district~.,data=X2) # QDA
```

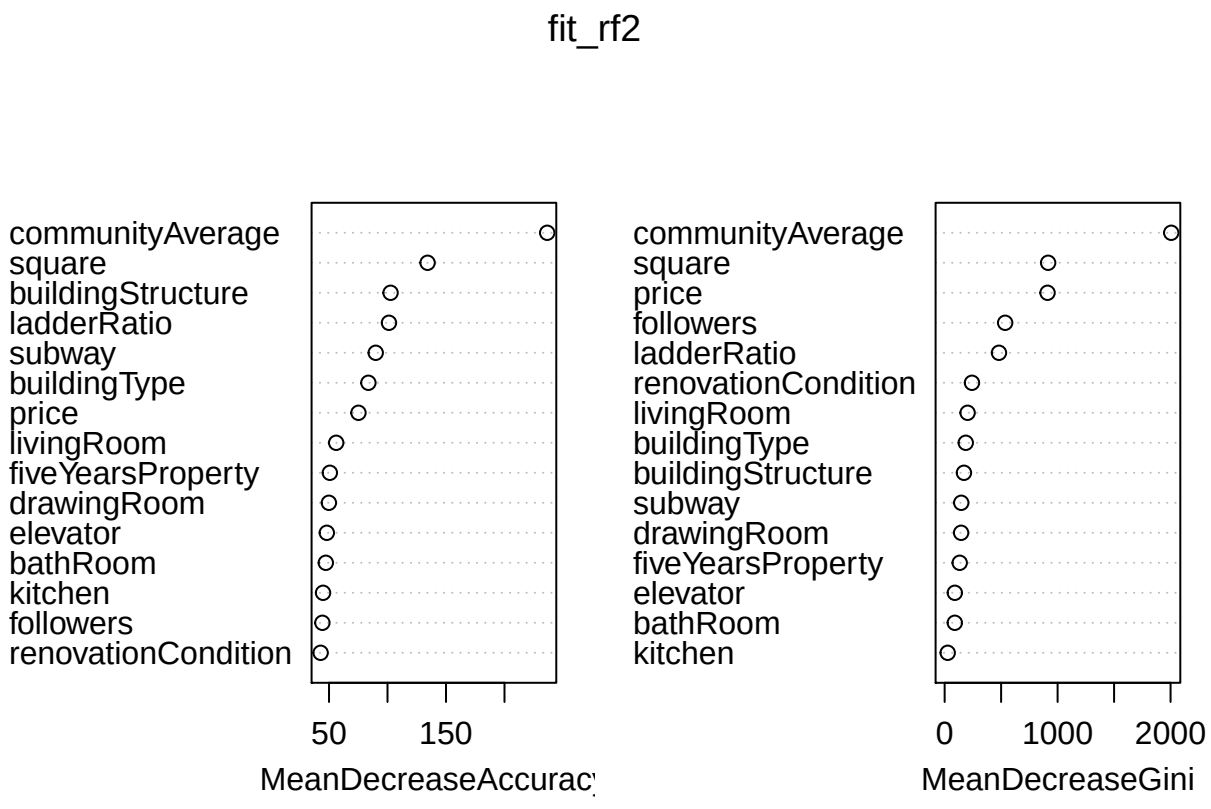
```r
# decision tree

set.seed(1)

full_tree = tree(district ~., data = data1sub2)

candidates = cv.tree(full_tree)

fit_tree = prune.misclass(full_tree, best = candidates$size[which.min(candidates$dev)])

plot(fit_tree)

text(fit_tree, pretty = 0)
```



```r
# Random Forest

fit_rf2 = randomForest(district ~., data = data1sub2, mtry = 4, importance = T, ntree = 1000)

importtab2 = importance(fit_rf2) # Table with variable importance

importtab2[,14:15]
```

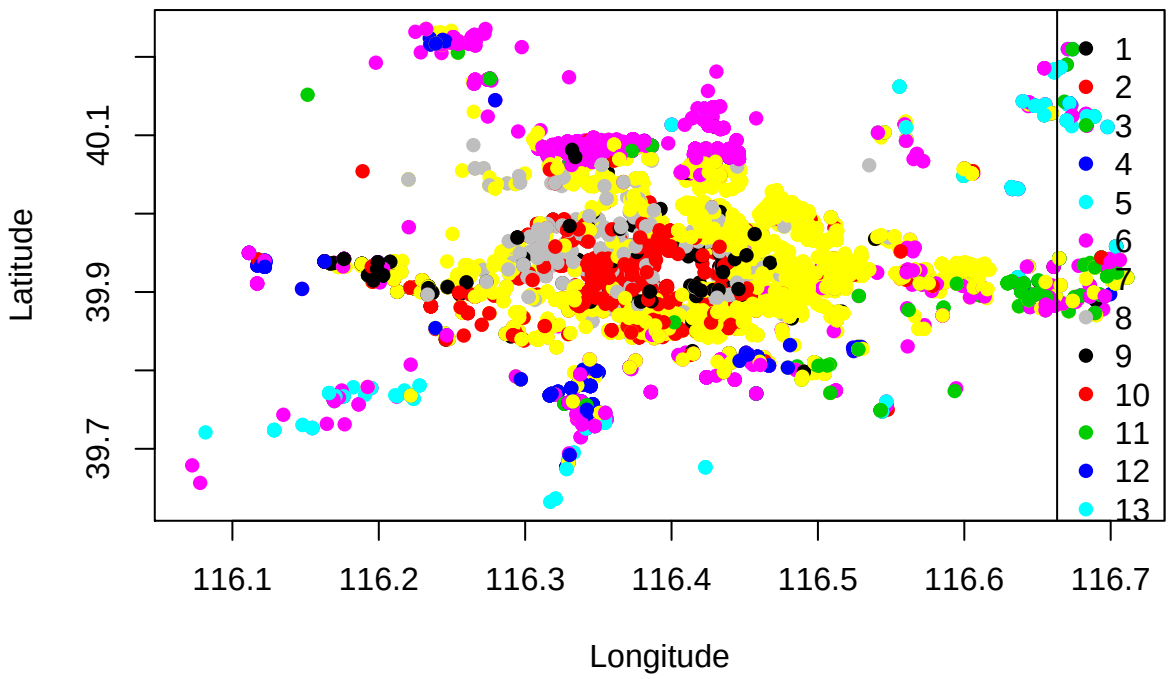
```
##                       MeanDecreaseAccuracy MeanDecreaseGini
## followers                          44.32459        535.58342
## price                              75.16272        911.01447
## square                            134.26489        916.04187
## livingRoom                         56.13373        203.17549
## drawingRoom                        49.88471        145.80672
## kitchen                            44.89811         26.99078
## bathRoom                           47.24067         89.76896
## buildingType                       83.64263        186.93092
## renovationCondition                42.82866        241.86552
## buildingStructure                 102.62459        170.76058
## ladderRatio                       101.23627        480.78071
## elevator                           48.10930         89.99056
## fiveYearsProperty                  50.69679        133.33969
## subway                             89.90532        146.93525
## communityAverage                  236.52593       2005.88907
```

```r
varImpPlot(fit_rf2)
```




```r
# Plot of longitude and latitude with colors given by district predicted by random forest
```

```
plot(data2sub$Lng,data2sub$Lat,col= (predict(fit_rf2,data2sub2,type="class")),pch=16,xlab="Longitude",y
legend("topright",legend=sort(unique(predict(fit_rf2,data2sub2,type="class"))),col=sort(unique(predict(
```




```
# Total Test Error for multinomial logistic regression

predlog1 = predict(fit_nom,data2sub2,type="class")

telog1 = mean(predlog1!=data2sub2$district)
```

```
# Total Test Error for QDA

predqda1 = predict(mod2,data2sub2,type="class")

teqda1 = mean(predqda1$class!=data2sub2$district)
```

```
# Total Test Error for decision Tree

predtree1 = predict(fit_tree,data2sub2,type="class")

tetree1 = mean(predtree1!=data2sub2$district)
```

```
# Total Test Error for Random Forest

predrf1 = predict(fit_rf2,data2sub2,type="class")

terf1 = mean(predrf1!=data2sub2$district)
```

```
# Table for test errors for different classification methods

te1 = matrix(c(telog1,teqda1,tetree1,terf1),ncol=1,byrow=TRUE)

rownames(te1) <- c("Multinomial Logistic","QDA","Tree","Random Forest (m=4)")
```

```r
colnames(te1) <- c("Test Total Error")

te1tab <- as.table(te1) # create table

te1tab
```

```
##                       Test Total Error
## Multinomial Logistic         0.5178865
## QDA                          0.5628871
## Tree                         0.5549235
## Random Forest (m=4)          0.3716344
```