Model Prediction on House Prices in Ames, Iowa

Quinn He

2023-05-25

Motivations

House prices are difficult to predict due to the countless number of variables and outside influences on the housing market as a whole. With data available from a 2016 Kaggle competition to predict house prices in Ames, Iowa, I plan to use supervised learning methods to predict the final sale price of houses. I have recently been interested in what goes into a home's sale price. As buying a home becomes more of a distant dream than a grounded reality for many young Americans, I am curious to see if I can create a model that helps predict sale prices in certain areas. While the data is only a sample of homes in Ames, Iowa, I hope I am able to figure out which models perform best on house prices with data that is a mix of quantitative and qualitative variables.

There are too many variables that combine together to inform a final sale price of a home. For a human to complete a project by hand that accounts for all these variables would be near impossible. By using machine learning principles I hope to accurately predict the prices of houses in Ames, Iowa. The models I create in the project will not be applicable to other house locations and that is a huge limitation to the project that should be accounted for. Certain geographic areas tend to me more expensive than others due to a variety of outside factors. Many of the homes in this data set are one story, single family homes so a model run here would not be sufficient to use to predict other areas, but it can be a start. With housing prices, the market is influenced by outside sources unaccounted for in the data set. Random financial fluctuations or recent natural disasters can be difficult to predict and is far outside the scope of this project.

The goal of this project is to use supervised learning methods to predict house sale prices given the 80 variables in the data set. I will implement forms of random forests, K-nearest neighbors, Ridge Regression, and Lasso machine learning models to make predictions on a testing data set. These models will help predict house prices given variables are constant and related to the house features.

More information about the data set can be found at the link below. https://www.kaggle.com/competitions/house-prices-advanced-regression-techniques/overview

EDA

There is an evenly balanced make up of discrete and continuous variables within the data. With almost 7000 missing variables, I will have to find a way of getting rid of them since some of the models do not perform well with NAs. With 1460 rows, I can conclude that number of houses should be sufficient to train a model.

```
introduce(train.house)
```

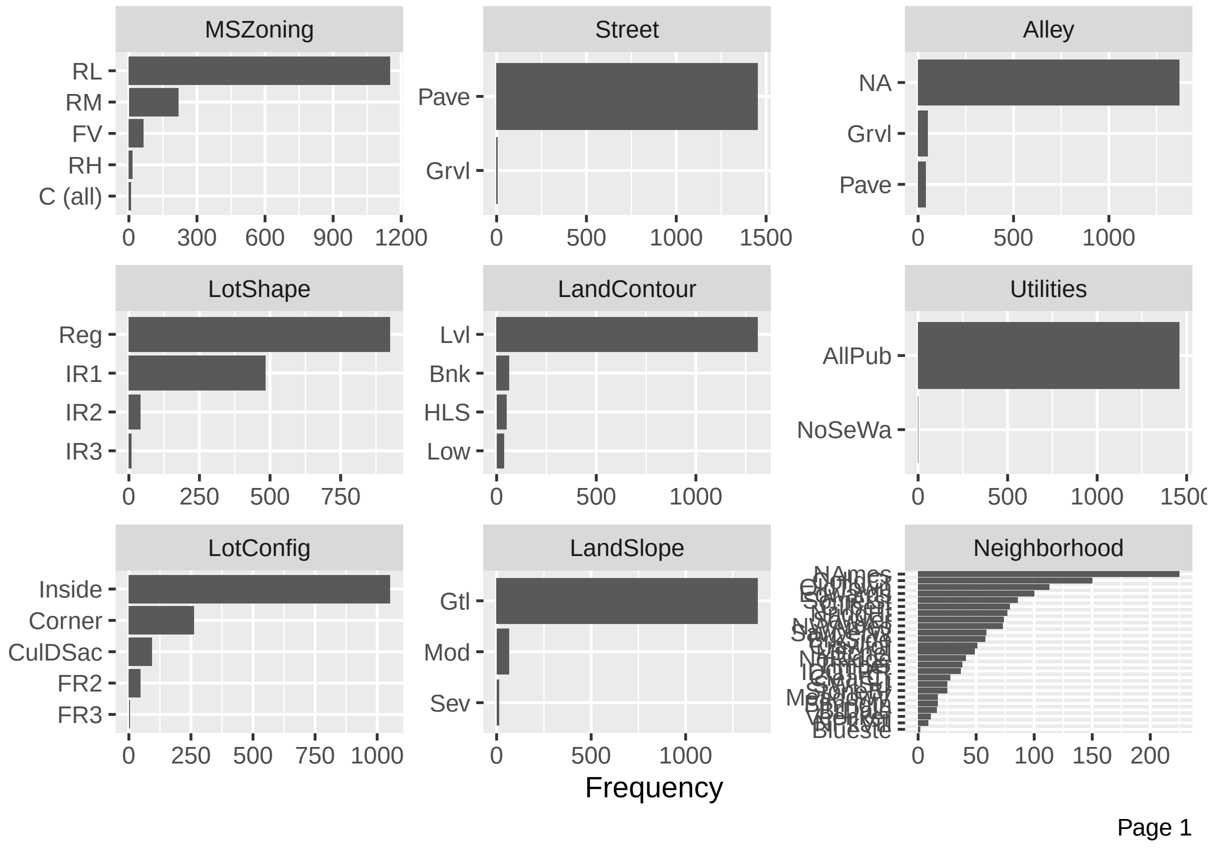
```
## # A tibble: 1 x 9
##    rows columns discrete_columns continuous_columns all_missing_columns
##   <int>   <int>            <int>              <int>               <int>
## 1  1460      81               43                 38                   0
## # i 4 more variables: total_missing_values <int>, complete_rows <int>,
## #   total_observations <int>, memory_usage <dbl>
```

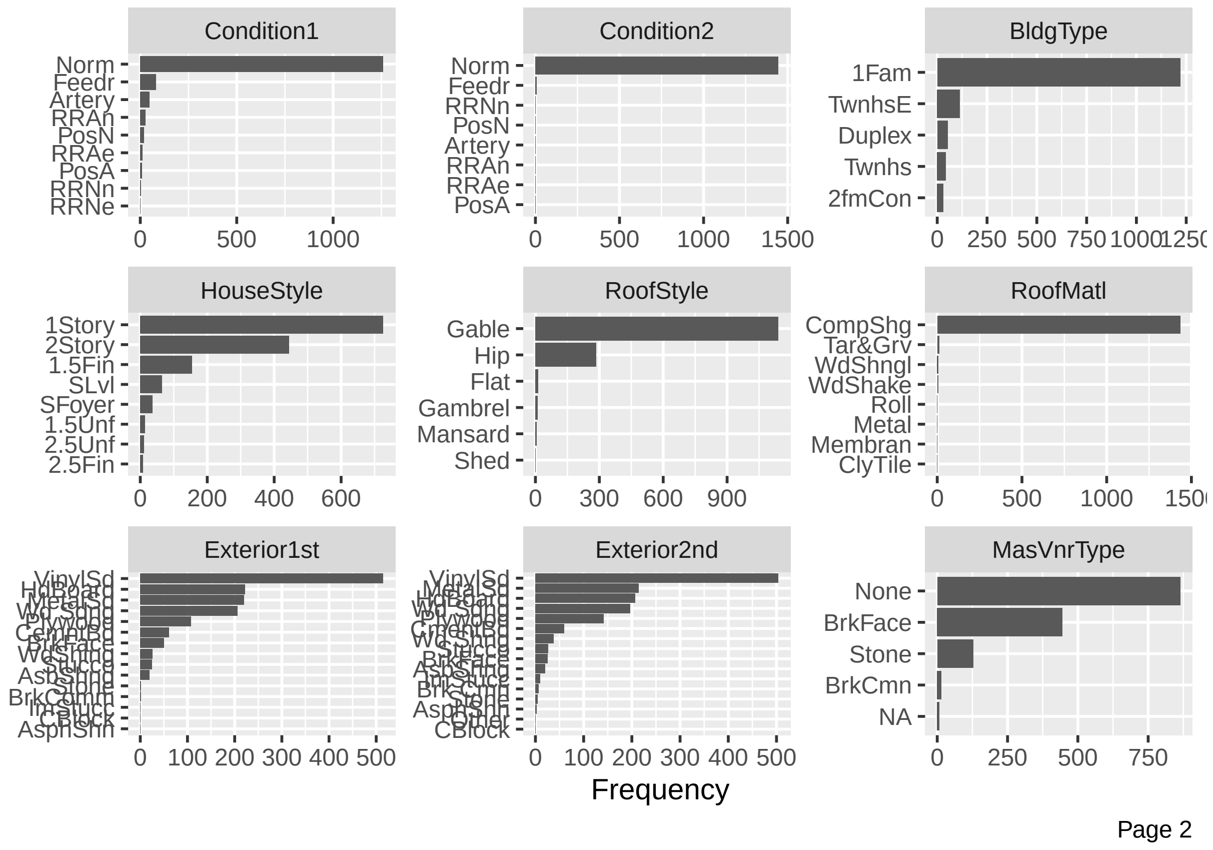
I can conclude most of the houses in the training set are 1 family, 1 story homes with a gable roof. The foundations are primarily of poured concrete or cinderblock construction. Most of the homes do not have a fence. Many variables pertain to the quality of certain aspects of the home (foundation, roof, fence, etc.) and a majority of them are of normal/average quality with little variation. Most of the homes are registered as Residential Low Density homes.

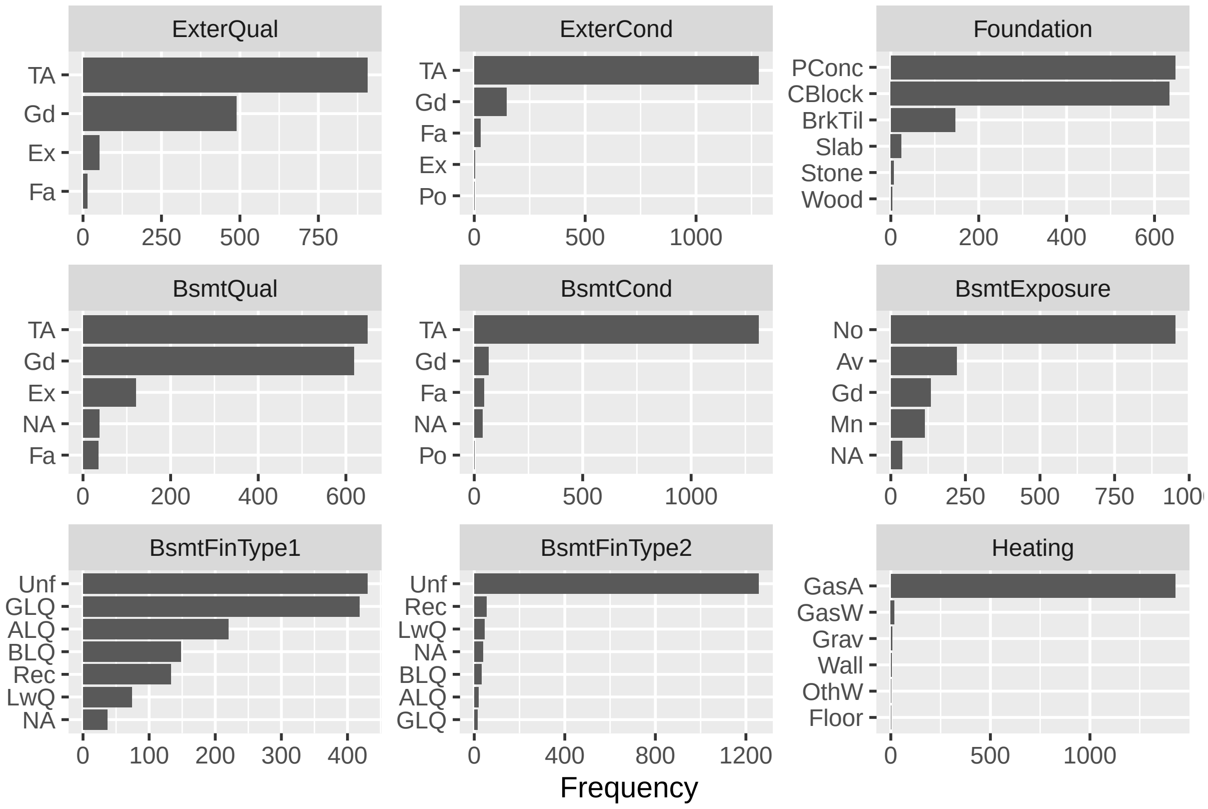
Most of the homes do not have a pool, fence, or other features such as a shed. Based on the distribution of almost all of the variables, many of them consist of just one type of observation.

Below, I am able to see the frequency of the categorical variables.

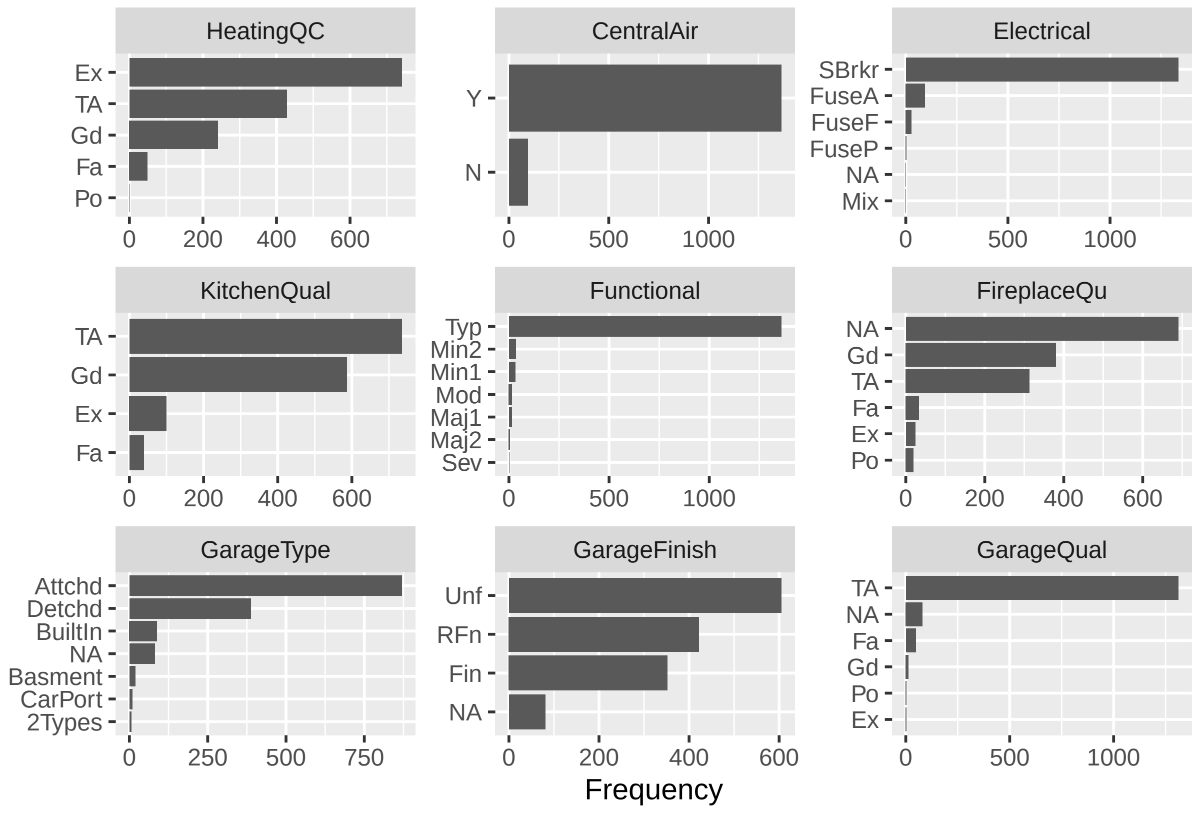
```
plot_bar(train.house)
```



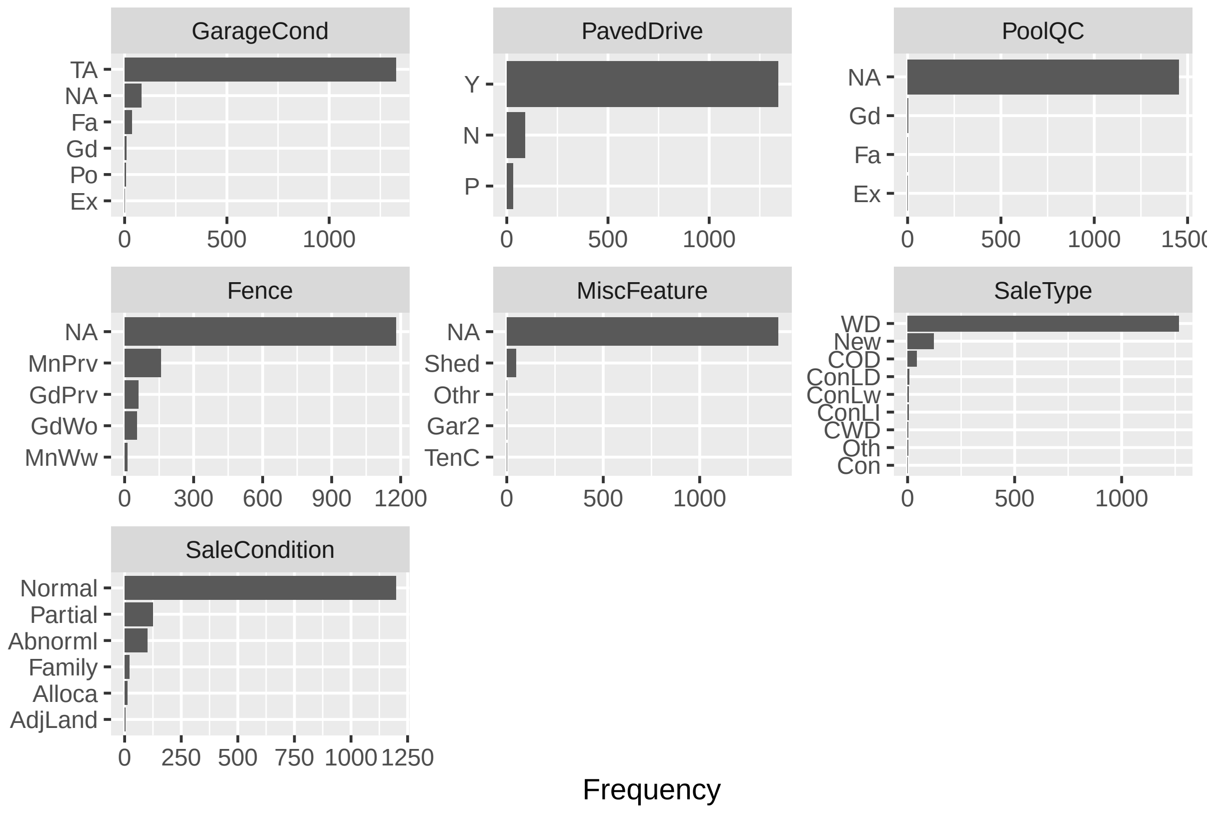

Condition1
Norm
Feedr
Artery
RRAn
PosN
RRAe
PosA
RRNn
RRNe
0
500
1000
Condition2
Norm
Feedr
RRNn
PosN
Artery
RRAn
RRAe
PosA
0
500
1000
1500
BldgType
1Fam
TwnhsE
Duplex
Twnhs
2fmCon
0
250
500
750
1000
1250
HouseStyle
1Story
2Story
1.5Fin
SLvl
SFoyer
1.5Unf
2.5Unf
2.5Fin
0
200
400
600
RoofStyle
Gable
Hip
Flat
Gambrel
Mansard
Shed
0
300
600
900
RoofMatl
CompShg
Tar&Grv
WdShngl
WdShake
Roll
Metal
Membran
ClyTile
0
500
1000
1500
Exterior1st
VinylSd
HdBoard
MetalSd
Wd Sdng
Plywood
CemntBd
BrkFace
WdShing
Stucco
AsbShng
Stone
BrkComm
ImStucc
CBlock
AsphShn
0
100
200
300
400
500
Exterior2nd
VinylSd
MetalSd
HdBoard
Wd Sdng
Plywood
CmentBd
Wd Shng
Stucco
BrkFace
AsbShng
ImStucc
Brk Cmn
Stone
AsphShn
Other
CBlock
0
100
200
300
400
500
MasVnrType
None
BrkFace
Stone
BrkCmn
NA
0
250
500
750
Frequency
Page 2


ExterQual
TA
Gd
Ex
Fa
0
250
500
750
ExterCond
TA
Gd
Fa
Ex
Po
0
500
1000
Foundation
PConc
CBlock
BrkTil
Slab
Stone
Wood
0
200
400
600
BsmtQual
TA
Gd
Ex
NA
Fa
0
200
400
600
BsmtCond
TA
Gd
Fa
NA
Po
0
500
1000
BsmtExposure
No
Av
Gd
Mn
NA
0
250
500
750
1000
BsmtFinType1
Unf
GLQ
ALQ
BLQ
Rec
LwQ
NA
0
100
200
300
400
BsmtFinType2
Unf
Rec
LwQ
NA
BLQ
ALQ
GLQ
0
400
800
1200
Heating
GasA
GasW
Grav
Wall
OthW
Floor
0
500
1000
Frequency

Page 3


HeatingQC
Ex
TA
Gd
Fa
Po
CentralAir
Y
N
Electrical
SBrkr
FuseA
FuseF
FuseP
NA
Mix
KitchenQual
Functional
Typ
Min2
Min1
Mod
Maj1
Maj2
Sev
FireplaceQu
GarageType
Attchd
Detchd
BuiltIn
Basment
CarPort
2Types
GarageFinish
Unf
RFn
Fin
GarageQual
Frequency

Page 4



The initial data set pre-split has 1460 observations with 81 variables (79 of which are explanatory variables).

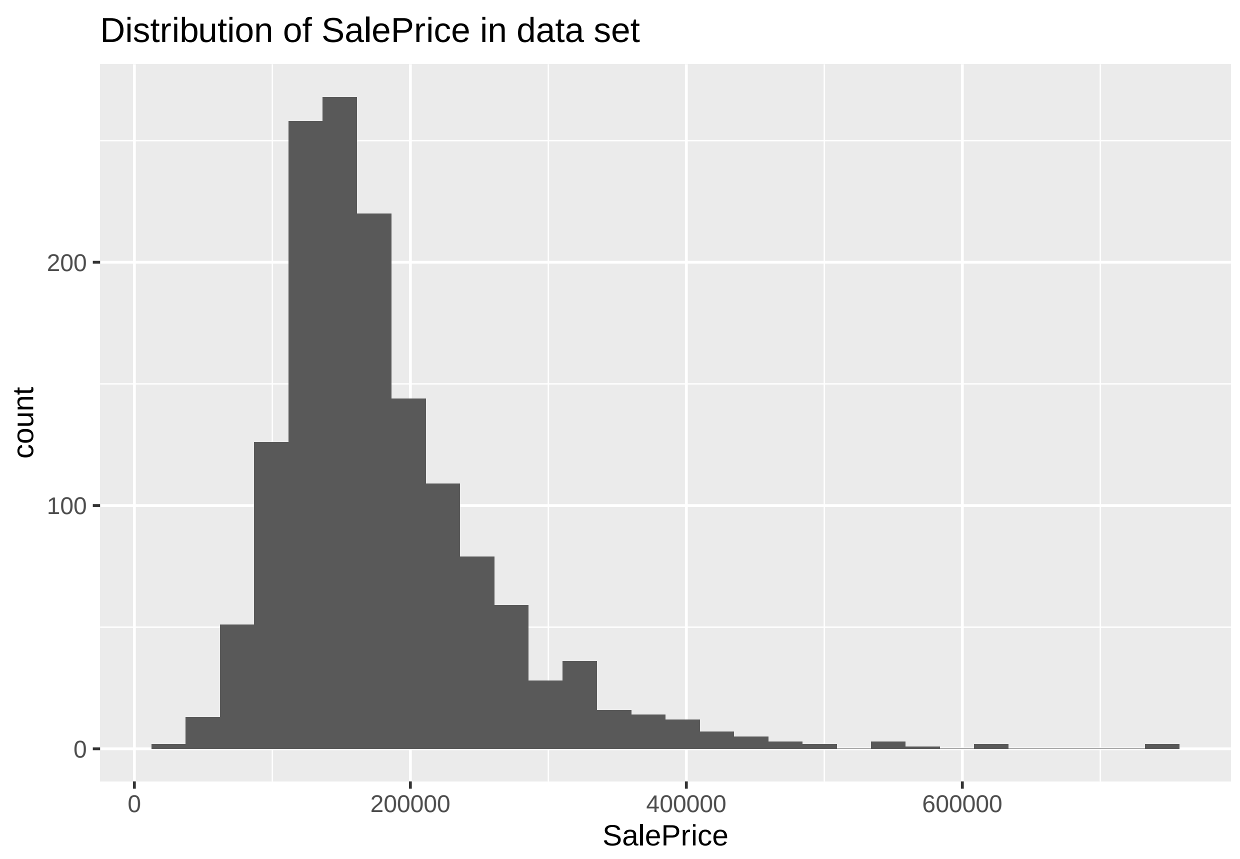
```
dim(train.house)
```

```
## [1] 1460   81
```

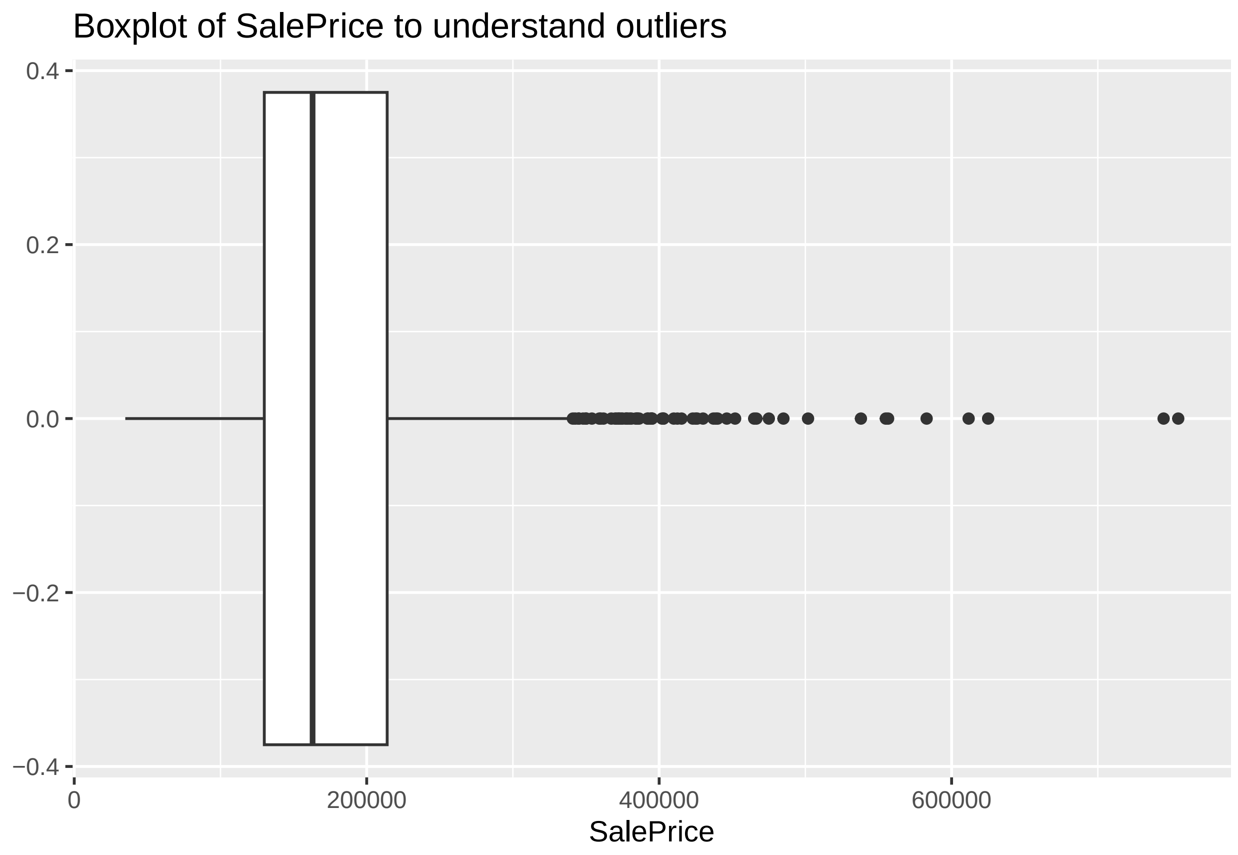
Based on this distribution, it appears most of the homes have a final sale price of just under $200,000 in the training data set. It may be beneficial to remove some of the extreme outliers if possible.

```
options(scipen = 999)#to get rid of scientific notation in graphs
ggplot(train.house, aes(x = SalePrice))+
  geom_histogram()+
  labs(title = "Distribution of SalePrice in data set")
```

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

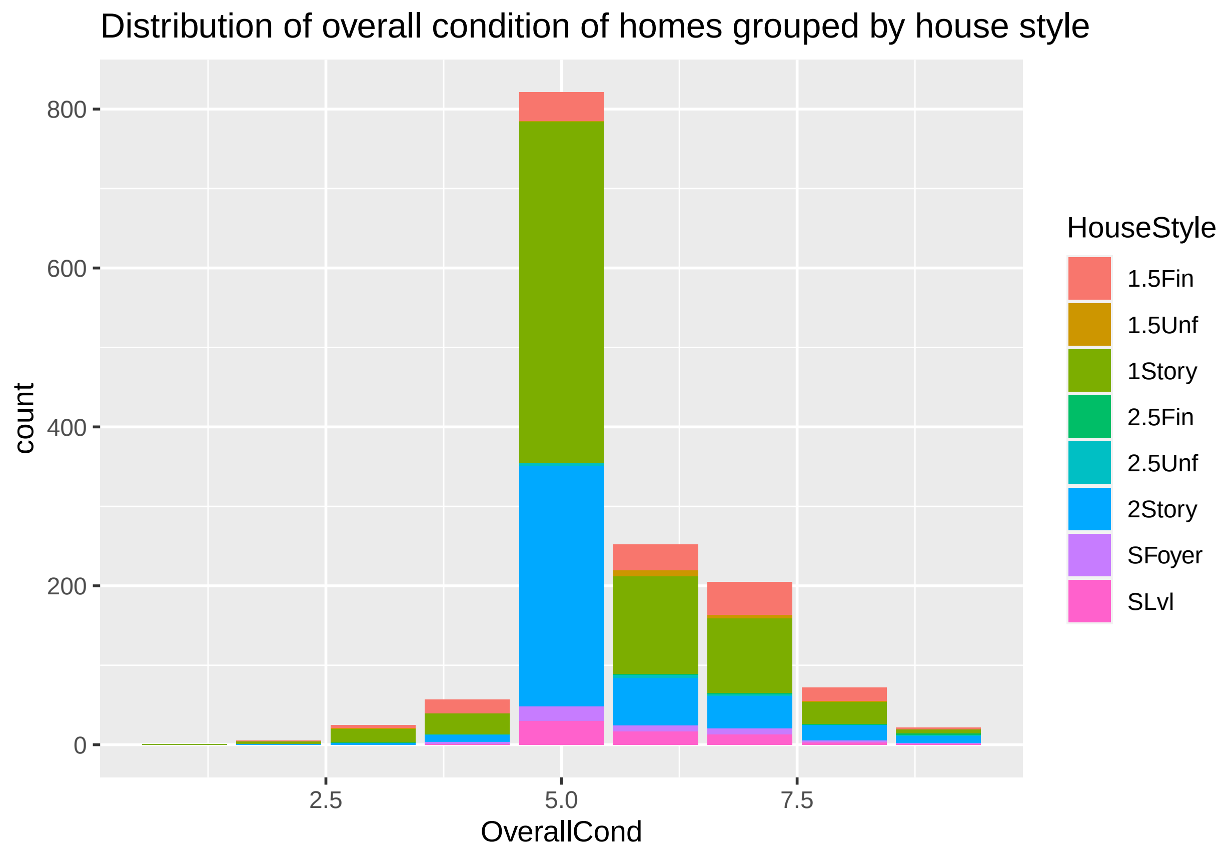


```
ggplot(train.house, aes(x = SalePrice))+
  geom_boxplot()+
  labs(title="Boxplot of SalePrice to understand outliers")
```



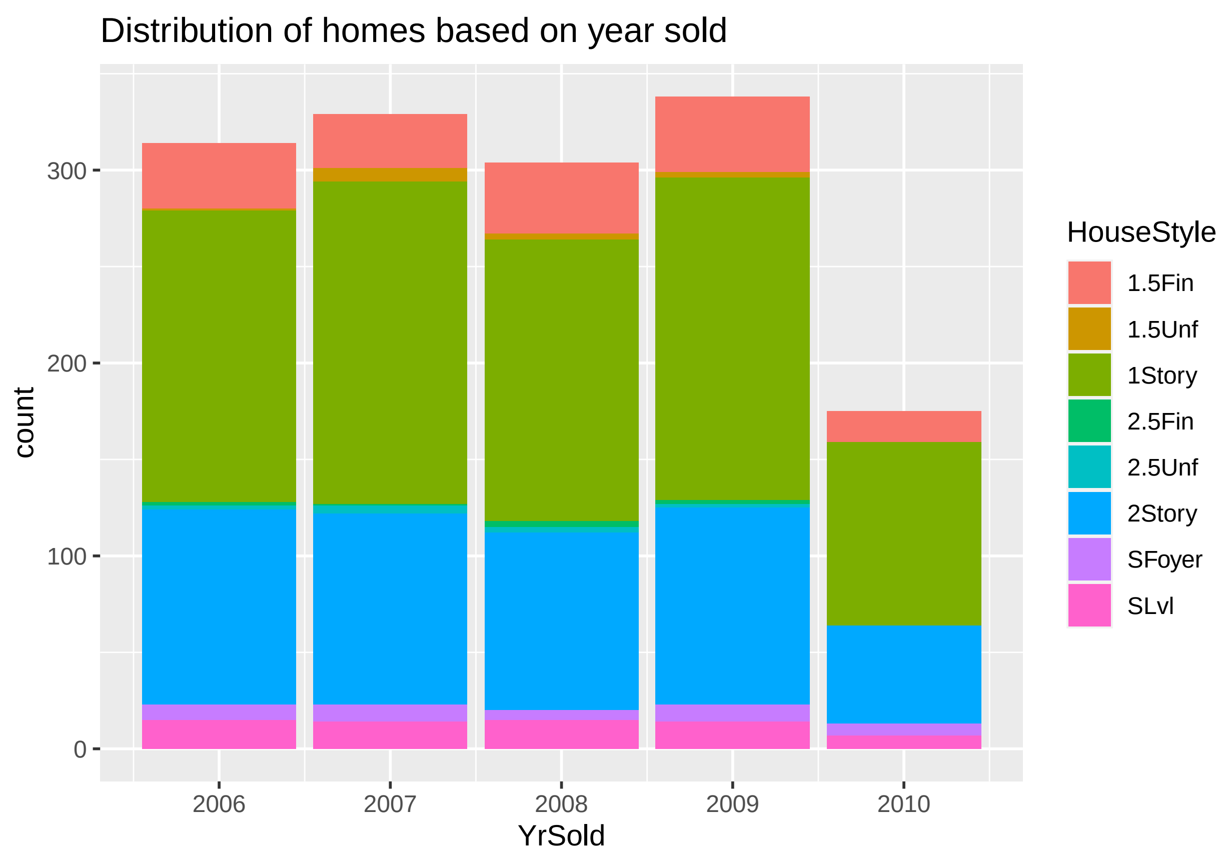
With this graph, I am trying to see if a particular style of house tends to be of an overall condition, but I am not seeing much of a relationship as there is a an even distribution of house stypes among the conditions.

```
options(scipen = 999)#to get rid of scientific notation in graphs
ggplot(train.house, aes(x = OverallCond, fill = HouseStyle))+
  geom_bar()+
  labs(title = "Distribution of overall condition of homes grouped by house style")
```



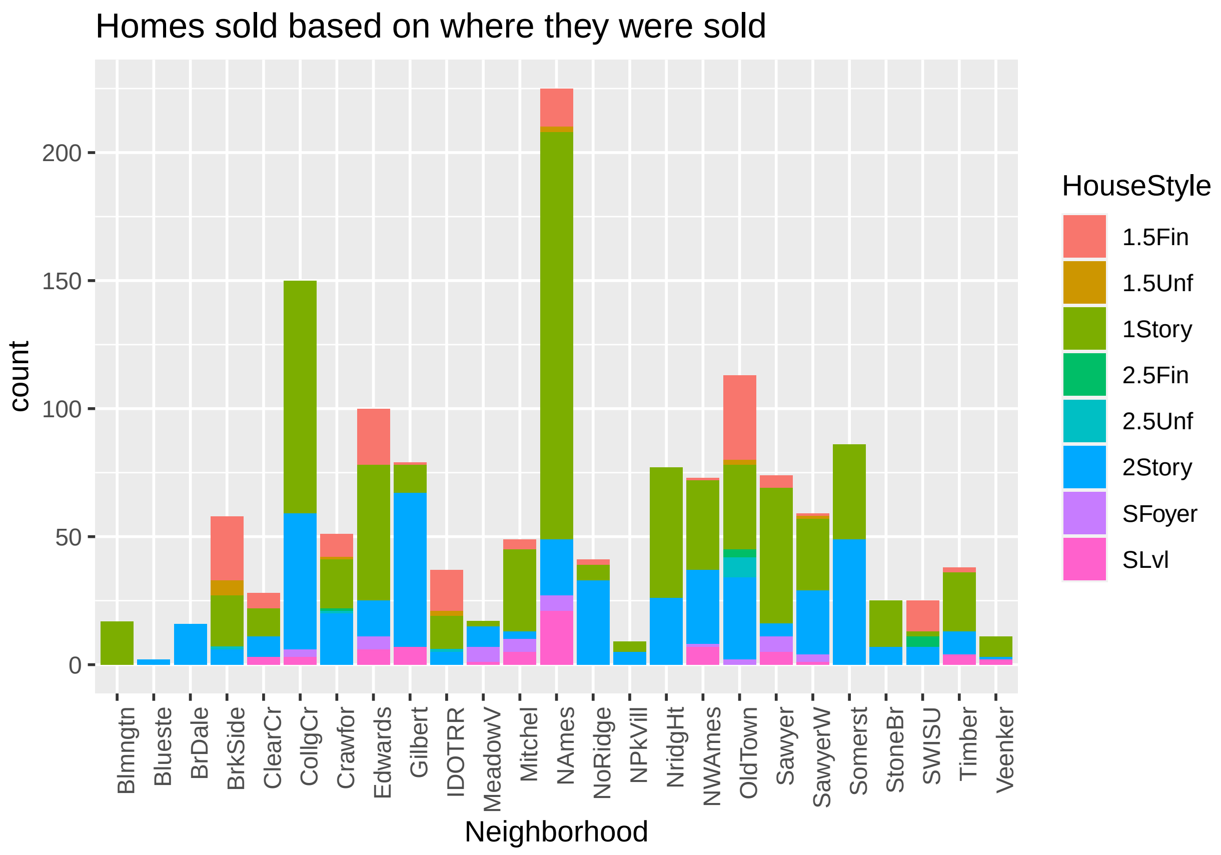
It looks like there is a slight dip in sales in 2008, but I am surprised the very next year sales returned to normal. The drop in 2010 is probably caused by the data collection ending. The house style is relatively evenly distributed across the years.

```r
train.house %>%
  ggplot(aes(x = YrSold, fill = HouseStyle))+
  geom_bar()+
  labs(title = "Distribution of homes based on year sold")
```



Most of the homes are located in the NAmes neighborhood. The second most popular location is the CollgCr area. Most of the homes in both of these neighborhoods are going to be 1 story.

```
train.house %>%
  ggplot(aes(x = Neighborhood, fill = HouseStyle))+
  geom_bar()+
  theme(axis.text.x = element_text(angle = 90, hjust =1))+
  labs(title = "Homes sold based on where they were sold")
```



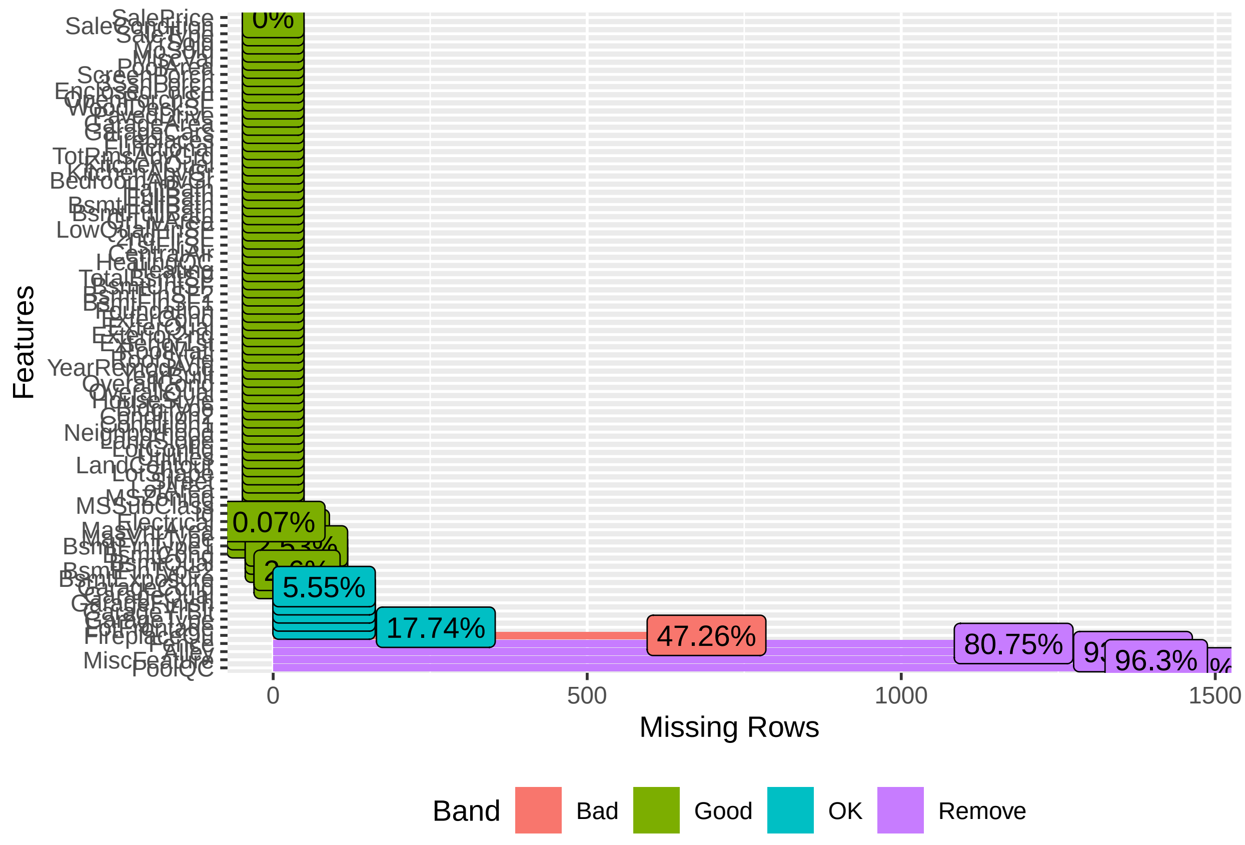
Evaluation Metric

From looking at online literature and by observing previous classroom work, I concluded I will use root mean squared log error to calculate and compare different Random Forest, k-NN, Ridge Regression models, and Lasso. RMSE is usually considered the standard for regression type models, but RMSLE is up-and-coming and I wanted to give it a try. RMSLE is an especially useful metric for this project because it deals with outliers very well. RMSE is drastically impacted by outliers, but RMSLE has a robustness to them and is not impacted much. RMSLE also penalizes underestimation of the actual value more severely than it does for overestimating of values.

PreProcessing

With functions from DataExplorer, I can observe PoolQC has 99.5% NA values, MiscFeature has 96.3%, Alley has 93%, and FirePlaceQu has 80% missing values. I may elect to get rid of those all together because I have more than enough explanatory variables to run with my models. LotFrontage is missing about 50% of the values in its columns. I will create dummy variables for all the columns where NA means that a feature is not present at the property.

```
plot_missing(train.house)
```



Here is another look at the NA values in the data set.

```
colSums(is.na(train.house))
```

```
##            Id    MSSubClass      MSZoning   LotFrontage       LotArea
##             0             0             0           259             0
##        Street         Alley      LotShape   LandContour     Utilities
##             0          1369             0             0             0
##     LotConfig     LandSlope  Neighborhood    Condition1    Condition2
##             0             0             0             0             0
##      BldgType    HouseStyle   OverallQual   OverallCond     YearBuilt
##             0             0             0             0             0
##  YearRemodAdd     RoofStyle      RoofMatl   Exterior1st   Exterior2nd
##             0             0             0             0             0
##    MasVnrType    MasVnrArea     ExterQual     ExterCond    Foundation
##             8             8             0             0             0
##      BsmtQual      BsmtCond  BsmtExposure  BsmtFinType1    BsmtFinSF1
##            37            37            38            37             0
##  BsmtFinType2    BsmtFinSF2     BsmtUnfSF   TotalBsmtSF       Heating
##            38             0             0             0             0
##     HeatingQC    CentralAir    Electrical      1stFlrSF      2ndFlrSF
##             0             0             1             0             0
##  LowQualFinSF     GrLivArea  BsmtFullBath  BsmtHalfBath      FullBath
##             0             0             0             0             0
##      HalfBath  BedroomAbvGr  KitchenAbvGr   KitchenQual  TotRmsAbvGrd
##             0             0             0             0             0
##    Functional    Fireplaces   FireplaceQu    GarageType   GarageYrBlt
##             0             0           690            81            81
##  GarageFinish    GarageCars    GarageArea    GarageQual    GarageCond
##            81             0             0            81            81
##    PavedDrive    WoodDeckSF   OpenPorchSF EnclosedPorch     3SsnPorch
##             0             0             0             0             0
##   ScreenPorch      PoolArea        PoolQC         Fence   MiscFeature
##             0             0          1453          1179          1406
##       MiscVal        MoSold        YrSold      SaleType SaleCondition
##             0             0             0             0             0
##     SalePrice
##             0
```

For MiscFeature, I plan to just create a new column that is a binary for indicating if a misc feature is present or not and the same goes for Alley. The NA values in PoolQC indicate "No Pool", so I will make NA into "No Pool". FirePlaceQu stands for Fire Place Quality with NA indicating no fire place. Again, the same set of procedures mentioned will have to be implemented here.

No matter what I do, I cannot get rid of the PoolQC NA values so I will be making binary outcomes and then getting rid of the other columns. The following variables below I turned into dummy variables to deal with NAs and then the original variables are deleted.

```r
train.house <- train.house %>%
  mutate(pool_avail = if_else(is.na(PoolQC), 0, 1)) %>%
  mutate(misc_feature_avail = if_else(is.na(MiscFeature), 0,1)) %>%
  mutate(fireplacequ = if_else(is.na(FireplaceQu), 0,1)) %>%
  mutate(fence_avail = if_else(is.na(Fence), 0, 1))

train.house$pool_avail <- as.factor(train.house$pool_avail)
train.house$misc_feature_avail <- as.factor(train.house$misc_feature_avail)
train.house$fireplacequ <- as.factor(train.house$fireplacequ)
```

```r
#Remove the Alley column
train.house$Alley = NULL
#Remove the PoolQC column
train.house$PoolQC = NULL
#Remove the MiscFeature column
train.house$MiscFeature = NULL
#Remove FireplaceQU
train.house$FireplaceQu = NULL
#remove Fense
train.house$Fence = NULL

#Function to see what the percentage of NA values exist in the data frame.
sum(is.na(train.house))/prod(dim(train.house))
```

```
## [1] 0.007431507
```

```r
colSums(is.na(train.house))
```

```
##                  Id          MSSubClass            MSZoning         LotFrontage
##                   0                   0                   0                 259
##             LotArea              Street            LotShape         LandContour
##                   0                   0                   0                   0
##           Utilities           LotConfig           LandSlope        Neighborhood
##                   0                   0                   0                   0
##          Condition1          Condition2            BldgType          HouseStyle
##                   0                   0                   0                   0
##        OverallQual        OverallCond          YearBuilt       YearRemodAdd
##                  0                  0                  0                  0
##          RoofStyle           RoofMatl        Exterior1st        Exterior2nd
##                  0                  0                  0                  0
##         MasVnrType         MasVnrArea          ExterQual          ExterCond
##                  8                  8                  0                  0
##         Foundation           BsmtQual           BsmtCond       BsmtExposure
##                  0                 37                 37                 38
##       BsmtFinType1         BsmtFinSF1       BsmtFinType2         BsmtFinSF2
##                 37                  0                 38                  0
##          BsmtUnfSF        TotalBsmtSF            Heating          HeatingQC
##                  0                  0                  0                  0
##         CentralAir         Electrical           1stFlrSF           2ndFlrSF
##                  0                  1                  0                  0
##       LowQualFinSF          GrLivArea       BsmtFullBath       BsmtHalfBath
##                  0                  0                  0                  0
##           FullBath           HalfBath       BedroomAbvGr       KitchenAbvGr
##                  0                  0                  0                  0
##        KitchenQual       TotRmsAbvGrd         Functional         Fireplaces
##                  0                  0                  0                  0
##         GarageType        GarageYrBlt       GarageFinish         GarageCars
##                 81                 81                 81                  0
##         GarageArea         GarageQual         GarageCond         PavedDrive
##                  0                 81                 81                  0
##         WoodDeckSF        OpenPorchSF      EnclosedPorch          3SsnPorch
##                  0                  0                  0                  0
##        ScreenPorch           PoolArea            MiscVal             MoSold
##                  0                  0                  0                  0
##             YrSold           SaleType      SaleCondition          SalePrice
##                  0                  0                  0                  0
##         pool_avail misc_feature_avail        fireplacequ        fence_avail
##                  0                  0                  0                  0
```

Since there are still NA values in a few of the columns, I will go through them and switch NA to "No…" since an NA value in one of these columns would signify the absence of a feature. NA values will make future model predictions difficult so I will get rid of them before as part of preprocessing.

```
train.house$LotFrontage[is.na(train.house$LotFrontage)] <- 0

train.house$BsmtQual <- ifelse(is.na(train.house$BsmtQual), "No basement", train.house$BsmtQual)
train.house$BsmtCond <- ifelse(is.na(train.house$BsmtCond), "No basement", train.house$BsmtCond)
train.house$BsmtExposure <- ifelse(is.na(train.house$BsmtExposure ), "No basement", train.house$BsmtExpo
train.house$BsmtFinType1 <- ifelse(is.na(train.house$BsmtFinType1), "No basement", train.house$BsmtFinTy
train.house$BsmtFinType2 <- ifelse(is.na(train.house$BsmtFinType2), "No basement", train.house$BsmtFinTy

train.house$GarageType <- ifelse(is.na(train.house$GarageType), "No garage", train.house$GarageType)
train.house$GarageYrBlt <- ifelse(is.na(train.house$GarageYrBlt), "No garage", train.house$GarageYrBlt)
train.house$GarageFinish <- ifelse(is.na(train.house$GarageFinish), "No garage", train.house$GarageFinis
train.house$GarageQual <- ifelse(is.na(train.house$GarageQual), "No garage", train.house$GarageQual)
train.house$GarageCond <- ifelse(is.na(train.house$GarageCond), "No garage", train.house$GarageCond)
```

Since most of the NA values are gone at this point, I am comfortable just completely omitting any additional NA values since it no longer makes up a significant portion of the data.

```
train.house <- na.omit(train.house)

dim(train.house)
```

```
## [1] 1451   80
```

As part of a tidy data practice, variables that begin with numbers will always prove to cause issues with R so I am just doing a column rename below.

```
train.house <- train.house %>%
  rename("first_floor_SF" = "1stFlrSF",
         "second_floor_SF" = "2ndFlrSF",
         "three_ssn_porch" = "3SsnPorch")
```

I also need to check if categorical variables are labeled correctly as a factor.

```
str(train.house)
```

```
## tibble [1,451 x 80] (S3: tbl_df/tbl/data.frame)
##  $ Id                   : num [1:1451] 1 2 3 4 5 6 7 8 9 10 ...
##  $ MSSubClass           : num [1:1451] 60 20 60 70 60 50 20 60 50 190 ...
##  $ MSZoning             : chr [1:1451] "RL" "RL" "RL" "RL" ...
##  $ LotFrontage          : num [1:1451] 65 80 68 60 84 85 75 0 51 50 ...
##  $ LotArea              : num [1:1451] 8450 9600 11250 9550 14260 ...
##  $ Street               : chr [1:1451] "Pave" "Pave" "Pave" "Pave" ...
##  $ LotShape             : chr [1:1451] "Reg" "Reg" "IR1" "IR1" ...
##  $ LandContour          : chr [1:1451] "Lvl" "Lvl" "Lvl" "Lvl" ...
##  $ Utilities            : chr [1:1451] "AllPub" "AllPub" "AllPub" "AllPub" ...
##  $ LotConfig            : chr [1:1451] "Inside" "FR2" "Inside" "Corner" ...
##  $ LandSlope            : chr [1:1451] "Gtl" "Gtl" "Gtl" "Gtl" ...
##  $ Neighborhood         : chr [1:1451] "CollgCr" "Veenker" "CollgCr" "Crawfor" ...
##  $ Condition1           : chr [1:1451] "Norm" "Feedr" "Norm" "Norm" ...
##  $ Condition2           : chr [1:1451] "Norm" "Norm" "Norm" "Norm" ...
##  $ BldgType             : chr [1:1451] "1Fam" "1Fam" "1Fam" "1Fam" ...
##  $ HouseStyle           : chr [1:1451] "2Story" "1Story" "2Story" "2Story" ...
##  $ OverallQual          : num [1:1451] 7 6 7 7 8 5 8 7 7 5 ...
##  $ OverallCond          : num [1:1451] 5 8 5 5 5 5 5 6 5 6 ...
##  $ YearBuilt            : num [1:1451] 2003 1976 2001 1915 2000 ...
##  $ YearRemodAdd         : num [1:1451] 2003 1976 2002 1970 2000 ...
##  $ RoofStyle            : chr [1:1451] "Gable" "Gable" "Gable" "Gable" ...
##  $ RoofMatl             : chr [1:1451] "CompShg" "CompShg" "CompShg" "CompShg" ...
##  $ Exterior1st          : chr [1:1451] "VinylSd" "MetalSd" "VinylSd" "Wd Sdng" ...
##  $ Exterior2nd          : chr [1:1451] "VinylSd" "MetalSd" "VinylSd" "Wd Shng" ...
##  $ MasVnrType           : chr [1:1451] "BrkFace" "None" "BrkFace" "None" ...
##  $ MasVnrArea           : num [1:1451] 196 0 162 0 350 0 186 240 0 0 ...
##  $ ExterQual            : chr [1:1451] "Gd" "TA" "Gd" "TA" ...
##  $ ExterCond            : chr [1:1451] "TA" "TA" "TA" "TA" ...
##  $ Foundation           : chr [1:1451] "PConc" "CBlock" "PConc" "BrkTil" ...
##  $ BsmtQual             : chr [1:1451] "Gd" "Gd" "Gd" "TA" ...
##  $ BsmtCond             : chr [1:1451] "TA" "TA" "TA" "Gd" ...
##  $ BsmtExposure         : chr [1:1451] "No" "Gd" "Mn" "No" ...
##  $ BsmtFinType1         : chr [1:1451] "GLQ" "ALQ" "GLQ" "ALQ" ...
##  $ BsmtFinSF1           : num [1:1451] 706 978 486 216 655 ...
##  $ BsmtFinType2         : chr [1:1451] "Unf" "Unf" "Unf" "Unf" ...
##  $ BsmtFinSF2           : num [1:1451] 0 0 0 0 0 0 0 32 0 0 ...
##  $ BsmtUnfSF            : num [1:1451] 150 284 434 540 490 64 317 216 952 140 ...
##  $ TotalBsmtSF       : num [1:1451] 856 1262 920 756 1145 ...
##  $ Heating           : chr [1:1451] "GasA" "GasA" "GasA" "GasA" ...
##  $ HeatingQC         : chr [1:1451] "Ex" "Ex" "Ex" "Gd" ...
##  $ CentralAir        : chr [1:1451] "Y" "Y" "Y" "Y" ...
##  $ Electrical        : chr [1:1451] "SBrkr" "SBrkr" "SBrkr" "SBrkr" ...
##  $ first_floor_SF    : num [1:1451] 856 1262 920 961 1145 ...
##  $ second_floor_SF   : num [1:1451] 854 0 866 756 1053 ...
##  $ LowQualFinSF      : num [1:1451] 0 0 0 0 0 0 0 0 0 0 ...
##  $ GrLivArea         : num [1:1451] 1710 1262 1786 1717 2198 ...
##  $ BsmtFullBath      : num [1:1451] 1 0 1 1 1 1 1 1 0 1 ...
##  $ BsmtHalfBath      : num [1:1451] 0 1 0 0 0 0 0 0 0 0 ...
##  $ FullBath          : num [1:1451] 2 2 2 1 2 1 2 2 2 1 ...
##  $ HalfBath          : num [1:1451] 1 0 1 0 1 1 0 1 0 0 ...
##  $ BedroomAbvGr      : num [1:1451] 3 3 3 3 4 1 3 3 2 2 ...
##  $ KitchenAbvGr      : num [1:1451] 1 1 1 1 1 1 1 1 2 2 ...
##  $ KitchenQual       : chr [1:1451] "Gd" "TA" "Gd" "Gd" ...
##  $ TotRmsAbvGrd      : num [1:1451] 8 6 6 7 9 5 7 7 8 5 ...
##  $ Functional        : chr [1:1451] "Typ" "Typ" "Typ" "Typ" ...
##  $ Fireplaces        : num [1:1451] 0 1 1 1 1 0 1 2 2 2 ...
##  $ GarageType        : chr [1:1451] "Attchd" "Attchd" "Attchd" "Detchd" ...
##  $ GarageYrBlt       : chr [1:1451] "2003" "1976" "2001" "1998" ...
##  $ GarageFinish      : chr [1:1451] "RFn" "RFn" "RFn" "Unf" ...
##  $ GarageCars        : num [1:1451] 2 2 2 3 3 2 2 2 2 1 ...
##  $ GarageArea        : num [1:1451] 548 460 608 642 836 480 636 484 468 205 ...
##  $ GarageQual        : chr [1:1451] "TA" "TA" "TA" "TA" ...
##  $ GarageCond        : chr [1:1451] "TA" "TA" "TA" "TA" ...
##  $ PavedDrive        : chr [1:1451] "Y" "Y" "Y" "Y" ...
##  $ WoodDeckSF        : num [1:1451] 0 298 0 0 192 40 255 235 90 0 ...
##  $ OpenPorchSF       : num [1:1451] 61 0 42 35 84 30 57 204 0 4 ...
##  $ EnclosedPorch     : num [1:1451] 0 0 0 272 0 0 0 228 205 0 ...
##  $ three_ssn_porch   : num [1:1451] 0 0 0 0 0 320 0 0 0 0 ...
##  $ ScreenPorch       : num [1:1451] 0 0 0 0 0 0 0 0 0 0 ...
##  $ PoolArea          : num [1:1451] 0 0 0 0 0 0 0 0 0 0 ...
##  $ MiscVal           : num [1:1451] 0 0 0 0 0 700 0 350 0 0 ...
##  $ MoSold            : num [1:1451] 2 5 9 2 12 10 8 11 4 1 ...
##  $ YrSold            : num [1:1451] 2008 2007 2008 2006 2008 ...
##  $ SaleType          : chr [1:1451] "WD" "WD" "WD" "WD" ...
##  $ SaleCondition     : chr [1:1451] "Normal" "Normal" "Normal" "Abnorml" ...
##  $ SalePrice         : num [1:1451] 208500 181500 223500 140000 250000 ...
##  $ pool_avail        : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ misc_feature_avail: Factor w/ 2 levels "0","1": 1 1 1 1 1 2 1 2 1 1 ...
##  $ fireplacequ       : Factor w/ 2 levels "0","1": 1 2 2 2 2 1 2 2 2 2 ...
##  $ fence_avail       : num [1:1451] 0 0 0 0 0 1 0 0 0 0 ...
##  - attr(*, "na.action")= 'omit' Named int [1:9] 235 530 651 937 974 978 1244 1279 1380
##   ..- attr(*, "names")= chr [1:9] "235" "530" "651" "937" ...
```

Most of the categorical variables are incorrectly labeled as characters. It is common practice for a categorical variable to be a factor so, manually, I make those adjustments below. When I attempted to do complete this task with a function, I never got the intended results, hence why I end up doing it by hand.

```
train.house$MSZoning <- as.factor(train.house$MSZoning)
train.house$Street <- as.factor(train.house$Street)
train.house$LotShape <- as.factor(train.house$LotShape)
train.house$LandContour  <- as.factor(train.house$LandContour)
train.house$LotConfig <- as.factor(train.house$LotConfig)
train.house$LandSlope <- as.factor(train.house$LandSlope)
train.house$Condition1 <- as.factor(train.house$Condition1)
train.house$Condition2 <- as.factor(train.house$Condition2)
train.house$BldgType <- as.factor(train.house$BldgType)
train.house$HouseStyle <- as.factor(train.house$HouseStyle)

train.house$RoofStyle <- as.factor(train.house$RoofStyle)
train.house$RoofMatl <- as.factor(train.house$RoofMatl)
train.house$MasVnrType <- as.factor(train.house$MasVnrType)
train.house$ExterQual <- as.factor(train.house$ExterQual)
train.house$ExterCond <- as.factor(train.house$ExterCond)
train.house$Foundation <- as.factor(train.house$Foundation)
train.house$BsmtCond <- as.factor(train.house$BsmtCond)
train.house$BsmtQual <- as.factor(train.house$BsmtQual)
train.house$BsmtExposure <- as.factor(train.house$BsmtExposure)
train.house$BsmtFinType1 <- as.factor(train.house$BsmtFinType1)
train.house$BsmtFinType2 <- as.factor(train.house$BsmtFinType2)
train.house$Heating <- as.factor(train.house$Heating)
train.house$HeatingQC <- as.factor(train.house$HeatingQC)
train.house$Electrical <- as.factor(train.house$Electrical)
train.house$CentralAir <- as.factor(train.house$CentralAir)

train.house$KitchenQual <- as.factor(train.house$KitchenQual)
train.house$Functional <- as.factor(train.house$Functional)
train.house$GarageType <- as.factor(train.house$GarageType)
train.house$GarageFinish <- as.factor(train.house$GarageFinish)
train.house$GarageQual <- as.factor(train.house$GarageQual)
train.house$GarageCond <- as.factor(train.house$GarageCond)
train.house$PavedDrive <- as.factor(train.house$PavedDrive)
train.house$SaleType <- as.factor(train.house$SaleType)
train.house$SaleCondition <- as.factor(train.house$SaleCondition)
```

I got an error when I ran models the first time around with predictors that are categorical or factors with only one value. It looks like the Utilities column only contains 1 unique variable, which means this may be the source of my errors. I will have to get rid of it so that models can run smoothly.

```
sapply(lapply(train.house, unique), length)
```

```
##            Id    MSSubClass      MSZoning   LotFrontage
##          1451            15             5           111
##       LotArea        Street      LotShape   LandContour
##          1066             2             4             4
##     Utilities     LotConfig     LandSlope  Neighborhood
##             2             5             3            25
##    Condition1    Condition2      BldgType    HouseStyle
##             9             8             5             8
##   OverallQual   OverallCond     YearBuilt  YearRemodAdd
##            10             9           112            61
##     RoofStyle      RoofMatl   Exterior1st   Exterior2nd
##             6             8            15            16
##    MasVnrType    MasVnrArea     ExterQual     ExterCond
##             4           327             4             5
##    Foundation      BsmtQual      BsmtCond  BsmtExposure
##             6             5             5             5
##       BsmtFinType1         BsmtFinSF1       BsmtFinType2         BsmtFinSF2
##                  7                633                  7                144
##          BsmtUnfSF        TotalBsmtSF            Heating          HeatingQC
##                777                717                  6                  5
##         CentralAir         Electrical     first_floor_SF    second_floor_SF
##                  2                  5                748                414
##       LowQualFinSF          GrLivArea       BsmtFullBath       BsmtHalfBath
##                 24                858                  4                  3
##           FullBath           HalfBath       BedroomAbvGr       KitchenAbvGr
##                  4                  3                  8                  4
##        KitchenQual       TotRmsAbvGrd         Functional         Fireplaces
##                  4                 12                  7                  4
##         GarageType        GarageYrBlt       GarageFinish         GarageCars
##                  7                 98                  4                  5
##         GarageArea         GarageQual         GarageCond         PavedDrive
##                438                  6                  6                  3
##         WoodDeckSF        OpenPorchSF      EnclosedPorch    three_ssn_porch
##                274                201                119                 20
##        ScreenPorch           PoolArea            MiscVal             MoSold
##                 76                  8                 21                 12
##             YrSold           SaleType      SaleCondition          SalePrice
##                  5                  9                  6                657
##         pool_avail misc_feature_avail        fireplacequ         fence_avail
##                  2                  2                  2                  2
```

```
train.house$Utilities = NULL
```

That was the final step and the preprocessing should lead smoothly into model fitting.

Fit Models

Validation and training set of train.house

For the project, I am using the training set, but splitting it into a training and testing set.

```
set.seed(12356)

split <- initial_split(train.house, prop = 0.75)

training.house <- training(split)
testing.house <- testing(split)

dim(training.house)
```

```
## [1] 1088   79
```

```
dim(testing.house)
```

```
## [1] 363  79
```

Random Forest

This initial model is run on default value parameters for regression problems. The next model is run by increasing a few of the numbers to see if another Random Forest model would fit the data better.

```
set.seed(1233)
```

```r
rf1 <- randomForest(SalePrice ~.,
                    ntree = 300,
                    mtry = 26,
                    nodesize = 5,
                    data = training.house)

rf1
```

```
##
## Call:
##  randomForest(formula = SalePrice ~ ., data = training.house,      ntree = 300, mtry = 26, nodesize
##                Type of random forest: regression
##                      Number of trees: 300
## No. of variables tried at each split: 26
##
##           Mean of squared residuals: 890059284
##                     % Var explained: 85.68
```

```r
set.seed(1238343)

rf2 <- randomForest(SalePrice ~.,
                    ntree = 400,
                    mtry = 40,
                    nodesize = 11,
                    data = training.house)

rf2
```

```
##
## Call:
##  randomForest(formula = SalePrice ~ ., data = training.house,      ntree = 400, mtry = 40, nodesize
##                Type of random forest: regression
##                      Number of trees: 400
## No. of variables tried at each split: 40
##
##           Mean of squared residuals: 893424434
##                     % Var explained: 85.63
```

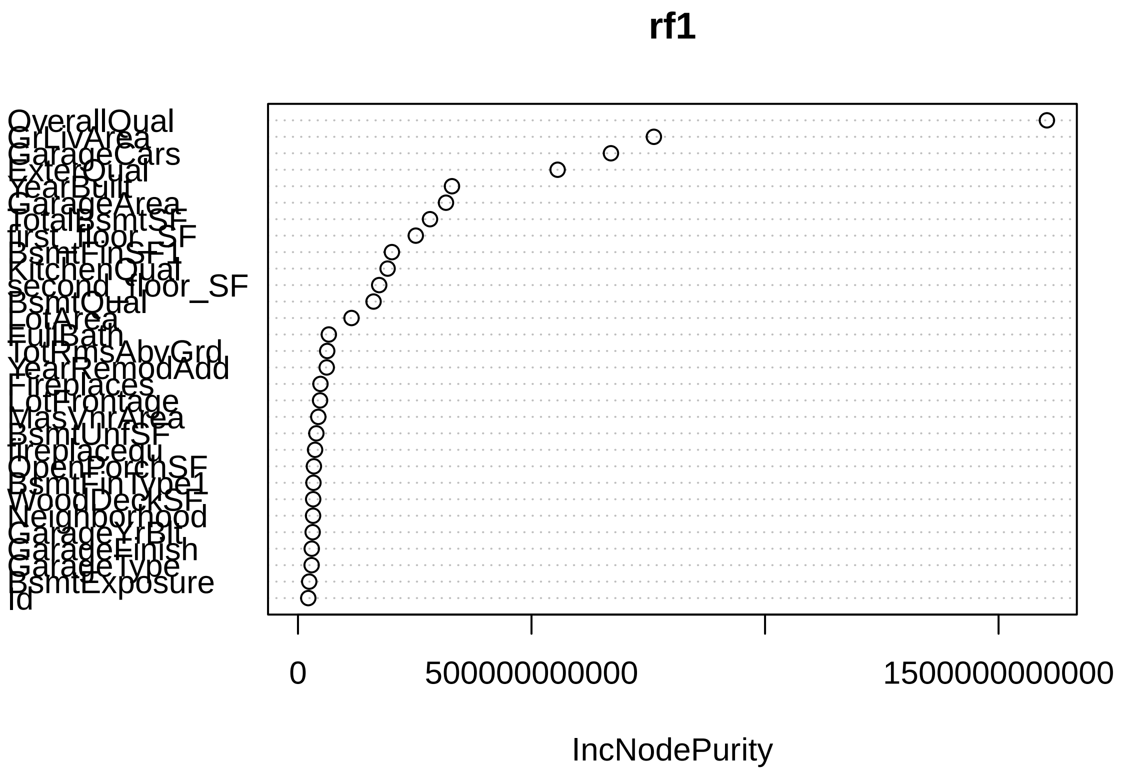
```r
set.seed(69348)

rf3 <- randomForest(SalePrice ~.,
                    ntree = 300,
                    mtry = 58,
                    nodesize = 19,
                    data = training.house)

rf3
```

```
##
## Call:
##  randomForest(formula = SalePrice ~ ., data = training.house,      ntree = 300, mtry = 58, nodesize
##                Type of random forest: regression
##                      Number of trees: 300
## No. of variables tried at each split: 58
##
##           Mean of squared residuals: 908003577
##                     % Var explained: 85.39
```

It looks like the first model has the best outcome so I will be continuing with rf1.

The chart shows the most important variables as measured by Random Forest. It appears overall quality of the home and if there is a garage are the most important.

```
varImpPlot(rf1)
```



Predict with Random Forest

```
rf.pred <- predict(rf1,newdata = testing.house)

random.forest.rmsle <- rmsle(testing.house$SalePrice, rf.pred)

random.forest.rmsle
```

```
## [1] 0.1573435
```

K-Nearest Neighbors Regression

The KNN output uses RMSE to determine that k = 9 is the optimal k for the model.

```
fit.knn
```

```
## k-Nearest Neighbors
##
## 1451 samples
##   78 predictor
##
```

```
## Pre-processing: centered (341), scaled (341)
## Resampling: Cross-Validated (10 fold, repeated 5 times)
## Summary of sample sizes: 1306, 1306, 1305, 1305, 1306, 1306, ...
## Resampling results across tuning parameters:
##
##   k  RMSE      Rsquared   MAE
##   5  48868.06  0.6342148  31984.70
##   7  47634.48  0.6582056  31365.57
##   9  47091.70  0.6730250  30813.22
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was k = 9.
```

Predict with KNN

```
knn.pred <- predict(fit.knn, testing.house)

knn.rmsle <- rmsle(knn.pred, testing.house$SalePrice)

knn.rmsle
```

```
## [1] 0.2283874
```

Ridge Regression

Creating the training and testing matrices for Ridge and Lasso

```
grid = 10^seq(10,-2, length = 100)

x.matrix.train <- data.matrix(training.house[1:79])

y.matrix.train <- data.matrix(training.house[,"SalePrice"])

x.matrix.test <- data.matrix(testing.house)

y.matrix.test <- data.matrix(testing.house[,"SalePrice"])

ridge.mod.cv <- cv.glmnet(x.matrix.train, y.matrix.train, alpha = 0, lambda = grid)

bestlam <- ridge.mod.cv$lambda.min
```

Predict with Ridge Regression

```
ridge.pred <- predict(ridge.mod.cv, s = bestlam, newx = x.matrix.test)

ridge.pred <- ifelse(ridge.pred < 0, 0, ridge.pred)

ridge.rmsle <- rmsle(ridge.pred, y.matrix.test)

ridge.rmsle
```

```
## [1] 0.0003046695
```

Lasso

```
lasso.mod.cv <- cv.glmnet(x.matrix.train, y.matrix.train, alpha = 1, lambda = grid)
```

```
bestlam <- lasso.mod.cv$lambda.min
```

Predict with Lasso

```
lasso.pred <- predict(lasso.mod.cv, s = bestlam, newx = x.matrix.test)

lasso.pred <- ifelse(lasso.pred < 0, 0, lasso.pred)

lasso.rmsle <- rmsle(lasso.pred, y.matrix.test)

lasso.rmsle
```

```
## [1] 0.00000007546139
```

Model Comparison

With root mean squared log error, the lower the value, the better the performance fit. RMSLE is a common model performance metric to use on regression problems like house price prediction, especially when dealing with outliers. Below I have listed the RMSLE metrics for each model (KNN, Ridge Regression, Lasso, and Random Forest). The models used in this project are mostly highly interpretative, but highly inflexible methods. Random Forest finds itself in the middle by being flexible, but also interpretable.

Striking a balance between bias and variance in machine learning models can be tricky, but it is necessary in building applicable and scaleable models. The goal is to have a model that is low in both variance and bias in so that it does not over or undrefit on test data. Regression models tend to have a lower variance, but a higher bias. With tree based decision models, they tend to work pretty well, but if there are too many trees, there is a tendency to overfit the data. This may explain why the Random Forest, an ensemble method model, performed best. Tree based decision models usually perform well in this balancing act, but not always.

The Lasso produces the lowest RMSLE score and so does ridge regression. That almost seems too low and I fear that something may have gone wrong in model performance. Random Forest and KNN seem more realistic. It looks like Random Forest performed the best with an RMSLE score of 0.15.

```
lasso.rmsle
```

```
## [1] 0.00000007546139
```

```
#0.00000007546139

ridge.rmsle
```

```
## [1] 0.0003046695
```

```
#0.0003046695

random.forest.rmsle
```

```
## [1] 0.1573435
```

```
#0.1573435

knn.rmsle
```

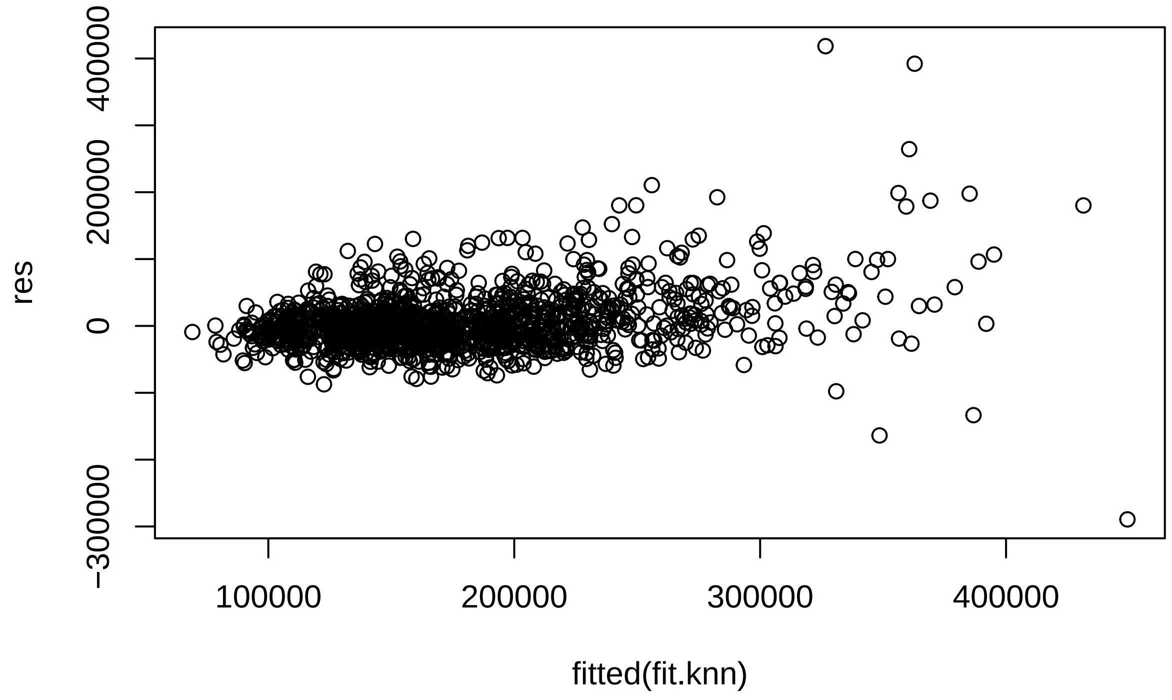
```
## [1] 0.2283874
```

```
#0.2283874
```

KNN Plot

Here are the residuals of the KNN model plotted, which leads toward heteroscedasicity.

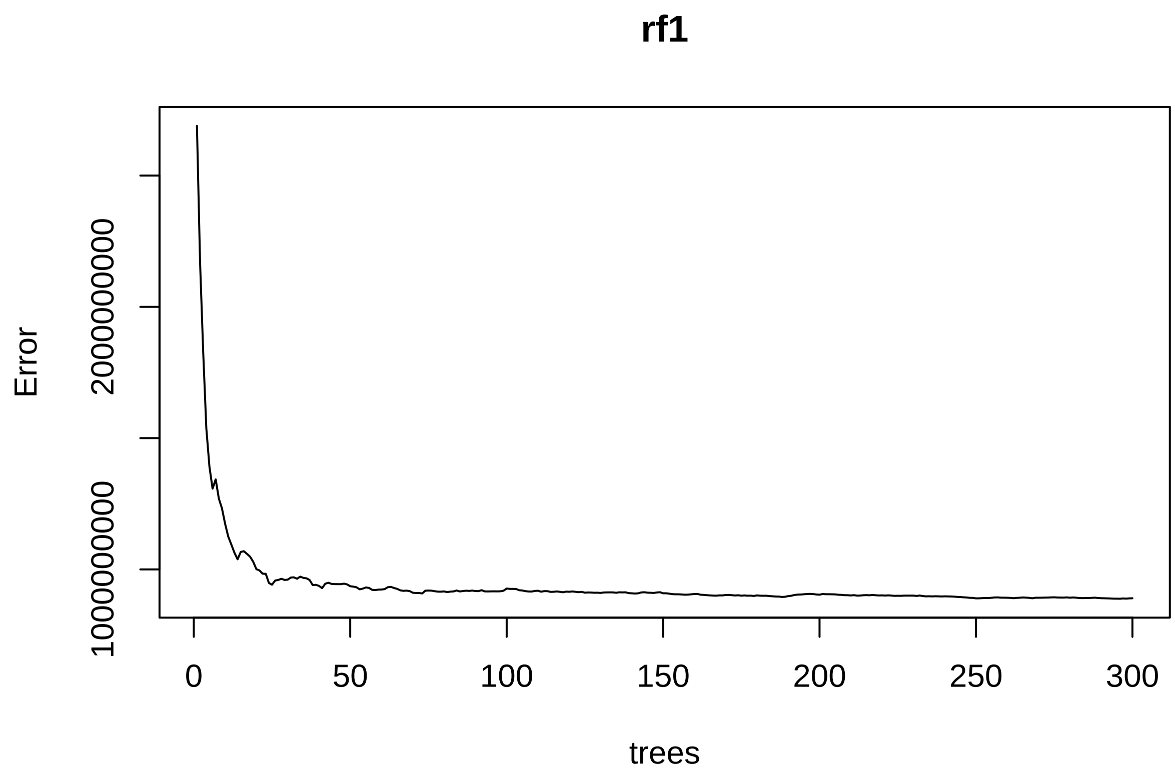
```
res <- resid(fit.knn)

plot(fitted(fit.knn), res)
```



Random Forest plot

The error rate decreases significantly at around 50 trees is sustained indefinitely.

```
plot(rf1)
```



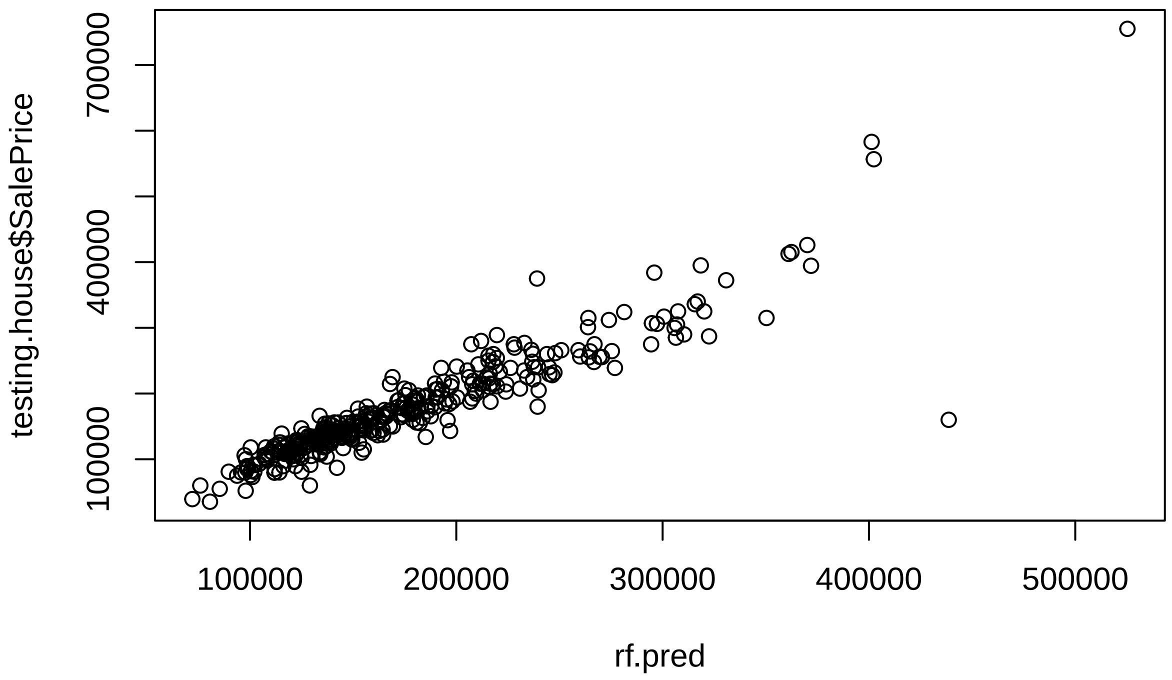
Predicted values plotted against actual values

These plots below represent performance of models by plotting the actual vs predicted values in the testing data set to observe how well they fit each other.

Random Forest

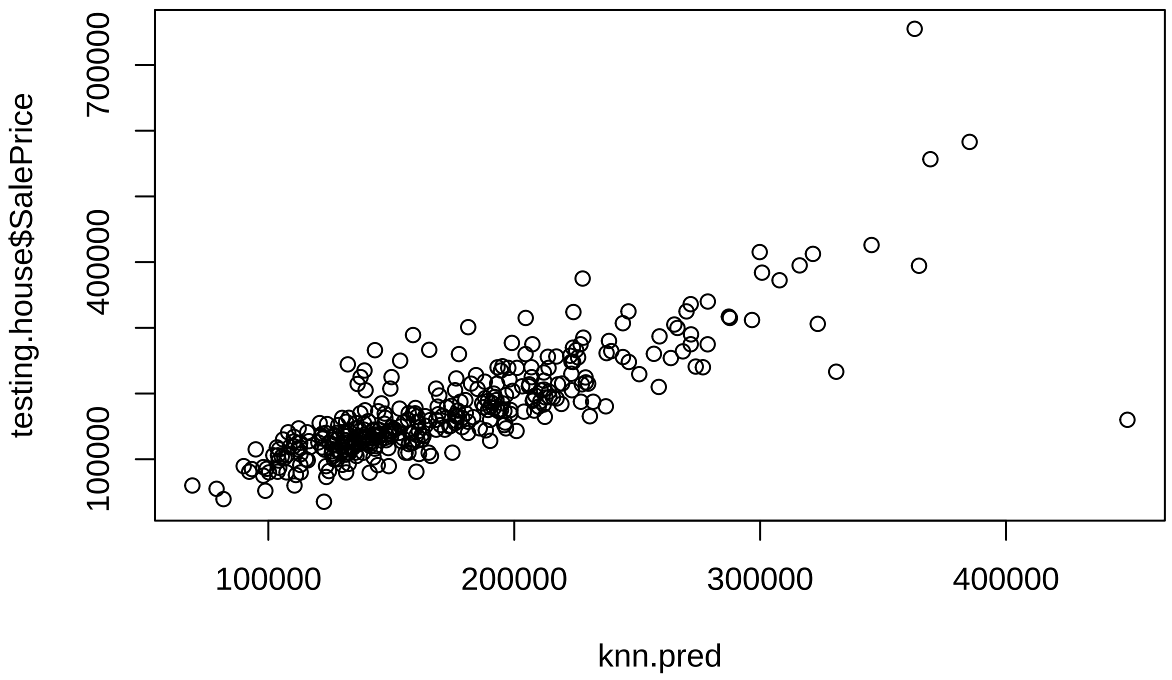
Generally, we want a linear line when comparing the predicted vs actualy values so this Random Forest plot and the next KNN plot look acceptable even though there are a few outliers. Overall, Random Forest is the best with the tightest cluster towards a normal linear line. As mentioned, I wonder if there is some unforseen error in my predictions or creating a matrix for Ridge and Lasso, because the predicted and actual values are right on top of each other.

```
plot(rf.pred, testing.house$SalePrice)
```



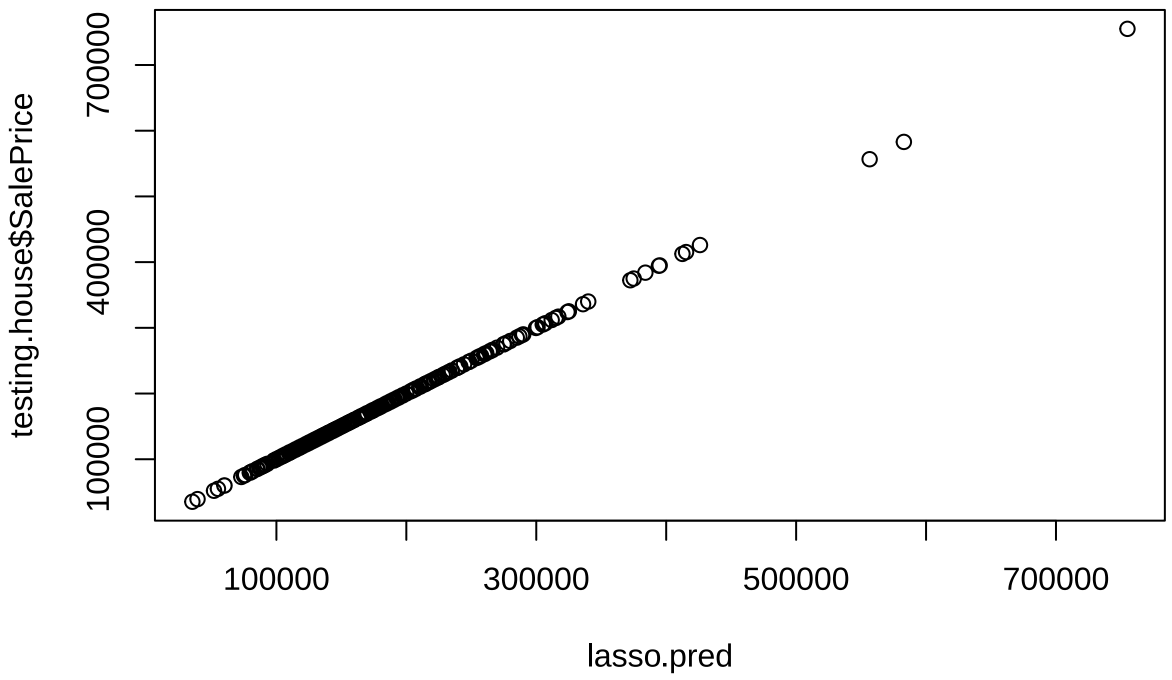
KNN

```
plot(knn.pred, testing.house$SalePrice)
```



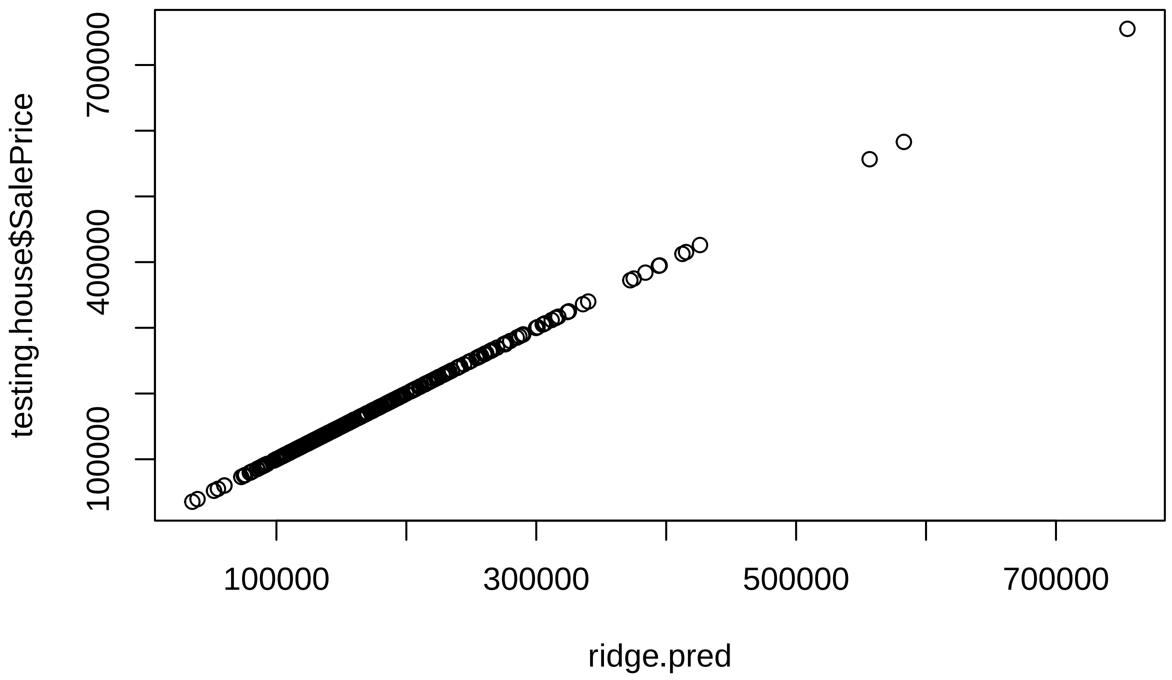
Lasso

```
plot(lasso.pred, testing.house$SalePrice)
```



Ridge regression

```
plot(ridge.pred, testing.house$SalePrice)
```



Conclusion: Ethical Implications

With any machine learning algorithm or model, there is cause for ethical concerns over outcomes of these models, especially in the case of housing price predictions. Like every model, the data used in the training process can lead to some undesirable outcomes. There could be a few ethical concerns about attempting to predict house prices with machine learning models.

1. If data sets to predict house price contain information on race or socioeconomic status, the model may end up further perpetuating discrimination against marginalized groups. It is important to make sure data used in these models is sourced appropriately and does not contain any variables that could damage or harm other individuals.
2. Homeowner privacy may also be of concern. The data set I used was curated and anonymous, but if an organization or individual chose to seek personal information of homeowners (credit score, race, income, etc.), concerns of data privacy may become an issue.
3. Transparency with results must be noted in any research or algorithm that involves a topic as complex as house price prediction. Unforeseen events such as natural disasters or financial crisis's cannot be taken into account when building data sets for prediction. Organizations and researchers must ensure users are aware of how data was collected, on what locations it was collected, and that most models can only predict prices on variables that are known.

The data set I am currently using does not have cause for concern. It was created/curated by Dean De Cock for data science educational purposes. No variables within the data set pertain to any issue of race or socioeconomic status. Variables are restricted to physical features of homes and their immediate location.