Kaggle Housing Prices Prediction

Ryan Moore

September 23, 2016

## Instructions from Kaggle

*"Ask a home buyer to describe their dream house, and they probably won't begin with the height of the basement ceiling or the proximity to an east-west railroad. But this playground competition's dataset proves that much more influences price negotiations than the number of bedrooms or a white-picket fence."*

*"With 79 explanatory variables describing (almost) every aspect of residential homes in Ames, Iowa, this competition challenges you to predict the final price of each home."*

### Aside from Author

This data set is extremely interesting due to fact that most people's biggest purchase in their lives is a house. Who wouldn't want to know what exactly what feaures matter and to what degree. Everyone assumes if they have been in the real estate industry that they can eye and price a house based on experience. Some experts may be able to, but due to time constraints and the size of the housing market scaling a prediction model is key. I chose this competition because it would be one of the best demonstrations of my skillset with a real life example. One of which that has a data science problem that is widely applicable to everyday businesses.

Data can be found here <https://www.kaggle.com/c/house-prices-advanced-regression-techniques/data>

My code can be found here <https://github.com/Alaroc?tab=repositories>

### Table of Contents

* EDA
* Data Transformation / NA Handling
* Prediction/Validation

#Loading necessary libraries  
library(ggplot2)  
library(data.table)  
library(gridExtra)  
library(dplyr)  
library(corrplot)  
library(randomForest)  
library(xgboost)  
library(leaps)  
library(class)  
library(moments)  
library(RColorBrewer)  
  
  
test<-fread("test.csv")  
train<-fread("train.csv")  
test[, ':='(SalePrice=NA, test\_flag=1)]  
train[, test\_flag:=0]  
  
all\_data<-rbind(train, test)

## Exploratory Data Analysis

### Summary / Structure of the data

(The output from this code is very large and I've decided to exclude this.)

str(all\_data)  
summary(all\_data)

We have mostly factor and numeric variables.

### Number of NA's by Variable

#creating a table that can be looped through to fix na values  
na.Table<-data.table(colnames(train),sapply(all\_data, function(y) sum(length(which(is.na(y))))))  
#vector with features to be handled.  
na.names<-na.Table[V2>0,]$V1

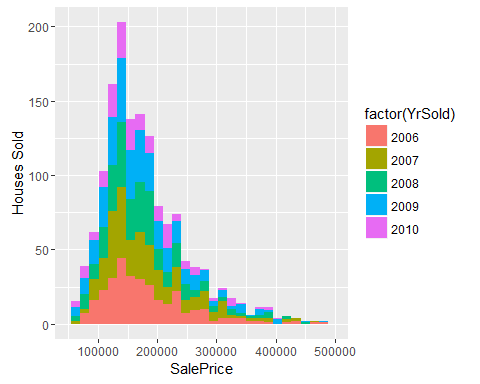
*Alley*, *PoolQC*, *Fence*, and *MiscFeatures* have a very large number of NA's.

#### What is the distribution of the house prices?

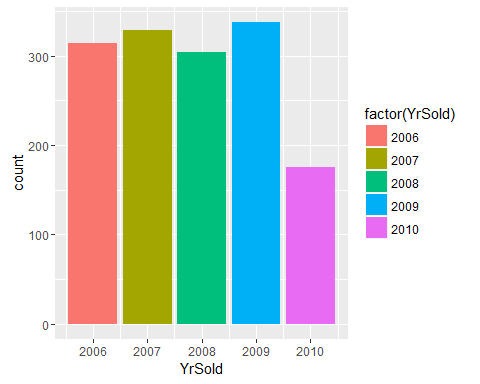
(Note that the colors are to be intrepreted as stacked and not overlaid)

#### How many houses were sold in each year?

ggplot(data=train, aes(x=SalePrice, fill=factor(YrSold)))+  
 geom\_histogram(position="stack")+  
 scale\_x\_continuous(limits=c(50000,500000),labels=function(n){format(n, scientific = FALSE)})+ ylab("Houses Sold")



ggplot(data=train, aes(x=YrSold, fill=factor(YrSold)))+  
 geom\_bar()



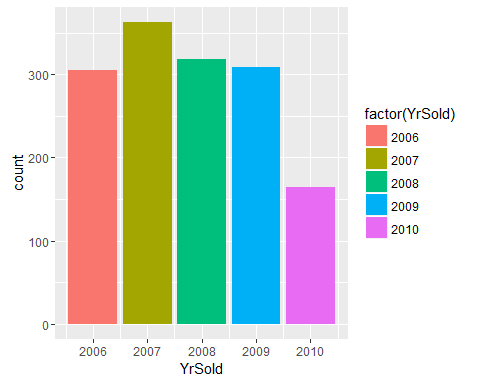
It looks like 2010 hasn't ended by the time of the data collection or the test data is the end of 2010. Let's test this.

summary(train[YrSold==2010,]$MoSold)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 1.000 3.000 5.000 4.326 5.000 7.000

Confirmed! The collection of the data stopped July 2010. Let's check if test follows the same pattern.

ggplot(data=test, aes(x=YrSold, fill=factor(YrSold)))+  
 geom\_bar()



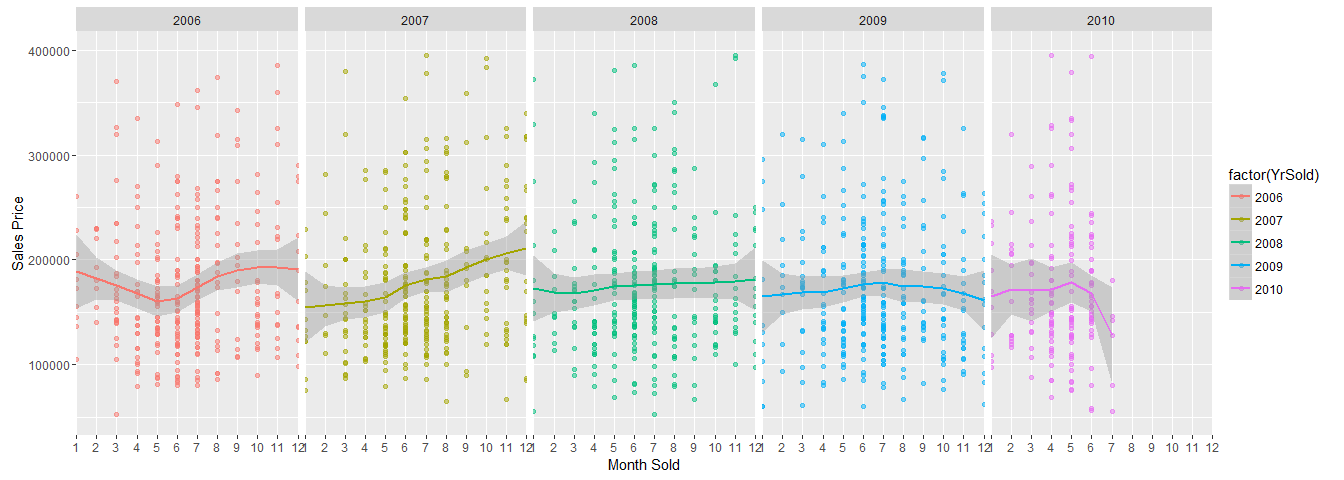
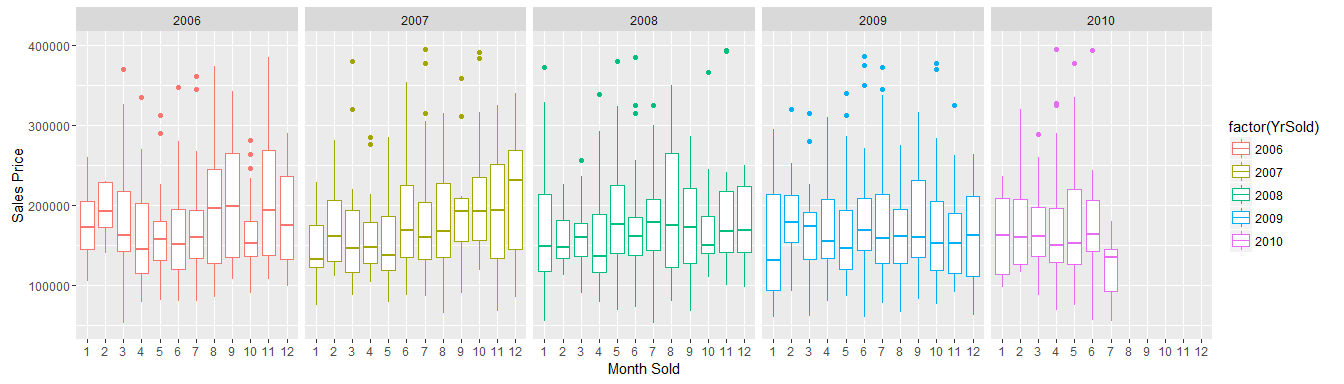
summary(test[YrSold==2010,]$MoSold)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 1.000 3.000 4.000 4.201 6.000 7.000

It does!

### Have prices of houses increased over time in Iowa?

ggplot(data=train, aes(x=MoSold, y=SalePrice, col=factor(YrSold)))+  
 geom\_point(alpha=.5)+  
 geom\_smooth()+  
 xlab("Month Sold")+  
 ylab("Sales Price")+  
 scale\_y\_continuous(limits=c(50000,400000),labels=function(n){format(n, scientific = FALSE)})+  
 scale\_x\_discrete(limits=1:12)+  
 facet\_grid(.~YrSold)

descriptive<-train[, list(mean=mean(SalePrice),  
 min=min(SalePrice),  
 lower=quantile(SalePrice, .25, na.rm=TRUE),  
 median=quantile(SalePrice, .50, na.rm=TRUE),  
 upper=quantile(SalePrice, .75, na.rm=TRUE),  
 max=max(SalePrice)), by=YrSold]  
  
descriptive<-descriptive[order(descriptive$YrSold)]  
descriptive

## YrSold mean min lower median upper max  
## 1: 2006 182549.5 35311 131375 163995 218782.5 625000  
## 2: 2007 186063.2 39300 129900 167000 219500.0 755000  
## 3: 2008 177360.8 40000 131250 164000 207000.0 446261  
## 4: 2009 179432.1 34900 125250 162000 212750.0 582933  
## 5: 2010 177393.7 55000 128100 155000 213250.0 611657

Based on this I would conclude that the prices of houses around the mean and median have dropped while the higher end prices have dipped in 2008 with the housing crisis and recovered slighly.

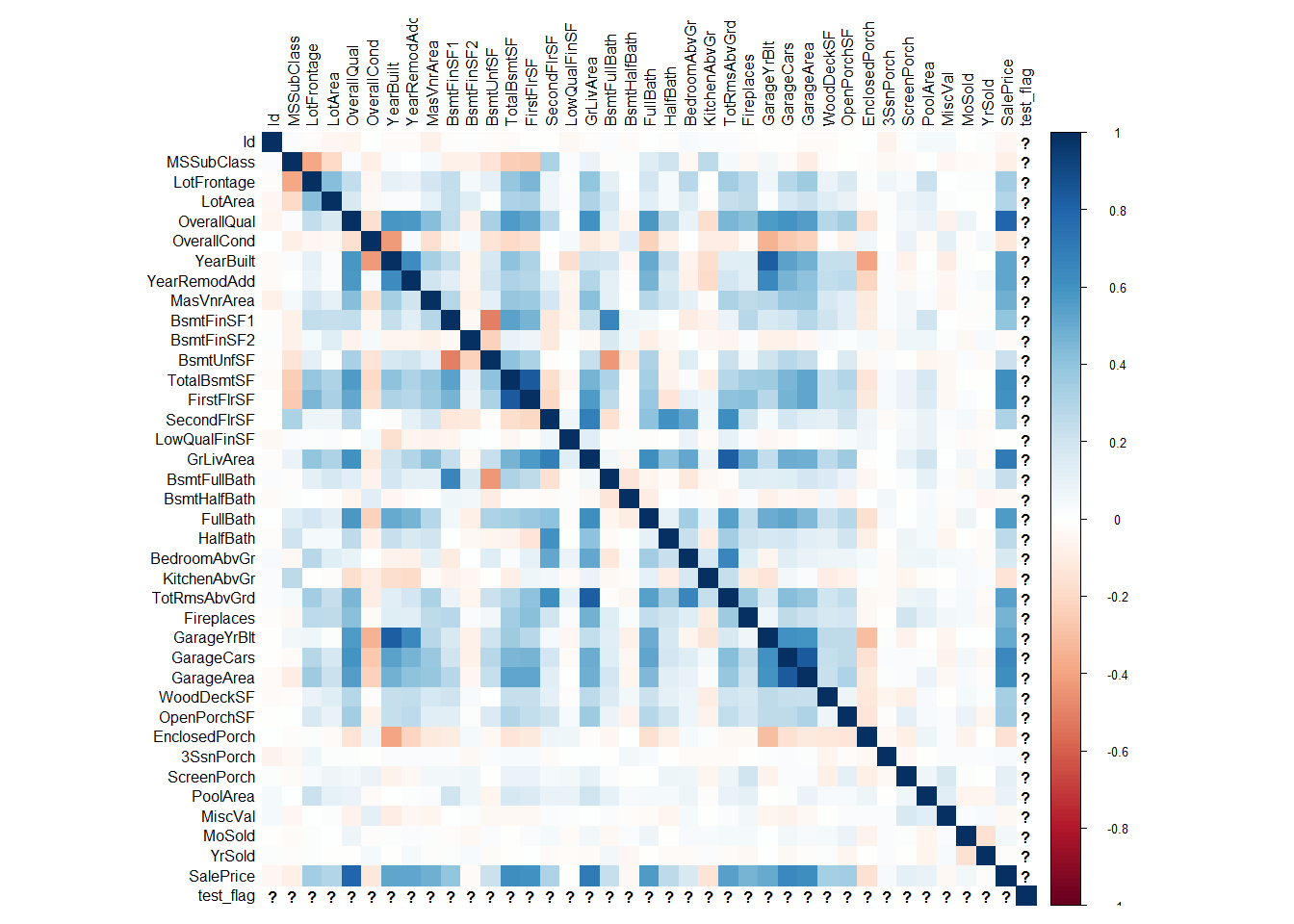
### What features correlate most with sale price?

This will be done only using numeric variables.

setnames(train, c("1stFlrSF","2ndFlrSF"), c("FirstFlrSF","SecondFlrSF"))  
numeric<-sapply(train, is.numeric)  
#numeric defined by previous code.  
cormatrix<-cor(x=train[, numeric,with=FALSE],use='complete')

## Warning in cor(x = train[, numeric, with = FALSE], use = "complete"): the  
## standard deviation is zero

par(mfrow=c(1,1))  
corrplot(cormatrix, method="color",mar=c(1,1,1,1),rect.col="black", tl.col="black")

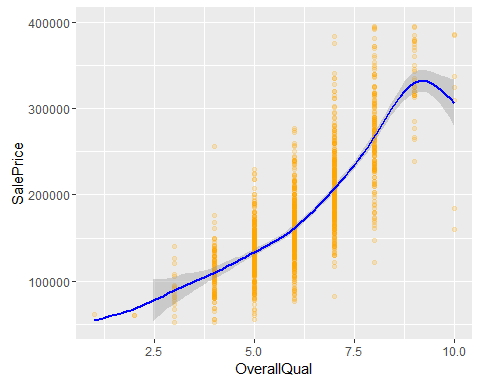


topcor<-data.table(Features=rownames(cormatrix),cormatrix[,35])  
setnames(topcor, names(topcor), c("Features","Correlation\_with\_Sale\_Price"))  
setorder(topcor, -Correlation\_with\_Sale\_Price)  
topcor

## Features Correlation\_with\_Sale\_Price  
## 1: test\_flag NA  
## 2: MiscVal 1.0000000000  
## 3: ScreenPorch 0.1698568742  
## 4: PoolArea 0.1286840123  
## 5: OverallCond 0.1197721471  
## 6: Fireplaces 0.0548263163  
## 7: BedroomAbvGr 0.0465228967  
## 8: Id 0.0457990995  
## 9: YrSold 0.0341060793  
## 10: OpenPorchSF 0.0288425004  
## 11: EnclosedPorch 0.0287954966  
## 12: BsmtFullBath 0.0276411019  
## 13: SecondFlrSF 0.0270465356  
## 14: TotRmsAbvGrd 0.0264950274  
## 15: 3SsnPorch 0.0246136793  
## 16: MoSold 0.0200670615  
## 17: LowQualFinSF 0.0152312894  
## 18: BsmtFinSF2 0.0142901786  
## 19: LotArea 0.0127899922  
## 20: BsmtFinSF1 0.0030266034  
## 21: LotFrontage 0.0014707377  
## 22: GrLivArea 0.0010669684  
## 23: KitchenAbvGr -0.0006403304  
## 24: WoodDeckSF -0.0071006971  
## 25: BsmtHalfBath -0.0109465220  
## 26: FullBath -0.0293972414  
## 27: FirstFlrSF -0.0309091885  
## 28: TotalBsmtSF -0.0310755000  
## 29: SalePrice -0.0360412372  
## 30: GarageArea -0.0369929442  
## 31: BsmtUnfSF -0.0389153564  
## 32: YearRemodAdd -0.0404198692  
## 33: MSSubClass -0.0406886729  
## 34: GarageYrBlt -0.0532954488  
## 35: MasVnrArea -0.0540440976  
## 36: HalfBath -0.0553793731  
## 37: OverallQual -0.0620638160  
## 38: GarageCars -0.0695923062  
## 39: YearBuilt -0.0969733924  
## Features Correlation\_with\_Sale\_Price

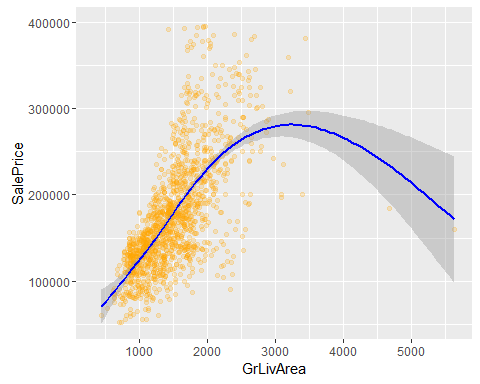
#### OverallQual

ggplot(data=train, aes(x=OverallQual, y=SalePrice))+  
 geom\_point(col="orange",alpha=.2)+  
 geom\_smooth(col="blue")+  
scale\_y\_continuous(limits=c(50000,400000),labels=function(n){format(n, scientific = FALSE)})



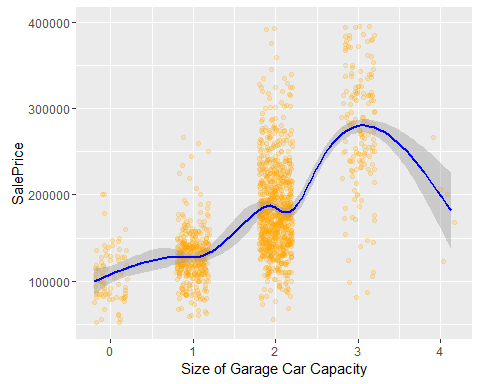
#### GrLivArea

ggplot(data=train, aes(x=GrLivArea, y=SalePrice))+  
 geom\_point(col="orange",alpha=.2)+  
 geom\_smooth(col="blue")+  
scale\_y\_continuous(limits=c(50000,400000),labels=function(n){format(n, scientific = FALSE)})

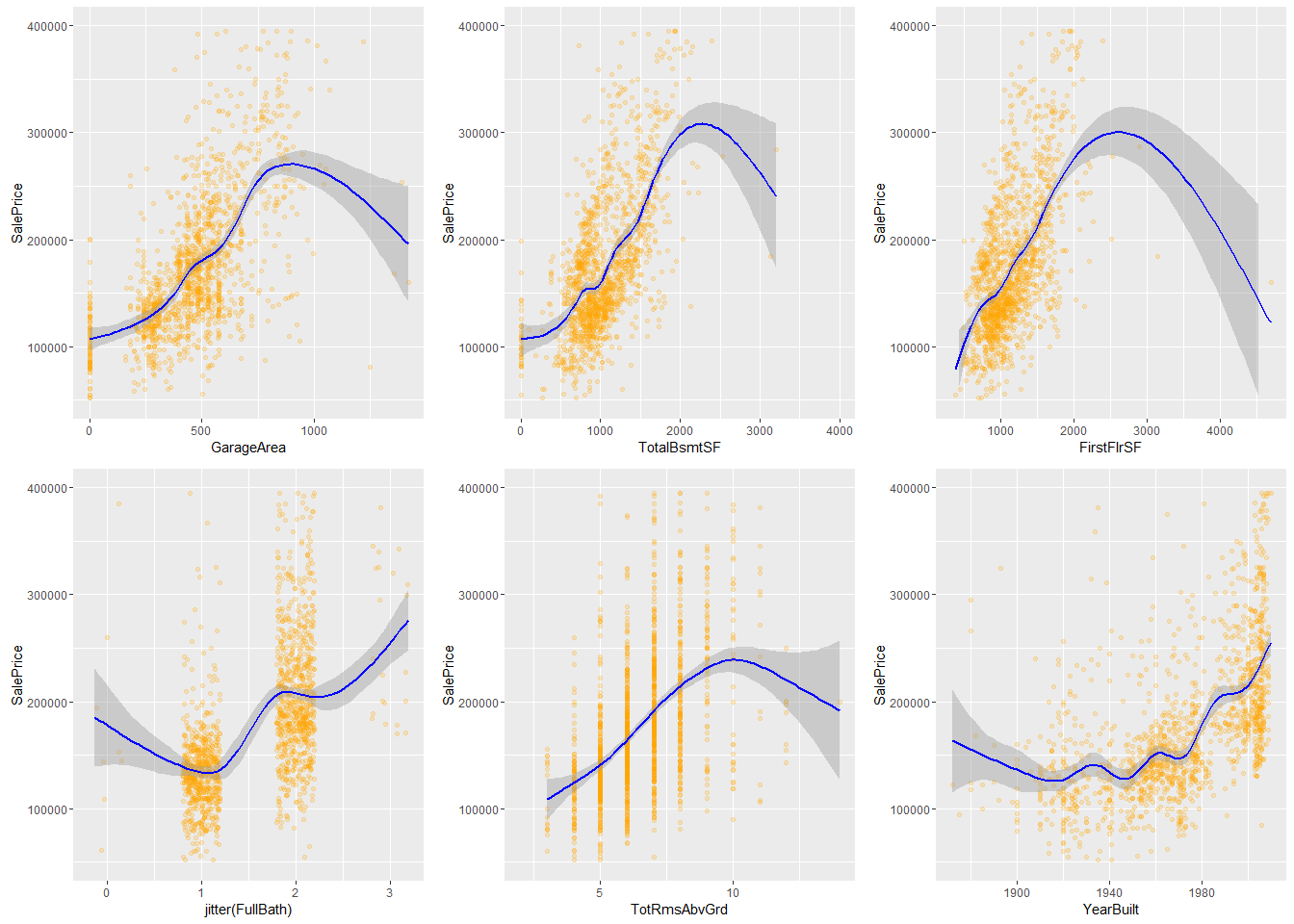


#### GarageCars

ggplot(data=train, aes(x=jitter(GarageCars), y=SalePrice))+  
 geom\_point(col="orange",alpha=.2)+  
 stat\_smooth(col="blue")+  
 xlab("Size of Garage Car Capacity")+  
scale\_y\_continuous(limits=c(50000,400000),labels=function(n){format(n, scientific = FALSE)})



### Various Other Continuous Variables



#### How do categorical variables affect sale price?

In this section I'll be using all of the categorical variables in a linear regression to determine which plots might be the most interesting.

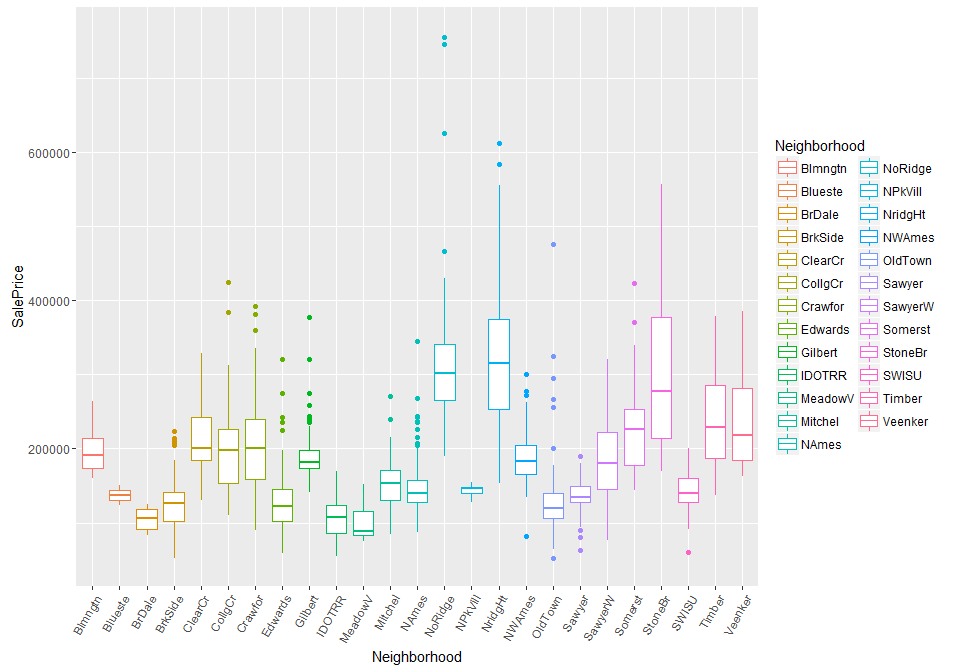
cat\_vars<-sapply(train, is.character)  
train\_cat<-train[,cat\_vars,with=FALSE]  
##Full Summary  
fits <- train[, list(MyFits = lapply(.SD[, cat\_vars, with = F], function(x) summary(lm(SalePrice ~ x))))]  
  
adjRSquared <- train[, list(MyFits = lapply(.SD[, cat\_vars, with = F], function(x) summary(lm(SalePrice ~ x))$adj.r.squared))]  
  
Rsqtable<-as.data.table(cbind(names(train[,cat\_vars,with=F]),as.numeric(unlist(adjRSquared))))  
setnames(Rsqtable, names(Rsqtable),c("cat\_variable", "adjRsquared"))  
Rsqtable[order(-adjRsquared)]

## cat\_variable adjRsquared  
## 1: Neighborhood 0.537974851283033  
## 2: ExterQual 0.476310966829465  
## 3: KitchenQual 0.455478978753147  
## 4: BsmtQual 0.452601216568323  
## 5: Alley 0.277468599280082  
## 6: GarageFinish 0.266211351296465  
## 7: Foundation 0.253811208963466  
## 8: GarageType 0.203749250269241  
## 9: HeatingQC 0.193288803326939  
## 10: BsmtFinType1 0.189748610154178  
## 11: MasVnrType 0.186210851690581  
## 12: Exterior2nd 0.145040004101391  
## 13: Exterior1st 0.144564696653506  
## 14: SaleType 0.132530974156944  
## 15: SaleCondition 0.132524634631605  
## 16: BsmtExposure 0.128575817738272  
## 17: PoolQC 0.115439809961231  
## 18: FireplaceQu 0.108504732590094  
## 19: MiscFeature 0.106595926165987  
## 20: MSZoning 0.105106239277029  
## 21: HouseStyle 0.0819077983726157  
## 22: LotShape 0.074472635499068  
## 23: CentralAir 0.0625232985084951  
## 24: Electrical 0.057086183258484  
## 25: RoofStyle 0.0544562471506006  
## 26: PavedDrive 0.0532419105246406  
## 27: Fence 0.0405848899597387  
## 28: BldgType 0.0318798249008438  
## 29: Condition1 0.0272971079951354  
## 30: BsmtCond 0.0267551979646207  
## 31: RoofMatl 0.0267436270803472  
## 32: GarageQual 0.0242738830799599  
## 33: GarageCond 0.024193102378254  
## 34: LandContour 0.0237867930549778  
## 35: ExterCond 0.0209334079453943  
## 36: LotConfig 0.0183280088392003  
## 37: Functional 0.0124190521455775  
## 38: Heating 0.0110479921506154  
## 39: BsmtFinType2 0.00595465405866302  
## 40: Condition2 0.00512594730394333  
## 41: LandSlope 0.00131262299126422  
## 42: Street 0.000999199066213152  
## 43: Utilities -0.000480831447705832  
## cat\_variable adjRsquared

We can see from above that from the adjusted R squares that **Neighborhood**, **Exterior Quality**, **Kitchen Quality**, **Basement Quality** are our top features for explaining variance in sale price. Let's graph them to visualize how each of these affect sale price.

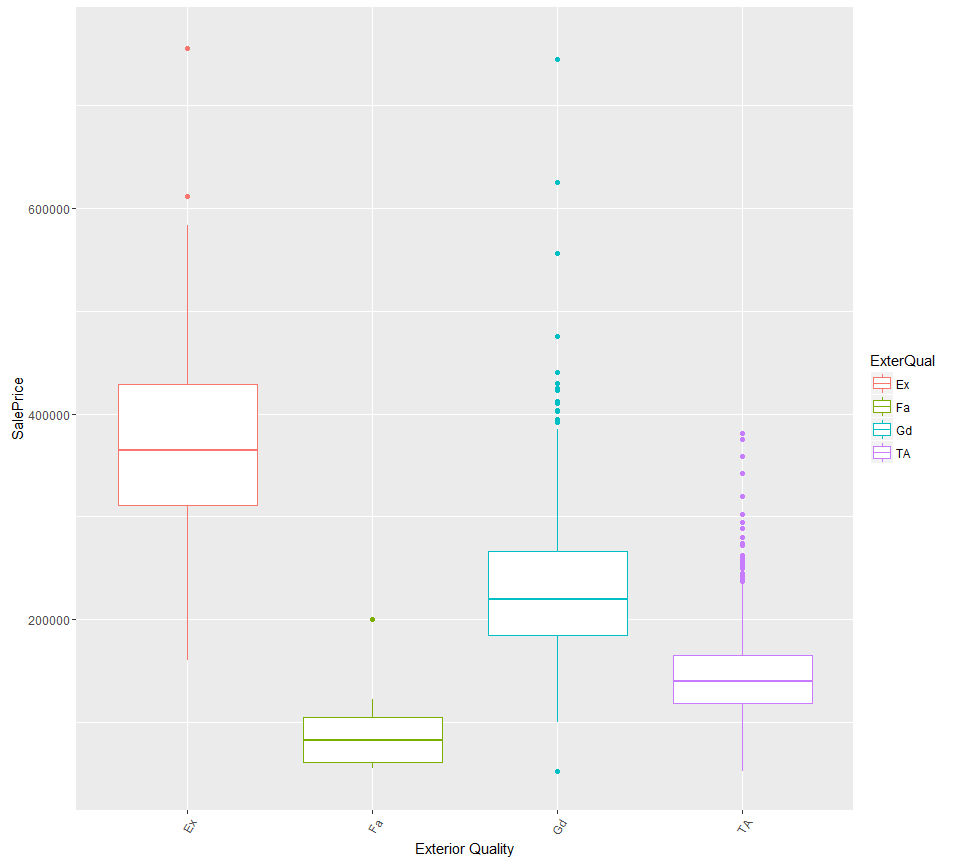
#### Neighborhood

ggplot(data=train, aes(x=as.factor(Neighborhood), y=SalePrice, col=Neighborhood))+  
 geom\_boxplot()+  
 scale\_y\_continuous(limits=c(50000,760000),labels=function(n){format(n, scientific = FALSE)})+theme(axis.text.x = element\_text(angle = 60, hjust = 1))+  
 xlab("Neighborhood")



#### Exterior Quality

ggplot(data=train, aes(x=as.factor(ExterQual), y=SalePrice, col=ExterQual))+  
 geom\_boxplot()+  
 scale\_y\_continuous(limits=c(50000,760000),labels=function(n){format(n, scientific = FALSE)})+theme(axis.text.x = element\_text(angle = 60, hjust = 1))+  
 xlab("Exterior Quality")



table(train$ExterQual, useNA = "always")

##   
## Ex Fa Gd TA <NA>   
## 52 14 488 906 0

table(test$ExterQual, useNA = "always")

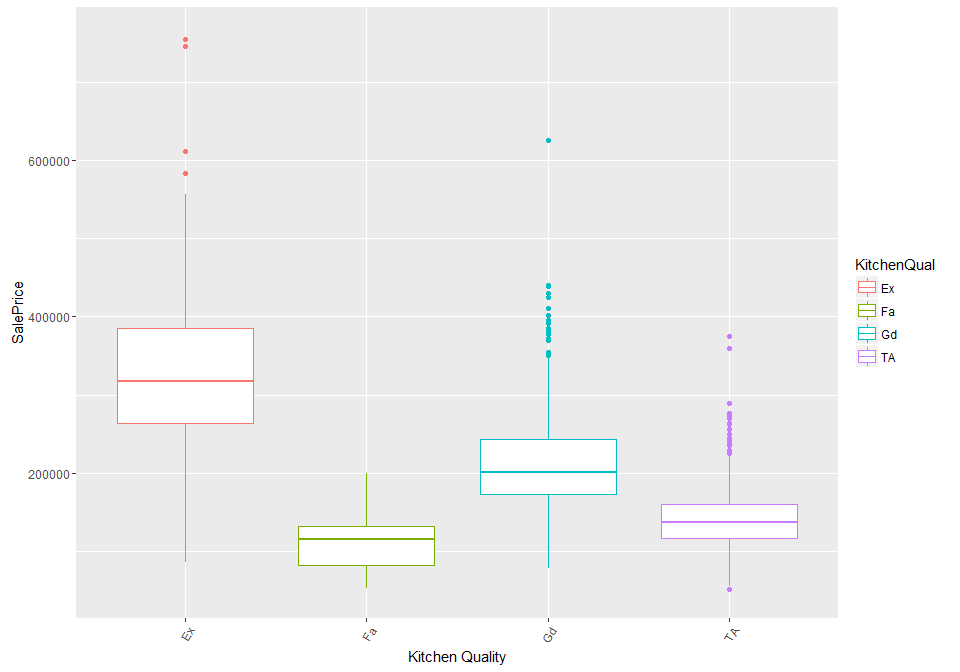
##   
## Ex Fa Gd TA <NA>   
## 55 21 491 892 0

* ExterQual: Evaluates the quality of the material on the exterior
  + Ex Excellent
  + Gd Good
  + TA Average/Typical
  + Fa Fair
  + Po Poor

**Exterior quality** shows that there should be 5 options but it only seems that there are 4 within our train set. It's missing PO for Poor. Test also follows this pattern.

#### Kitchen Quality

ggplot(data=train, aes(x=as.factor(KitchenQual), y=SalePrice, col=KitchenQual))+  
 geom\_boxplot()+  
 scale\_y\_continuous(limits=c(50000,760000),labels=function(n){format(n, scientific = FALSE)})+theme(axis.text.x = element\_text(angle = 60, hjust = 1))+  
 xlab("Kitchen Quality")



table(train$KitchenQual, useNA = "always")

##   
## Ex Fa Gd TA <NA>   
## 100 39 586 735 0

table(test$KitchenQual, useNA = "always")

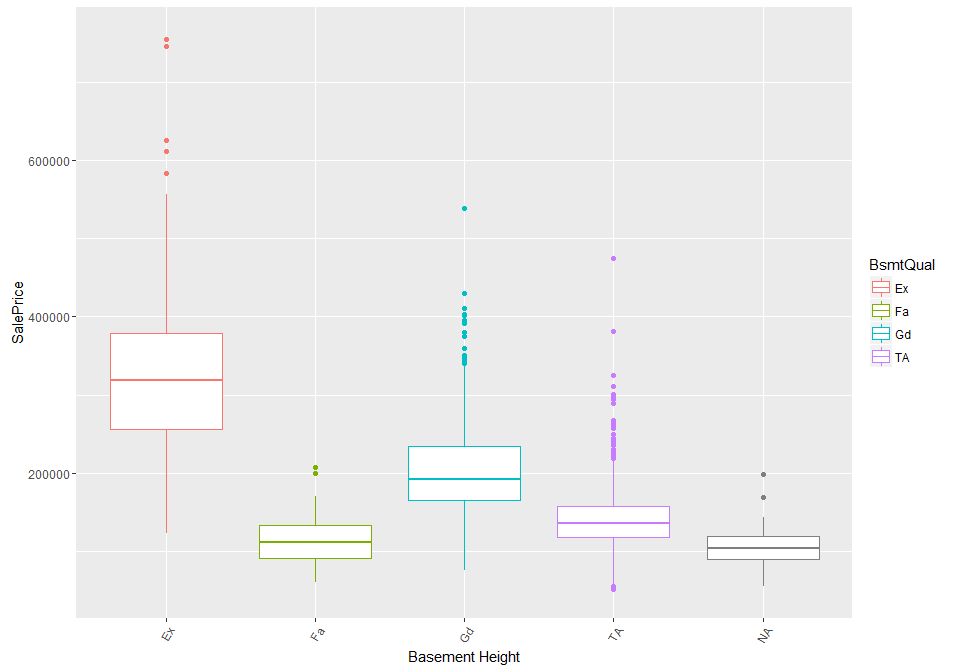
##   
## Ex Fa Gd TA <NA>   
## 105 31 565 757 1

* KitchenQual: Kitchen quality
  + Ex Excellent
  + Gd Good
  + TA Average/Typical
  + Fa Fair
  + Po Poor

**Kitchen Quality** shows that there should be 5 options but it only seems that there are 4 within our train set. It's missing PO for Poor. Test seems to have 1 NA.

#### Basement Height

ggplot(data=train,na.rm=FALSE, aes(x=as.factor(BsmtQual), y=SalePrice, col=BsmtQual))+  
 geom\_boxplot()+  
 scale\_y\_continuous(limits=c(50000,760000),labels=function(n){format(n, scientific = FALSE)})+theme(axis.text.x = element\_text(angle = 60, hjust = 1))+  
 xlab("Basement Height")



table(test$BsmtQual, useNA = "always")

##   
## Ex Fa Gd TA <NA>   
## 137 53 591 634 44

table(train$BsmtQual, useNA = "always")

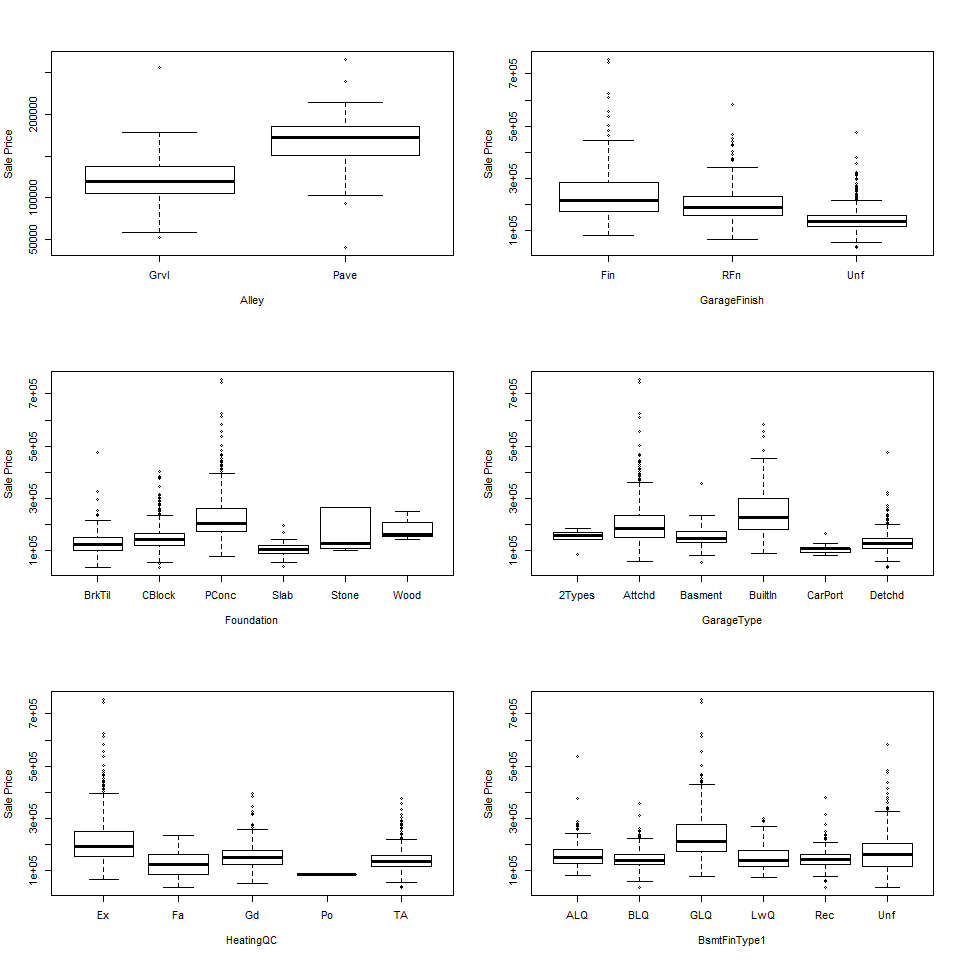
##   
## Ex Fa Gd TA <NA>   
## 121 35 618 649 37

* BsmtQual: Evaluates the height of the basement
  + Ex (100+ inches)
  + Gd Good (90-99 inches)
  + TA Typical (80-89 inches)
  + Fa Fair (70-79 inches)
  + Po Poor (<70 inches

**Basement Height** shows that there should be 5 options but PO or Poor is missing. Test and Train have multiple NA's so this may be where Poor was allocated.

### Various other categorical variables.

par(mfrow=c(3,2))  
boxplot(SalePrice ~ as.factor(train$Alley), train, xlab = "Alley", ylab = "Sale Price")  
boxplot(SalePrice ~ as.factor(train$GarageFinish), train, xlab = "GarageFinish", ylab = "Sale Price")  
boxplot(SalePrice ~ as.factor(train$Foundation), train, xlab = "Foundation", ylab = "Sale Price")  
boxplot(SalePrice ~ as.factor(train$GarageType), train, xlab = "GarageType", ylab = "Sale Price")  
boxplot(SalePrice ~ as.factor(train$HeatingQC), train, xlab = "HeatingQC", ylab = "Sale Price")  
boxplot(SalePrice ~ as.factor(train$BsmtFinType1), train, xlab = "BsmtFinType1", ylab = "Sale Price")



## Data Treatment

Data treatment will be relegated to assigning values to NA's and transforming skewed variables to more normal distributions.

### NA Features

Let's see what the distribution is of features with NA's. *I have hidden the results to save space.*

#creating a function to show the count and percentage of a given feature.  
tblFun <- function(x){  
 tbl <- table(x, useNA="always")  
 res <- cbind(tbl,round(prop.table(tbl)\*100,2))  
 colnames(res) <- c('Count','Percentage')  
 res  
}  
  
tblFun(all\_data$LotFrontage)  
tblFun(all\_data$Alley)  
tblFun(all\_data$MasVnrType)  
tblFun(all\_data$MasVnrArea)  
tblFun(all\_data$BsmtQual)  
tblFun(all\_data$BsmtCond)  
tblFun(all\_data$BsmtExposure)  
tblFun(all\_data$BsmtFinType1)  
tblFun(all\_data$BsmtFinType2)  
tblFun(all\_data$Electrical)  
tblFun(all\_data$FireplaceQu)  
tblFun(all\_data$GarageType)  
tblFun(all\_data$GarageYrBlt)  
tblFun(all\_data$GarageFinish)  
tblFun(all\_data$GarageQual)  
tblFun(all\_data$GarageCond)  
tblFun(all\_data$PoolQC)  
tblFun(all\_data$Fence)  
tblFun(all\_data$MiscFeature)

After looking at these features in the data dictionary you can conclude the majority have just disgnated NA to mean that this feature is absence from the home. This means that these can just be changed to none. However, *Electrical*, and *LotFrontage* look like they could use adjustment.

### Handling NA's

Let's assign 0 to NA's as the majority of the NA's are signifying None for the feature. LotFrontage will be an example of changing a feature to match its neighborhood's average.

all\_data[all\_data=="",]<-0  
all\_data[is.na(all\_data),]<-0  
all\_data[Electrical=="None",]$Electrical<-"SBrkr"  
all\_data[]$LotFrontage<-as.numeric(all\_data$LotFrontage)  
all\_data[, meanLotFrontage:=mean(LotFrontage,na.rm=T), by=Neighborhood]  
all\_data[,LotFrontage2:=ifelse(is.na(all\_data$LotFrontage), meanLotFrontage,LotFrontage)]  
  
drop<-c("meanLotFrontage","LotFrontage")  
all\_data[,(drop):=NULL]  
rm(drop)

### Transforming Skewed Variables.

Transforming skewed features will lead to better predictions as linear models perform better on normal distributions.

numeric<-sapply(all\_data, is.numeric)  
abs\_skew<-abs(apply(all\_data[,numeric,with=F],2,skewness))  
abs\_skew

## Id MSSubClass LotArea OverallQual OverallCond   
## 0.000000e+00 1.375457e+00 1.282243e+01 1.971104e-01 5.703121e-01   
## YearBuilt YearRemodAdd MasVnrArea BsmtFinSF1 BsmtFinSF2   
## 5.998055e-01 4.510204e-01 2.613592e+00 1.425230e+00 4.146143e+00   
## BsmtUnfSF TotalBsmtSF 1stFlrSF 2ndFlrSF LowQualFinSF   
## 9.193392e-01 1.156894e+00 1.469604e+00 8.616747e-01 1.208876e+01   
## GrLivArea BsmtFullBath BsmtHalfBath FullBath HalfBath   
## 1.269358e+00 6.248321e-01 3.931594e+00 1.676057e-01 6.945665e-01   
## BedroomAbvGr KitchenAbvGr TotRmsAbvGrd Fireplaces GarageYrBlt   
## 3.263243e-01 4.302254e+00 7.583669e-01 7.334946e-01 3.906205e+00   
## GarageCars GarageArea WoodDeckSF OpenPorchSF EnclosedPorch   
## 2.195810e-01 2.392569e-01 1.842433e+00 2.535114e+00 4.003891e+00   
## 3SsnPorch ScreenPorch PoolArea MiscVal MoSold   
## 1.137606e+01 3.946694e+00 1.689833e+01 2.194719e+01 1.958839e-01   
## YrSold SalePrice test\_flag LotFrontage2   
## 1.323988e-01 1.098792e+00 6.851662e-04 2.201286e-02

abs\_skew<-abs\_skew[abs\_skew>1]  
name\_skew<-names(abs\_skew)  
all\_data.skew<-all\_data[,name\_skew,with=F]  
all\_data.skew.m <- log1p(as.matrix(all\_data.skew))  
  
  
#Add in the new columns and delete old  
all\_data.norm<-cbind(all\_data,all\_data.skew.m)  
all\_data.norm[,(name\_skew):=NULL]  
  
#Rename the features starting with numbers  
setnames(all\_data.norm, c("1stFlrSF","2ndFlrSF","3SsnPorch"), c("FirstFlrSF","SecondFlrSF","ThreeSsnPorch"))  
  
#Change the character features to factor  
upd.cols = sapply(all\_data.norm, is.character)  
all\_data.norm[, names(all\_data.norm)[upd.cols] := lapply(.SD, as.factor), .SDcols = upd.cols]  
  
#GarageYrBlt relegated GarageYrBlt to the last decade or not  
all\_data.norm$GarageYrBlt<-as.numeric(all\_data.norm$GarageYrBlt)  
all\_data.norm$GarageYrBlt<-ifelse(all\_data.norm$GarageYrBlt>2000,1,0)  
  
#Split test and train since transformation is complete  
train.final<-all\_data.norm[test\_flag==0,]  
test.final<-all\_data.norm[test\_flag==1,]

## Prediction

Let's now use linear regression, random forest and extreme gradient boosting to predict house prices with k-folds CV.

#### Linear Regression

set.seed(10)  
folds<-sample(rep(1:10, length=nrow(train.final)))  
table(folds)

## folds  
## 1 2 3 4 5 6 7 8 9 10   
## 146 146 146 146 146 146 146 146 146 146

cv.lm.errors<-matrix(NA, nrow=10,ncol=2)  
  
  
for(k in 1:10)  
{  
 lm.fit<-lm(SalePrice~OverallQual+Neighborhood+GrLivArea+GarageCars+BsmtFinSF1+YearBuilt+LotArea+OverallCond+MSZoning, data=train.final[folds!=k,])  
 pred<-predict(lm.fit, train.final[folds==k,])  
 cv.lm.errors[k,1]<-sqrt(mean((train.final$SalePrice[folds==k]-pred)^2, na.rm=T))  
 cv.lm.errors[k,2]<-summary(lm.fit)$adj.r.squared  
}  
  
#RMSE of each fold  
cv.lm.errors

## [,1] [,2]  
## [1,] 0.1127420 0.8859220  
## [2,] 0.1132323 0.8877849  
## [3,] 0.1403066 0.8902759  
## [4,] 0.1266468 0.8907725  
## [5,] 0.1354475 0.8916943  
## [6,] 0.1380004 0.8924642  
## [7,] 0.1654739 0.8971195  
## [8,] 0.1404707 0.8912237  
## [9,] 0.1348698 0.8911803  
## [10,] 0.1400355 0.8899768

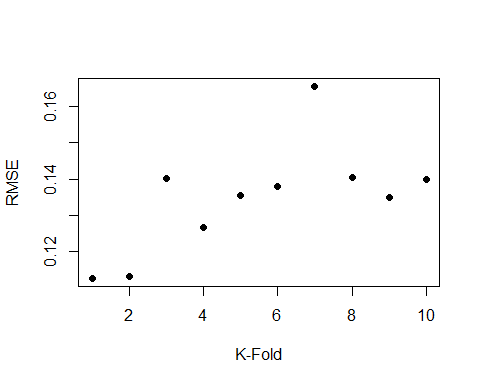
#Average RMSE  
mean(cv.lm.errors[,1])

## [1] 0.1347225

#Average adj.R.Squared  
mean(cv.lm.errors[,2])

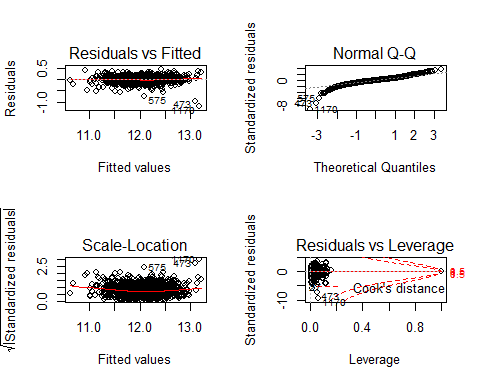
## [1] 0.8908414

#Distribution of RMSE by fold  
plot(cv.lm.errors[,1], pch=19, ylab="RMSE", xlab="K-Fold")



#plotting one of the folds  
layout(matrix(c(1,2,3,4), 2, 2, byrow = TRUE))  
plot(lm.fit)

## Warning in sqrt(crit \* p \* (1 - hh)/hh): NaNs produced  
  
## Warning in sqrt(crit \* p \* (1 - hh)/hh): NaNs produced



#### Random Forest

Looking for the optimal value for mtry which determines how many variables each decision tree subsets from the pool of paramters.

oob.err<-matrix(NA, nrow=10,ncol=16)  
test.err<-matrix(NA, nrow=10,ncol=16)  
colindex<-1  
folds<-sample(rep(1:10, length=nrow(train.norm)))  
  
#trying to find the correct tuning parameters  
for(mtry in 10:25){  
 for (k in 1:10){  
 train.folds<-train.final[,.I[folds!=k]]  
 train.val<-train.final[folds==k,]  
 fit<-randomForest(SalePrice~.-GarageYrBlt-Id-binnum, data=train.final,subset=train.folds, mtry=mtry, ntree=300)  
 oob.err[k,colindex]<-fit$mse[300]  
 pred<-predict(fit,train.val)  
 test.err[k,colindex]<-with(train.val,sqrt(mean((SalePrice-pred)^2,na.rm=T)))  
 #cat(mtry,k," ")  
 }  
 colindex<-colindex+1  
}  
test.err2<-as.data.table(test.err)  
g.test.err<-sapply(test.err2, mean)  
  
oob.err2<-as.data.table(oob.err)  
g.oob.err<-sapply(oob.err2, mean)  
  
par(mfrow=c(1,2))  
#Note that these are on different scales.  
plot(1:11,y=g.test.err, ylab="RMSE",type="b", pch=19)  
plot(1:11,y=g.oob.err, ylab="RMSE",type="b", pch=19)  
  
par(mfrow=c(1,1))

Now we can attempt to opimize the value for ntree which determines how many trees are built with random forest.

set.seed(10)  
folds<-sample(rep(1:10, length=nrow(train.final)))  
oob.err.kag<-matrix(NA, nrow=10,ncol=4)  
test.err.kag<-matrix(NA, nrow=10,ncol=4)  
colindex<-1  
  
for(ntree in seq(400,700,by=100) )  
{  
 for (k in 1:10){  
 train.folds<-train.norm[,.I[folds!=k]]  
 train.val<-train.norm[folds==k,]  
 fit<-randomForest(SalePrice~.-GarageYrBlt-Id-binnum, data=train.norm,subset=train.folds, mtry=30, ntree=ntree)  
 oob.err.kag[k,colindex]<-fit$mse[300]  
 pred<-predict(fit,train.val)  
 test.err.kag[k,colindex]<-with(train.val,sqrt(mean((SalePrice-pred)^2,na.rm=T)))  
 #cat(mtry,ntree," ")  
 }  
 colindex<-colindex+1  
}  
  
test.err.kag2<-as.data.table(test.err.kag)  
g.test.err.kag<-sapply(test.err.kag2, mean)  
  
oob.err.kag2<-as.data.table(oob.err.kag)  
g.oob.err.kag<-sapply(oob.err.kag2, mean)  
  
par(mfrow=c(1,2))  
plot(1:4, g.test.err.kag, pch=19, col=brewer.pal(12,"Paired"), type="b", ylab="Mean Squared Error", xlab="Trees built(400:700)",xlim=c(1,4), main="Test.err.RMSE")  
plot(1:4, g.oob.err.kag, pch=19, col=brewer.pal(12,"Paired"), type="b", ylab="Mean Squared Error", xlab="Trees built(400:700)",xlim=c(1,4),main="Out of Bag Error")  
par(mfrow=c(1,1))  
#Variable importance plot of the last random forest model  
varImpPlot(fit)

Using What we learned from the tuning (leaving us with RMSE .14287 on the leaderboard)

fit<-randomForest(SalePrice~., data=train.final, mtry=9, ntree=500)  
pred<-predict(fit,test.final)  
  
test.final[,SalePrice:=pred]  
sub9\_23\_2016<-test.final[,.(Id,SalePrice)]  
sub9\_23\_2016<-sub9\_23\_2016[,SalePrice:=expm1(SalePrice)]  
write.csv(sub9\_23\_2016, "sub9\_23\_2016.csv",row.names=F)

#### Extreme Gradient Boosting (xgb)

features=names(train.final)[!(names(train.final) %in% c('Id',"SalePrice"))]   
  
wltst=sample(nrow(train.final),146)   
  
dval<-xgb.DMatrix(data=data.matrix(train.final[wltst,features,with=FALSE]),label=data.matrix(train.final[wltst,SalePrice]),missing=NA)  
watchlist<-list(dval=dval)  
  
clf <- xgb.train(params=list( objective="reg:linear",   
 booster = "gbtree",  
 eta=0.008,   
 max\_depth=6,   
 subsample=0.85,  
 colsample\_bytree=1) ,  
 data = xgb.DMatrix(data=data.matrix(train.final[-wltst,features,with=FALSE]),  
 label=data.matrix(train.final[-wltst,SalePrice]),missing=NA),   
 nrounds = 5000,   
 verbose = 1,  
 print\_every\_n=5,  
 early\_stopping\_rounds = 10,  
 watchlist = watchlist,  
 maximize = FALSE,  
 eval\_metric='rmse'  
)  
#These take a significant amount of time and I have commented them out.  
#importance\_matrix <- xgb.importance(features, model = clf)  
#xgb.plot.importance(importance\_matrix)  
  
pred<-predict(clf,xgb.DMatrix(data.matrix(test.final[,features,with=FALSE]),missing=NA))  
test.final[,SalePrice:=pred]  
sub9\_23\_2016<-test.final[,.(Id,SalePrice)]  
sub9\_23\_2016\_xgb<-sub9\_23\_2016[,SalePrice:=expm1(SalePrice)]  
write.csv(sub9\_23\_2016\_xgb, "sub9\_23\_2016\_xgb.csv",row.names=F)

### Final Words

This model yields a RMSE of .12631 on the leaderboard(test set) and a rank of 130.

Thank you for reading!