**Data Driven Real Estate Analysis**

Reality tv shows about real estate are all the rage lately, and they drive me nuts because of how unscientific and illogical they are. (Well, that's not the only reason, but it's the relevant one.) People with more money than they know what to do with going 10 rounds about the kind of drawer pulls they want to have in their kitchen and where exactly their imported bathroom tile should come from. Blech.

This dataset gives us a chance to look into the data on what really influences the value of a house without the garbage impulses of reality tv getting in the way, so I am excited to take a look!

library(data.table)

library(FeatureHashing)

library(Matrix)

library(xgboost)

require(randomForest)

require(caret)

require(dplyr)

require(ggplot2)

library(pROC)

library(stringr)

library(dummies)

library(Metrics)

library(kernlab)

library(mlbench)

###Plan

* Assemble the data and explore it
* Clean variables, build what is needed
* Three Models: Linear, randomForest, and xgboost
* Choose the best model and make the prediction for entry

###Clean the Data

So, what do we have here?

train <- read.csv("../input/train.csv", stringsAsFactors=FALSE)

test <- read.csv("../input/test.csv", stringsAsFactors=FALSE)

#train <- read.csv("U:/Programming/R/Kaggle/houses/train.csv", stringsAsFactors=FALSE)

#test <- read.csv("U:/Programming/R/Kaggle/houses/test.csv", stringsAsFactors=FALSE)

names(train)

I think the best steps to start with would be reformatting some character variables that we can easily convert to numeric. What's the street type about?

table(train$Street)

Not exactly fancy, let's just make that paved or not. What about Lot Shape?

train$paved[train$Street == "Pave"] <- 1

train$paved[train$Street != "Pave"] <- 0

table(train$LotShape)

```

I assume these are something like variations on "irregular". So let's go with regular or not, and then we'll have this shape variable still if we want to go more granular later.

Taking up land contour as the next.

```{r formatting\_shape}

train$regshape[train$LotShape == "Reg"] <- 1

train$regshape[train$LotShape != "Reg"] <- 0

table(train$LandContour)

In order to save space, I'll just go through the rest of the categoricals using the provided codebook and pick up narrating again when it's done.

Cue muzak. Go look at the code tab if you want to read all of this. It's about 300 lines. Have a good time.

###Poking Around

train$flat[train$LandContour == "Lvl"] <- 1

train$flat[train$LandContour != "Lvl"] <- 0

train$pubutil[train$Utilities == "AllPub"] <- 1

train$pubutil[train$Utilities != "AllPub"] <- 0

train$gentle\_slope[train$LandSlope == "Gtl"] <- 1

train$gentle\_slope[train$LandSlope != "Gtl"] <- 0

# summarize(group\_by(train, LotConfig),

# mean(SalePrice, na.rm=T))

train$culdesac\_fr3[train$LandSlope %in% c("CulDSac", "FR3")] <- 1

train$culdesac\_fr3[!train$LandSlope %in% c("CulDSac", "FR3")] <- 0

nbhdprice <- summarize(group\_by(train, Neighborhood),

mean(SalePrice, na.rm=T))

#nbhdprice[order(nbhdprice$`mean(SalePrice, na.rm = T)`),]

nbhdprice\_lo <- filter(nbhdprice, nbhdprice$`mean(SalePrice, na.rm = T)` < 140000)

nbhdprice\_med <- filter(nbhdprice, nbhdprice$`mean(SalePrice, na.rm = T)` < 200000 &

nbhdprice$`mean(SalePrice, na.rm = T)` >= 140000 )

nbhdprice\_hi <- filter(nbhdprice, nbhdprice$`mean(SalePrice, na.rm = T)` >= 200000)

train$nbhd\_price\_level[train$Neighborhood %in% nbhdprice\_lo$Neighborhood] <- 1

train$nbhd\_price\_level[train$Neighborhood %in% nbhdprice\_med$Neighborhood] <- 2

train$nbhd\_price\_level[train$Neighborhood %in% nbhdprice\_hi$Neighborhood] <- 3

# summarize(group\_by(train, Condition1),

# mean(SalePrice, na.rm=T))

train$pos\_features\_1[train$Condition1 %in% c("PosA", "PosN")] <- 1

train$pos\_features\_1[!train$Condition1 %in% c("PosA", "PosN")] <- 0

# summarize(group\_by(train, Condition2),

# mean(SalePrice, na.rm=T))

train$pos\_features\_2[train$Condition1 %in% c("PosA", "PosN")] <- 1

train$pos\_features\_2[!train$Condition1 %in% c("PosA", "PosN")] <- 0

# summarize(group\_by(train, BldgType),

# mean(SalePrice, na.rm=T))

train$twnhs\_end\_or\_1fam[train$BldgType %in% c("1Fam", "TwnhsE")] <- 1

train$twnhs\_end\_or\_1fam[!train$BldgType %in% c("1Fam", "TwnhsE")] <- 0

housestyle\_price <- summarize(group\_by(train, HouseStyle),

mean(SalePrice, na.rm=T))

housestyle\_lo <- filter(housestyle\_price, housestyle\_price$`mean(SalePrice, na.rm = T)` < 140000)

housestyle\_med <- filter(housestyle\_price, housestyle\_price$`mean(SalePrice, na.rm = T)` < 200000 &

housestyle\_price$`mean(SalePrice, na.rm = T)` >= 140000 )

housestyle\_hi <- filter(housestyle\_price, housestyle\_price$`mean(SalePrice, na.rm = T)` >= 200000)

train$house\_style\_level[train$HouseStyle %in% housestyle\_lo$HouseStyle] <- 1

train$house\_style\_level[train$HouseStyle %in% housestyle\_med$HouseStyle] <- 2

train$house\_style\_level[train$HouseStyle %in% housestyle\_hi$HouseStyle] <- 3

roofstyle\_price <- summarize(group\_by(train, RoofStyle),

mean(SalePrice, na.rm=T))

train$roof\_hip\_shed[train$RoofStyle %in% c("Hip", "Shed")] <- 1

train$roof\_hip\_shed[!train$RoofStyle %in% c("Hip", "Shed")] <- 0

roofmatl\_price <- summarize(group\_by(train, RoofMatl),

mean(SalePrice, na.rm=T))

train$roof\_matl\_hi[train$RoofMatl %in% c("Membran", "WdShake", "WdShngl")] <- 1

train$roof\_matl\_hi[!train$RoofMatl %in% c("Membran", "WdShake", "WdShngl")] <- 0

price <- summarize(group\_by(train, Exterior1st),

mean(SalePrice, na.rm=T))

matl\_lo\_1 <- filter(price, price$`mean(SalePrice, na.rm = T)` < 140000)

matl\_med\_1<- filter(price, price$`mean(SalePrice, na.rm = T)` < 200000 &

price$`mean(SalePrice, na.rm = T)` >= 140000 )

matl\_hi\_1 <- filter(price, price$`mean(SalePrice, na.rm = T)` >= 200000)

train$exterior\_1[train$Exterior1st %in% matl\_lo\_1$Exterior1st] <- 1

train$exterior\_1[train$Exterior1st %in% matl\_med\_1$Exterior1st] <- 2

train$exterior\_1[train$Exterior1st %in% matl\_hi\_1$Exterior1st] <- 3

price <- summarize(group\_by(train, Exterior2nd),

mean(SalePrice, na.rm=T))

matl\_lo <- filter(price, price$`mean(SalePrice, na.rm = T)` < 140000)

matl\_med <- filter(price, price$`mean(SalePrice, na.rm = T)` < 200000 &

price$`mean(SalePrice, na.rm = T)` >= 140000 )

matl\_hi <- filter(price, price$`mean(SalePrice, na.rm = T)` >= 200000)

train$exterior\_2[train$Exterior2nd %in% matl\_lo$Exterior2nd] <- 1

train$exterior\_2[train$Exterior2nd %in% matl\_med$Exterior2nd] <- 2

train$exterior\_2[train$Exterior2nd %in% matl\_hi$Exterior2nd] <- 3

price <- summarize(group\_by(train, MasVnrType),

mean(SalePrice, na.rm=T))

train$exterior\_mason\_1[train$MasVnrType %in% c("Stone", "BrkFace") | is.na(train$MasVnrType)] <- 1

train$exterior\_mason\_1[!train$MasVnrType %in% c("Stone", "BrkFace") & !is.na(train$MasVnrType)] <- 0

price <- summarize(group\_by(train, ExterQual),

mean(SalePrice, na.rm=T))

train$exterior\_cond[train$ExterQual == "Ex"] <- 4

train$exterior\_cond[train$ExterQual == "Gd"] <- 3

train$exterior\_cond[train$ExterQual == "TA"] <- 2

train$exterior\_cond[train$ExterQual == "Fa"] <- 1

price <- summarize(group\_by(train, ExterCond),

mean(SalePrice, na.rm=T))

train$exterior\_cond2[train$ExterCond == "Ex"] <- 5

train$exterior\_cond2[train$ExterCond == "Gd"] <- 4

train$exterior\_cond2[train$ExterCond == "TA"] <- 3

train$exterior\_cond2[train$ExterCond == "Fa"] <- 2

train$exterior\_cond2[train$ExterCond == "Po"] <- 1

price <- summarize(group\_by(train, Foundation),

mean(SalePrice, na.rm=T))

train$found\_concrete[train$Foundation == "PConc"] <- 1

train$found\_concrete[train$Foundation != "PConc"] <- 0

price <- summarize(group\_by(train, BsmtQual),

mean(SalePrice, na.rm=T))

train$bsmt\_cond1[train$BsmtQual == "Ex"] <- 5

train$bsmt\_cond1[train$BsmtQual == "Gd"] <- 4

train$bsmt\_cond1[train$BsmtQual == "TA"] <- 3

train$bsmt\_cond1[train$BsmtQual == "Fa"] <- 2

train$bsmt\_cond1[is.na(train$BsmtQual)] <- 1

price <- summarize(group\_by(train, BsmtCond),

mean(SalePrice, na.rm=T))

train$bsmt\_cond2[train$BsmtCond == "Gd"] <- 5

train$bsmt\_cond2[train$BsmtCond == "TA"] <- 4

train$bsmt\_cond2[train$BsmtCond == "Fa"] <- 3

train$bsmt\_cond2[is.na(train$BsmtCond)] <- 2

train$bsmt\_cond2[train$BsmtCond == "Po"] <- 1

price <- summarize(group\_by(train, BsmtExposure),

mean(SalePrice, na.rm=T))

train$bsmt\_exp[train$BsmtExposure == "Gd"] <- 5

train$bsmt\_exp[train$BsmtExposure == "Av"] <- 4

train$bsmt\_exp[train$BsmtExposure == "Mn"] <- 3

train$bsmt\_exp[train$BsmtExposure == "No"] <- 2

train$bsmt\_exp[is.na(train$BsmtExposure)] <- 1

price <- summarize(group\_by(train, BsmtFinType1),

mean(SalePrice, na.rm=T))

train$bsmt\_fin1[train$BsmtFinType1 == "GLQ"] <- 5

train$bsmt\_fin1[train$BsmtFinType1 == "Unf"] <- 4

train$bsmt\_fin1[train$BsmtFinType1 == "ALQ"] <- 3

train$bsmt\_fin1[train$BsmtFinType1 %in% c("BLQ", "Rec", "LwQ")] <- 2

train$bsmt\_fin1[is.na(train$BsmtFinType1)] <- 1

price <- summarize(group\_by(train, BsmtFinType2),

mean(SalePrice, na.rm=T))

train$bsmt\_fin2[train$BsmtFinType2 == "ALQ"] <- 6

train$bsmt\_fin2[train$BsmtFinType2 == "Unf"] <- 5

train$bsmt\_fin2[train$BsmtFinType2 == "GLQ"] <- 4

train$bsmt\_fin2[train$BsmtFinType2 %in% c("Rec", "LwQ")] <- 3

train$bsmt\_fin2[train$BsmtFinType2 == "BLQ"] <- 2

train$bsmt\_fin2[is.na(train$BsmtFinType2)] <- 1

price <- summarize(group\_by(train, Heating),

mean(SalePrice, na.rm=T))

train$gasheat[train$Heating %in% c("GasA", "GasW")] <- 1

train$gasheat[!train$Heating %in% c("GasA", "GasW")] <- 0

price <- summarize(group\_by(train, HeatingQC),

mean(SalePrice, na.rm=T))

train$heatqual[train$HeatingQC == "Ex"] <- 5

train$heatqual[train$HeatingQC == "Gd"] <- 4

train$heatqual[train$HeatingQC == "TA"] <- 3

train$heatqual[train$HeatingQC == "Fa"] <- 2

train$heatqual[train$HeatingQC == "Po"] <- 1

price <- summarize(group\_by(train, CentralAir),

mean(SalePrice, na.rm=T))

train$air[train$CentralAir == "Y"] <- 1

train$air[train$CentralAir == "N"] <- 0

price <- summarize(group\_by(train, Electrical),

mean(SalePrice, na.rm=T))

train$standard\_electric[train$Electrical == "SBrkr" | is.na(train$Electrical)] <- 1

train$standard\_electric[!train$Electrical == "SBrkr" & !is.na(train$Electrical)] <- 0

price <- summarize(group\_by(train, KitchenQual),

mean(SalePrice, na.rm=T))

train$kitchen[train$KitchenQual == "Ex"] <- 4

train$kitchen[train$KitchenQual == "Gd"] <- 3

train$kitchen[train$KitchenQual == "TA"] <- 2

train$kitchen[train$KitchenQual == "Fa"] <- 1

price <- summarize(group\_by(train, FireplaceQu),

mean(SalePrice, na.rm=T))

train$fire[train$FireplaceQu == "Ex"] <- 5

train$fire[train$FireplaceQu == "Gd"] <- 4

train$fire[train$FireplaceQu == "TA"] <- 3

train$fire[train$FireplaceQu == "Fa"] <- 2

train$fire[train$FireplaceQu == "Po" | is.na(train$FireplaceQu)] <- 1

price <- summarize(group\_by(train, GarageType),

mean(SalePrice, na.rm=T))

train$gar\_attach[train$GarageType %in% c("Attchd", "BuiltIn")] <- 1

train$gar\_attach[!train$GarageType %in% c("Attchd", "BuiltIn")] <- 0

price <- summarize(group\_by(train, GarageFinish),

mean(SalePrice, na.rm=T))

train$gar\_finish[train$GarageFinish %in% c("Fin", "RFn")] <- 1

train$gar\_finish[!train$GarageFinish %in% c("Fin", "RFn")] <- 0

price <- summarize(group\_by(train, GarageQual),

mean(SalePrice, na.rm=T))

train$garqual[train$GarageQual == "Ex"] <- 5

train$garqual[train$GarageQual == "Gd"] <- 4

train$garqual[train$GarageQual == "TA"] <- 3

train$garqual[train$GarageQual == "Fa"] <- 2

train$garqual[train$GarageQual == "Po" | is.na(train$GarageQual)] <- 1

price <- summarize(group\_by(train, GarageCond),

mean(SalePrice, na.rm=T))

train$garqual2[train$GarageCond == "Ex"] <- 5

train$garqual2[train$GarageCond == "Gd"] <- 4

train$garqual2[train$GarageCond == "TA"] <- 3

train$garqual2[train$GarageCond == "Fa"] <- 2

train$garqual2[train$GarageCond == "Po" | is.na(train$GarageCond)] <- 1

price <- summarize(group\_by(train, PavedDrive),

mean(SalePrice, na.rm=T))

train$paved\_drive[train$PavedDrive == "Y"] <- 1

train$paved\_drive[!train$PavedDrive != "Y"] <- 0

train$paved\_drive[is.na(train$paved\_drive)] <- 0

price <- summarize(group\_by(train, Functional),

mean(SalePrice, na.rm=T))

train$housefunction[train$Functional %in% c("Typ", "Mod")] <- 1

train$housefunction[!train$Functional %in% c("Typ", "Mod")] <- 0

price <- summarize(group\_by(train, PoolQC),

mean(SalePrice, na.rm=T))

train$pool\_good[train$PoolQC %in% c("Ex")] <- 1

train$pool\_good[!train$PoolQC %in% c("Ex")] <- 0

price <- summarize(group\_by(train, Fence),

mean(SalePrice, na.rm=T))

train$priv\_fence[train$Fence %in% c("GdPrv")] <- 1

train$priv\_fence[!train$Fence %in% c("GdPrv")] <- 0

price <- summarize(group\_by(train, MiscFeature),

mean(SalePrice, na.rm=T))

#This doesn't seem worth using at the moment. May adjust later.

price <- summarize(group\_by(train, SaleType),

mean(SalePrice, na.rm=T))

# price[order(price$`mean(SalePrice, na.rm = T)`),]

train$sale\_cat[train$SaleType %in% c("New", "Con")] <- 5

train$sale\_cat[train$SaleType %in% c("CWD", "ConLI")] <- 4

train$sale\_cat[train$SaleType %in% c("WD")] <- 3

train$sale\_cat[train$SaleType %in% c("COD", "ConLw", "ConLD")] <- 2

train$sale\_cat[train$SaleType %in% c("Oth")] <- 1

price <- summarize(group\_by(train, SaleCondition),

mean(SalePrice, na.rm=T))

# price[order(price$`mean(SalePrice, na.rm = T)`),]

train$sale\_cond[train$SaleCondition %in% c("Partial")] <- 4

train$sale\_cond[train$SaleCondition %in% c("Normal", "Alloca")] <- 3

train$sale\_cond[train$SaleCondition %in% c("Family","Abnorml")] <- 2

train$sale\_cond[train$SaleCondition %in% c("AdjLand")] <- 1

price <- summarize(group\_by(train, MSZoning),

mean(SalePrice, na.rm=T))

# price[order(price$`mean(SalePrice, na.rm = T)`),]

train$zone[train$MSZoning %in% c("FV")] <- 4

train$zone[train$MSZoning %in% c("RL")] <- 3

train$zone[train$MSZoning %in% c("RH","RM")] <- 2

train$zone[train$MSZoning %in% c("C (all)")] <- 1

price <- summarize(group\_by(train, Alley),

mean(SalePrice, na.rm=T))

# price[order(price$`mean(SalePrice, na.rm = T)`),]

train$alleypave[train$Alley %in% c("Pave")] <- 1

train$alleypave[!train$Alley %in% c("Pave")] <- 0

Done. Now, time to drop off the variables that have been made numeric and are no longer needed.

train$Street <- NULL

train$LotShape <- NULL

train$LandContour <- NULL

train$Utilities <- NULL

train$LotConfig <- NULL

train$LandSlope <- NULL

train$Neighborhood <- NULL

train$Condition1 <- NULL

train$Condition2 <- NULL

train$BldgType <- NULL

train$HouseStyle <- NULL

train$RoofStyle <- NULL

train$RoofMatl <- NULL

train$Exterior1st <- NULL

train$Exterior2nd <- NULL

train$MasVnrType <- NULL

train$ExterQual <- NULL

train$ExterCond <- NULL

train$Foundation <- NULL

train$BsmtQual <- NULL

train$BsmtCond <- NULL

train$BsmtExposure <- NULL

train$BsmtFinType1 <- NULL

train$BsmtFinType2 <- NULL

train$Heating <- NULL

train$HeatingQC <- NULL

train$CentralAir <- NULL

train$Electrical <- NULL

train$KitchenQual <- NULL

train$FireplaceQu <- NULL

train$GarageType <- NULL

train$GarageFinish <- NULL

train$GarageQual <- NULL

train$GarageCond <- NULL

train$PavedDrive <- NULL

train$Functional <- NULL

train$PoolQC <- NULL

train$Fence <- NULL

train$MiscFeature <- NULL

train$SaleType <- NULL

train$SaleCondition <- NULL

train$MSZoning <- NULL

train$Alley <- NULL

Another thing I want to do is build some interactions that may be worth looking at. For example, if the house has a pool, is it more important that it has a big deck, or something like that? I used correlation visuals like this to do it- you can choose what you'd want to put in and how many variations you want to make.

library(corrplot)

correlations <- cor(train[,c(5,6,7,8, 16:25)], use="everything")

corrplot(correlations, method="circle", type="lower", sig.level = 0.01, insig = "blank")

correlations <- cor(train[,c(5,6,7,8, 26:35)], use="everything")

corrplot(correlations, method="circle", type="lower", sig.level = 0.01, insig = "blank")

correlations <- cor(train[,c(5,6,7,8, 66:75)], use="everything")

corrplot(correlations, method="circle", type="lower", sig.level = 0.01, insig = "blank")

Anyway, the correlations that both have to do with square footage I am going to discount, because size of the total and size of a floor, for example, are obvious correlations.

pairs(~YearBuilt+OverallQual+TotalBsmtSF+GrLivArea,data=train,

main="Simple Scatterplot Matrix")

This is fun too- I picked a few of the variables that had a lot of correlation strengths. Basements have been getting bigger over time, apparently. As have the sizes of the living areas. Good to know!

I'm also interested in the relationship between sale price and some numeric variables, but these can be tougher to visualize.

library(car)

scatterplot(SalePrice ~ YearBuilt, data=train, xlab="Year Built", ylab="Sale Price", grid=FALSE)

scatterplot(SalePrice ~ YrSold, data=train, xlab="Year Sold", ylab="Sale Price", grid=FALSE)

scatterplot(SalePrice ~ X1stFlrSF, data=train, xlab="Square Footage Floor 1", ylab="Sale Price", grid=FALSE)

Prices are higher for new houses, that makes sense. Also, we can see that sale prices dropped when we would expect (thanks, housing crisis). We also have some loopy outliers on first floor square footage- probably bad data but it's not going to have a huge influence.

#Fix some NAs

train$GarageYrBlt[is.na(train$GarageYrBlt)] <- 0

train$MasVnrArea[is.na(train$MasVnrArea)] <- 0

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

#Interactions based on correlation

train$year\_qual <- train$YearBuilt\*train$OverallQual #overall condition

train$year\_r\_qual <- train$YearRemodAdd\*train$OverallQual #quality x remodel

train$qual\_bsmt <- train$OverallQual\*train$TotalBsmtSF #quality x basement size

train$livarea\_qual <- train$OverallQual\*train$GrLivArea #quality x living area

train$qual\_bath <- train$OverallQual\*train$FullBath #quality x baths

train$qual\_ext <- train$OverallQual\*train$exterior\_cond #quality x exterior

#names(train)

###Model Prepping

Then, partition! I always like to use the caret partitioning function.

outcome <- train$SalePrice

partition <- createDataPartition(y=outcome,

p=.5,

list=F)

training <- train[partition,]

testing <- train[-partition,]

###A Linear Model

Finally, we have our data and can build some models. Since our outcome is a continuous numeric variable, we want a linear model, not a GLM. First, let's just toss it all in there. I always like to use a proper regression model as my first examination of the data, to get a feel for what's there.

lm\_model\_15 <- lm(SalePrice ~ ., data=training)

summary(lm\_model\_15)

Lots of stuff we can drop right off, that's good. Some multicollinearity is making the model drop a few variables, but that's ok.

Also, our R-squared is not too bad! In case you're unfamiliar, that indicates what percent of the variation in the outcome is explained using the model we designed.

lm\_model\_15 <- lm(SalePrice ~ MSSubClass+LotArea+BsmtUnfSF+

X1stFlrSF+X2ndFlrSF+GarageCars+

WoodDeckSF+nbhd\_price\_level+

exterior\_cond+pos\_features\_1+

bsmt\_exp+kitchen+housefunction+pool\_good+sale\_cond+

qual\_ext+qual\_bsmt, data=training)

summary(lm\_model\_15)

That's our model with the important stuff, more or less. How does the RMSE turn out? That is our outcome of interest, after all.

prediction <- predict(lm\_model\_15, testing, type="response")

model\_output <- cbind(testing, prediction)

model\_output$log\_prediction <- log(model\_output$prediction)

model\_output$log\_SalePrice <- log(model\_output$SalePrice)

#Test with RMSE

rmse(model\_output$log\_SalePrice,model\_output$log\_prediction)

###A Random Forest

Not too bad, given that this is just an LM. Let's try training the model with an RF. Let's use all the variables and see how things look, since randomforest does its own feature selection.

model\_1 <- randomForest(SalePrice ~ ., data=training)

# Predict using the test set

prediction <- predict(model\_1, testing)

model\_output <- cbind(testing, prediction)

model\_output$log\_prediction <- log(model\_output$prediction)

model\_output$log\_SalePrice <- log(model\_output$SalePrice)

#Test with RMSE

rmse(model\_output$log\_SalePrice,model\_output$log\_prediction)

###An xgboost Nice! Try it with xgboost?

#Assemble and format the data

training$log\_SalePrice <- log(training$SalePrice)

testing$log\_SalePrice <- log(testing$SalePrice)

#Create matrices from the data frames

trainData<- as.matrix(training, rownames.force=NA)

testData<- as.matrix(testing, rownames.force=NA)

#Turn the matrices into sparse matrices

train2 <- as(trainData, "sparseMatrix")

test2 <- as(testData, "sparseMatrix")

#####

#colnames(train2)

#Cross Validate the model

vars <- c(2:37, 39:86) #choose the columns we want to use in the prediction matrix

trainD <- xgb.DMatrix(data = train2[,vars], label = train2[,"SalePrice"]) #Convert to xgb.DMatrix format

#Cross validate the model

cv.sparse <- xgb.cv(data = trainD,

nrounds = 600,

min\_child\_weight = 0,

max\_depth = 10,

eta = 0.02,

subsample = .7,

colsample\_bytree = .7,

booster = "gbtree",

eval\_metric = "rmse",

verbose = TRUE,

print\_every\_n = 50,

nfold = 4,

nthread = 2,

objective="reg:linear")

#Train the model

#Choose the parameters for the model

param <- list(colsample\_bytree = .7,

subsample = .7,

booster = "gbtree",

max\_depth = 10,

eta = 0.02,

eval\_metric = "rmse",

objective="reg:linear")

#Train the model using those parameters

bstSparse <-

xgb.train(params = param,

data = trainD,

nrounds = 600,

watchlist = list(train = trainD),

verbose = TRUE,

print\_every\_n = 50,

nthread = 2)

Predict and test the RMSE.

testD <- xgb.DMatrix(data = test2[,vars])

#Column names must match the inputs EXACTLY

prediction <- predict(bstSparse, testD) #Make the prediction based on the half of the training data set aside

#Put testing prediction and test dataset all together

test3 <- as.data.frame(as.matrix(test2))

prediction <- as.data.frame(as.matrix(prediction))

colnames(prediction) <- "prediction"

model\_output <- cbind(test3, prediction)

model\_output$log\_prediction <- log(model\_output$prediction)

model\_output$log\_SalePrice <- log(model\_output$SalePrice)

#Test with RMSE

rmse(model\_output$log\_SalePrice,model\_output$log\_prediction)

Nice, that's pretty good stuff. I'll take the xgboost I think, let's call that good and make up the submission. Honestly, this is where the interesting stuff basically ends, unless you want to see the retraining and submission formatting.

##Retrain on the full sample

rm(bstSparse)

#Create matrices from the data frames

retrainData<- as.matrix(train, rownames.force=NA)

#Turn the matrices into sparse matrices

retrain <- as(retrainData, "sparseMatrix")

param <- list(colsample\_bytree = .7,

subsample = .7,

booster = "gbtree",

max\_depth = 10,

eta = 0.02,

eval\_metric = "rmse",

objective="reg:linear")

retrainD <- xgb.DMatrix(data = retrain[,vars], label = retrain[,"SalePrice"])

#retrain the model using those parameters

bstSparse <-

xgb.train(params = param,

data = retrainD,

nrounds = 600,

watchlist = list(train = trainD),

verbose = TRUE,

print\_every\_n = 50,

nthread = 2)

##Prepare the prediction data

Here I just repeat the same work I did on the training set, check the code tab to see all the details.

test$paved[test$Street == "Pave"] <- 1

test$paved[test$Street != "Pave"] <- 0

test$regshape[test$LotShape == "Reg"] <- 1

test$regshape[test$LotShape != "Reg"] <- 0

test$flat[test$LandContour == "Lvl"] <- 1

test$flat[test$LandContour != "Lvl"] <- 0

test$pubutil[test$Utilities == "AllPub"] <- 1

test$pubutil[test$Utilities != "AllPub"] <- 0

test$gentle\_slope[test$LandSlope == "Gtl"] <- 1

test$gentle\_slope[test$LandSlope != "Gtl"] <- 0

test$culdesac\_fr3[test$LandSlope %in% c("CulDSac", "FR3")] <- 1

test$culdesac\_fr3[!test$LandSlope %in% c("CulDSac", "FR3")] <- 0

test$nbhd\_price\_level[test$Neighborhood %in% nbhdprice\_lo$Neighborhood] <- 1

test$nbhd\_price\_level[test$Neighborhood %in% nbhdprice\_med$Neighborhood] <- 2

test$nbhd\_price\_level[test$Neighborhood %in% nbhdprice\_hi$Neighborhood] <- 3

test$pos\_features\_1[test$Condition1 %in% c("PosA", "PosN")] <- 1

test$pos\_features\_1[!test$Condition1 %in% c("PosA", "PosN")] <- 0

test$pos\_features\_2[test$Condition1 %in% c("PosA", "PosN")] <- 1

test$pos\_features\_2[!test$Condition1 %in% c("PosA", "PosN")] <- 0

test$twnhs\_end\_or\_1fam[test$BldgType %in% c("1Fam", "TwnhsE")] <- 1

test$twnhs\_end\_or\_1fam[!test$BldgType %in% c("1Fam", "TwnhsE")] <- 0

test$house\_style\_level[test$HouseStyle %in% housestyle\_lo$HouseStyle] <- 1

test$house\_style\_level[test$HouseStyle %in% housestyle\_med$HouseStyle] <- 2

test$house\_style\_level[test$HouseStyle %in% housestyle\_hi$HouseStyle] <- 3

test$roof\_hip\_shed[test$RoofStyle %in% c("Hip", "Shed")] <- 1

test$roof\_hip\_shed[!test$RoofStyle %in% c("Hip", "Shed")] <- 0

test$roof\_matl\_hi[test$RoofMatl %in% c("Membran", "WdShake", "WdShngl")] <- 1

test$roof\_matl\_hi[!test$RoofMatl %in% c("Membran", "WdShake", "WdShngl")] <- 0

test$exterior\_1[test$Exterior1st %in% matl\_lo\_1$Exterior1st] <- 1

test$exterior\_1[test$Exterior1st %in% matl\_med\_1$Exterior1st] <- 2

test$exterior\_1[test$Exterior1st %in% matl\_hi\_1$Exterior1st] <- 3

test$exterior\_2[test$Exterior2nd %in% matl\_lo$Exterior2nd] <- 1

test$exterior\_2[test$Exterior2nd %in% matl\_med$Exterior2nd] <- 2

test$exterior\_2[test$Exterior2nd %in% matl\_hi$Exterior2nd] <- 3

test$exterior\_mason\_1[test$MasVnrType %in% c("Stone", "BrkFace") | is.na(test$MasVnrType)] <- 1

test$exterior\_mason\_1[!test$MasVnrType %in% c("Stone", "BrkFace") & !is.na(test$MasVnrType)] <- 0

test$exterior\_cond[test$ExterQual == "Ex"] <- 4

test$exterior\_cond[test$ExterQual == "Gd"] <- 3

test$exterior\_cond[test$ExterQual == "TA"] <- 2

test$exterior\_cond[test$ExterQual == "Fa"] <- 1

test$exterior\_cond2[test$ExterCond == "Ex"] <- 5

test$exterior\_cond2[test$ExterCond == "Gd"] <- 4

test$exterior\_cond2[test$ExterCond == "TA"] <- 3

test$exterior\_cond2[test$ExterCond == "Fa"] <- 2

test$exterior\_cond2[test$ExterCond == "Po"] <- 1

test$found\_concrete[test$Foundation == "PConc"] <- 1

test$found\_concrete[test$Foundation != "PConc"] <- 0

test$bsmt\_cond1[test$BsmtQual == "Ex"] <- 5

test$bsmt\_cond1[test$BsmtQual == "Gd"] <- 4

test$bsmt\_cond1[test$BsmtQual == "TA"] <- 3

test$bsmt\_cond1[test$BsmtQual == "Fa"] <- 2

test$bsmt\_cond1[is.na(test$BsmtQual)] <- 1

test$bsmt\_cond2[test$BsmtCond == "Gd"] <- 5

test$bsmt\_cond2[test$BsmtCond == "TA"] <- 4

test$bsmt\_cond2[test$BsmtCond == "Fa"] <- 3

test$bsmt\_cond2[is.na(test$BsmtCond)] <- 2

test$bsmt\_cond2[test$BsmtCond == "Po"] <- 1

test$bsmt\_exp[test$BsmtExposure == "Gd"] <- 5

test$bsmt\_exp[test$BsmtExposure == "Av"] <- 4

test$bsmt\_exp[test$BsmtExposure == "Mn"] <- 3

test$bsmt\_exp[test$BsmtExposure == "No"] <- 2

test$bsmt\_exp[is.na(test$BsmtExposure)] <- 1

test$bsmt\_fin1[test$BsmtFinType1 == "GLQ"] <- 5

test$bsmt\_fin1[test$BsmtFinType1 == "Unf"] <- 4

test$bsmt\_fin1[test$BsmtFinType1 == "ALQ"] <- 3

test$bsmt\_fin1[test$BsmtFinType1 %in% c("BLQ", "Rec", "LwQ")] <- 2

test$bsmt\_fin1[is.na(test$BsmtFinType1)] <- 1

test$bsmt\_fin2[test$BsmtFinType2 == "ALQ"] <- 6

test$bsmt\_fin2[test$BsmtFinType2 == "Unf"] <- 5

test$bsmt\_fin2[test$BsmtFinType2 == "GLQ"] <- 4

test$bsmt\_fin2[test$BsmtFinType2 %in% c("Rec", "LwQ")] <- 3

test$bsmt\_fin2[test$BsmtFinType2 == "BLQ"] <- 2

test$bsmt\_fin2[is.na(test$BsmtFinType2)] <- 1

test$gasheat[test$Heating %in% c("GasA", "GasW")] <- 1

test$gasheat[!test$Heating %in% c("GasA", "GasW")] <- 0

test$heatqual[test$HeatingQC == "Ex"] <- 5

test$heatqual[test$HeatingQC == "Gd"] <- 4

test$heatqual[test$HeatingQC == "TA"] <- 3

test$heatqual[test$HeatingQC == "Fa"] <- 2

test$heatqual[test$HeatingQC == "Po"] <- 1

test$air[test$CentralAir == "Y"] <- 1

test$air[test$CentralAir == "N"] <- 0

test$standard\_electric[test$Electrical == "SBrkr" | is.na(test$Electrical)] <- 1

test$standard\_electric[!test$Electrical == "SBrkr" & !is.na(test$Electrical)] <- 0

test$kitchen[test$KitchenQual == "Ex"] <- 4

test$kitchen[test$KitchenQual == "Gd"] <- 3

test$kitchen[test$KitchenQual == "TA"] <- 2

test$kitchen[test$KitchenQual == "Fa"] <- 1

test$fire[test$FireplaceQu == "Ex"] <- 5

test$fire[test$FireplaceQu == "Gd"] <- 4

test$fire[test$FireplaceQu == "TA"] <- 3

test$fire[test$FireplaceQu == "Fa"] <- 2

test$fire[test$FireplaceQu == "Po" | is.na(test$FireplaceQu)] <- 1

test$gar\_attach[test$GarageType %in% c("Attchd", "BuiltIn")] <- 1

test$gar\_attach[!test$GarageType %in% c("Attchd", "BuiltIn")] <- 0

test$gar\_finish[test$GarageFinish %in% c("Fin", "RFn")] <- 1

test$gar\_finish[!test$GarageFinish %in% c("Fin", "RFn")] <- 0

test$garqual[test$GarageQual == "Ex"] <- 5

test$garqual[test$GarageQual == "Gd"] <- 4

test$garqual[test$GarageQual == "TA"] <- 3

test$garqual[test$GarageQual == "Fa"] <- 2

test$garqual[test$GarageQual == "Po" | is.na(test$GarageQual)] <- 1

test$garqual2[test$GarageCond == "Ex"] <- 5

test$garqual2[test$GarageCond == "Gd"] <- 4

test$garqual2[test$GarageCond == "TA"] <- 3

test$garqual2[test$GarageCond == "Fa"] <- 2

test$garqual2[test$GarageCond == "Po" | is.na(test$GarageCond)] <- 1

test$paved\_drive[test$PavedDrive == "Y"] <- 1

test$paved\_drive[!test$PavedDrive != "Y"] <- 0

test$paved\_drive[is.na(test$paved\_drive)] <- 0

test$housefunction[test$Functional %in% c("Typ", "Mod")] <- 1

test$housefunction[!test$Functional %in% c("Typ", "Mod")] <- 0

test$pool\_good[test$PoolQC %in% c("Ex")] <- 1

test$pool\_good[!test$PoolQC %in% c("Ex")] <- 0

test$priv\_fence[test$Fence %in% c("GdPrv")] <- 1

test$priv\_fence[!test$Fence %in% c("GdPrv")] <- 0

test$sale\_cat[test$SaleType %in% c("New", "Con")] <- 5

test$sale\_cat[test$SaleType %in% c("CWD", "ConLI")] <- 4

test$sale\_cat[test$SaleType %in% c("WD")] <- 3

test$sale\_cat[test$SaleType %in% c("COD", "ConLw", "ConLD")] <- 2

test$sale\_cat[test$SaleType %in% c("Oth")] <- 1

test$sale\_cond[test$SaleCondition %in% c("Partial")] <- 4

test$sale\_cond[test$SaleCondition %in% c("Normal", "Alloca")] <- 3

test$sale\_cond[test$SaleCondition %in% c("Family","Abnorml")] <- 2

test$sale\_cond[test$SaleCondition %in% c("AdjLand")] <- 1

test$zone[test$MSZoning %in% c("FV")] <- 4

test$zone[test$MSZoning %in% c("RL")] <- 3

test$zone[test$MSZoning %in% c("RH","RM")] <- 2

test$zone[test$MSZoning %in% c("C (all)")] <- 1

test$alleypave[test$Alley %in% c("Pave")] <- 1

test$alleypave[!test$Alley %in% c("Pave")] <- 0

test$Street <- NULL

test$LotShape <- NULL

test$LandContour <- NULL

test$Utilities <- NULL

test$LotConfig <- NULL

test$LandSlope <- NULL

test$Neighborhood <- NULL

test$Condition1 <- NULL

test$Condition2 <- NULL

test$BldgType <- NULL

test$HouseStyle <- NULL

test$RoofStyle <- NULL

test$RoofMatl <- NULL

test$Exterior1st <- NULL

test$Exterior2nd <- NULL

test$MasVnrType <- NULL

test$ExterQual <- NULL

test$ExterCond <- NULL

test$Foundation <- NULL

test$BsmtQual <- NULL

test$BsmtCond <- NULL

test$BsmtExposure <- NULL

test$BsmtFinType1 <- NULL

test$BsmtFinType2 <- NULL

test$Heating <- NULL

test$HeatingQC <- NULL

test$CentralAir <- NULL

test$Electrical <- NULL

test$KitchenQual <- NULL

test$FireplaceQu <- NULL

test$GarageType <- NULL

test$GarageFinish <- NULL

test$GarageQual <- NULL

test$GarageCond <- NULL

test$PavedDrive <- NULL

test$Functional <- NULL

test$PoolQC <- NULL

test$Fence <- NULL

test$MiscFeature <- NULL

test$SaleType <- NULL

test$SaleCondition <- NULL

test$MSZoning <- NULL

test$Alley <- NULL

#Fix some NAs

test$GarageYrBlt[is.na(test$GarageYrBlt)] <- 0

test$MasVnrArea[is.na(test$MasVnrArea)] <- 0

test$LotFrontage[is.na(test$LotFrontage)] <- 0

test$BsmtFinSF1[is.na(test$BsmtFinSF1)] <- 0

test$BsmtFinSF2[is.na(test$BsmtFinSF2)] <- 0

test$BsmtUnfSF[is.na(test$BsmtUnfSF)] <- 0

test$TotalBsmtSF[is.na(test$TotalBsmtSF)] <- 0

test$BsmtFullBath[is.na(test$BsmtFullBath)] <- 0

test$BsmtHalfBath[is.na(test$BsmtHalfBath)] <- 0

test$GarageCars[is.na(test$GarageCars)] <- 0

test$GarageArea[is.na(test$GarageArea)] <- 0

test$pubutil[is.na(test$pubutil)] <- 0

#Interactions based on correlation

test$year\_qual <- test$YearBuilt\*test$OverallQual #overall condition

test$year\_r\_qual <- test$YearRemodAdd\*test$OverallQual #quality x remodel

test$qual\_bsmt <- test$OverallQual\*test$TotalBsmtSF #quality x basement size

test$livarea\_qual <- test$OverallQual\*test$GrLivArea #quality x living area

test$qual\_bath <- test$OverallQual\*test$FullBath #quality x baths

test$qual\_ext <- test$OverallQual\*test$exterior\_cond #quality x exterior

Then, format it for xgboost, I'm just using my boilerplate code for that.

# Get the supplied test data ready #

predict <- as.data.frame(test) #Get the dataset formatted as a frame for later combining

#Create matrices from the data frames

predData<- as.matrix(predict, rownames.force=NA)

#Turn the matrices into sparse matrices

predicting <- as(predData, "sparseMatrix")

Make sure your training sample and prediction sample have the same variables. I have been including this in code lately because I was making silly mistakes on variable choice.

colnames(train[,c(2:37, 39:86)])

vars <- c("MSSubClass","LotFrontage","LotArea","OverallQual","OverallCond","YearBuilt",

"YearRemodAdd","MasVnrArea","BsmtFinSF1","BsmtFinSF2","BsmtUnfSF","TotalBsmtSF" ,

"X1stFlrSF","X2ndFlrSF","LowQualFinSF","GrLivArea","BsmtFullBath","BsmtHalfBath" ,

"FullBath","HalfBath","BedroomAbvGr","KitchenAbvGr","TotRmsAbvGrd","Fireplaces" ,

"GarageYrBlt","GarageCars","GarageArea","WoodDeckSF","OpenPorchSF","EnclosedPorch" ,

"X3SsnPorch","ScreenPorch","PoolArea","MiscVal","MoSold","YrSold",

"paved","regshape","flat","pubutil","gentle\_slope","culdesac\_fr3" ,

"nbhd\_price\_level" , "pos\_features\_1","pos\_features\_2","twnhs\_end\_or\_1fam","house\_style\_level", "roof\_hip\_shed" ,

"roof\_matl\_hi","exterior\_1","exterior\_2","exterior\_mason\_1","exterior\_cond","exterior\_cond2" ,

"found\_concrete","bsmt\_cond1","bsmt\_cond2","bsmt\_exp","bsmt\_fin1","bsmt\_fin2" ,

"gasheat","heatqual","air","standard\_electric", "kitchen","fire",

"gar\_attach","gar\_finish","garqual","garqual2","paved\_drive","housefunction",

"pool\_good","priv\_fence","sale\_cat","sale\_cond","zone","alleypave",

"year\_qual","year\_r\_qual","qual\_bsmt","livarea\_qual","qual\_bath", "qual\_ext")

#colnames(predicting)

colnames(predicting[,vars])

Actually do the predicting.

#Column names must match the inputs EXACTLY

prediction <- predict(bstSparse, predicting[,vars])

prediction <- as.data.frame(as.matrix(prediction)) #Get the dataset formatted as a frame for later combining

colnames(prediction) <- "prediction"

model\_output <- cbind(predict, prediction) #Combine the prediction output with the rest of the set

sub2 <- data.frame(Id = model\_output$Id, SalePrice = model\_output$prediction)

length(model\_output$prediction)

write.csv(sub2, file = "sub3.csv", row.names = F)

head(sub2$SalePrice)