**Introduction**

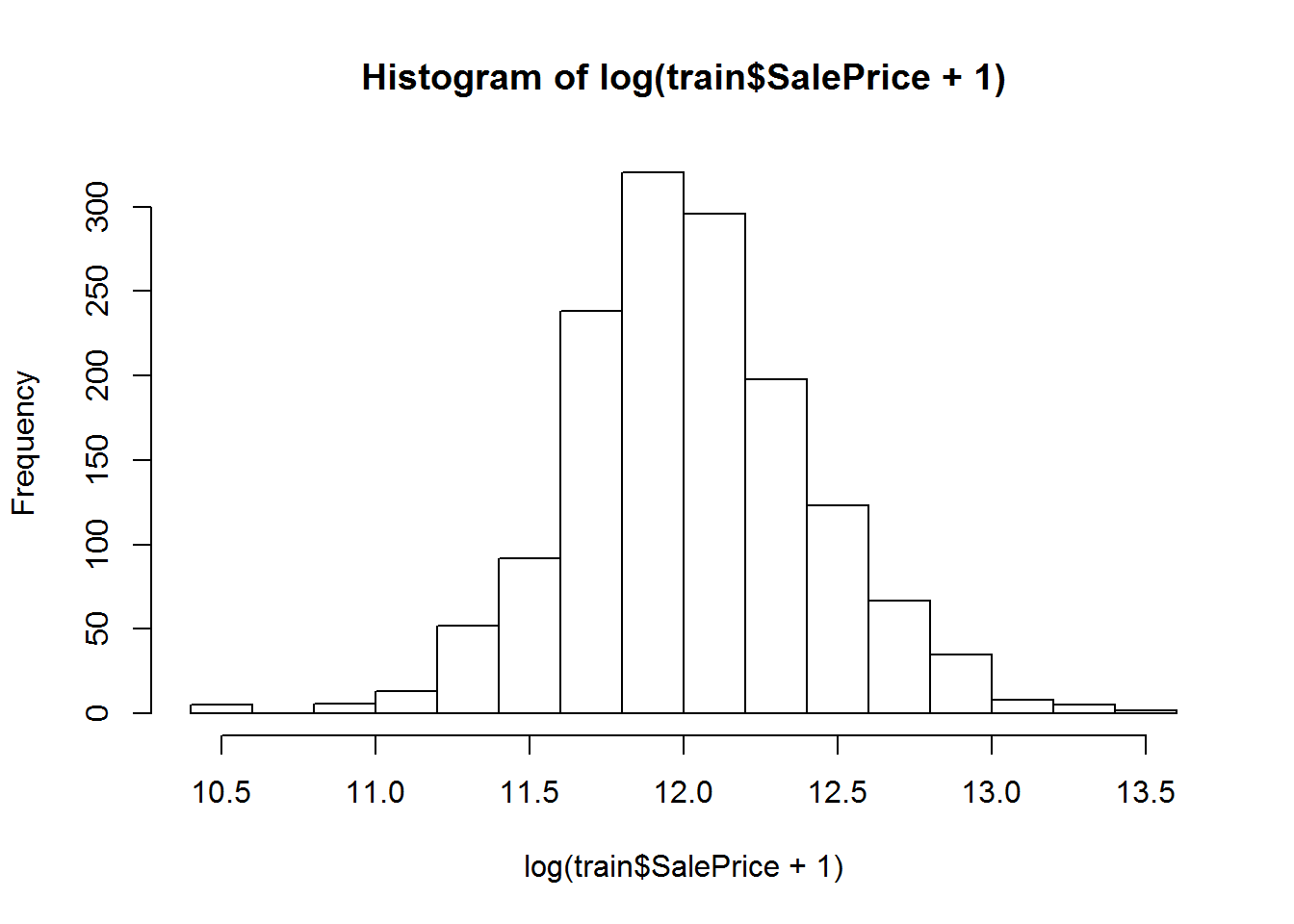
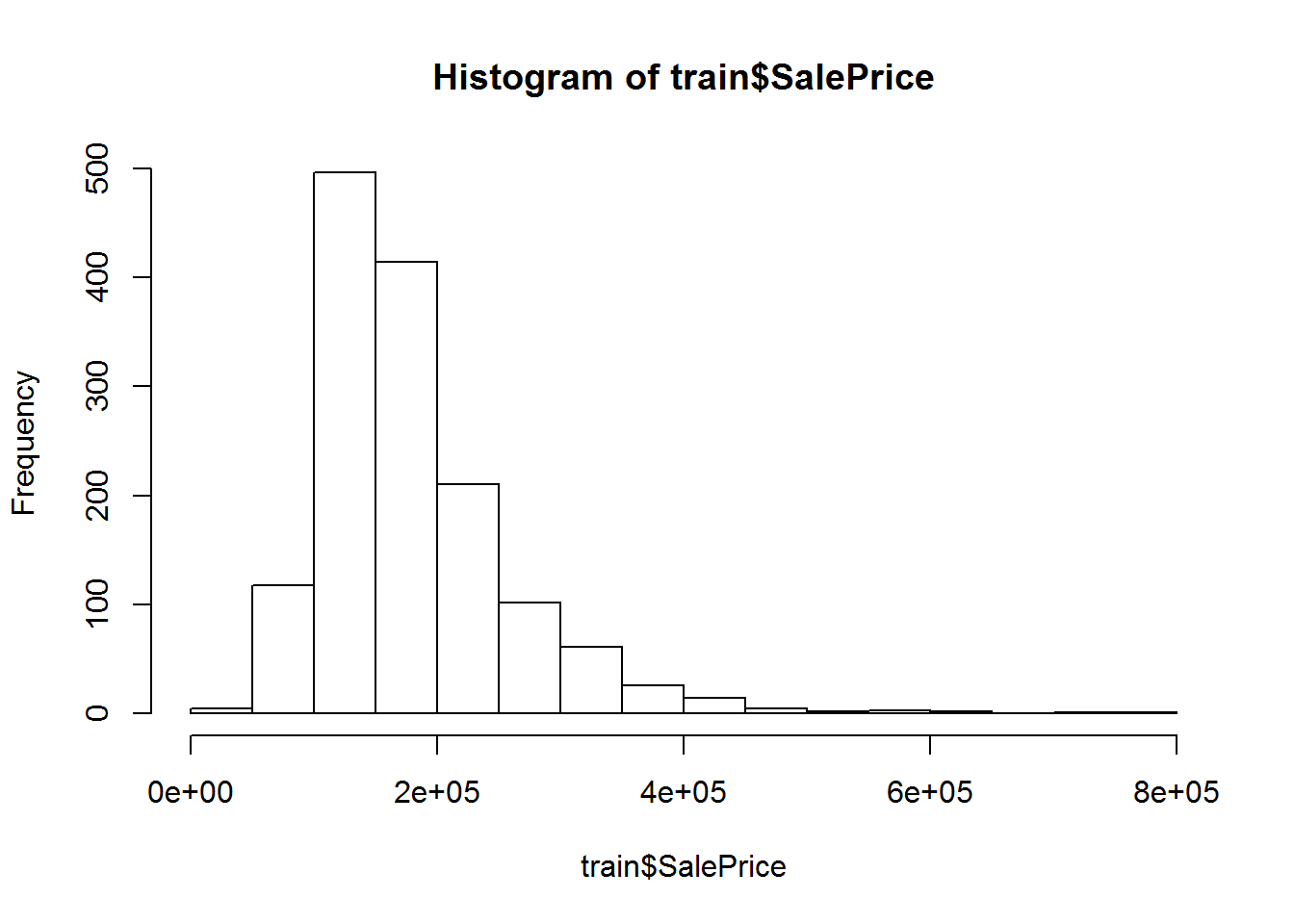
The problem we are trying to solve here is to build models to predict house prices, given the Ames Housing dataset, with high degree of predictive accuracy. The problem does not call for specific algorithms or techniques to be used. Just know that machine learning is a street brawl. The goal of the problem was to utilize any and all machine learning tools or time series forecasting techniques to make the best possible prediction of house prices. This is an interesting problem because most people will eventually buy/sell a home. This problem allows us, as data scientists, to learn more about the housing market and helps with making more informed decisions. This project encouraged getting our hands dirty to clean, transform, and engineer features that enabled better predictions. This problem also necessitated learning popular algorithms and tools. The use of Kaggle as a platform for data science competition helped motivate us to keep improving.

**Data**

The Ames Housing dataset was retrieved from <https://www.kaggle.com/c/house-prices-advanced-regression-techniques/data>. The dataset represents residential properties in Ames, Iowa from 2006 to 2010. There is a train and a test file. The train file has 1460 observations and the test file has 1459 observations. Both datasets contain 79 explanatory variables composed of 46 categorical and 33 continuous variables that describe house features such as neighborhood, square footage, number of full bathrooms, and many more. The train file contains a response variable column, SalePrice, which is what we will predict in the test set. There is also a unique ID for each house sold, but were not used in fitting the models.

**Data Set Explanation**

Below is a historgram of the SalePrice. Notice that the SalePrice is heavily skewed to the right.



A log transformation was made to normalize the variable. This would allow algorithms such as linear regression, which rely on the assumption of linear relationships, to make better predictions.

**Data Cleansing**

* It is important to clean the data with some specific rules, otherwise the precision of result can be jeopardized. After summarizing training set, it is not difficult to find that some data columns got too many missing values. We first have look on the number of missing values in every variable.
* Among 1460 variables, 'Alley', 'PoolQC', 'Fence' and 'MiscFeature' have amazingly high number of missing value. Therefore, I have decided to remove those variables. After that, the number of effective variables has shrunken to 75 (excluding id).
* Then, I transferred dummy variables into numeric form. Due to the intimidating size of dummy variables, I decided to transfer them directly by implementing 'as.integer' method. This is why I let the string as factor when reading the data file. The numeric variables are sorted out in particular for the convenience of descriptive analysis.
* Finally, for the remaining missing values, I replaced them with zero directly. The data cleansing procedure ends here.

**Descriptive Analysis**

* Exploring dataset could be difficult when the quantity of variables is quite huge. Therefore, I mainly focused on the exploration of numeric variables in this report. The descriptive analysis of dummy variables are mostly finished by drawing box plots. Some dummy variables, like 'Street', are appeared to be ineffective due to the extreme box plot. The numeric variables are sorted out before turning dummy variables into numeric form.
* We first draw a corrplot of numeric variables. Those with strong correlation with sale price are examined.'OverallQual','TotalBsmtSF','GarageCars' and 'GarageArea' have relative strong correlation with each other. Therefore, as an example, we plot the correlation among those four variables and SalePrice.
* The dependent variable (SalePrice) looks having decent linearity when plotting with other variables. However, it is also obvious that some independent variables also have linear relationship with others. The problem of multicollinearity is obvious and should be treated when the quantity of variables in regression formula is huge. The final descriptive analysis I put here would be the relationship between the variable 'YearBu' and Sale Price.
* It is not difficult to find that the price of house increases generally with the year built, the trend is obvious. The workload of data exploration is huge so I decide to end it at here. More details can be digged out by performing descriptive analysis.

**Model Selection**

* Before implementing models, one should first split the training set of data into 2 parts: a training set within the training set and a test set that can be used for evaluation.

**Model 1: Linear Regression**

* The first and simplest but useful model is linear regression model. As the first step, I put all variables into the model. R Square is not bad, but many variables do not pass the Hypothesis Testing, so the model is not perfect. Potential overfitting will occur if someone insist on using it. Therefore, the variable selection process should be involved in model construction. I prefer to use Step AIC method.
* Several variables still should not be involved in model. By checking the result of Hypothesis Test, I manually build the final linear regression model. The R Square is not bad, and all variables pass the Hypothesis Test. The diagnosis of residuals is also not bad. We check the performance of linear regression model with RMSE value.

**Model 2: LASSO Regression**

* For the avoidance of multicollinearity, implementing LASSO regression is not a bad idea. Transferring the variables into the form of matrix, we can automate the selection of variables by implementing 'lars' method in Lars package.

**Model 3: Random Forest**

* The other model I chose to fit in the training set is Random Forest model. Obviously, Random Forest may produce the best result within the training set so far.

**Model 4: XGBoost**

This amazing package really impressed me! And I have enthusiasm to explore it. The first step of XGBoost is to transform the dataset into Sparse matrix.Then I tune the parameters of xgboost model by building a 20-iteration for-loop. Not sure whether this method is reliable but really time-consuming.

**R Code**

# Load Packages

library(MASS)

library(Metrics)

library(corrplot)

library(randomForest)

library(lars)

library(ggplot2)

# Read Data

Training <- read.csv("/Users/lovepreetkaur/Desktop/HouseRent/train.csv")

Test <- read.csv("/Users/lovepreetkaur/Desktop/HouseRent/test.csv")

# Test whether data is successfully loaded

names(Training)

names(Test) #Does Not have SalesPrice(Y) values so we can't use that

#View(Training) #1460 Observations

#View(Test) #1459 Observations

summary(Training)

hist(Training$SalePrice)

hist(log(Training$SalePrice+1)) #A log transformation was made to normalize the variable. This would allow algorithms such as linear regression, which rely on the assumption of linear relationships, to make better predictions.

###########Data Cleansing######################################

Num\_NA<-sapply(Training,function(y)length(which(is.na(y)==T)))

NA\_Count<- data.frame(Item=colnames(Training),Count=Num\_NA)

Training<- Training[,-c(7,73,74,75)] #Remove 'Alley', 'PoolQC', 'Fence' and 'MiscFeature' variables with amazingly high number of missing value.

names(Training) #1460 observations

str(Training)

#View(Training)

# Numeric Variables

Num<-sapply(Training,is.numeric)

Num<-Training[,Num]

for(i in 1:77){

if(is.factor(Training[,i])){

Training[,i]<-as.integer(Training[,i])

}

}

str(Training)

Training[is.na(Training)]<-0 #Replacing missing values with 0

summary(Training)

n <- nrow(Training) #scaling all variables

n\_func <- function(n)

{

return((n - min(n))/(max(n)-min(n)))

}

Training <- as.data.frame(lapply(Training, n\_func))

summary(Training)

#####################Descriptive Analysis#######################

correlations<- cor(Num[,-1],use="everything")

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

pairs(~SalePrice+OverallQual+TotalBsmtSF+GarageCars+GarageArea,data=Training,main="Scatterplot Matrix")

p<- ggplot(Training,aes(x= YearBuilt,y=SalePrice))+geom\_point()+geom\_smooth() #It is not diffcult to find that the price of house increases generally with the year built, the trend is obvious.

###################PCA############################################

PCA <- princomp(Training[,-77], cor = FALSE, scores = TRUE) #cor=FALSE to base the principal components on the covariance matrix.

summary(PCA)

plot(PCA,type="lines")

biplot(PCA)

##############Split the data into Training and Test Set #################

Training\_Inner<- Training[1:floor(length(Training[,1])\*0.8),] #80% Training Data

Test\_Inner<- Training[(length(Training\_Inner[,1])+1):1460,] #20% Test Data

# Test

Num[is.na(Num)]<-0

str(Test)

names(Test) #80 Variables

###############Model 1: Linear Regression########################

reg1<- lm(SalePrice~., data = Training\_Inner)

summary(reg1)

#R Square is not bad, but many variables do not pass the Hypothesis Testing, so the model is not perfect. Potential overfitting will occur if someone insist on using it. Therefore,

#the variable selection process should be involved in model construction. I prefer to use Step AIC method.

reg1\_modified<-stepAIC(reg1,direction = "both")

reg1\_modified$anova

#Several variables still should not be involved in model. By checking the result of Hypothesis Test, I mannually build the final linear regression model.

reg1\_Modified\_2<-lm(formula = SalePrice ~ MSSubClass + LotArea +

Condition2 + OverallQual + OverallCond +

YearBuilt + RoofMatl + ExterQual +

BsmtQual + BsmtCond + BsmtFinSF1 + BsmtFinSF2 +

BsmtUnfSF + X1stFlrSF + X2ndFlrSF + BedroomAbvGr + KitchenAbvGr +

KitchenQual + TotRmsAbvGrd + Functional + Fireplaces + FireplaceQu +

GarageYrBlt + GarageCars + SaleCondition,

data = Training\_Inner)

summary(reg1\_Modified\_2)

#The R Square is not bad, and all variables pass the Hypothesis Test. The diagonsis of residuals is also not bad.

layout(matrix(c(1,2,3,4), 2, 2, byrow = TRUE))

plot(reg1\_Modified\_2)

par(mfrow=c(1,1)) # We check the performance of linear regression model with RMSE value.

Prediction\_1<- predict(reg1\_Modified\_2, newdata= Test\_Inner)

rmse(log(Test\_Inner$SalePrice),log(Prediction\_1))

###############Model 2: LASSO Regression###########################

#For the avoidance of multicollinearity, implementing LASSO regression is not a bad idea. Transferring the variables into the form of matrix, we can automate

#the selection of variables by implementing 'lars' method in Lars package.

Independent\_variable<- as.matrix(Training\_Inner[,1:76])

Dependent\_Variable<- as.matrix(Training\_Inner[,77])

laa<- lars(Independent\_variable,Dependent\_Variable,type = 'lasso')

plot(laa)

best\_step<- laa$df[which.min(laa$Cp)]

Prediction\_2<- predict.lars(laa,newx =as.matrix(Test\_Inner[,1:76]), s=best\_step, type= "fit")

rmse(log(Test\_Inner$SalePrice),log(Prediction\_2$fit))

#rmse(Test\_Inner$SalePrice,Prediction\_2$fit)

################# Model 3: Random Forest#########################

for\_1<- randomForest(SalePrice~.,data= Training\_Inner)

Prediction\_3 <- predict(for\_1, newdata= Test\_Inner)

rmse(log(Test\_Inner$SalePrice),log(Prediction\_3)) #Obviously, Random Forest may produce the best result within the training set. Seems I have had no other choice yet... So just implement it directly.

########## Model 4: Log Linear Regression##########################

Prediction\_1m<- predict(loglm\_1, newdata= Test\_Inner)

rmse(Test\_Inner$SalePrice,Prediction\_1m)

**Discussion & Conclusion**



Best Model so far is Log Linear Regression which gives us high R2 value and least RMSE value.