---

title: "DS 6371 Regression Project_Second Draft" output: html_document date: "2023-11-27" authors:

- "Joel Laskow"
- "Christopher Johnson"

The purpose of this document serves to walk readers through our analysis of housing data found on Kaggle

```
library(leaps)
library(ggfortify)
library(ggcorrplot)
library(base)
library(visdat)
library(tidyverse)
library(ggplot2)
library(olsrr)
library(caret)
library(dplyr)
library(corrplot)
library(ggpubr)
library(GGally)
```

Analysis 1 - Chris Johnson

Read in data and view column names

```
url = "https://raw.githubusercontent.com/cjohnson4510/Housing-Project/main/train.csv"
Htrain = read.csv(url)
colnames(Htrain)
```

Change neighborhood to factor

```
class(Htrain$Neighborhood)
Htrain$Neighborhood=as.factor(Htrain$Neighborhood)
levels(Htrain$Neighborhood)
```

Create 'Ames' Dataframe with neighborhoods of interest

```
am=grep("NAmes|Edwards|BrkSide", Htrain$Neighborhood, ignore.case = TRUE)
Ames=Htrain[am,]
Ames$Neighborhood
```

Use Naniar library to find missing values in variables of interest

```
library(naniar)
mv=miss_var_summary(Ames)
print(mv, n=100)
Ames$GrLivArea
```

No Missing values in variables of interest Create model and Plot to See if Data is normally Distributed

```
plot(Ames$GrLivArea,Ames$SalePrice)
hist(Ames$GrLivArea)
hist(Ames$SalePrice)
model=lm(SalePrice~Neighborhood+GrLivArea, Ames)
summary(model)
plot(model)
```

GrlivArea and Sale Price non-normally distributed, log transformation performed and plotted

```
logLiv=log(Ames$GrLivArea)
logSale=log(Ames$SalePrice)
hist(logLiv)
hist(logSale)
plot(logLiv, logSale)
```

Create new data frame with log transformations. Model coeffecients, confidence intervals and adj r-sqaured. Assumptions: QQ plot looks normal, residuals and leverages address below

```
logAmes=cbind(Ames,logSale, logLiv)
logModel=lm(logSale~Neighborhood+logLiv, logAmes)
summary(logModel)
confint(logModel)
plot(logModel)
```

Calculate CV Press of the model

```
set.seed(123)
k <- 5
fold_size <- nrow(logAmes) / k
cv_press <- 0
for (i in 1:k) {
  test_indices <- ((i-1) * fold_size + 1):(i * fold_size)
  test_data <- logAmes[test_indices, ]
  train_data <- logAmes[-test_indices, ]
  model <- lm(logSale ~ Neighborhood + logLiv, data = train_data)
  predictions <- predict(model, test_data)
  cv_press[i] <- cv_press + sum((test_data$logSale - predictions) ^ 2)
}
mean(cv_press)
```

To address leverages and residuals we omitted the 3 largest leverage and the 3 largest residual data points Created new model and plot for comparison Leverages and residuals looks normally distributed after datapoints omitted

```
leverages=hatvalues(logModel)
order(leverages, decreasing=TRUE)

resid=abs(resid(logModel))
order(resid, decreasing=TRUE)

outlier_remove_df=logAmes[-c(339, 136, 131, 190, 104, 186  ),]
ordfModel=lm(logSale~Neighborhood+logLiv, outlier_remove_df)
summary(ordfModel)
confint(ordfModel)
plot(ordfModel)
```

Orginal data, Plot log Model, all neighborhoods

```
library(ggplot2)
ggplot(logAmes, aes(x = logLiv, y = logSale)) +
  geom_point() +
  geom_smooth(method = "lm", se = TRUE) +
  labs(title = "Scatter Plot with Regression Lines all neighborhoods", x = "log of Living Are
  theme_minimal()
```

Omitted data, Plot log model, all neighborhoods

```
ggplot(outlier_remove_df, aes(x = logLiv, y = logSale)) +
  geom_point() +
  geom_smooth(method = "lm", se = TRUE) +
  labs(title = "Scatter Plot with Regression Lines all neighborhoods", x = "log of Living Are
  theme_minimal()
```

Original data, Plot Log Data by Neighboorhood

```
ggplot(logAmes, aes(x = logLiv, y = logSale, color=Neighborhood)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE) +
  labs(title = "Scatter Plot with Regression Lines all neighborhoods", x = "log of Living Are
  theme_minimal()
```

Omitted data, Plot Log Data by Neighboorhood

```
ggplot(outlier_remove_df, aes(x = logLiv, y = logSale, color=Neighborhood)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE) +
  labs(title = "Scatter Plot with Regression Lines all neighborhoods", x = "log of Living Are
  theme_minimal()
```

Original data, Separate plots for each Neighboorhood

```
ggplot(logAmes, aes(x = logLiv, y = logSale)) +
  geom_point() +
  geom_smooth(method = "lm", se = TRUE, aes(color = Neighborhood)) +
  labs(title = "Regression Scatterplots by Neighborhood (Outliers Included)", x = "log of Liv
  theme_minimal() +
  facet_wrap(~Neighborhood, scales = "fixed")
```

Omitted data, Separate plots for each Neighboorhood

```
ggplot(outlier_remove_df, aes(x = logLiv, y = logSale)) +
  geom_point() +
  geom_smooth(method = "lm", se = TRUE, aes(color = Neighborhood)) +
  labs(title = "Regression Scatterplots by Neighborhood (Outliers Omitted)", x = "log of Livi
  theme_minimal() +
  facet_wrap(~Neighborhood, scales = "fixed")
```

Transform logmodel back to original scale for final interpretation with confidence intervals

```
summary(logModel)
exp(confint(logModel))
```

Transform omitted model back to original scale for final interpretation with confidence intervals

The data suggest that a doubling of GrLivArea with equates to a multiplicative change of 2^0.55579 in the median of the SalePrice. A 9% confidence interval for the Brookside neighborhood multiplicative increase is (2^1.6358194, 2^1.857874) ; for Edwards, we expect (2^0.9190889, 2^1.044457); and for NAmes we expect (2^1.0785822, 2^1.209165))

```
summary(ordfModel)
exp(confint(ordfModel))
```

The outlier-omitted data suggest that a doubling of GrLivArea with equates to a multiplicative change of 2^0.56470 in the median of the SalePrice. A 95% confidence interval for the Brookside neighborhood multiplicative increase is (2^1.6485055, 2^1.876736) ; for Edwards, we expect (2^0.9134727, 2^1.030956); and for NAmes we expect (2^1.0623958, 2^1.183289))

Analysis 2 - Joel Laskow

```
# Train Dataset

train <- data.frame(read.csv("/cloud/project/DS 6371 Housing Project/train.csv", header=TRUE)


# Test Dataset

test <- read.csv("/cloud/project/DS 6371 Housing Project/test.csv", header=TRUE)

test<-data.frame(test)
```

Replace any instances of "NA" as a class level with "None" to avoid confusion

```
# train

train<- train %>%
  mutate_all(~ ifelse(. == "NA", "None", .))
```

```
# test

test<- test %>%
  mutate_all(~ ifelse(. == "NA", "None", .))
```

Assessing Numeric NA values within the datasets

```
# training set

numerictrain<-train[sapply(train,is.numeric)]
vis_miss(numerictrain)
vis_miss(train)
```

```
# test set

numerictest<-test[sapply(test,is.numeric)]
vis_miss(test)
vis_miss(numerictest)
```

We see from missing value tests that while we're only missing 0.6% of our training set and 0.6% of our test set, we're missing 18% of our LotsFrontage data in training and 16% in testing. We're further missing 6% of our GarageYrBlt data in the training set and 5% in the testing set. This quantity of missing errors could wildly skew our results.

Assessing Categorical NA values within the datasets

```
# train

nonnumerictrain<-train[sapply(train,is.character)]

vis_miss(nonnumerictrain)
```

```
# test

nonnumerictest<-test[sapply(test,is.character)]

vis_miss(nonnumerictest)
```

10.6% of our training dataset has missing values. Due to the severity of missing values in Alley (93%), FireplaceQu (50%), PoolQC (100%), Fence (80%), and MiscFeat we cannot impute missing values with the most common categorical level. For this reason we will remove these columns from the dataset.

```
# Removing Alley, Fireplace, PookQC, Fence, and MiscFeature from our dataset


# train

trainprime<-subset(train, select= -c(Alley, FireplaceQu, PoolQC, Fence, MiscFeature))


# test

testprime<-subset(test, select= -c(Alley, FireplaceQu, PoolQC, Fence, MiscFeature))
```

Addressing remaining NA values:

```
##### Numeric Datasets

# train

## Find the mean of columns to impute:

### LotsFrontage

x<-mean(train$LotFrontage, na.rm=TRUE)

### GarageYrBlt

y<-mean(train$GarageYrBlt, na.rm=TRUE)
```

```
### MasVnrArea

z<-mean(train$MasVnrArea, na.rm=TRUE)


## Impute mean for each column

### LotFrontage

trainprime$LotFrontage<-replace(train$LotFrontage, is.na(train$LotFrontage), x)

### GarageYrBlt

trainprime$GarageYrBlt<-replace(train$GarageYrBlt, is.na(train$GarageYrBlt), y)

### MasVnrArea

trainprime$MasVnrArea<-replace(train$MasVnrArea, is.na(train$MasVnrArea), z)

# test

## Find the mean of columns to impute:

### LotsFrontage

x<-mean(testprime$LotFrontage, na.rm=TRUE)

### GarageYrBlt

y<-mean(testprime$GarageYrBlt, na.rm=TRUE)

### MasVnrArea

z<-mean(testprime$MasVnrArea, na.rm=TRUE)

# BsmtFinSF1
d<-mean(testprime$BsmtFinSF1, na.rm=TRUE)

# BsmFinSF2
e<-mean(as.numeric(testprime$BsmFinSF2), na.rm=TRUE)

# BsmtUnfSF
f<-mean(testprime$BsmtUnfSF, na.rm=TRUE)
```

```
# TotalBsmtSF
g<-mean(testprime$TotalBsmtSF, na.rm=TRUE)

# BsmtFullBath
h<-mean(testprime$BsmtFullBath, na.rm=TRUE)

# BsmtHalfBath
i<-mean(testprime$BsmtHalfBath, na.rm=TRUE)

# GarageCars
j<-mean(testprime$GarageCars, na.rm=TRUE)

# GarageArea
k<-mean(testprime$GarageArea, na.rm=TRUE)

## Impute mean for each column

### LotFrontage

testprime$LotFrontage<-replace(test$LotFrontage, is.na(test$LotFrontage), x)

### GarageYrBlt

testprime$GarageYrBlt<-replace(test$GarageYrBlt, is.na(test$GarageYrBlt), y)

### MasVnrArea

testprime$MasVnrArea<-replace(test$MasVnrArea, is.na(test$MasVnrArea), z)

# BsmtFinSF1
testprime$MasVnrArea<-replace(test$BsmtFinSF1, is.na(test$MasVnrArea), d)

# BsmFinSF2
testprime$MasVnrArea<-replace(as.numeric(test$BsmFinSF2), is.na(test$MasVnrArea), e)

# BsmtUnfSF
testprime$MasVnrArea<-replace(test$BsmtUnfSF, is.na(test$MasVnrArea), f)



# TotalBsmtSF
testprime$MasVnrArea<-replace(test$TotalBsmtSF, is.na(test$MasVnrArea), g)
```

```
# BsmtFullBath
testprime$MasVnrArea<-replace(test$BsmtFullBath, is.na(test$MasVnrArea), h)


# BsmtHalfBath
testprime$MasVnrArea<-replace(test$BsmtHalfBath, is.na(test$MasVnrArea), i)


# GarageCars

testprime$MasVnrArea<-replace(test$GarageCars, is.na(test$MasVnrArea), j)

# GarageArea

testprime$MasVnrArea<-replace(test$GarageArea, is.na(test$MasVnrArea), k)

##### Categorical (Non-numeric)

# trainnew

get_mode <- function(v) {
  uniqv <- unique(v)
  uniqv[which.max(tabulate(match(v, uniqv)))]
}

for (col in names(trainprime)) {
  if (is.character(trainprime[[col]]) && anyNA(trainprime[[col]])) {
    trainprime[[col]][is.na(trainprime[[col]])] <- get_mode(trainprime[[col]])
  }
}

# testnew
get_mode <- function(v) {
  uniqv <- unique(v)
  uniqv[which.max(tabulate(match(v, uniqv)))]
}

for (col in names(testprime)) {
  if (is.character(testprime[[col]]) && anyNA(testprime[[col]])) {
    testprime[[col]][is.na(testprime[[col]])] <- get_mode(testprime[[col]])
  }
}

trainduplicate<-trainprime
testduplicate<-testprime



vis_miss(trainprime)


vis_miss(testprime)



# trainnew

get_mode <- function(v) {
  uniqv <- unique(v)
  uniqv[which.max(tabulate(match(v, uniqv)))]
}

for (col in names(trainprime)) {
  if (is.character(trainprime[[col]]) && anyNA(trainprime[[col]])) {
    trainprime[[col]][is.na(trainprime[[col]])] <- get_mode(trainprime[[col]])
  }
}

# Mark MSSubClass as factor

trainprime$MSSubClass<-factor(trainprime$MSSubClass)
testprime$MSSubClass<-factor(testprime$MSSubClass)


vis_miss(trainprime)
```

Our datasets are now clear of NA values

Unequal level between train and test data

If we compare the levels of testprime and trainprime within the MSSubClass feature, we see testprime has a level that does not appear in the training set.

```
levels(testprime$MSSubClass)
```

```
levels(trainprime$MSSubClass)
```

Level 150 appears only in the test set

```
subset(testprime, MSSubClass == "150")
```

Only one row in the test set contains MSSubClass of 150. Descriptor: 150 1-1/2 STORY PUD - ALL AGES

Because of the descriptor similarity to level 50 (50 1-1/2 STORY FINISHED ALL AGES), the level of this row was changed

```
# Before change
table(testprime$MSSubClass)

row_to_change <- which(testprime$MSSubClass == "150")

# Change the level to 50
testprime$MSSubClass[row_to_change] <- "50"

# Verify the change
table(testprime$MSSubClass)
```

If we are to build an efficient model, we must be selective with our variables. We have 3 methods to select our variables, all of which will be discussed later: Forward Selection, Backward Selection, and Stepwise Selection.

```
trainnew <- trainprime %>%
  mutate(log_SalePrice = log(SalePrice)) %>%
  mutate(log_LotFrontage = log(LotFrontage)) %>%
  mutate(log_LotArea = log(LotArea))

selected_vars <- c("log_SalePrice", "log_LotFrontage", "log_LotArea", "YearBuilt", "YearRemoc

# Calculating correlations

subset_data <- trainnew[, selected_vars]

# Create correlation plot using ggpairs
ggpairs(subset_data, upper=list (continuous="points"), title = "Correlation Matrix")
```

High correlation between LotArea and LotFrontage, YearBuilt and YearRemodAdd.

Variables to remove: LotFrontage, YearBuilt

Next Matrix Batch

```
trainnew <- trainprime %>%
  mutate(log_SalePrice = log(SalePrice)) %>%
  mutate(log_MasVnrArea = log(MasVnrArea)) %>%
  mutate(log_TotalBsmtSF = log(TotalBsmtSF))%>%
  mutate(log_X1stFlrSF = log(X1stFlrSF))%>%
  mutate(log_X2ndFlrSF = log(X2ndFlrSF))

selected_vars <- c("log_SalePrice", "log_MasVnrArea", "log_TotalBsmtSF", "log_X1stFlrSF", "lc

# Calculating correlations

subset_data <- trainnew[, selected_vars]

# Create correlation plot using ggpairs
ggpairs(subset_data, upper=list (continuous="points"), title = "Correlation Matrix")
```

High correlation between total TotalBsmntSF and 1stFlrSF. Minimal correlation between MasVnrArea and Sale Price.

Variables to remove: TotalBsmntSF, MasVnrArea

3rd Matrix Batch

```
# Deal with outliers by logging large-value columns

# NoteL LowQualFinSF ignored

# GrLivArea ignored, similar to 1stFlrSF but only accounts for living area

# BedroomAbvGr and KitchenAbvGr ignored, similar to TotRmsAbvGr


# Log transform 'SalePrice' column
numerictrainnew <- trainprime %>%
  mutate(log_SalePrice = log(SalePrice))

# Select necessary variables
selected_vars <- c("log_SalePrice", "BsmtFullBath", "BsmtHalfBath", "FullBath", "HalfBath", '

# Create a subset of data with selected variables
subset_data <- numerictrainnew[, selected_vars]

# Create correlation plot using ggpairs
ggpairs(subset_data, upper=list (continuous="points"), title = "Correlation Matrix")
```

No correlation seen between log(SalePrice) and BsmtFullBath, BsmtHalfBath, and HalfBath. Possible positive correlation observed between FullBath and TotRmsAbvGrd

Variables to remove: BsmtFullBath, BsmtHalfBath, and HalfBath

4th Matrix Batch

```
# Log transform 'SalePrice' column
numerictrainnew <- trainprime %>%
  mutate(log_SalePrice = log(SalePrice))%>% mutate(log_GarageArea = log(GarageArea)) %>% muta
```

```
# Select necessary variables

## GarageCars removed, similar to GarageArea

selected_vars <- c("log_SalePrice", "Fireplaces", "GarageYrBlt", "log_GarageArea", "log_WoodD

# Create a subset of data with selected variables
subset_data <- numerictrainnew[, selected_vars]

# Create correlation plot using ggpairs
ggpairs(subset_data, upper=list (continuous="points"), title = "Correlation Matrix")
```

High correlation between GargeYrBlt and and log_GarageArea; both show high correlation with log_SalePrice. log_GarageArea removed for simplicity.

We also see evidence of correlation between Fireplaces and log_SalePrice

Slight correlation observed between log_WoodDeckSF and log_SalePrice, and log_OpenPorchSF and log_SalePrice. Due to the weak correlation, both WoodDeckSF and OpenPorchSF will be removed.

The following variables were also ignored due to the high prevlance of 0 (i.e., "Not Applicable" or "None") within the column. There does not appear to be a sufficient quantity of values to make an informed interpretation from these variables.

- EnclosedPorch
- X3SsnPorch
- PoolArea
- MiscVal

At this point we have the following numerical variables left:

- FirePlaces
- GarageYrBlt
- FullBath

- TotRmsAbvGrd
- log(1stFlrSF)
- log(LotArea)
- YearRemodAdd

  We will move forward with the following categorical variable from the training set:

- MSSubClass

Subdividing our training set (80:20) to produce a validation and training set

```
# We will split out training dataset to train our model and test its quality
indices <- sample(1:nrow(trainprime), 0.8 * nrow(trainprime))

# Creating the training and validation sets
training_set <- trainprime[indices, ]  # 80% of data for training
validation_set <- trainprime[-indices, ]  # Remaining 20% for validation
```

All variable selection was performed in SAS using the trainprime dataset. Final CV Press and Adjusted R2 Values are shown below for each selection method.

Forward:

- No Variables Removed
- CV Press: 54.84692
- Adj. R2: 0.7691

Backward:

- TotRmsAbvGrd suggested for removal
- Candidate CV Press: 55.1482
- Compare CV Press: 55.0320
- Adj. R2: 0.7691

Stepwise:

- No variables removed

- CV Press: 55.05540
- Adj. R2: 0.7691

We will move forward with the full model (no variables removed). Removal of TotRmsAbvGrd provides miminimal change in CV Press.

Full Model:

log(SalePrice) = Fireplaces + GarageYrBlt + FullBath + TotRmsAbvGrd + log(X1stFlrSF) + log(LotArea) + YearRemodAdd + MSSubClass

Assumption checks for final model

```
fitfull<-lm(log(SalePrice) ~ Fireplaces + GarageYrBlt + FullBath + TotRmsAbvGrd + log(X1stFl


# Residuals plot

res<-resid(fitfull)

plot(fitted(fitfull), res, xlab="Fitted Values", ylab="Residual", main = "Residual vs. Fiite
abline(0,0)



# QQ plot
qqnorm(resid(fitfull))
qqline(resid(fitfull))


# Histogram of residuals
hist(resid(fitfull), xlab="Residuals", ylab="Frequency", main="Histogram of Residuals")
```

Visualizing Leverage

```
ols_plot_cooksd_bar(fitfull)
```

```
ols_plot_resid_stand(fitfull)
```

```
ols_plot_resid_lev(fitfull)
```

Testing Adjusted R Square Change with Leverage Points Removed

```
# Make new dataet with high-leverage outliers removed

trainnew_no_leverage <- trainprime[-51, -513, -859]


# Perform 80:20 split with edited dataset
train_indices <- sample(nrow(trainnew_no_leverage), 0.8 * nrow(trainnew_no_leverage))  # 80%
train_data_no_outliers <- trainnew_no_leverage[train_indices, ]
test_data_no_outliers <- trainnew_no_leverage[-train_indices, ]  # Remaining 20% for testing


# Refit the model without high leverage points

fit_no_leverage <- lm(fitfull, data = train_data_no_outliers)

fit_no_leverage

summary(fit_no_leverage)


fit_with_leverage <- lm(fitfull, data = training_set)

summary(fit_with_leverage)
```

No significant change in RSE or Adjusted R Square with removal of high-leverage outliers.

Building Predictions

Final MLR model

```
trainprime3<-trainprime
testprime3<-testprime
# Backward

fitfull <- lm(log(SalePrice) ~ Fireplaces + GarageYrBlt + FullBath + TotRmsAbvGrd + log(X1stF

prediction<-predict(fitfull, newdata=testprime3)

testprime3$logSalePrice<-prediction

testprime3$SalePrice<-exp(testprime3$logSalePrice)

houseprices3<-testprime3[c("Id", "SalePrice")]


write.csv(houseprices3,"testhouseprices_final.csv", row.names=TRUE)
```

Final Kaggle Score: 0.20089

Simple Linear Regression Model (YearBuilt)

```
# SLR


slr <- lm(log(SalePrice) ~ YearBuilt, data = trainprime)

predicted <- predict(slr, newdata = testprime)

testprime3$logSalePrice<-predicted

testprime3$SalePrice<-exp(testprime3$logSalePrice)

houseprices3<-testprime3[c("Id", "SalePrice")]


write.csv(houseprices3,"testhouseprices_SLR.csv", row.names=TRUE)
```

Explanatory variables: GrLivArea + FullBath (Custom)

```
testprime$FullBath<-as.numeric(testprime$FullBath)

custom <- lm(log(SalePrice) ~ log(GrLivArea) + FullBath, data = trainprime)

predicted <- predict(custom, newdata = testprime)

testprime3$logSalePrice<-predicted

testprime3$SalePrice<-exp(testprime3$logSalePrice)

houseprices3<-testprime3[c("Id", "SalePrice")]


write.csv(houseprices3,"testhouseprices_Custom.csv", row.names=TRUE)
```

SAS Code

/* Generated Code (IMPORT) / / Source File: trainprime.csv / / Source Path: /home/u63533968 / / Code generated on: 11/30/23, 1:47 AM */

%web_drop_table(trainprime);

FILENAME REFFILE '/home/u63533968/trainprime.csv';

PROC IMPORT DATAFILE=REFFILE DBMS=CSV OUT=trainprime; GETNAMES=YES; RUN;

PROC CONTENTS DATA=trainprime; RUN;

%web_open_table(trainprime);

data trainprime_log; set trainprime;

```
/* Log transformation for SalePrice, X1stFlrSF, and LotArea */
log_SalePrice = log(SalePrice);
log_X1stFlrSf = log(X1stFlrSf);
log_LotArea = log(LotArea);
log_GrLivArea = log(GrLivArea);
```

run;

proc glmselect data=trainprime_log; class MSSubClass; model log_SalePrice = FirePlaces GarageYrBlt FullBath TotRmsAbvGrd log_X1stFlrSf log_LotArea YearRemodAdd MSSubClass /

selection=Backward(stop=CV) cvmethod=random(10) stats=adjrsq; run;

proc glmselect data=trainprime_log; class MSSubClass; model log_SalePrice = FirePlaces GarageYrBlt FullBath TotRmsAbvGrd log_X1stFlrSf log_LotArea YearRemodAdd MSSubClass / selection=Forward(stop=CV) cvmethod=random(10) stats=adjrsq; run;

proc glmselect data=trainprime_log; class MSSubClass; model log_SalePrice = FirePlaces GarageYrBlt FullBath TotRmsAbvGrd log_X1stFlrSf log_LotArea YearRemodAdd MSSubClass / selection=Stepwise(stop=CV) cvmethod=random(10) stats=adjrsq; run;

/* SLR: YearBuilt only */

proc glmselect data=trainprime_log; class MSSubClass; model log_SalePrice = YearBuilt / selection=Backward(stop=CV) cvmethod=random(10) stats=adjrsq; run;

proc glmselect data=trainprime_log; class MSSubClass; model log_SalePrice = FirePlaces GarageYrBlt FullBath TotRmsAbvGrd log_X1stFlrSf log_LotArea YearRemodAdd MSSubClass / selection=Stepwise(stop=CV) cvmethod=random(10) stats=adjrsq; run;