MSDS 6372 Project 1

Zackary Gill, Tej Tenmattam, Limin Zheng

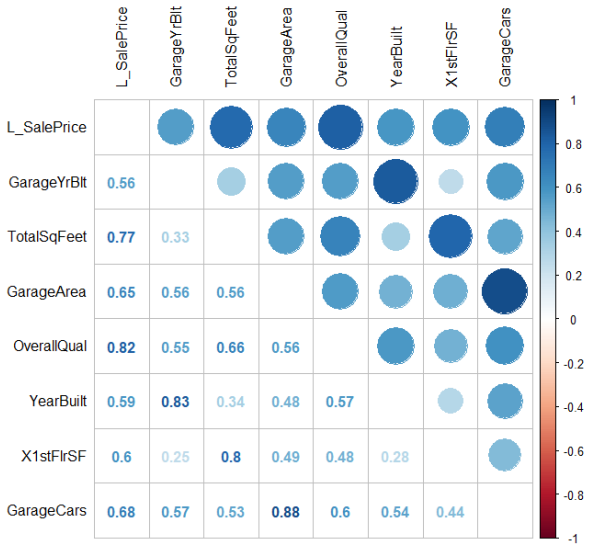
1. Introduction

With the large number of houses available on the market it is difficult to have a professional go through a home and come up with a reasonable price for a home. Our goal is to create a few models that will allow houses to be quickly and accurately priced. To accomplish this goal, we have worked on some detailed EDA and many different modeling techniques to identify an algorithm that performs better with a train/test sets RMSE-score. Two of our models will be complex with the third being easy to explain in order to allow people to quickly see what the most important things are that relate to the SalePrice of their home.

2. Data Description

This dataset is from the Ames area of Idaho and contains 1460 observations with 79 explanatory variables and one response variable called "SalePrice". Each one of these explanatory variables describes nearly every aspect of the residential homes in that area. For more information about the dataset go to Kaggle's website (https://www.kaggle.com/c/house-prices-advanced-regression-techniques).

3. Exploratory Analysis



[**CHART 3.1**]

The initial examination of the data resulted in finding approximately 19 columns have missing data. We examined each of these and fixed those with logical values (EX: with Fence being NA, it is assumed that there is no fence). Next we removed categorical columns that had problems with their levels (EX: Utilities has two levels, 1459 of the rows were of one level and the final row was of the other level). Examining a scatterplot of the variables led us to take a log transform the response variable 'SalePrice'. Removal of columns that had too much missing data and the consolidation of redundant columns were next. We examined the correlation plots **[CHART 3.1]** and removed a few highly correlated columns that were describing similar attributes. At this point the data was clean so we moved on to the model building analysis.

4. Objective 1

a. Problem Restatement

This report presents the analysis and modeling done on the House Pricing dataset provided from Kaggle for advanced regression technique. It starts by acquiring and defining input data, data cleaning, feature engineering, and finally by model training, selection, and prediction. The end result is in the creation of a model that explains/predicts the SalePrice of a home.

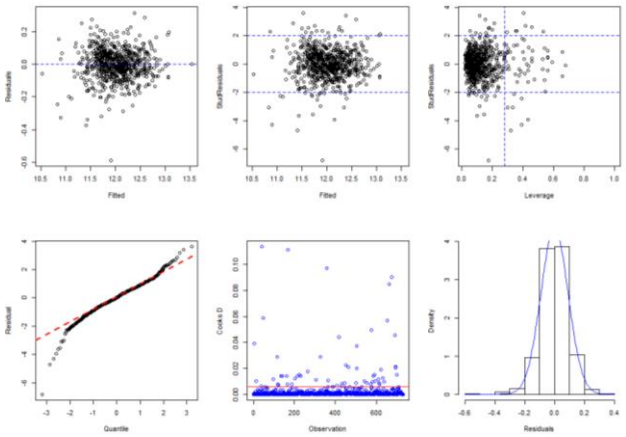
b. Build and Fit Models

During our analysis we built three different models. Two of them were built using automatic selection algorithms: stepwise and lasso. The other model was custom and was designed to be easily interpretable.

Stepwise

For the creation of the stepwise model we removed a few categorical parameters that did not have enough of certain levels to do testing upon them. Also we included the interaction terms: Neighborhood*LotArea, Neighborhood*GarageCars, TotalSqFeet*FullBath, TotalSqFeet *GarageCars, TotalSqFeet *BedroomAbvGr, and Neighborhood*OverallCond. When doing the selection we also ran into an issue with the categorical variables BsmtQual/BsmtExposure being highly correlated and GarageQual/GarageCond being highly correlated. We removed BsmtQual and GarageCond to fix this. In the end the stepwise model selected 33 of the variables and 1 interaction. Those variables are: MSZoning, LotArea, Street, LandContour, LandSlope, Neighborhood, Condition1, BldgType, OverallQual, OverallCond, YearBuilt, RoofMatl, ExterQual, Foundation, BsmtExposure, BsmtFinSF1, BsmtFinSF2, BsmtUnfSF, CentralAir, LowQualFinSF, AllFullBath, AllHalfBaths, KitchenAbvGr, KitchenQual, FireplaceQu, GarageCars, GarageQual, WoodDeckSF, OpenPorchSF, ScreenPorch, MiscVal, SaleCondition, TotalSqFeet, and one interaction term GarageCars*TotalSqFeet.

The initial model seemed to have a small problem with normality with an outlier. We ran it again without those few outliers and looked at the results. Normality and constant variance were slightly better, but the Adjusted R-Squared value was slightly worse. We decided to leave the points in and continue with our analysis.

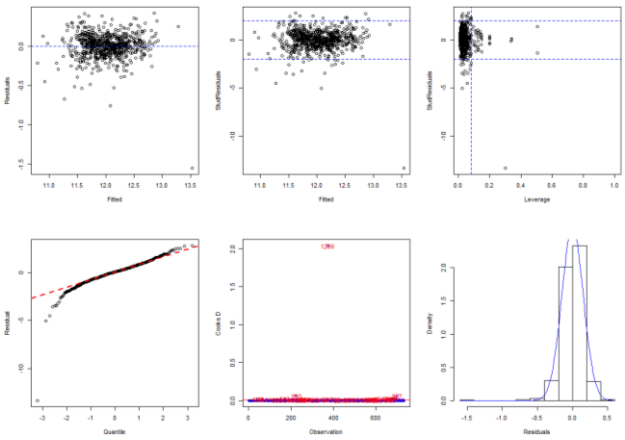


[CHART 4.b.1]

Checking the VIF's resulted in the values all being between 1.06 and 6.9 which indicated no multicollinearity. The residual plot indicates that except for a few outliers (which we decided to keep in the initial stage of analyzing this model) there is constant variance. The data is a bit non-normally distributed with a small tail at the beginning. This is fine because (as seen in the histogram) there few outliers and plenty of observations. We will assume independence, although due to the nature of house pricing this is in suspect. As mentioned earlier there are a few minor outliers. However because these outliers are minor and out of 1480 total points, their influence is low so we included them.

Custom

The custom model was designed to be simple to understand and interpret. For this model we took the variables that were the most significantly correlated with L_SalePrice and also a few sensible ones. The model consists of OverallQual, TotalSqFeet, GarageCars, AllFullBath, Neighborhood, and OverallCond. The initial model had a severely outlying value. After analyzing the row there is no error in that recording so we proceeded to do the test without it to see the result. Without the point normality and the adjusted r-squared appeared worse but Cook's D and Leverage was be better. Because this is just a single point out of 1480, its influence is low so we continued our analysis with the point included.

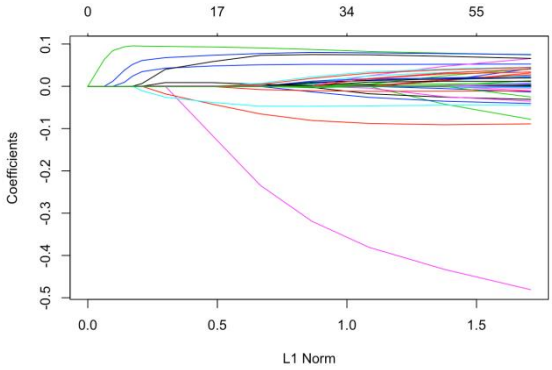


[CHART 4.b.2]

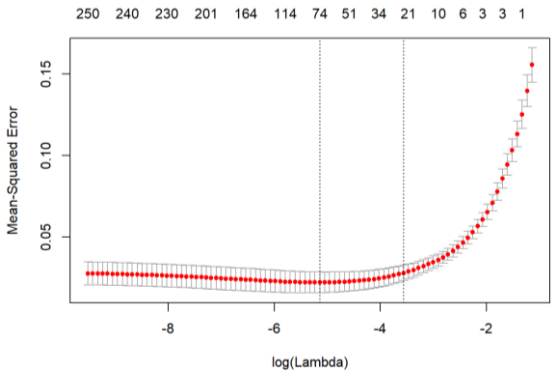
The VIF's are very low and nearly identical with their values ranging from 1.06 to 2.0. This indicates that there is no multicollinearity present. The residual plot indicates that except for a few outliers (which we decided to keep in the initial stage of analyzing this model) there is constant variance. The data is nearly normally distributed, with a bit of a tail at the beginning. This is fine because (as seen in the histogram) there few outliers. We will assume independence, although due to the nature of house pricing this is in suspect. As mentioned earlier there is one major and a few minor outliers. However these points are out of 1480 total, their influence is low so we continued our analysis with them included.

Lasso

Because we are dealing with high-dimensional data set, we have used the shrinkage model Lasso to obtain the model with the least effect of predictors variance and collinearity. Lasso penalizes the model coefficient estimates using Lambda tuning parameter to reduce their variance which results in better model fitting. Lasso's GLM-NET performs 10-fold CV to determine an optimal penalty parameter. The coefficients are easy to extract and making predictions are straight forward.

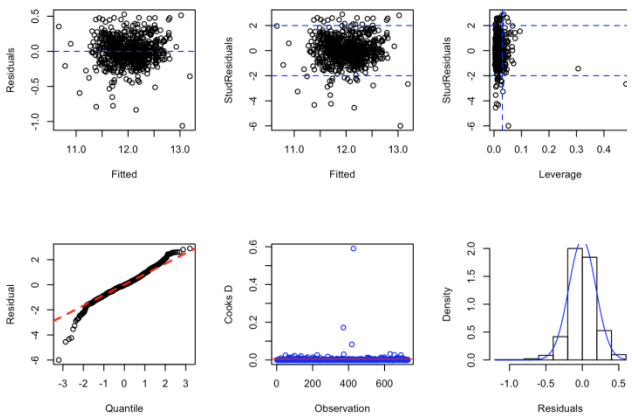


[CHART 4.b.3]



[CHART 4.b.4]

We can see from the coefficient plot **[CHART 4.b.3]** that depending on the choice of tuning parameter, some of the coefficients will be exactly equal to zero. We performed cross validation and examined the test error. The number of non-zero coefficients that the Lasso Regression picked is 56 **[CHART 4.b.4]**. From this list of coefficients we selected the top ten variables that were the most significantly correlated with L_SalePrice and ran a LM model on those. The model consists of MSSubClass, LotArea, OverallQual, OverallCond, YearBuilt, YearRemodAdd, X2ndFlrSF, BedroomAbvGr, KitchenAbvGr, GarageCars.



[CHART 4.b.5]

On this model the VIF's are very low and nearly identical (between 1.06 and 2.0), which indicates that there is no multicollinearity. The residual plot indicates that except for a few outliers (which we decided to keep in the initial stage of analyzing this model) there is constant variance. The data is nearly normally distributed, with a bit of a tail at the beginning. This is fine because (as seen in the histogram) there are few outliers. We will assume independence, although due to the nature of house pricing this is in suspect. As mentioned earlier there is one major and one minor outlier. However this is out of 1480 total rows which makes its influence low. Because of this so we continued our analysis with the points included.

c. Comparing Models

	RMSE	Adj-R2
Stepwise	36687	0.813
Lasso	40740	0.76
Custom	31489	0.86

[CHART 4.c.1]

For each of the models we generated an adjusted r-squared (Adj-R2) and the root mean squared error (RMSE) values. On the left **[CHART 4.c.1]** are those results. As you can see the Custom model performed best (RMSE: 31489). This is most likely due to our domain knowledge.

d. Parameter Interpretation

	Estimate	t value	Pr(>|t|)
(Intercept)	39757.5238	141.7791	0.0000
OverallQual	1.0833	10.9183	0.0000
TotalSqFeet	1.0001	12.1196	0.0000
GarageCars	1.0954	8.7560	0.0000
AllFullBath	1.0706	6.7110	0.0000
NeighborhoodBlueste	0.8496	-0.9844	0.3252
NeighborhoodBrDale	0.8183	-2.4459	0.0147
NeighborhoodBrkSide	0.8558	-2.3792	0.0176
NeighborhoodClearCr	1.1651	2.1893	0.0289
NeighborhoodCollgCr	1.0850	1.3893	0.1652
NeighborhoodCrawfor	1.0746	1.1191	0.2635
NeighborhoodEdwards	0.8990	-1.7550	0.0797
NeighborhoodGilbert	1.0814	1.2879	0.1982
NeighborhoodIDOTRR	0.7568	-4.1038	0.0000
NeighborhoodMeadowV	0.8088	-1.9710	0.0491
NeighborhoodMitchel	0.9599	-0.6196	0.5358
NeighborhoodNAmes	0.9572	-0.7437	0.4573
NeighborhoodNoRidge	1.2337	3.1815	0.0015
NeighborhoodNPkVill	0.8760	-1.4827	0.1386
NeighborhoodNridgHt	1.2245	3.3153	0.0010
NeighborhoodNWAmes	0.9699	-0.5017	0.6161
NeighborhoodOldTown	0.8014	-3.6244	0.0003
NeighborhoodSawyer	0.9271	-1.1791	0.2388
NeighborhoodSawyerW	1.0350	0.5584	0.5767
NeighborhoodSomerst	1.1230	1.9274	0.0543
NeighborhoodStoneBr	1.2366	2.9998	0.0028
NeighborhoodSWISU	0.8833	-1.6249	0.1046
NeighborhoodTimber	1.1565	2.2488	0.0248
NeighborhoodVeenker	1.2166	1.5870	0.1130
OverallCond	1.0582	9.5980	0.0000

[CHART 4.d.1]

	Lower	Upper
(Intercept)	34334.167768	46037.542139
OverallQual	1.067826	1.099001
TotalSqFeet	1.000108	1.000150
GarageCars	1.073252	1.118021
AllFullBath	1.049434	1.092161
NeighborhoodBlueste	0.613785	1.175973
NeighborhoodBrDale	0.696680	0.961215
NeighborhoodBrkSide	0.752574	0.973147
NeighborhoodClearCr	1.015894	1.336218
NeighborhoodCollgCr	0.966846	1.217614
NeighborhoodCrawfor	0.947152	1.219245
NeighborhoodEdwards	0.798027	1.012722
NeighborhoodGilbert	0.959800	1.218363
NeighborhoodIDOTRR	0.662386	0.864748
NeighborhoodMeadowV	0.654704	0.999181
NeighborhoodMitchel	0.843146	1.092825
NeighborhoodNAmes	0.852712	1.074425
NeighborhoodNoRidge	1.083752	1.404508
NeighborhoodNPkVill	0.735099	1.043860
NeighborhoodNridgHt	1.086106	1.380581
NeighborhoodNWAmes	0.860609	1.093096
NeighborhoodOldTown	0.710817	0.903509
NeighborhoodSawyer	0.817250	1.051659
NeighborhoodSawyerW	0.917166	1.167899
NeighborhoodSomerst	0.997840	1.263808
NeighborhoodStoneBr	1.076133	1.421016
NeighborhoodSWISU	0.760333	1.026183
NeighborhoodTimber	1.018632	1.313071
NeighborhoodVeenker	0.954556	1.550695
OverallCond	1.046045	1.070547

[CHART 4.d.2]

As seen in **[CHART 4.c.1]** the most accurate model is the Custom model. The Custom model is also the simplest and therefore it is also the one that is the easiest to interpret. The model consists of OverallQual, TotalSqFeet, GarageCars, AllFullBath, Neighborhood, and OverallCond with the response variable being log(SalePrice). The results from this model **[CHARTS 4.d.1 and 4.d.2]** have been converted back

from the log form and can be interpreted as follows. A one unit increase in TotalSqFeet is associated with a multiplicative median change of 1.01 **[CHART 4.d.1]** in the SalePrice (in essence an increase) with everything else held constant (p-value < 0.0001). The 95% confidence interval **[CHART 4.d.2]** for this median multiplicative change is between [1.0108, 1.015]. All of the other numerical parameters can be interpreted accordingly.

Neighborhood is the only categorical parameter. The intercept value is in reference to Neighborhood 'Blmngtn'. Being in the Neighborhood 'IDOTRR' is associated with a **[CHART 4.d.1]** median multiplicative change of 0.76 in SalePrice (in essence a decrease) relative to Neighborhood 'Blmngtn' (p-value < 0.0001). The 95% confidence interval **[CHART 4.d.2]** for this median multiplicative change is between [0.66, 0.84]. All of the other Neighborhoods can be interpreted accordingly.

e. Conclusion

This report elaborated on the details of how the house prices were predicted. Missing data was first imputed, a combination of Stepwise, Lasso and Custom models were used. Of the three models, custom model performed better with an RMSE of 31489. Often, ensemble technique outperforms the single best model. However in this instance, a creating a model using domain knowledge enabled our Custom model to be a better predict the SalePrice.

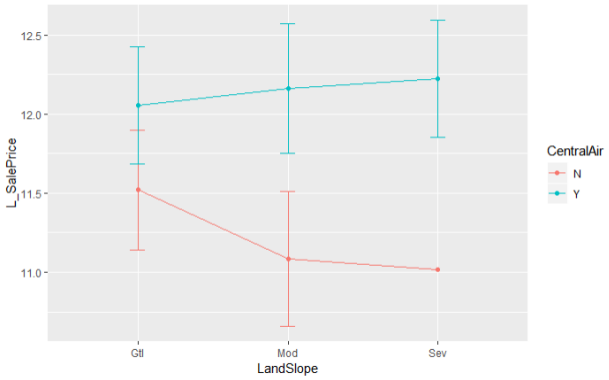
To improve upon this project and to give the automatic selection algorithms a chance to do better than a custom model, we would need to do a bit more on front end of the data analysis. There are many aspects of homes (such as having a basement) that are not required for a home to be worth more. If we had dealt with that zero inflation before fitting those parameters in our model it is likely that there would have been a few more highly significant variables. This could have allowed the automatic selection techniques model's to beat out our Custom model.

5. Objective 2

a. Goal of Two Way ANOVA

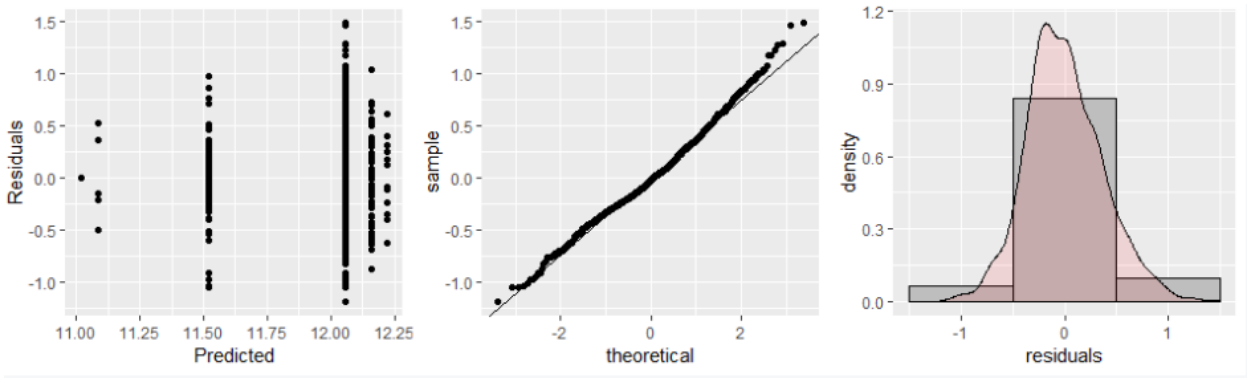
For this objective we will pretend LandSlope and CentralAir are the only two categorical variables and perform a Two Way ANOVA analysis. We will compare the mean differences between the combination of groups to understand if there is an interaction between the two independent categorical variables.

b. Analysis of Two Way ANOVA



[CHART 5.b.1]

First of all we checked if the two independent variables are interacting by using the mean profile plot **[CHART 5.b.1]** with standard deviation as the error bar. From the profile plot it is obvious that the lines are not parallel which means the model is non-additive. Therefore there is likely an interaction between LandSlope and CentralAir.



[CHART 5.b.2]

Next we created the two way ANOVA model and checked the diagnostic plots **[CHART 5.b.2]** to see if there were any violation of the assumptions. The residual plot shows the standard deviations are roughly the same. The qq-plot and histogram do not violate the assumption of normally distribution of the residual. Thus, the residual diagnostics do not provide any concern about the assumptions of a two way ANOVA analysis.

Response: L_SalePrice

	Sum Sq	Df	F value	Pr(>F)
(Intercept)	11811.2	1	85034.5039	< 2.2e-16
LandSlope	1.1	2	4.0269	0.018028
CentralAir	23.8	1	171.6115	< 2.2e-16
LandSlope:CentralAir	1.6	2	5.9186	0.002754
Residuals	202.0	1454		

[CHART 5.b.3]

Examining the type-III sums of squares F table **[CHART 5.b.3]** shows there is a statistically significant interaction between LandSlope and CentralAir (p-value = 0.0028).

Tukey multiple comparisons of means
95% family-wise confidence level

Fit: aov(formula = L_SalePrice ~ LandSlope + CentralAir + LandSlope:CentralAir, data = df.clean)

$`LandSlope:CentralAir`

	diff	lwr	upr	p adj
Mod:N-Gtl:N	-0.43416167	-0.92294203	0.0546187	0.1148034
Sev:N-Gtl:N	-0.50135665	-1.57079580	0.5680825	0.7639044
Gtl:Y-Gtl:N	0.53503509	0.41849118	0.6515790	0.0000000
Mod:Y-Gtl:N	0.63946706	0.46182232	0.8171118	0.0000000
Sev:Y-Gtl:N	0.70323591	0.37619288	1.0302789	0.0000000
Sev:N-Mod:N	-0.06719498	-1.23218029	1.0977903	0.9999833
Gtl:Y-Mod:N	0.96919676	0.49267481	1.4457187	0.0000001
Mod:Y-Mod:N	1.07362872	0.57860515	1.5686523	0.0000000
Sev:Y-Mod:N	1.13739757	0.57131646	1.7034787	0.0000002
Gtl:Y-Sev:N	1.03639174	-0.02750065	2.1002841	0.0612924
Mod:Y-Sev:N	1.14082370	0.06851676	2.2131306	0.0293765
Sev:Y-Sev:N	1.20459255	0.09768620	2.3114989	0.0237365
Mod:Y-Gtl:Y	0.10443196	-0.03601224	0.2448762	0.2764611
Sev:Y-Gtl:Y	0.16820082	-0.14022108	0.4766227	0.6276867
Sev:Y-Mod:Y	0.06376885	-0.27253344	0.4000711	0.9944605

[CHART 5.b.4]

Thirdly we adjust for multiple tests using Tukey's procedure. The Tukey method can be used to compare all combination of groups [CHART 5.b.4]. However, including all possible combinations of comparisons is too much and of little interest to the actual study. In order to compare a specific group of interest, we can use the Bonforroni adjustment. For example, we are interested to see if the SalePrice will be different with and without CentralAir in each LandSlope level. In this scenario, we only need to compare groups of "GtlY-GtlN", "ModY-ModN", "SevY-SevN". The result [CHART 5.b.5] shows that influence of CentralAir to SalePrice is significant in each specific LandSlope level, with all p-values < 0.05.

contrast	estimate	SE	df	t.ratio	p.value	bonf
GtlY-GtlN	0.535	0.0408	1454	13.100	<.0001	0.00e+00
ModY-ModN	1.074	0.1735	1454	6.189	<.0001	2.00e-09
SevY-SevN	1.205	0.3879	1454	3.105	0.0019	5.81e-03

[CHART 5.b.5]

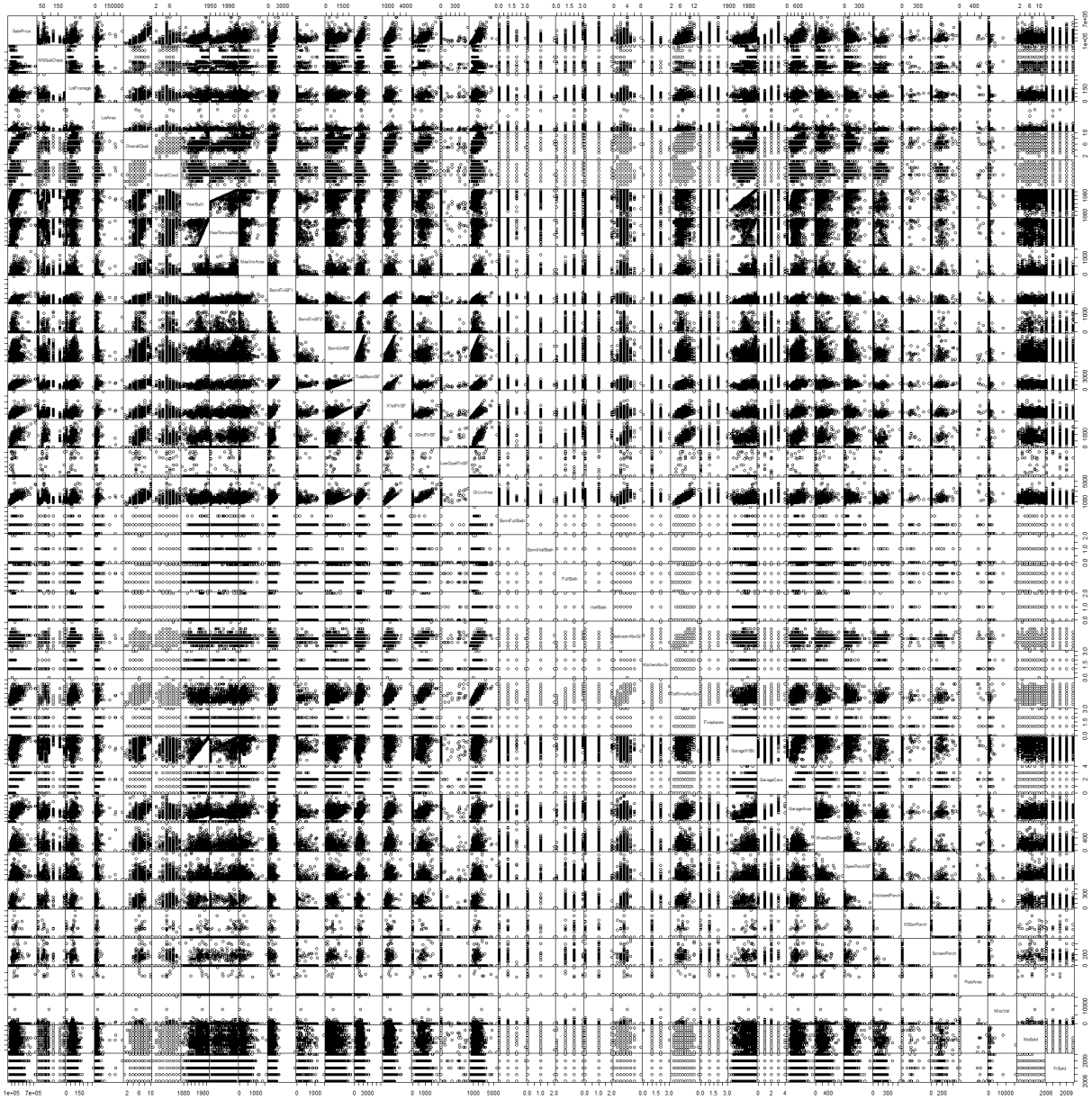
c. Conclusion/Discussion

The two way ANOVA analysis shows there is a statistically significant interaction between the two independent categorical variables CentralAir and LandSlope. The mean profile plot and the type III sum of squares F table can both exam the existing of interaction. The residual diagnostics do not provide any concern about the assumptions of a two way ANOVA analysis. Tukey adjustment can be used to perform comparison of all the combination of groups. However, we do not need to To check if the SalePrice will be different with and without CentralAir in each LandSlope level, we only need to compare groups of "GtlY-GtlN","ModY-ModN","SevY-SevN". Bonforroni adjustment can serve the purpose to compare each specific groups. The result shows that influence of CentralAir to SalePrice is significant in each specific LandSlope level.

6. Appendix

a. Charts and Figures

Pairs plot of all the columns in the model (no transformations or columns removed).



b. R Code

```
#### 1. Load and clean the Train Data:
#Read in the data---------------------------------
df.orig <- read.csv("data/AMES_train.csv", stringsAsFactors=FALSE)
#Check number of NA's
missing <- colSums(is.na(df.orig))
missing

#CLEAN THE DATA-----------------------------------
#df.clean is where we store the cleaned data
df.clean <- df.orig

#The actual cleaning of the data that has NA's
df.clean$LotFrontage[is.na(df.clean$LotFrontage)] <- 1  #1 in case we take the log
df.clean$Alley[is.na(df.clean$Alley)] <- "NoAlley"
df.clean$MasVnrType[is.na(df.clean$MasVnrType)] <- "None"
df.clean$MasVnrArea[is.na(df.clean$MasVnrArea)] <- 0
df.clean$BsmtQual[is.na(df.clean$BsmtQual)] <- "NoBsmt"
df.clean$BsmtCond[is.na(df.clean$BsmtCond)] <- "NoBsmt"
```

```
df.clean$BsmtExposure[is.na(df.clean$BsmtExposure)] <- "NoBsmt"
df.clean$BsmtFinType1[is.na(df.clean$BsmtFinType1)] <- "NoBsmt"
df.clean$BsmtFinType2[is.na(df.clean$BsmtFinType2)] <- "NoBsmt"
df.clean$Electrical[is.na(df.clean$Electrical)] <- "Unknown"
df.clean$FireplaceQu[is.na(df.clean$FireplaceQu)] <- "NoFireplace"
df.clean$GarageType[is.na(df.clean$GarageType)] <- "NoGarage"
#Sets NA's ty the average between the two dates: YearBuilt, YearRemodAdd
df.clean$GarageYrBlt <- ifelse( is.na(df.clean$GarageYrBlt),
                        round((df.clean$YearBuilt + df.clean$YearRemodAdd)/2),
                        df.clean$GarageYrBlt )
df.clean$GarageFinish[is.na(df.clean$GarageFinish)] <- "NoGarage"
df.clean$GarageQual[is.na(df.clean$GarageQual)] <- "NoGarage"
df.clean$GarageCond[is.na(df.clean$GarageCond)] <- "NoGarage"
df.clean$PoolQC[is.na(df.clean$PoolQC)] <- "NoPool"
df.clean$Fence[is.na(df.clean$Fence)] <- "NoFence"
df.clean$MiscFeature[is.na(df.clean$MiscFeature)] <- "None"

#Print out number of NA's per row to ensure no NA's
colSums(is.na(df.clean))

#Set all charcter columns to factors
charindexes <- sapply(df.clean, is.character)
df.clean[charindexes] <- lapply(df.clean[charindexes], factor)

#Remove Utilities because it is a Factor with 2 levels, and one level
#only has 1 entry (the other 1459 are the other entry)
df.clean$Utilities <- NULL

#Remove ID because its literally just the ID
df.clean$Id <- NULL

#50% of the total number of rows
length(df.orig$Id)/2

#Column names with 50% or more data missing:
#Alley, PoolQC, Fence, MiscFeature
#Remove those columns?... I did
df.clean$Alley <- NULL
df.clean$PoolQC <- NULL
df.clean$Fence <- NULL
df.clean$MiscFeature <- NULL

#Number of bathrooms in general are more important than their location
#plus the number.
df.clean$AllHalfBaths <- df.clean$BsmtHalfBath + df.clean$HalfBath
df.clean$AllFullBath <- df.clean$BsmtFullBath + df.clean$FullBath
df.clean$HalfBath <- NULL
df.clean$FullBath <- NULL
df.clean$BsmtFullBath <- NULL
df.clean$BsmtHalfBath <- NULL

#The total square footage of the house is better than SQ/FT of parts
df.clean$TotalSqFeet <- df.clean$GrLivArea + df.clean$TotalBsmtSF
df.clean$GrLivArea <- NULL
df.clean$TotalBsmtSF <- NULL

#Finds rows [A,B,123], [B,A,123] and removes them (duplicates)
rmCorDup <- function(res)
{
 rmrow <- numeric()
 len <- length(res$Var1)
 for( i in c(1:(len-1)) )
 {
   num <- i+1
   for(j in c(num:len))
   {
     if( (res[i,1] == res[j,2]) & (res[i,2] == res[j,1]) )
     {
       rmrow <- c(rmrow, j)
     }
   }
```

```
 }
 res <- res[-rmrow,]
 res
}

#### 2. EDA - Identify the important numeric predictors:
##### 2a. Transformation of data:
#Gets only the numeric values for the scatterplots
df.clean.numeric <- df.clean[, sapply(df.clean, is.numeric)]

#Transformations-------------------------------------------------------
#In plots tab Export-Image, Save as a png: 2048x2048, If you don't it is way too small to see
#pairs(df.clean.numeric[c(37, 1:36)], gap = 0)#, pch=".")

#Based on the pairs plot SalePrice should have a log transformation
df.clean.numeric$L_SalePrice <- log(df.clean.numeric$SalePrice)
df.clean$L_SalePrice <- log(df.clean$SalePrice)
#Plot again (commented out again because long run time)
#pairs(df.clean.numeric[c(38, 1:36)], gap = 0)#, pch=".")
#End Transforming------------------------------------------------------

##### 2b. Correlation Removal:
#Correlation list and plotting----------------------------------------
library(corrplot)
#Correlations of all numeric variables
#NOTE: -c(31) removes the unlogged SalePrice
df.clean.allcor <- cor(df.clean.numeric[,-c(31)], use="pairwise.complete.obs")
#The cutoff point for correlation, currently randomly assigned
corr_amt <- 0.7
#Gets a list of all correlations with higher than 'corr_amt' of correlation
df.clean.highcor <- rmCorDup(subset(as.data.frame(as.table(df.clean.allcor)), (abs(Freq) > corr_amt) & (abs(Freq) < 1)))
df.clean.highcor
#Vector of the names of the columns with high correlation
df.clean.highcor.names <- unique( c(as.vector(df.clean.highcor$Var1), as.vector(df.clean.highcor$Var2)) )

#Creates a matrix of high correlation for the graphic
df.clean.highcor.matrix <- df.clean.allcor[df.clean.highcor.names, df.clean.highcor.names]
#Creates the high correlation graphic
corrplot.mixed(df.clean.highcor.matrix, tl.col="black", tl.pos = "lt")

#Remove columns with ultra high correlation---------------------------
#The NA's in GarageYrBlt were a combo of YearBuilt and YearRemodAdd
#GarageYrBlt (0.83) YearBuilt
df.clean$GarageYrBlt <- NULL
df.clean.numeric$GarageYrBlt <- NULL

#Both of these are indicating nearly the same thing
#GarageCars is more commonly used than GarageArea
#GarageArea (0.88) GarageCars
df.clean$GarageArea <- NULL
df.clean.numeric$GarageArea <- NULL

#More space generally = more rooms
#GrLivArea (0.825) TotRmsAbvGrd
df.clean$TotRmsAbvGrd <- NULL
df.clean.numeric$TotRmsAbvGrd <- NULL
```

2c. Make correlation plots again:
#MAKE CORRELATION PLOTS AGAIN WITH LOWER CORRELATION-------------------
#NOTE: -c(30) removes the unlogged SalePrice
df.clean.allcor <- cor(df.clean.numeric[,-c(28)], use="pairwise.complete.obs")
#The cutoff point for correlation, currently randomly assigned
corr_amt <- 0.6
#Gets a list of all correlations with higher than 'corr_amt' of correlation
df.clean.highcor <- rmCorDup(subset(as.data.frame(as.table(df.clean.allcor)), (abs(Freq) > corr_amt) & (abs(Freq) < 1)))
df.clean.highcor
#Vector of the names of the columns with high correlation
df.clean.highcor.names <- unique(c(as.vector(df.clean.highcor$Var1), as.vector(df.clean.highcor$Var2)))

#Creates a matrix of high correlation for the graphic

```
df.clean.highcor.matrix <- df.clean.allcor[df.clean.highcor.names, df.clean.highcor.names]
#Creates the high correlation graphic
corrplot.mixed(df.clean.highcor.matrix, tl.col="black", tl.pos = "lt")

#All pairs after removal of all heavily correlated ones
#pairs(df.clean.numeric[,c(length(df.clean.numeric), 1:(length(df.clean.numeric)-2))], gap = 0)#, pch=".")

#Function to print out the plots for the multiple linear regression
mlrplots <- function(fit, hidenum = TRUE)
{
 #library(MASS)
 sres <- rstudent(fit)
 res <- resid(fit)
 leverage <- hatvalues(fit)

 par(mfrow=c(2,3))

 #Plot residuals
 plot(fitted(fit), res, xlab = "Fitted", ylab = "Residuals")
 abline(h=0, col="blue", lty=2)

 #Plot studentized residuals
 plot(fitted(fit), sres, xlab = "Fitted", ylab = "StudResiduals")
 abline(h=-2, col="blue", lty=2)
 abline(h=2, col="blue", lty=2)
 if(!hidenum)
  text(sres~fitted(fit), y=sres, labels=ifelse( abs(sres) >= 2, names(sres),""), col="red")

 #Plot Leverage - examine any observations ~2-3 times greater than the average hat value
 plot(x = leverage, y = sres, xlab = "Leverage", ylab = "StudResiduals")
 abline(h=-2, col="blue", lty=2)
 abline(h=2, col="blue", lty=2)
 abline(v = mean(leverage)*2, col="blue", lty=2) #line is at 2x mean

 #QQ Plot
 qqnorm(sres, xlab="Quantile", ylab="Residual", main = NULL)
 qqline(sres, col = 2, lwd = 2, lty = 2)

 #Cooks D
 cooksd <- cooks.distance(fit)
 sample_size <- length(fit$model[,1])
 plot(cooksd, xlab = "Observation", ylab = "Cooks D", col = c("blue"))
 abline(h = 4/sample_size, col="red")  # add cutoff line
 if(!hidenum)
  text(x=1:length(cooksd)+1, y=cooksd, labels=ifelse(cooksd>4/sample_size, names(cooksd),""), col="red")  # add labels

 #Histogram of residuals with normal curve
 #If the curve looks wrong try using the studentized residuals
 hist(res, freq=FALSE, xlab = "Residuals", main = NULL)
 curve(dnorm(x, mean=mean(res), sd=sd(res)), add=TRUE, col = "blue")
}

#Makes the VIF plot for the LM models
makevif <- function(df, yresp)
{
 library(mctest)
 imcdiag(x = as.matrix(df[, sapply(df, is.numeric)]), y=yresp, method="VIF")
}

#Adjusted r-squared
adjr <- function(rval, n, p)
{
 rr <- 1-(1-rval*rval)*((n-1)/(n-p-1))
 rr
}

### Creates the test and training sets
library(MASS)
library(Metrics) # RMSE
library(glmnet)  # Package to fit ridge/lasso/elastic net models
```

```
set.seed(123)
df.clean.smp_size <- floor(0.5 * nrow(df.clean))
df.clean.train_ind <- sample(seq_len(nrow(df.clean)), size = df.clean.smp_size)
df.clean.train <- df.clean[df.clean.train_ind, ]
df.clean.test <- df.clean[-df.clean.train_ind, ]

### Stepwise
#Stepwise----------------------------------------
#Fit for Stepwise
#REMOVED HeatingQC, Exterior1st, Functional, Heating, Electrical because they didn't have enough of each level
# + X1stFlrSF + X2ndFlrSF, and TotalSqFeet are similar so the first two were removed
df.clean.train.fit <- lm(L_SalePrice ~ MSSubClass + MSZoning + LotFrontage + LotArea + Street + LotShape + LandContour + LotConfig + LandSlope + Neighborhood + Condition1 + Condition2 + BldgType + HouseStyle + OverallQual + OverallCond + YearBuilt + YearRemodAdd + RoofStyle + RoofMatl + Exterior2nd + MasVnrType + MasVnrArea + ExterQual + ExterCond + Foundation + BsmtCond + BsmtExposure + BsmtFinType1 + BsmtFinSF1 + BsmtFinType2 + BsmtFinSF2 + BsmtUnfSF + CentralAir + LowQualFinSF + AllFullBath + AllHalfBaths + BedroomAbvGr + KitchenAbvGr + KitchenQual + Fireplaces + FireplaceQu + GarageType + GarageFinish + GarageCars + GarageQual + PavedDrive + WoodDeckSF + OpenPorchSF + EnclosedPorch + X3SsnPorch + ScreenPorch + PoolArea + MiscVal + MoSold + YrSold + SaleType + SaleCondition + TotalSqFeet + TotalSqFeet*GarageCars, data = df.clean.train)
#[!(row.names(df.clean.train) %in% c("633", "94", "534", "689", "589")),]
# + BsmtQual + GarageCond

#Stepwise call
df.clean.train.step <- stepAIC(df.clean.train.fit, direction="both", trace=FALSE)
#Summary of stepwise with p-values
summary(df.clean.train.step)
#The VIF's for the data (only numeric), the -c(1) removes the y component (L_SalePrice)
df.clean.train.step.vif <- makevif(df.clean.train.step$model[,-c(1)], df.clean.train.step$model$L_SalePrice)
df.clean.train.step.vif
#Overall AIC for the model
df.clean.train.step.aic <- df.clean.train.step$anova$AIC[ length(df.clean.train.step$anova$AIC) ]
#Plots for the model
mlrplots(df.clean.train.step)

for(i in c(2:length(df.clean.train.step$model) ) )
{
  plot(df.clean.train.step$model[,i], df.clean.train.step$residuals, xlab = names(df.clean.train.step$model)[i])
}

#Prediction
df.clean.pred.step <- predict(df.clean.train.step, df.clean.test, type="response")
#RMSE
rmse(df.clean.test$SalePrice, exp(df.clean.pred.step))
#Adjusted r squared on test
adjr(cor(df.clean.test$SalePrice, exp(df.clean.pred.step)), length(df.clean.train$L_SalePrice), length(names(df.clean.train.step$model)))
#Plot of the predicted vs actual
plot(exp(df.clean.pred.step), df.clean.test$SalePrice)

### Custom Model
#Custom Model------------------------------------------------------------
#df.clean.train.cust <- lm(L_SalePrice ~ LotArea + Neighborhood + OverallQual + OverallCond + YearBuilt + GrLivArea + GarageCars + YearRemodAdd + BsmtFinSF1, data = df.clean.train) #to remove a row: df.clean.train[row.names(df.clean.train) != "1299",]
df.clean.train.cust <- lm(L_SalePrice ~ OverallQual + TotalSqFeet + GarageCars + AllFullBath + Neighborhood + OverallCond, data = df.clean.train)
summary(df.clean.train.cust)
#The VIF's for the data (only numeric), the -c(1) removes the y component (L_SalePrice)
df.clean.train.cust.vif <- makevif(df.clean.train.cust$model[,-c(1)], df.clean.test$L_SalePrice)
df.clean.train.cust.vif
#Plots for the model
mlrplots(df.clean.train.cust)

#pairs(df.clean.train[-c(1299, 890, 327),c(73, 4, 10, 15, 16, 17, 32, 44, 57, 18)])

for(i in c(2:length(df.clean.train.cust$model) ) )
{
  plot(df.clean.train.cust$model[,i], df.clean.train.cust$residuals, xlab = names(df.clean.train.cust$model)[i])
}
#  plot(df.clean.train.cust$residuals, df.clean.train.cust$model$AllFullBath)

#Predicted Values
df.clean.pred.cust <- predict(df.clean.train.cust, df.clean.test, type="response")
#RMSE
```

```
rmse(df.clean.test$SalePrice, exp(df.clean.pred.cust))
#Adjusted r squared on test
adjr(cor(df.clean.test$SalePrice, exp(df.clean.pred.cust)), length(df.clean.train$L_SalePrice), length(names(df.clean.train.cust$model)))
#Plot of the predicted vs actual
plot(exp(df.clean.pred.cust), df.clean.test$SalePrice)

### LASSO Model
library(glmnet)

df.clean.numeric <- df.clean[, sapply(df.clean, is.numeric)]
df.clean.factors <- df.clean[, !(names(df.clean) %in% names(df.clean.numeric))]

DFdummies <- as.data.frame(model.matrix(~.-1, df.clean.factors))
dim(DFdummies)

#Also taking out variables with less than 10 'ones' in the train set.
fewOnes <- which(colSums(DFdummies[1:nrow(df.clean),]) < 10)
#Removing predictors with < 10
DFdummies <- DFdummies[,-fewOnes]
#combining all (now numeric) predictors into one dataframe
df.clean.combined <- cbind(df.clean.numeric, DFdummies)

set.seed(123)
df.clean.smp_size <- floor(0.5 * nrow(df.clean.combined))
df.clean.train_ind <- sample(seq_len(nrow(df.clean)), size = df.clean.smp_size)
df.clean.combined.train <- df.clean.combined[df.clean.train_ind, ]
df.clean.combined.test <- df.clean.combined[-df.clean.train_ind, ]

#--------------------------------------------------------------------------------

#Formatting data for GLM net
x = model.matrix(df.clean.combined.train$L_SalePrice ~ ., data = df.clean.combined.train[, -which(names(df.clean.combined.train) %in% c("SalePrice","L_SalePrice"))])
y = df.clean.combined.train$L_SalePrice

xtest = model.matrix(df.clean.combined.test$L_SalePrice ~ ., data = df.clean.combined.test[, -which(names(df.clean.combined.test) %in% c("SalePrice","L_SalePrice"))])
ytest <- df.clean.combined.test$L_SalePrice

#creating a huge range for the penalty parameter
grid=10^seq(10,-2, length =100)

lasso.mod=glmnet(x,y,alpha=1, lambda=grid)
plot(lasso.mod)
#We can see from the coefficient plot that depending on the choice of tuning parameter, some of the coefficients will be exactly equal to zero.
Let’s perform cross validation and see the test error.
set.seed(123)
#alpha=1 performs LASSO
cv.out=cv.glmnet(x,y,alpha=1)
plot(cv.out)
#Optimal penalty parameter.  You can make this call visually.
bestlambda <- cv.out$lambda.min
lasso.pred = predict(lasso.mod, s=bestlambda, newx=xtest)
lasso.rmse <- sqrt(mean((lasso.pred-ytest)^2))
lasso.rmse
out=glmnet(x,y,alpha=1,lambda=grid)
lasso.coef=predict(out,type="coefficients",s=bestlambda)[1:20,]
lasso.coef
lasso.coef[lasso.coef!=0]

df.clean.train.lasso <- lm(L_SalePrice ~ MSSubClass + LotArea + OverallQual + OverallCond + YearBuilt + YearRemodAdd + X2ndFlrSF + BedroomAbvGr + KitchenAbvGr + GarageCars , data = df.clean.train)
summary(df.clean.train.lasso)

#The VIF's for the data (only numeric), the -c(1) removes the y component (L_SalePrice)
df.clean.train.lasso.vif <- makevif(df.clean.train.lasso$model[,-c(1)], df.clean.test$L_SalePrice)
df.clean.train.cust.vif
#Plots for the model
mlrplots(df.clean.train.lasso)

#pairs(df.clean.train[-c(1299, 890, 327),c(73, 4, 10, 15, 16, 17, 32, 44, 57, 18)])
```

```
for(i in c(2:length(df.clean.train.lasso$model) ) )
{
  plot(df.clean.train.lasso$model[,i], df.clean.train.lasso$residuals, xlab = names(df.clean.train.lasso$model)[i])
}

#Predicted Values
df.clean.pred.lasso <- predict(df.clean.train.lasso, df.clean.test, type="response")

#RMSE
rmse(df.clean.test$SalePrice, exp(df.clean.pred.lasso))
#Adjusted r squared on test
adjr(cor(df.clean.test$SalePrice, exp(df.clean.pred.lasso)), length(df.clean.train$L_SalePrice), length(names(df.clean.train.lasso$model)))
#Plot of the predicted vs actual
plot(exp(df.clean.pred.lasso), df.clean.test$SalePrice)

###Two Way ANOVA

mysummary<-function(x){
  result<-c(length(x),mean(x),sd(x))
  names(result)<-c("N","Mean","SD")
  return(result)
}
sumstats<-aggregate(L_SalePrice~LandSlope*CentralAir,data=df.clean,mysummary)
sumstats<-cbind(sumstats[,1:2],sumstats[,-(1:2)])
sumstats
library(ggplot2)
ggplot(sumstats,aes(x=LandSlope,y=Mean,group=CentralAir,colour=CentralAir))+
  ylab("L_SalePrice")+
  geom_line()+
  geom_point()+
  geom_errorbar(aes(ymin=Mean-SD,ymax=Mean+SD),width=.1)

#Fit the two way ANOVA model
model.aov<-aov(L_SalePrice~LandSlope+CentralAir+LandSlope:CentralAir,data=df.clean)

#check the residual diagnostic plots to see if there is any violation of the assumptions.
library(gridExtra)
library(ggplot2)
myfits<-data.frame(fitted.values=model.aov$fitted.values,residuals=model.aov$residuals)

#Residual vs Fitted
plot1<-ggplot(myfits,aes(x=fitted.values,y=residuals))+ylab("Residuals")+
  xlab("Predicted")+geom_point()

#QQ plot of residuals  #Note the diagonal abline is only good for qqplots of normal data.
plot2<-ggplot(myfits,aes(sample=residuals))+
  stat_qq()+geom_abline(intercept=mean(myfits$residuals), slope = sd(myfits$residuals))

#Histogram of residuals
plot3<-ggplot(myfits, aes(x=residuals)) +
  geom_histogram(aes(y=..density..),binwidth=1,color="black", fill="gray")+
  geom_density(alpha=.1, fill="red")

grid.arrange(plot1, plot2,plot3, ncol=3)

#Examining the type-III sums of squares F table
library(car)
Anova(model.aov,type=3)

#Tukey adjustment for comparison of all combination of groups
TukeyHSD(model.aov,"LandSlope:CentralAir",conf.level=.95)
plot(TukeyHSD(model.aov,"LandSlope:CentralAir",conf.level=.95))

library(lsmeans)
#Define contrast.factor and mycontrast objects.
contrast.factor<-~LandSlope*CentralAir
mycontrast<-c("GtlY-GtlN","ModY-ModN","SevY-SevN")
dat<-df.clean

#Running a loop that determines the appropriate 0's and 1's for each
```

```
#contrast specified above.
library(limma)
final.result<-c()
for( j in 1:length(mycontrast)){
  contrast.factor.names<-gsub(" ", "", unlist(strsplit(as.character(contrast.factor),split = "*", fixed = T))[-1])
  contrast.factor.2 <- vector("list", length(contrast.factor.names))
  for (i in 1:length(contrast.factor.names)) {
    contrast.factor.2[[i]] <- levels(dat[, contrast.factor.names[i]])
  }
  new.factor.levels <- do.call(paste, c(do.call(expand.grid,
                                    contrast.factor.2), sep = ""))
  temp.cont<-mycontrast[j]
  contrast2 <- list(comparison = as.vector(do.call(makeContrasts,
                                     list(contrasts = temp.cont, levels = new.factor.levels))))

  contrast.result <- summary(contrast(lsmeans(model.aov,
                                    contrast.factor), contrast2, by = NULL))

  final.result<-rbind(final.result,contrast.result)
}
#Cleaning up and applying bonferroni correction to the number
#of total comparisons investigated.
final.result$contrast<-mycontrast
final.result$bonf<-length(mycontrast)*final.result$p.value
final.result$bonf[final.result$bonf>1]<-1
final.result
```