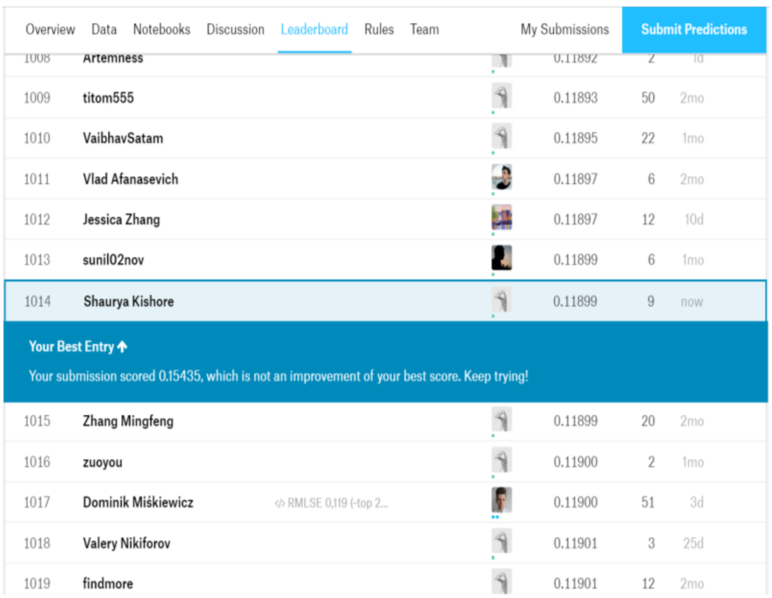

Overview Data Notebooks Discussion Leaderboard Rules Team My Submissions Submit Predictions

1008	Artemness	0.11892	2	1d
1009	titom555	0.11893	50	2mo
1010	VaibhavSatam	0.11895	22	1mo
1011	Vlad Afanasevich	0.11897	6	2mo
1012	Jessica Zhang	0.11897	12	10d
1013	sunil02nov	0.11899	6	1mo
1014	Shaurya Kishore	0.11899	9	now

Your Best Entry ↑
Your submission scored 0.15435, which is not an improvement of your best score. Keep trying!

1015	Zhang Mingfeng	0.11899	20	2mo
1016	zuoyou	0.11900	2	1mo
1017	Dominik Miśkiewicz (RMLSE 0,119 (-top 2...)	0.11900	51	3d
1018	Valery Nikiforov	0.11901	3	25d
1019	findmore	0.11901	12	2mo

MMA 867

Assignment - 2

ABSTRACT

The final position at the time of submission is 1014 which was achieved by using LASSO regression

Shaurya Kishore

Contents

1 Introduction: 2
2 Data Description and Pre-Processing: 2
2.1 Missing Value Analysis: 2
2.2 Data Transformation and Outlier Analysis: 3
2.3 Feature Engineering: 4
3 Modeling: 5
3.1 LASSO: 5
3.2 Ridge Regression: 6
4 Conclusion: 6

Table of Figures

Figure 1: Correlation 2
Figure 2: Missing values 3
Figure 3: Outliers in dataset 3
Figure 4: Normality error in SalePrice 4
Figure 5: Skewness 4
Figure 6: Dummy variables created 4
Figure 7: Lasso Regression 5
Figure 8: Ridge Regression 6

1 INTRODUCTION:

Buying a house remains one of the biggest decisions that individuals can make in their lifetime. With the data at hand, the goal is to predict the final price of each house in Ames, Iowa, by using every aspect of a residential home. The insights through this model could be used by every person to aid their decision-making process when purchasing a home. The dataset mostly contains categorical variables stores as factors or integers. We will be focusing on advance regression techniques such as LASSO and Ridge to create our model and accurately predict the house prices in Iowa.

2 DATA DESCRIPTION AND PRE-PROCESSING:

The training dataset consisted of 1460 observations and 81 features whereas the test dataset consisted of 1459 observations and 80 features. Before proceeding with pre-processing and cleaning of the dataset, basic analysis was performed to understand the relationship between different variables. The correlation between SalePrice and rest of the variables was seen to understand which variable had the most impact on Sales.

```
> cor
             Var1      Var2      Freq
1444    SalePrice SalePrice 1.0000000
1411  OverallQual SalePrice 0.7909816
1423    GrLivArea SalePrice 0.7086245
1433   GarageCars SalePrice 0.6404092
1434   GarageArea SalePrice 0.6234314
1419  TotalBsmtSF SalePrice 0.6135806
1420    X1stFlrSF SalePrice 0.6058522
1426     FullBath SalePrice 0.5606638
1430 TotRmsAbvGrd SalePrice 0.5337232
1413    YearBuilt SalePrice 0.5228973
```

Figure 1: Correlation

Overall quality had the highest correlation, and this shows that in our model we should be mindful of these variables.

2.1 MISSING VALUE ANALYSIS:

The first step in pre-processing the dataset was to perform a missing value analysis. The percentage of missing data for the variable with NA's is given below:

```
> as.data.frame(sum.na.percent)
             sum.na.percent
PoolQC         0.9952054795
MiscFeature    0.9630136986
Alley          0.9376712329
Fence          0.8075342466
FireplaceQu    0.4726027397
LotFrontage    0.1773972603
GarageType     0.0554794521
GarageYrBlt    0.0554794521
GarageFinish   0.0554794521
GarageQual     0.0554794521
GarageCond     0.0554794521
BsmtExposure   0.0260273973
BsmtFinType2   0.0260273973
BsmtQual       0.0253424658
BsmtCond       0.0253424658
BsmtFinType1   0.0253424658
MasVnrType     0.0054794521
MasVnrArea     0.0054794521
Electrical     0.0006849315
```

Figure 2: Missing values

The variables which had more than 90% of missing data were removed from the dataset. It was assumed that the remaining features have data missing completely at random and thus multiple imputation was performed to fill the remaining missing values.

```
#imputation with mean
imp <- mice(train, m=1, method="cart")
train_comp <- mice::complete(imp)
as.data.frame(colSums(is.na(train_comp)))
```

Cart method was used for multiple imputation to deal with the categorical variables in our dataset.

2.2 Data Transformation and Outlier Analysis:

In order to make the model efficient, it was important to remove outliers from the features which had the highest correlation with SalePrice.

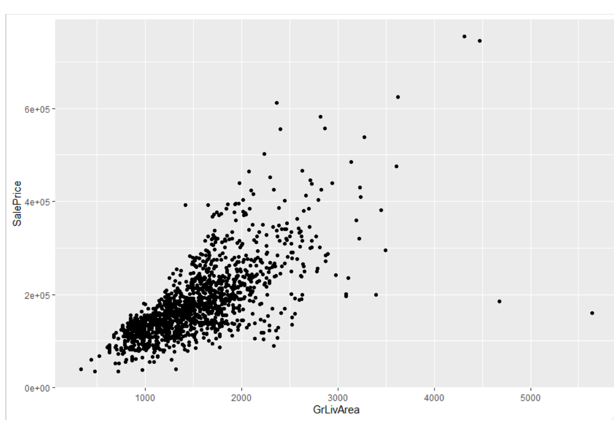

Figure 3: Outliers in dataset

Two outliers in GrLivArea were found and removed.

Once outliers were removed, the next step was data transformation. It was found that SalePrice is not normally distributed:

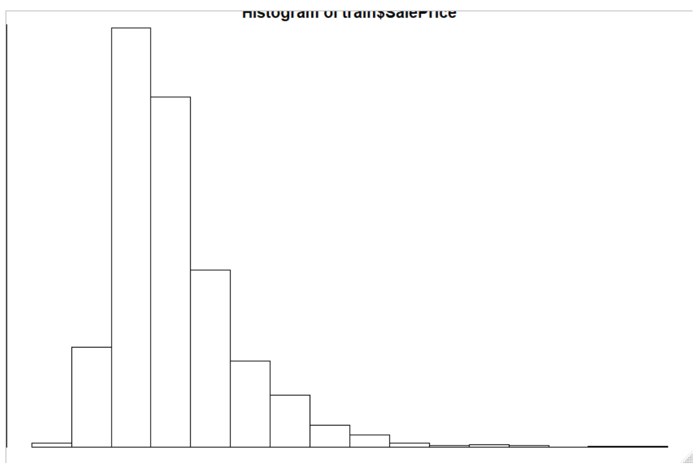

Figure 4: Normality error in SalePrice

Log Transformation was used to make SalePrice normal in nature. Also, all the variables which measured years were transformed into factors for ease of plotting and model complexity.

```
> as.table(skewed_feats)
   MSSubClass   LotFrontage       LotArea   OverallCond    MasVnrArea    BsmtFinSF1    BsmtFinSF2     BsmtUnfSF
    1.3747506     1.6595299    12.8158428     0.5700190     2.5935817     1.4247403     4.1440129     0.9187319
  TotalBsmtSF     X1stFlrSF     X2ndFlrSF  LowQualFinSF     GrLivArea  BsmtFullBath  BsmtHalfBath      HalfBath
    1.1621950     1.4688493     0.8612320    12.0825494     1.2687055     0.6245111     3.9295737     0.6942096
 KitchenAbvGr  TotRmsAbvGrd    Fireplaces    WoodDeckSF   OpenPorchSF EnclosedPorch    X3SsnPorch   ScreenPorch
    4.3000437     0.7579772     0.7331177     1.8414861     2.5338111     4.0018339    11.3702193     3.9446658
     PoolArea       MiscVal
   16.8896450    21.9359177
```

Figure 5: Skewness

The features which had more than 0.5 skewness were also transformed using log transformation to make the data normal.

2.3 Feature Engineering:

Since our dataset, consisted of mostly categorical variables, dummy variables were created to make the model more robust and accurate.

```
dmy <- dummyVars( ~ ., data = main_df)
dmy_predict <- data.frame(predict(dmy, newdata = main_df))
```

Figure 6: Dummy variables created

3 MODELING:

Finally, after prepping the data, we began to develop our predictive model using LASSO and ridge regression.

3.1 LASSO:

LASSO regression was used to predict the prices of the house. All the interactions were taken into consideration while creating the model matrix for LASSO. The LASSO plot can be seen below:

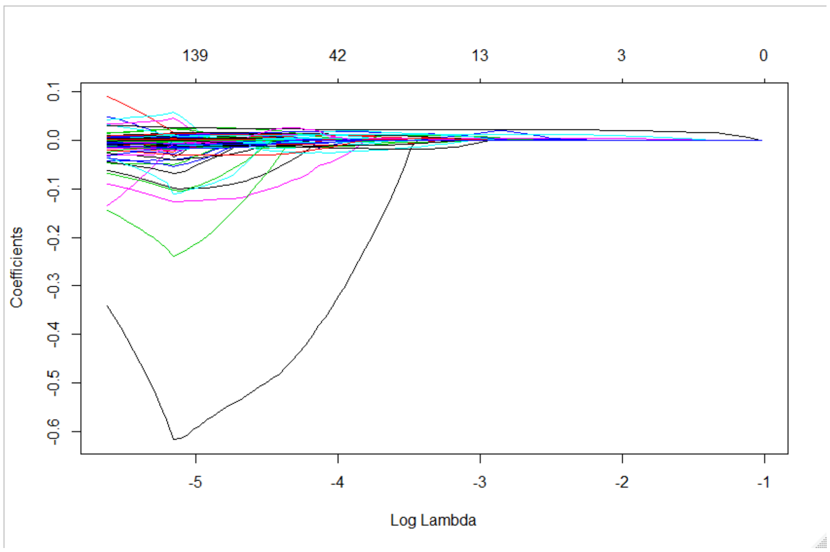

Figure 7: Lasso Regression

A final score of 0.11899 was obtained on Kaggle using LASSO regression

3.2 RIDGE REGRESSION:

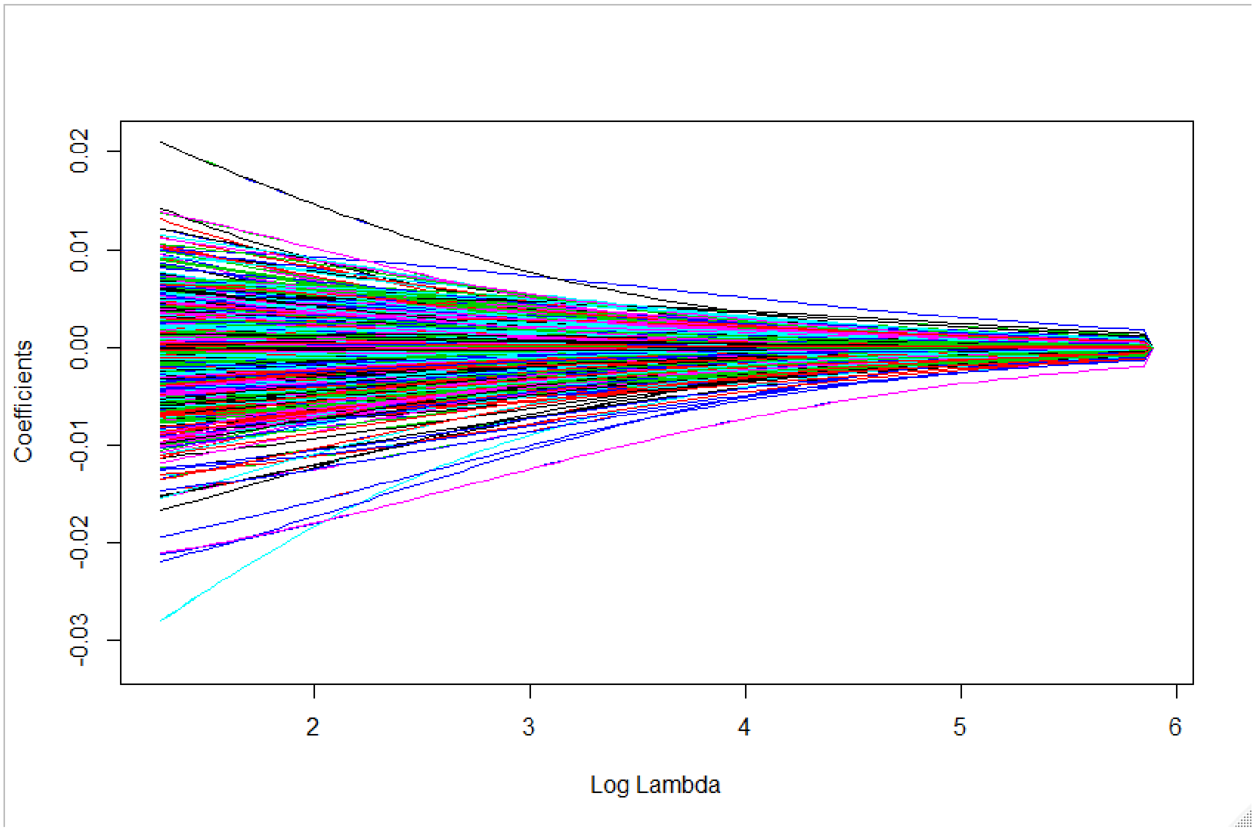

Figure 8: Ridge Regression

Using Ridge Regression for the model yielded a score of 0.15435 on Kaggle.

4 CONCLUSION:

It was found that LASSO regression performed better than Ridge regression. Lasso method overcomes the disadvantage of Ridge regression by not only punishing high values of the coefficients β but setting them to zero if they are not relevant. Therefore, we end up with fewer features included in the model than we started with, which is a huge advantage.

5 Appendix

5.1 R Code

```
library(readr)
library(tidyverse)
library(mice)
library(corrplot)
library(caret)
library(glmnet)

train <- read.csv("train.csv",stringsAsFactors=T, header=T)
test <- read.csv("test.csv",stringsAsFactors=T, header=T)


test$SalePrice <- NA

main_df <- rbind(train,test)
test$SalePrice <- NULL
str(train)


rapply(train, class = "factor", f = levels, how = "list")

md.pattern(main_df)
as.data.frame(colSums(is.na(main_df)))

sum.na <- sort(sapply(train, function(x) { sum(is.na(x))}), decreasing=TRUE)
sum.na
sum.na.percent <- sort(sapply(train, function(x) { sum(is.na(x)/dim(train)[1]
)}), decreasing=TRUE)
as.data.frame(sum.na.percent)

contVar <- names(train)[which(sapply(train, is.numeric))]
trainCont <- train[, contVar]
correlations <- cor(trainCont, use = "pairwise.complete.obs")
corrplot(correlations, method = "square")

# look up top 10 feature correlated to SalePrice
cor <- as.data.frame(as.table(correlations))
cor <- subset(cor, cor$Var2 == "SalePrice")
cor <- cor[order(cor$Freq, decreasing = T)[1:10],]
cor

#train set
train$PoolQC <- NULL
train$MiscFeature <- NULL
train$Alley <- NULL
```

```
train$Utilities <- NULL

ggplot(train, aes(x = GrLivArea, y = SalePrice)) + geom_point() + title("Outl
iers")
plot(train$GrLivArea,train$SalePrice)
order(train$GrLivArea,decreasing = T)[1:2]
train <- train[-1299,]
train <- train[-524,]


#imputation with mean
imp <- mice(train, m=1, method="cart")
train_comp <- mice::complete(imp)
as.data.frame(colSums(is.na(train_comp)))

#log transformation of SalePrice in train
hist(train$SalePrice)
train_comp$SalePrice <- log(train_comp$SalePrice)

#test Set
test$PoolQC <- NULL
test$MiscFeature <- NULL
test$Alley <- NULL
test$Utilities <- NULL

#imputation with mean
imp1 <- mice(test, m=1, method="cart")
test_comp <- mice::complete(imp1)
as.data.frame(colSums(is.na(test_comp)))


test_comp$SalePrice <- NA
train_comp <- train_comp %>% mutate(GarageYrBlt= as.factor(GarageYrBlt),YrSol
d=as.factor(YrSold), YearBuilt =as.factor(YearBuilt),
                                    YearRemodAdd = as.factor(YearRemodAdd))
test_comp <- test_comp %>% mutate(GarageYrBlt= as.factor(GarageYrBlt),YrSold=
as.factor(YrSold), YearBuilt =as.factor(YearBuilt),
                                  YearRemodAdd = as.factor(YearRemodAdd))


main_df <- rbind(train_comp,test_comp)
as.data.frame(colSums(is.na(main_df)))

df <- as.data.frame(main_df[,77])
main_df <- main_df[,-c(77)]

library(e1071)

classes <- lapply(main_df,function(x) class(x))
numeric_feats <- names(classes[classes=="integer" | classes=="numeric"])
```

```r
factor_feats <- names(classes[classes=="factor"| classes=="character"])

skewed_feats <- sapply(numeric_feats, function(x) skewness(main_df[[x]]))
skewed_feats <- skewed_feats[abs(skewed_feats) > .50]
as.table(skewed_feats)
hist(main_df$KitchenAbvGr)


for (x in names(skewed_feats)) {main_df[[x]] <- log(main_df[[x]]+1)}
main_df <- cbind(main_df, df = df$`main_df[, 77]`)
colnames(main_df)[77] <- "SalePrice"

dmy <- dummyVars( ~ ., data = main_df)
dmy_predict <- data.frame(predict(dmy, newdata = main_df))

master_df <- dmy_predict
master_df_train <- master_df[1:1458,]
master_df_test <- master_df[1459:2917,]

#LASSO Regression

options(na.action='na.pass')
y<- master_df_train$SalePrice

X <- model.matrix(SalePrice ~.^2, master_df)[,-c(1)]

X.training<- subset(X,X[,1]< 1461)
X.prediction<- subset(X,X[,1]>=1461)

nlasso.fit<-glmnet(x = X.training, y = y, alpha = 1)
plot(nlasso.fit, xvar = "lambda")

crossval <-  cv.glmnet(x = X.training, y = y, alpha = 1) #create cross-valida
tion data. By default, the function performs ten-fold cross-validation, thoug
h this can be changed using the argument nfolds.
plot(crossval)
penalty.lasso <- crossval$lambda.min #determine optimal penalty parameter, la
mbda
log(penalty.lasso) #see where it was on the graph
lasso.opt.fit <-glmnet(x = X.training, y = y, alpha = 1, lambda = penalty.las
so) #estimate the model with the optimal penalty
coef <- coef(lasso.opt.fit) #resultant model coefficients


# predicting the performance on the testing set
predicted.prices.log.i.lasso <- exp(predict(lasso.opt.fit, s = penalty.lasso,
 newx =X.prediction))
write.csv(predicted.prices.log.i.lasso, file = "Predicted Sale Prices.csv")

#Ridge Regression
```

```
ridge.fit<-glmnet(x = X.training, y = y, alpha = 0)
plot(ridge.fit, xvar = "lambda")

#selecting the best penalty lambda
crossval1 <-  cv.glmnet(x = X.training, y = y, alpha = 0)
plot(crossval1)
penalty.ridge <- crossval1$lambda.min
log(penalty.ridge)
ridge.opt.fit <-glmnet(x = X.training, y = y, alpha = 0, lambda = penalty.rid
ge) #estimate the model with that
coef(ridge.opt.fit)

ridge.testing <- exp(predict(ridge.opt.fit, s = penalty.ridge, newx =X.predic
tion))
write.csv(ridge.testing, file = "Predicted Sale Prices Ridge.csv")
```