MMA867

Predictive Modeling

Jue Wang

Individual Assignment 2

2019-09-05

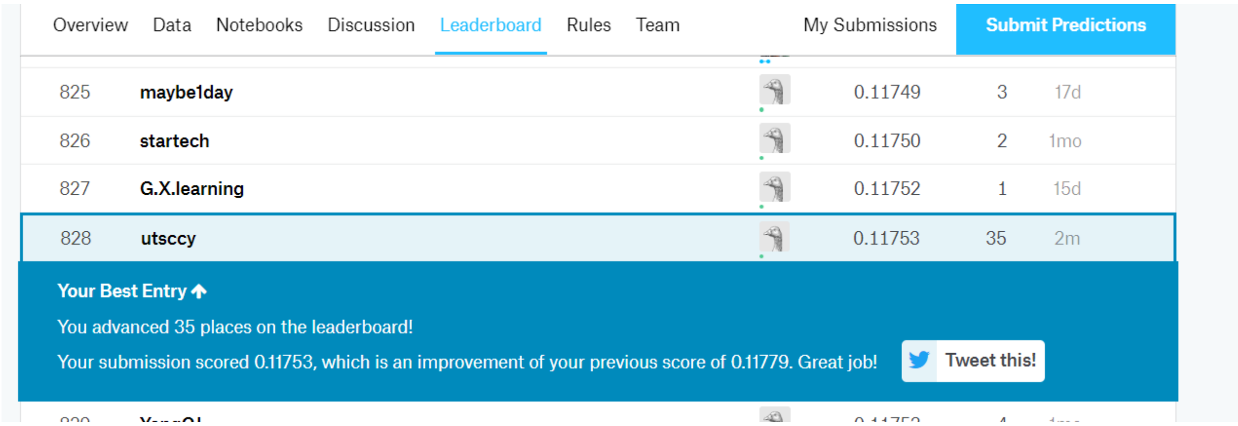
Chris Cao

Order of files:

Filename	Pages	Comments and/or Instructions
Individual Assignment 2 – Chris Cao	10	10 pages in total including cover page
submission.xlsx		Submission file for Kaggle
a2-kaggle code.r		R file for code

Additional Comments:

Leaderboard



House Price Prediction

Introduction

We would like to make prediction of house price based on 76 explanatory features provided in the train and test dataset. Due to the large number of variables, we decided to build advanced regression model (LASSO/RIDGE) to help with variables selection.

Load Data

We are provided two datasets: test.csv and train.csv. The first step is to load the dataset and remove ID columns. We only store IDs in the test data for our submission file.

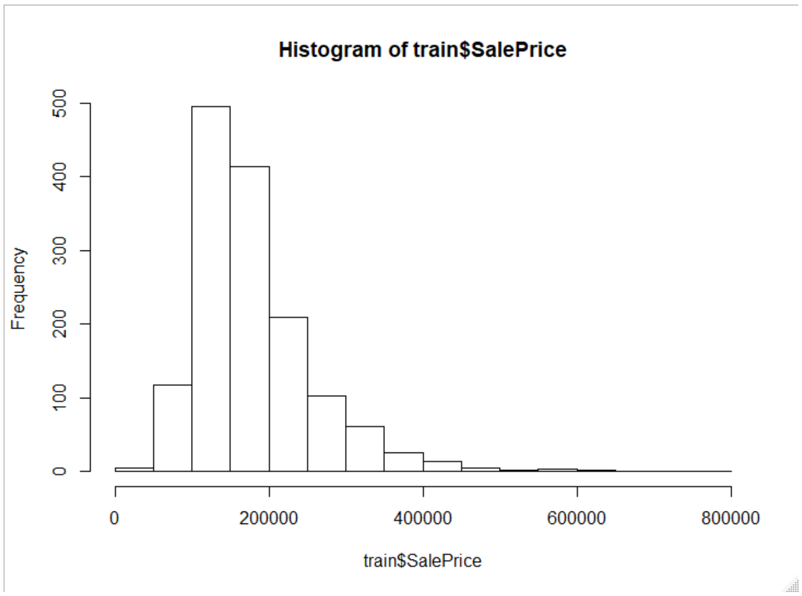
```
train<-read.csv('train.csv',stringsAsFactors = FALSE)

test<-read.csv('test.csv',stringsAsFactors = FALSE)

test_id <- test$Id

train<-select(train,-'Id')
```

Log Transformation

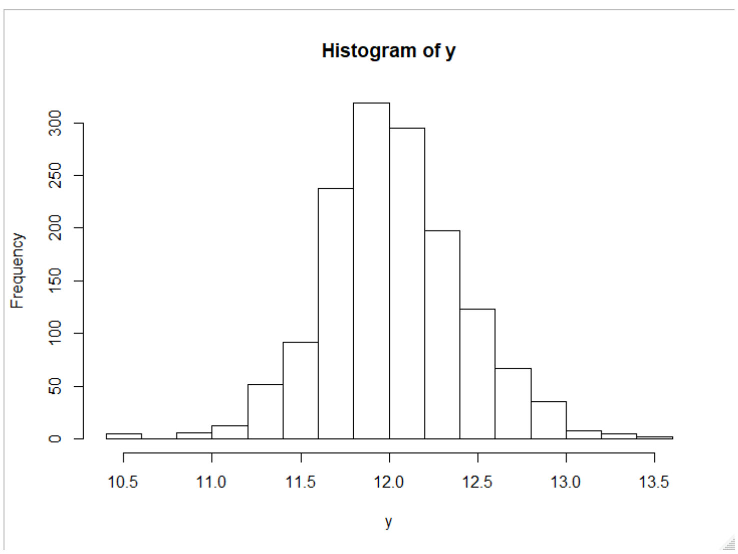
By plotting the histogram of sale price, we noticed that sale price distribution is right skewed, so we use log transform to make it to normally distributed.

```
hist(train_y$SalePrice)

y<-log(train_y+1)

y = y[[1]]
```

Before

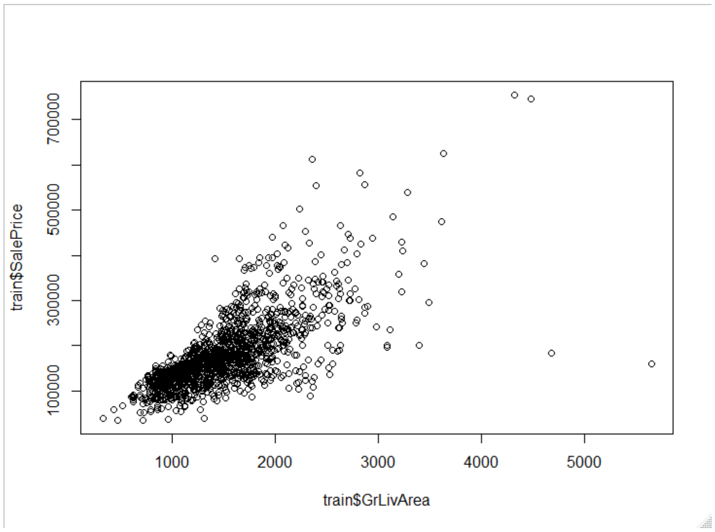


After



Outlier Analysis

After plotting chart of GrLiveArea and SalePrice, we noticed there are two observations having more than 4000 sqft living area and were sold less than 350000. We decided to remove these two observations as they will be too influential to our model.



Feature Engineering

We then combine our train and test dataset and perform feature engineering on the explanatory variables.

```
train_x<-select(train,-'SalePrice')
train_y<-select(train,'SalePrice')
test_x<-select(test,-'Id')
full<-rbind(train_x,test_x)
```

We added one new dummy variable isnew, for houses' year sold (YrSold) is within one year of year built (YearBuilt)

By looking at other two numeric variables: MoSold and MSSubclass, month should be factor variable and MSSubclass is a mapping from numeric code to subclasses, so it should be a factor variable as well. Here, we changed their data type to character and we will mass update all character variables to factor variables later.

```
full$IsNew <- as.character(ifelse(full$YrSold<=1+full$YearBuilt, 1, 0))
full$MoSold <- as.character(full$MoSold)
full$MSSubClass <- as.character(full$MSSubClass)
```

Missing Data

There are 33 variables having missing data.

```
which(colSums(is.na(full)) > 0)

   MSZoning  LotFrontage       Alley    Utilities  Exterior1st  Exterior2nd   MasVnrType   MasVnrArea     BsmtQual     BsmtCond
          2            3           6           9           23          24           25           26           30           31
BsmtExposure BsmtFinType1   BsmtFinSF1 BsmtFinType2   BsmtFinSF2    BsmtUnfSF  TotalBsmtSF   Electrical BsmtFullBath Bsmt
HalfBath
         32           33          34          35           36          37           38           42           46           47
 KitchenQual   Functional  FireplaceQu   GarageType  GarageYrBlt GarageFinish   GarageCars   GarageQual   GarageCond       Pool
QC
         52           53          55          56           57          58           59           60           61           69
      Fence  MiscFeature     SaleType
         70           71          75
```

Garage Variables

We filled in missing data in garage-related variables with 0 and 'None' as these houses are very likely to not have a garage.

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

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

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

full$GarageType[is.na(full$GarageType)]<-'None'

full$GarageFinish[is.na(full$GarageFinish)]<-'None'

full$GarageQual[is.na(full$GarageQual)]<-'None'

full$GarageCond[is.na(full$GarageCond)]<-'None'
```

Basement Variables

We filled in missing data in Basement-related variables with 0 and 'None'. The reason for these data get missing is because the house does not have basement or corresponding rooms in their basements.

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

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

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

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

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

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

full$BsmtQual[is.na(full$BsmtQual)]<-'None'

full$BsmtExposure[is.na(full$BsmtExposure)]<-'None'

full$BsmtFinType1[is.na(full$BsmtFinType1)]<-'None'

full$BsmtFinType2[is.na(full$BsmtFinType2)]<-'None'
```

```
full$BsmtCond[is.na(full$BsmtCond)]<-'None'
```

LotFrontage

We impute missing LotFrontage data using the sample median.

```
full$LotFrontage[is.na(full$LotFrontage)]<-median(full$LotFrontage,na.rm = TRUE)
```

Other Variables

We replaced missing data in other variables with 0 and 'None' depend on the data type as we wouldn't want to impute them with wrong data.

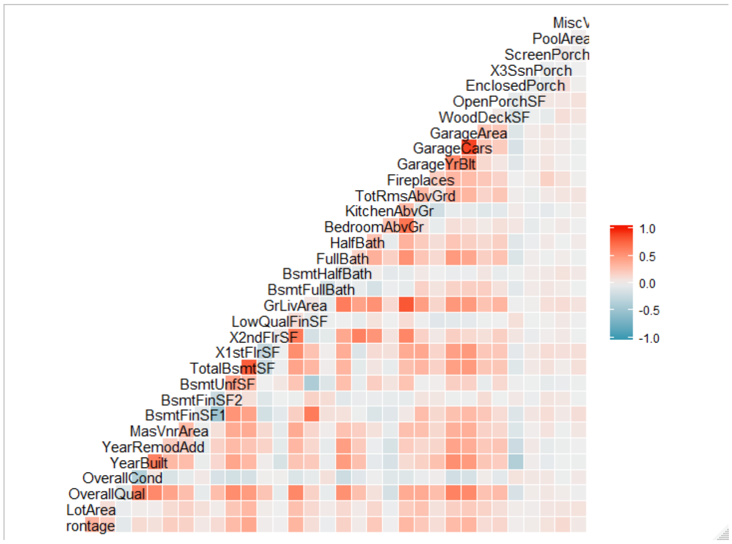
```
full$MasVnrArea[is.na(full$MasVnrArea)]<-0

full$MasVnrType[is.na(full$MasVnrType)]<-'None'

full$Alley[is.na(full$Alley)]<-'None'

full$PoolQC[is.na(full$PoolQC)]<-'None'

full$MiscFeature[is.na(full$MiscFeature)]<-'None'

full$Fence[is.na(full$Fence)]<-'None'

full$FireplaceQu[is.na(full$FireplaceQu)]<-'None'

full$MSZoning[is.na(full$MSZoning)]<-'None'

full$KitchenQual[is.na(full$KitchenQual)]<-'None'

full$Functional[is.na(full$Functional)]<-'None'

full$Electrical[is.na(full$Electrical)]<-'None'

full$Exterior1st[is.na(full$Exterior1st)]<-'None'

full$Exterior2nd[is.na(full$Exterior2nd)]<-'None'

full$SaleType[is.na(full$SaleType)]<-'None'
```

Correlation Analysis

After cleaning the data, we run the correlation matrix plot between independent variables to see if there is any multi-collinearity issue exists.

We noticed several variables have very high correlation between each other. So we decided to remove these variables from our dataset.

```
ggcorr(full)

full <- select(full, -c(TotRmsAbvGrd, GarageArea,X2ndFlrSF, X1stFlrSF))
```



Data Preprocess

Data Normalization

We would like our independent variables to be normally distributed, after we run skim function, we noticed that skewness issue exists in lots of variables, so we perform the boxcox transformation to all variables having skewness more than 0.5.

```
int_skew <- sapply(names(int_var), function(x) {

 skewness(full[[x]], na.rm = TRUE)

})

int_skew <- int_skew[abs(int_skew) > 0.5]

for (x in names(int_skew)) {

 bc = BoxCoxTrans(full[[x]], lambda = 0.1)

 int_var[[x]] = predict(bc, full[[x]])

} }
```

Standardization

We then further clean our data using preprocess function to standardize our numeric variables.

```
#normalize the data

preint <- preProcess(int_var, method=c("center", "scale"))

int_var <- predict(preint, int_var)
```

One-hot encoding

We create all the dummy variables for our character variables so they can be treat as factors in our model.

```
dummies <- dummyVars(~., full[names(chr_var)])
```

```
chr_var <- predict(dummies, full[names(chr_var)])
```

Now, we have the data cleaned and are ready for building the model.

Model Development

We would like to lasso and ridge regression method to perform variables selection since we have over 200 variables.

First, we start with dividing

LASSO MODEL

We run lasso regression model with default nfolds = 10 and get optimal lambda = 0.002821214 and RMSE = 0.09578228. The RMSE is quite small but we didn't know how this model performs on the test dataset. Thus, we are going to run ridge regression and then compare both results.

```
cv_lasso = cv.glmnet(x, y,alpha=1)

penalty.lasso <- cv.lasso$lambda.min

penalty.lasso

lasso.opt.fit <-glmnet(x = x, y = y, alpha = 1, lambda = penalty.lasso)

lasso.train <- predict(lasso.opt.fit, s = penalty.lasso, newx =x)

sqrt(mean((y - lasso.train)^2))

lasso_test <- predict(cv_lasso, newx = x_test, s = "lambda.min")

lasso.final<-cbind(test_id,(exp(lasso_test)-1))
```

RIDGE MODEL

We run the ridge model by changing alpha in the cv.glmnet function from 1 to 0 and we get optimal lambda = 0.06294838 and RMSE = 0.09366618. Looks like Ridge Model has better performance on our test dataset. However, RIDGE model has way more variables than LASSO (more likely to have over-fitting issues), both models' performance on test dataset is unknown at this point.

To compare the result, we upload result from both models and see which one has the higher score.

```
#ridge

cv.ridge = cv.glmnet(x, y, alpha = 0)

plot(cv.ridge)

penalty.ridge <- cv.ridge$lambda.min

ridge.opt.fit <-glmnet(x = x, y = y, alpha = 0, lambda = penalty.ridge)

coef(ridge.opt.fit) #resultant model coefficients

ridge.train <- predict(ridge.opt.fit, s = penalty.ridge, newx =x)
```

```
sqrt(mean((y - ridge.train)^2))

ridge.testing <- predict(ridge.opt.fit, s = penalty.ridge, newx =x_test)
```

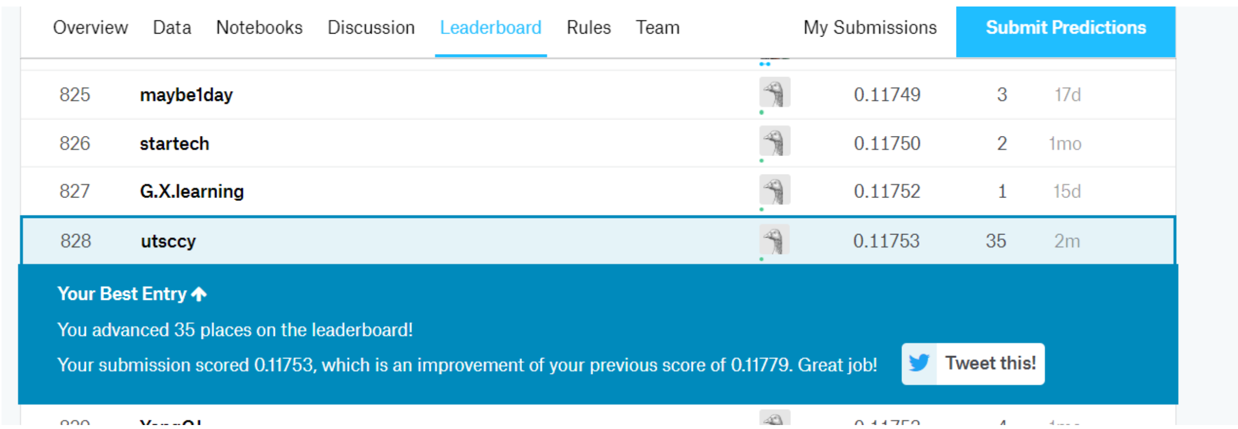
Submission

We run the model on test dataset, and we create the final submission file according to the format requirement.

It turns out our lasso regression model has the best score: 0.11753 (ridge model’s score is 0.124), so we choose LASSO model as our final model and we are able to reach top 20% in the competition.

Appendix

Leaderboard:



Final Model:

```
cv_lasso = cv.glmnet(x, y,alpha=1)

penalty.lasso <- cv.lasso$lambda.min

penalty.lasso

lasso.opt.fit <-glmnet(x = x, y = y, alpha = 1, lambda = penalty.lasso)

lasso.train <- predict(lasso.opt.fit, s = penalty.lasso, newx =x)

sqrt(mean((y - lasso.train)^2))

lasso_test <- predict(cv_lasso, newx = x_test, s = "lambda.min")

lasso.final<-cbind(test_id,(exp(lasso_test)-1))

colnames(lasso.final)<-c('Id','SalePrice')

write.csv(lasso.final, '../kaggle_houseprice/submission_lasso.csv',row.names=FALSE)
```