Predicting Sales Prices Kaggle Competition

Team LI202

Ramtin Boustani, Souptik Sen, Ashwin Murthy, Siddharth Nair

We had 10 submissions and the best one was the last one with public score 3253103545.49860. Our prediction of the sales price is average of predictions of Lasso and XGBoost models.

Techniques we used in our experiment in R code:

- Using handy R packages

  `caret` package provides an easy and unified interface for training and predicting different models.

  ```
  library(caret)
  ```

  Very easy to apply cross-validations for selecting the best hyperparameter
  ```
  trainControl(method = "cv", number = 10)
  ```

  Very easy to training any models
  ```
  train(x=.., y=..., method='glmnet', trControl=...) #LASSO or RIDGE
  train(x=.., y=..., method='gbm', trControl=...) #Boosting
  ```

  Very easy to predict
  ```
  predict( … , … )
  ```

  Very easy to get prediction accuracy
  ```
  postResample(...,...)
          RMSE     Rsquared          MAE
  3.984523e+04 9.834983e-01 3.366952e+04
  ```

  `fastDummies` package for creating dummy columns from categorical variables
  ```
  library(fastDummies)
  ```

- Combined train and test data sources for preprocessing steps

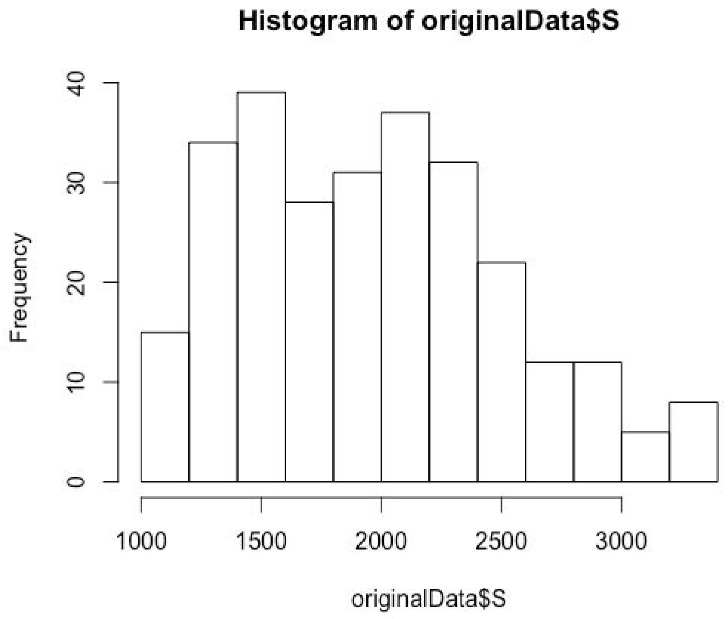
  It was much easier to combine train and test csv files and apply all preprocessing methods in one dataset. At the end separate them back to 2 train and test datasets.

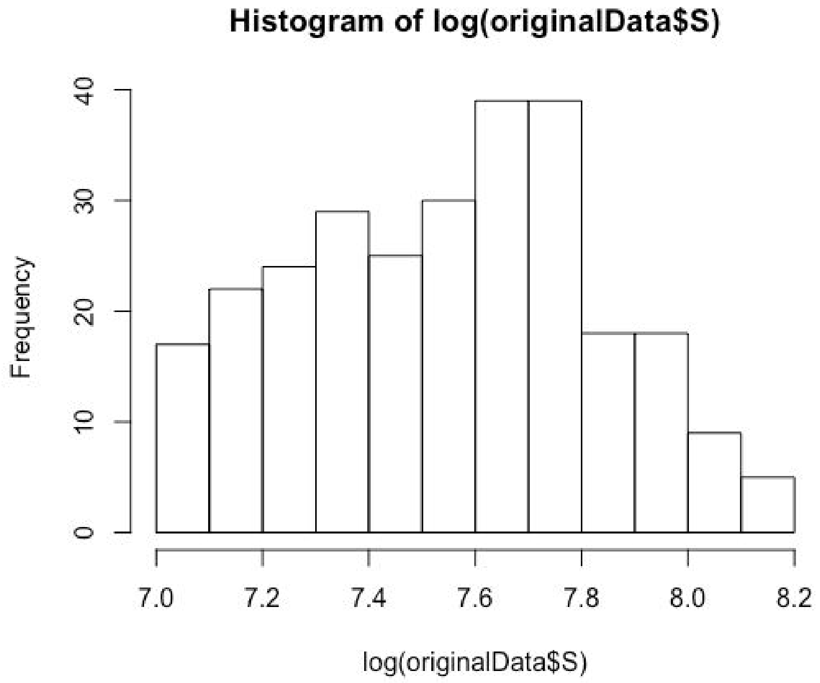
  ```
  all = rbind(data, test)
  #PreProcessing steps
  ```

```
...
data  = all[!is.na(all$P),]
test = all[is.na(all$P),]
```

- Normalizing Sales Price

  Using log() and exp() for solving sales price skewness problem





- Dummy variables for the categorical predictors

  Creating dummy columns for Sch (index of school), C (index of crime) and Y (year the house was sold)

```
dummySch = dummy_cols(data.frame( "Sch" = as.factor(all$Sch)))
dummyC = dummy_cols(data.frame( "C" = as.factor(all$C)))
dummyY = dummy_cols(data.frame( "Y" = as.factor(all$Y)))
```

  As a result the following 26 predictors added to the model

```
"Sch_0" "Sch_1" "Sch_2" "C_1" "C_2" "C_3" "C_4" "Y" "Y_2000" "Y_2001"
"Y_2002" "Y_2003" "Y_2004" "Y_2005" "Y_2006" "Y_2007" "Y_2008" "Y_2009"
"Y_2010" "Y_2011" "Y_2012" "Y_2013" "Y_2014" "Y_2015" "Y_2016" "Y_2017"
"Y_2018"
```

- Using Lasso

  We wanted to using least square fit and selecting best subset of predictors that all provided in Lasso. In our experiment from 496 predictors Lasso used 76 variables in its model, and did not select 420 variables.

```
#alpha 1 is Lasso
lassoCtrl = trainControl(method = "cv", number = 10)
lassoGrid = expand.grid(alpha = 1, lambda = seq(0.001,0.1, by = 0.0005))

lasso.mod = train( P ~
(R+PT+S+YB+L+Sch_0+Sch_1+Sch_2+C_1+C_2+C_3+C_4+Y_2000+Y_2001+Y_2002+Y_2003+Y
_2004+Y_2005+Y_2006+Y_2007+Y_2008+Y_2009+Y_2010+Y_2011+Y_2012+Y_2013+Y_2014+
Y_2015+Y_2016+Y_2017+Y_2018)^2,  data=data, method='glmnet',
trControl=lassoCtrl,tuneGrid=lassoGrid)

lasso.pred = predict(lasso.mod,test)
```

  With alpha =1 and lambda=0.0025

- Adding interactions between predictors improved Lasso prediction accuracy

  By adding interactions between predictors our testing result improved significantly. To having more bias, less variance and avoid overfitting we did not use polynomial degrees. After adding interactions to 8 predictors we end up training our model by using 496 predictors.

- Using XGBoost

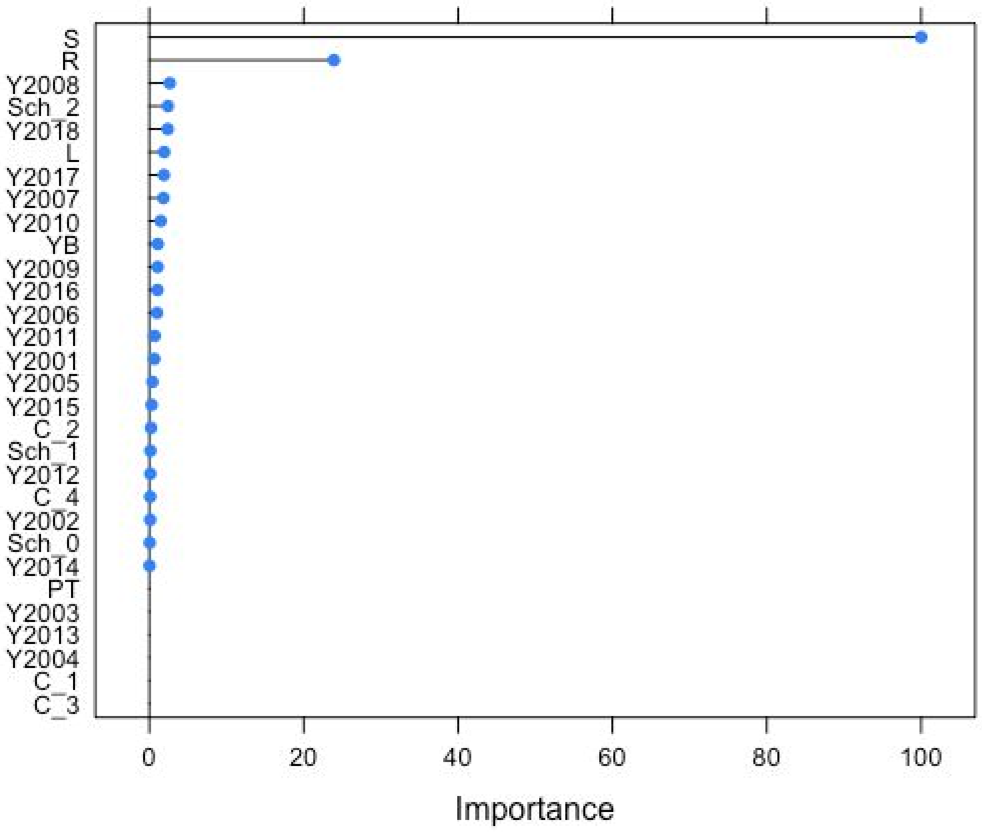
  We used extreme gradient boosting tree-based regression. It has better testing accuracy compared to Boosting (gbm) and very close to Lasso.

```
xgbCtrl = trainControl(method = "cv", number = 10)
```

```
xgb.mod = train( P ~ (R+PT+S+YB+L+Sch_0+Sch_1+Sch_2+C_1+C_2+C_3+C_4+Y),
data=data, method='xgbTree', trControl=xgbCtrl)
```

```
xgb.pred = predict(xgb.mod, test)
```

Following is variable importance plot



- Using Cross-Validation

  For finding the best hyperparameter we used cross validations with 10 folds
  `trainControl(method = "cv", number = 10)`

- Using different set of predictors for each modeling

  We train Lasso and xgboost with different set of predictors and observed better performance.
  E.g. in Lasso we used Y (year the house was sold) as a categorical predictor and added related dummy columns

  ```
  Y_2000+Y_2001+Y_2002+Y_2003+Y_2004+Y_2005+Y_2006+Y_2007+Y_2008+Y_2009+Y_2010
  +Y_2011+Y_2012+Y_2013+Y_2014+Y_2015+Y_2016+Y_2017+Y_2018
  ```

  But it did not work well for xgboost so when we tried Y as a numerical predictor it had a much better result.

- Averaging the Lasso and XGBoost predictions

  We found for some records Lasso had a better prediction and for some others XGBoost. Also in our experiment MSE for both of them were close enough.
  So when we only submit Lasso prediction we had 4,610,612,297 public score and when we submit the average prediction of Lasso and XGboost we got much better public score 325,3103,545 .

In addition to the above we also tried the following techniques but did not help or not required for the given data

- Feature engineering

  We tried to add the two following features but did not get much improvement. We also thinking they cause multicollinearity problem.

  - House Age

    Y (year the house was sold) - YB (year the house was built)

  - Price Per Square Foot

    P (sales price) / S (size of the house in square feet)

- Imputation Missing data

  We did not find any missing data.

- Removing outlier

  We checked the sales price predictor and did not find any significant outlier.

- Using Ridge Regression model

  We tried but Lasso had a better result

  ```
  ridgeCtrl = trainControl(method = "cv", number = 10)
  ridgeGrid = expand.grid(alpha = 0, lambda = seq(0.001,0.1, by = 0.0005))
  ```

- Using Boosting (GBM)

  We tried but XGBoost had a better result

  ```
  gbmCtrl = trainControl(method = "cv", number = 10)
  gbm.mod = train(x=data[,-1], y=data$P, method='gbm', trControl= gbmCtrl,
  verbose=FALSE)
  ```

References

1. Package ‘caret’
   https://cran.r-project.org/web/packages/caret/caret.pdf

2. House prices: Lasso, XGBoost, and a detailed EDA
   https://www.kaggle.com/erikbruin/house-prices-lasso-xgboost-and-a-detailed-eda/report