Code

Project 1: Predict the Housing Prices in Ames

Fall 2021

Ames Housing Data

Data set

Download the dataset, `Ames_data.csv` , from Campuswire. The dataset has 2930 rows (i.e., houses) and 83 columns.

- The first column is "PID", the Parcel identification number;
- The last column is the response variable, `Sale_Price` ;
- The remaining 81 columns are explanatory variables describing (almost) every aspect of residential homes.

Test IDs

Download `project1_testIDs.dat` from Campuswire. This file contains 879 rows and 10 columns, which will be used to generate **10 sets** of training/test splits from `Ames_data.csv` . Each column contains the 879 row-numbers of a test data.

Here is how you generate a split of training and test using the j-th column of `project1_testIDs.dat` in R.

Hide

```
data <- read.csv("Ames_data.csv")
testIDs <- read.table("project1_testIDs.dat")
j <- 2
train <- data[-testIDs[,j], ]
test <- data[testIDs[,j], ]
test.y <- test[, c(1, 83)]
test <- test[, -83]
write.csv(train,"train.csv",row.names=FALSE)
write.csv(test, "test.csv",row.names=FALSE)
write.csv(test.y,"test_y.csv",row.names=FALSE)
```

Save the training and test as csv files. In particular, the test data are saved as two seperate files: one containing just the feature vectors and the other one containing the response column.

For your remaining analysis, you need to read the training and test data from the csv files.

Hide

```
train <- read.csv("train.csv")
test <- read.csv("test.csv")
```

Goal

The goal is to predict the final price of a home (**in log scale**) with those explanatory variables. You need to build **Two** prediction models selected from the following two categories:

- one based on linear regression models with Lasso or Ridge or Elasticnet penalty;
- one based on tree models, such as randomForest or boosting tree;

The features for the two models do not need to be the same. Please check Campuswire for packages students are allowed to use for this project.

Source

- De Cock, D. (2011). “Ames, Iowa: Alternative to the Boston Housing Data as an End of Semester Regression Project,” Journal of Statistics Education, Volume 19, Number 3. [PDF (http://ww2.amstat.org/publications/jse/v19n3/decock.pdf)]
- Check variable description [Here (https://ww2.amstat.org/publications/jse/v19n3/decock/DataDocumentation.txt)]
- This data set has been used in a Kaggle competition (https://www.kaggle.com/c/house-prices-advanced-regression-techniques (https://www.kaggle.com/c/house-prices-advanced-regression-techniques)). You can check how others analyze this data and try some sample code on Kaggle. Note that our data set has two more explanatory variables, “Longitude” and “Latitude.”

What to Submit

Submit the following **two** items on Coursera/Canvas:

- R/Python code in a single file named **mymain.R** (or **mymain.py**) that takes a training data and a test data as input and outputs two submission files (the format of the submission file is given below). **No zip files; no R markdown files**.

The structure of your mymain.R (or mymain.py) should look like the following. **Note** that You have to pre-process your training and test data separately.

Hide

```
# Step 0: Load necessary libraries
##########################################
# Step 0: Load necessary libraries
#
# YOUR CODE
#


##########################################
# Step 1: Preprocess training data
#         and fit two models
#
train <- read.csv("train.csv")
#
# YOUR CODE
#


##########################################
# Step 2: Preprocess test data
#         and output predictions into two files
#
test <- read.csv("test.csv")
#
# YOUR CODE
#
```

- A report (3 pages maximum, PDF or HTML) that provides
    1. technical details (e.g., pre-processing, implementation details if not trivial) for the models you use, and
    2. any interesting findings from your analysis.
    3. In addition, report the accuracy on the test data (see evaluation metric given below), running time of your code and the computer system you use (e.g., Macbook Pro, 2.53 GHz, 4GB memory, or AWS t2.large) for the 10 training/test splits.
    4. You **do not** submit the part of the code that evaluates test accuracy.

  In your report, describe the techincal details of your code, summarize your findings, and **do not copy-and-paste your code to the report**.

Code Evaluation

We will run the command `source(mymain.R)` in a directory, in which there are only **two files**:

- **train.csv**: 83 columns;
- **test.csv**: 82 columns without the column “Sale_Price”.

The two files are from one of the 10 training/test splits. Note that **test_y.csv** is NOT in this directory):

After running your code, we should see **two** txt files in the same directory named `mysubmission1.txt` and `mysubmission2.txt`. Each submission file contains prediction on the test data from a model.

Submission File Format. The file should have the following format (do not forget the **comma** between PID and Sale_Price):

```
PID,  Sale_Price
528221060,  169000.7
535152150,  14523.6
533130020,  195608.2
```

Evaluation Metric. Submission are evaluated on Root-Mean-Squared-Error (RMSE) between the logarithm of the predicted price and the logarithm of the observed sales price. Our evaluation R code looks like the following:

```
# Suppose we have already read test_y.csv in as a two-column
# data frame named "test.y":
# col 1: PID
# col 2: Sale_Price

pred <- read.csv("mysubmission1.txt")
names(test.y)[2] <- "True_Sale_Price"
pred <- merge(pred, test.y, by="PID")
sqrt(mean((log(pred$Sale_Price) - log(pred$True_Sale_Price))^2))
```

Performance Target. Your performance is based on RMSE from the two models. Full credit for submissions with RMSEs less than

- **0.125** for the first 5 training/test splits and
- **0.135** for the remaining 5 training/test splits.

FAQ

1. Can I find the 10 training/test datasets on Campuswire?

   You generate them from `project1_testIDs.dat` and `Ames_data.csv`

2. Should we download the training/test datasets from Kaggle and upload our prediction on Kaggle for evaluation?

   No. You do not need to download/upload any dataset from/to Kaggle.

3. Should we include the split data part in the function we write?

   No, you do not need to include the split data part in your code. Your code should start with loading train.csv. Build your models based on train.csv, and then load test.csv and output the corresponding prediction. Store your output in txt files following the Submission File Format described above.

4. Do we need to do diagnostics? Is it important for this data set? If it is, should we delete any extreme values?

   Diagnostics or any pre-processing for missing values and extreme values are done by you. Please include your pre-processing procedure in the code and its description in the report.

   Please keep in mind that you are **NOT** asked to report detailed EDA (exploratory data analysis), but to build predictive models based on a training set.

5. The RMSE of the logarithm of price is infinity/nan for some observations. What should I do?

   Recall that log(y) is only defined for y > 0. If you train a model to predict y without any constraints it is possible that your model may predict a zero or negative value. These zero or negative predictions will cause infinities and nans to pop up in the metric calculations.

   Since what matters in the evaluation metric is log(y), predict log(y) or in this case log(Sales_Price) instead of the untransformed target. In other words, build a model to predict the logarithm of the price.

   Since you are asked to output the predicted Sales_Price in `mysubmission1.txt` and `mysubmission2.txt` , you need to transform the target back using the exponential –- exp(pred) –- when submitting your results.

6. How to specify tuning parameters?

   For Lasso or any algorithm that has just one or two tuning parameters, suggest to use their built-in CV procedures to select tuning parameters.

   Some algorithms such as boosting treess have quite a few tuning parameters. It would be time-consuming to tune them on a grid for each new training. For this assignment, it is okay to tune them based on the 10 splits of training/test, and then in your submitted code, fix the value of some (or even all) tuning parameter to save computation time.