Raj Kumar

Data Mining Final Project Report

Prof. Steven Bogaerts

October 1, 2021

Kaggle Competition: Housing Prices Advanced Regression Techniques

Data Preprocessing

All the data preprocessing was done in the provided transformData function. First, I checked the number of missing values in each column in the training set.

The following columns had the respective number of missing values:

['LotFrontage' (259), 'Alley'(1369), 'MasVnrType' and 'MasVnrArea'(8), 'BsmtQual', 'BsmtCond' and 'BsmtFinType1'(37), 'BsmtExposure'(38) and 'BsmtFinType2'(38), 'Electrical'(1), 'FireplaceQu'(690), 'GarageType', 'GarageYrBlt', 'GarageFinish', 'GarageQual', and 'GarageCond'(81), 'PoolQC'(1453), 'Fence'(1179), 'MiscFeature'(1406)]

Fixing data types in the raw data

Upon examining the data_description.txt file, it became apparent that some numeric attributes actually have categorical ranges such as 'MSSubClass' which has 16 types/categories of dwellings

MSSubClass: Identifies the type of dwelling involved in the sale.

20 1-STORY 1946 & NEWER ALL STYLES;
30 1-STORY 1945 & OLDER;
...
180 PUD - MULTILEVEL - INCL SPLIT LEV/FOYER,
190 2 FAMILY CONVERSION - ALL STYLES AND AGES

This column clearly was intended to be for categorical purposes.

Filling Missing Values

(a) Non-Numeric Attributes

While examining the data_description.txt file, I noticed that certain missing values actually communicated meaningful information. For example, Alley; Grvl = Gravel, Pave = Paved, NA = No alley access. So, filling Non-Numeric Missing Values isn't as simple as replacing NaN values with mode.

Only Non-Numeric Attributes with missing values:
['MSZoning','Alley','Utilities','Exterior1st','Exterior2nd','MasVnrType','BsmtQual','BsmtCond','BsmtExposure','BsmtFinType1','BsmtFinSF1','BsmtFinType2','Electrical','KitchenQual', 'Functional','FireplaceQu','GarageType','GarageFinish','GarageQual','GarageCond','PoolQC','Fence','MiscFeature','SaleType']

For each column name in the list above, I manually checked data_description.txt for any meaningful missing values

The following columns were found to have meaningful information contained in 'missing' values:

['Alley','BsmtQual','BsmtCond','BsmtExposure','BsmtFinType1','BsmtFinType2', 'FireplaceQu','GarageType','GarageFinish','GarageQual','GarageCond','PoolQC', 'Fence','MiscFeature']

The following columns were found to have no meaningful information contained in 'missing' values:

['MSZoning','Utilities','Exterior1st','Exterior2nd','MasVnrType','Electrical', 'KitchenQual','Functional','SaleType']

Filling Missing Values (b) Numeric Attributes

The following numeric columns were found to have missing values:

['LotFrontage', 'MasVnrArea', 'BsmtFinSF1', 'BsmtFinSF2', 'BsmtUnfSF', 'TotalBsmtSF', 'BsmtFullBath', 'BsmtHalfBath', 'GarageYrBlt', 'GarageCars', 'GarageArea']

I used KNNRegressor() to fill the missing values. Great care was taken to prevent data leakage.

There is an attribute in the data_description.txt called 'MoSold.' Any algorithm that we try to fit to our data would think that MoSold is a number, or that higher or lower values actually do mean something. Well, they do. Since, the values range from 1 to 12, it means that they are cyclical values based on time. So, in order for any model to capture the information this attribute actually represents, it needs to be converted to cyclical values. Cosine transformation was used on that column to fully capture the information and transform it to values that could be leveraged by the models.

Scaling

In order to scale, the skew was observed for each column. Log transformation was used to heavily skewed columns.

One-hot Encoding of Non-Numeric Variables

Due to disparities in some Non-Numeric columns in the test and train sets, for example, in the number of categories certain categorical attributes have in the train set exceeds the number of categories for the same attributes in the test set (Utilities; 'AllPub', 'NoSeWa' in trainDF but only 'AllPub' in testDF) and vice versa. So, it would be hard to maintain a consistent number of columns in both the sets after one-hot encoding.

While I am aware of the philosophical difference between the train and test set, it might be more orderly to merge the two sets (excluding the SalePrice column in the train set), and then spliting the sets to hide the test set from the train set.

I will then fit the model on the train set and predict the output for the test set.

Standardizing the dummy columns

I wanted to standardize after one hot encoding as well. The standardization method taught in class was giving missing values for a huge fraction of the dummy columns.

So, a built-in scikit preprocessing for standardization submodule called StandardScaler is the best bet to avoid that.

The transformations were separately applied to the train and test set.

Experiments

Gradient Boosting Regressor, Xtreme Gradient Boosting Regressor and Random Forrest regressor were fitted to the train set and used to predict the test output.

Since the train output was scaled using log. The predictions had be exponentiated before submission to Kaggle.

Experiment Results:

CV Average Score GBR: 1.133990795385413

CV Average Score XGB: 1.1461533816791263

CV Average Score Random Forest Regressor: 1.1506340763171994

GBR appears to result in the best performance of all three chosen models. So, it was chosen for Kaggle submission. The score on Kaggle was 0.13526.