Project 1: Predict the Housing Prices in Ames

CS598: Practical Statistical Learning

Naomi Bhagat - nbhagat3, Michael Miller - msmille3, Joe May - jemay3

01 October 2023

Assignment Data

Program: MCS-DS Assignment post: campuswire

Team contributions:

Person	Contribution
Naomi Bhagat	Data pre-procesing, training/testing general process, report Section 1
Michael Miller	Training/testing refinement and finalization, general process
Joe May	Report Section 1 and 2, debugging, and quality control

Section 1: Technical Details

The goal of this project is to predict the price of a home using certain features present in the Ames housing dataset. Importantly, ths prediction of Sale price is being done in the logarithmic scale. To achieve this goal, we built two prediction models: one using linear regression and one using a tree model in order to achieve an RSME of 0.125 for the first five splits and 0.135 on the final five splits.

Pre-Processing

The first step we took for this project was to clean up the data with some data pre-processing steps. The pre-processing steps described in this section are common between the linear and tree models that were built.

There were several pre-processing steps necessary for the linear model. First, we replaced NULL or missing values in the "Garage Year Built" column, as it was the only column in the data with missing values. Next, we removed a set of imbalanced categorical variables, and variables that don't offer additional insights. Most of these variables are highly biased, meaning that the entries for these vairables skew heavily in favor of one categry rather than a more normal distribution. The following is the set of removed variables:

- Street
- Utilities
- Condition_2
- Roof_Matl
- Heating
- Pool_QC
- Misc_Feature
- Low_Qual_Fin_SF

- Pool_Area
- Longitude
- Latitude

The next pre-processing step was Winsorization. Because the impact of some of the area-related variables need a ceiling, we calculated the 95% upper quantile of the chosen variables based on the training data, and any value in both the training and test datasets that exceeds this value is replaced with that value, effectively capping the possibe values in the feature at this 95% quantile value. The winsorized variables include:

- Lot_Frontage
- Lot_Area
- Mas_Vnr_Area
- BsmtFin_SF_2
- Bsmt_Unf_SF
- Total_Bsmt_SF
- Second_Flr_SF
- First_Flr_SF
- Gr_Liv_Area
- Garage_Area
- Wood_Deck_SF
- Open_Porch_SF
- Enclosed_Porch
- Three_season_porch
- Screen_Porch
- Misc_Val

Next, we converted our remaining categorical variables into 1-hot vectors. This step counts the unique categorical values for any character vector, creates a column for each value, and assigns a one to the column representing a row's category, and sets the rest equal to zero. Finally, the last step of our pre-processing function add the 'Sale_Price' column back into the output dataframe (if it was removed - this is mainly to benefit the training data) and returns the output dataframe.

In addition to the direct proe-processing of data, we include a function that which pads both dataframes with columns of zeros to ensure our training and test data have the same number of columns, making it easier to train models.

Linear Model

After running the training data through the described pre-processing, we create our linear model. First, we trained an initial model using glmnet with Lasso regression. The sole purpose of this model is to get the most optimal set of variables to use for prediction purposes. Lambda is set here to lambda.min. Then, using only this set of selected variables from the intial model, we created a second glmnet model with Ridge regression which is then used to make the final prediction on the test data.

Tree Model

For our tree model, we initially tried using a randomForest, but no matter how much we tweaked the model, we were not getting good results (as determined by RSME calculations on the 10 test folders) on the 10 splits. Therefore, we switched to using an xgboost model, largely using trial and error to determine which model was the most optimal for our purposes. Specifically, the xgboost model uses the following parameters:

- Maximum Depth: 6

- eta: 0.05
- nthread: 2
- rounds: 500

Section 2: Performance Metrics

The computer system this was run on was a Dell Inspiron 3501, 1.00 GHz with 8.00 GB of installed RAM for all 10 training/test splits. Below is a summary of the fold #, the model RMSE and the time it took to train the model.

Linear Model Performance

fold	RMSE	TIME (s)
1	0.1248	1.415
2	0.1211	1.919
3	0.1201	1.672
4	0.1208	1.730
5	0.1140	2.155
6	0.1341	1.473
7	0.1266	1.379
8	0.1192	1.756
9	0.1305	1.547
10	0.1257	1.528

Tree Model Performance

fold	RMSE	TIME (s)
1	0.1171	6.725
2	0.1211	10.118
3	0.1138	10.393
4	0.1181	9.897
5	0.1152	10.121
6	0.1308	10.330
7	0.1340	10.355
8	0.1283	10.378
9	0.1315	10.226
10	0.1269	10.348