Predicting House Price - Assignment

Jeffrey Josanne Michael

November 4, 2017

1 Pre-Processing

The objective of the assignment is to build and evaluate a simple model on the given data after appropriate pre-processing.
The instructions point out that the model should only be built with at most 3 independent variables. In our case, all the three variables (isLondon (bool), Property Type (categorical), Duration (categorical)) are unfortunately **categorical**. Having this concern, we cannot fit any functions like Radial basis or Gaussian functions to transform the data as they are discrete. Degree of Freedom is far less than the output observations(Price, in our case). Going deeper, **ANOVA (factorial)** is the only way to go through, but the assignment works on simpler models given the time constraint.

Anyways, the categorical data must be transformed to numerical in a discrete way (like categorical data). This assignment has adopted the method of **One-Hot encoding**.

2 Data Exploration

There was no much of an observable correlation between the suggested columns and the Price column.

- isLondon (Town) vs Price
- Duration (Lease / Free / U) vs Price
- Property Type vs Price

  The graphs from the output in the notebook says, there is no strong correlation between these categorical values and Price. There seems to exist a Gaussian distribution on property type and price. But, the problem is we have categorical data of property type. One specific observation is, for any property with Duration "U", the Price will be within 1*e8. This will be accommodated by the model.

3 Model and Evaluation

As it a good practice to compare a model with a baseline model, two kinds of regression models are implemented and compared.

Table 1: Evaluation and Performance metric comparison

Type	Metric	Linear regression	XGBooster
Negatively-oriented	**MAE**	173649.598543	206567.801635
Negatively-oriented	**RMS**	1163195.44077	1140490.83465
Positively-oriented	**CC**	0.141192564914	0.223089030475
Positively-oriented	**COD**	-0.00427021052886	0.0345522006582

- Linear regression model (regularized and non-regularized) is considered as the baseline model.
- XGBooster (gradient boosted trees) model is the one with better performance.

As we did not find any solid positive or negative correlations, linear regression model cannot perform well with the given data.

Linear regression and ridge regression models do not seem to vary as the dataset is clean and no penalty is incorporated into the model. But, XGBooster model seemed to dominate other models as they stack up weak leaner trees. Also, there were a few outliers (above 2*e8) and this would adversely affect the linear curve. But in the case of gradient trees, we consider all possible and eccentric data as a separate branch of the tree. And for this specific dataset, gradient boosted trees like XGBooster makes much sense for me to use.

4 Evaluation

Here, we have considered root mean square error(RMS), mean absolute error(MAE) as error metrics and correlation coefficient(CC), coefficient of determination(COD) as the performance metric. Negatively Oriented = Lower the better. Positively Oriented = Higher the better.

4.1 RMS and MAE

Since we have outliers in the test data, RMS is better to consider as RMS gives higher weight to large error, as they are squared. As we see in the table above, linear regression performs better in every metric. Mean Absolute error is lower in Linear regression than XGBooster as we have few outliers as already mentioned.

4.2 CC and COD

As we already saw in the data exploration, correlation is so weak and that is getting reflected in CC in the table. COD is the metric of that depicts the model from a horizontal baseline. And, even XGBooster seems to be a bit better than the horizontal curve.

Note: Date of buying seems to have a good positive correlation. Including this column would increase the accuracy of the model so much.

Important: I was struggling with the amount of data initially, so Prices were imported as np.int32 type instead of the default np.int64. Also, XGBooster uses tree data structure and exploits all possible CPU cores of the system.