Capstone Project: Zillow House Pricing Competition

By Diego Gallegos Garcia

Definition: Problem Statement and Metrics

The Capstone Project for Machine Learning is about the Zillow House Prediction Kaggle Competition. The competition posed the problem to improve the algorithm that Zillow uses to predict house prices based on a dataset that contains a previous period of house sales. The dataset not only contains house sales but also a series of features about the homes such as square footage, number of bedrooms and bathrooms along with the location of the house. The house prices that are going to be assessed is the last quarter of 2017 which was available after the public dataset phase was done. The fact that we have the sales which is the variable that must be predicted, it indicates that the problem at hand is classified as a supervised learning problem. A small set of algorithms have been chosen to be used on the competition such as Gradient Boosting Decision Trees and to improve the accuracy of the algorithm meta ensemble (stacking) has been implemented. The competition uses the logarithmic error between the Zillow House Sale algorithm prediction (Zestimate) and the real house sale price. Its worth to mention that real house sale prices are not available to competitors for some of the months. (Zillow, 2018)

The metric can be defined as follows:

$$Logerror = \log(Zestimate) - \log(SalePrice)$$

An important remark is to know a little about Kaggle competitions dynamics. The first stage of the competition consists in submitting prediction against the public dataset which contains limited data regarding newer house sales events and the private dataset that has the whole universe of training examples that assesses the ability of the predictor generalize to newer data. This is a very important step as there is a very thin line between creating accurate models and creating models that can only adapt to data that it already knows.

Data Exploration and Preprocessing

During the preprocessing phase, the variable types are considered to chose what techniques to apply. The chosen technique depends whether the model being used is tree-based or non-tree based. As the model being used is a tree based model, the numerical values are not scaled (min-max or standard scaling).

Exploration

The first step towards exploring data that was taken, was to analyze the target variable. The logerror distribution graph helps assess the skew-ness and normality of the target variable.

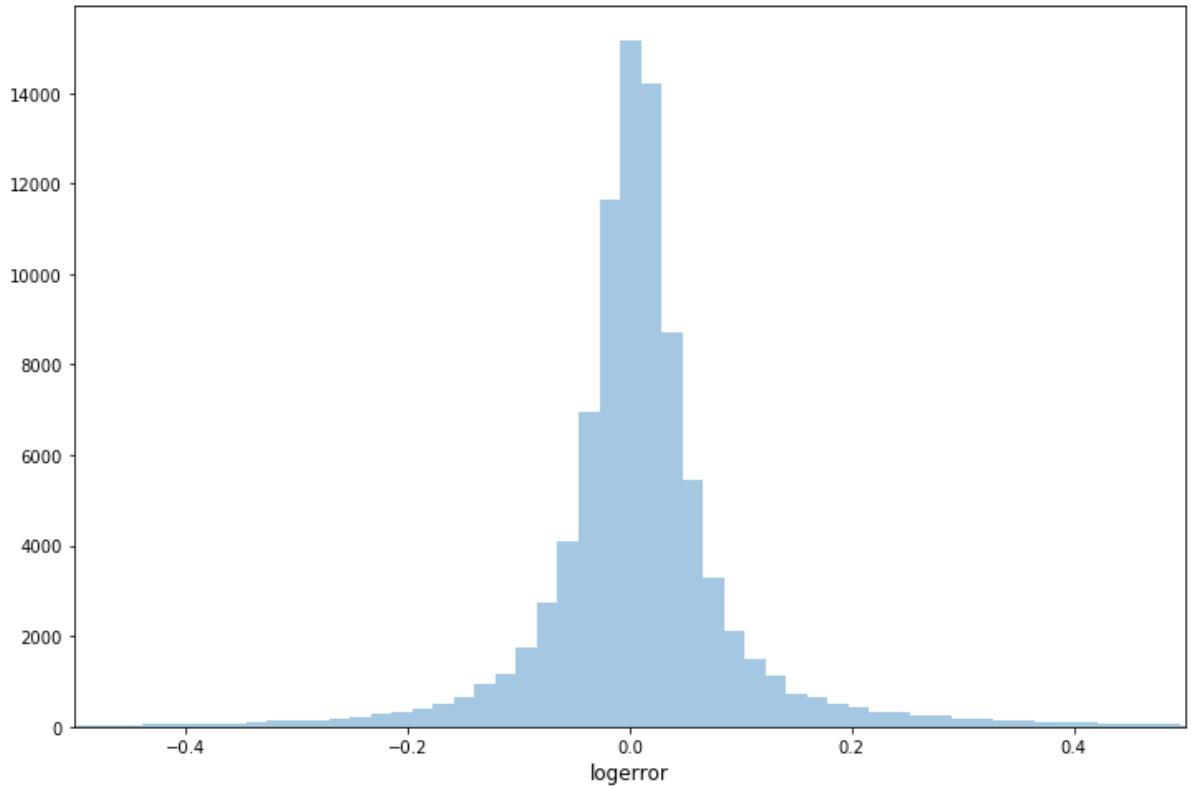

Figure 1 Logerror normal distribution

The data distribution across the dates of interest contains enough data points for each month to make learn accurately the last quarter of the year. One important thing to notice is the small amount of data points for the the last quarter, this is done on purpose as it was revealed for assessing the algorithms predictive power during the private dataset phase.

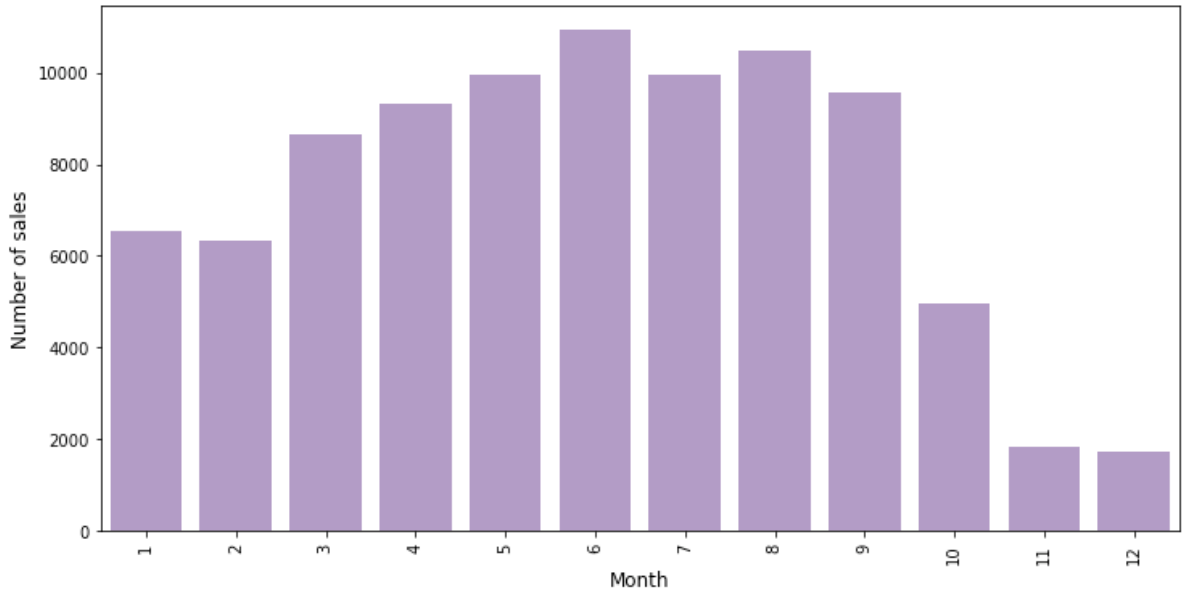

Figure 2 Monthly Sales Prices

Missing Values

Next, the training features must be analyzed to find out what pre-processing step do they need. First, the amount of values that are null are plotted to know the sparsity of the dataset. As it can be seen, there are variables that for most of the training examples the values are null. Training features with sparsity greater than 80% were discarded from the dataset.

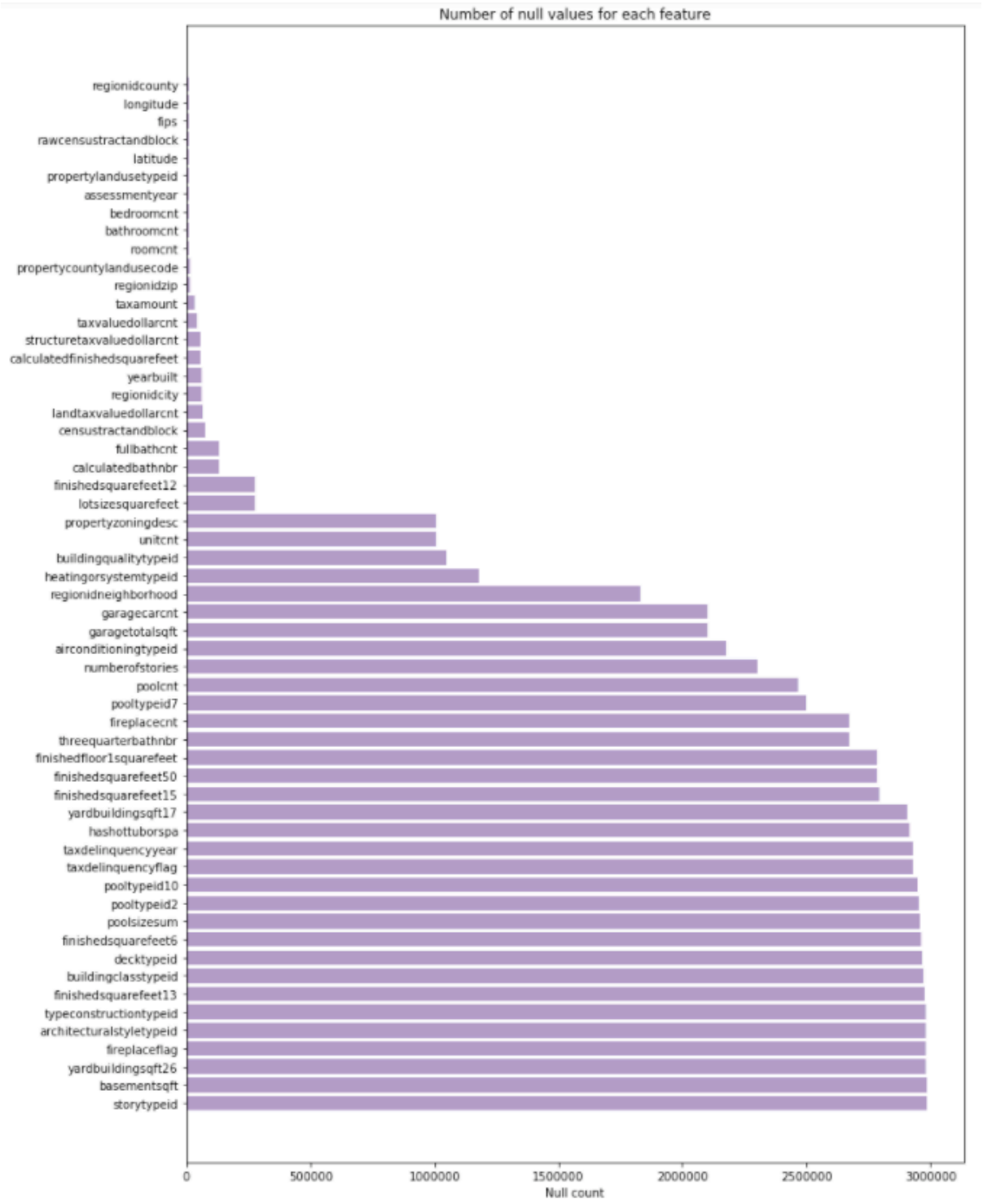

Figure 3 Number of missing values per feature

The next step, towards pre-processing the training features is imputation or also called handling missing values. Some of the methods tried were:

1. Filling the value with a number outside of the missing value range. (-1, -9999)
2. Fill the value for mean and median
3. Reconstruct value by interpolation.
4. Create a column that is called **isnull** (Boolean) that indicates whether a value is null or not.

We made experiments for the first two methods and stick with filling values with mean values. Filling values with median led to the same results.

Correlation

Correlation is used to investigate the relationship between continuous variables. This is analysis was conducted only to compare it with the results of the Feature Importance Analysis that uses Xgboost (Boosting Trees) to calculate by entropy techniques what features have greater weights towards the decision of predicting house pricing. As it can be seen on the plot of correlation of variables with the target variable it can give us an insight of which variables are important than others to predict the logerror correctly. Correlation must be handled with caution as it is known that correlation does not imply causation and that why it is used a comparative and exploratory analysis.

The values with greater positive and negative correlation are plotted.

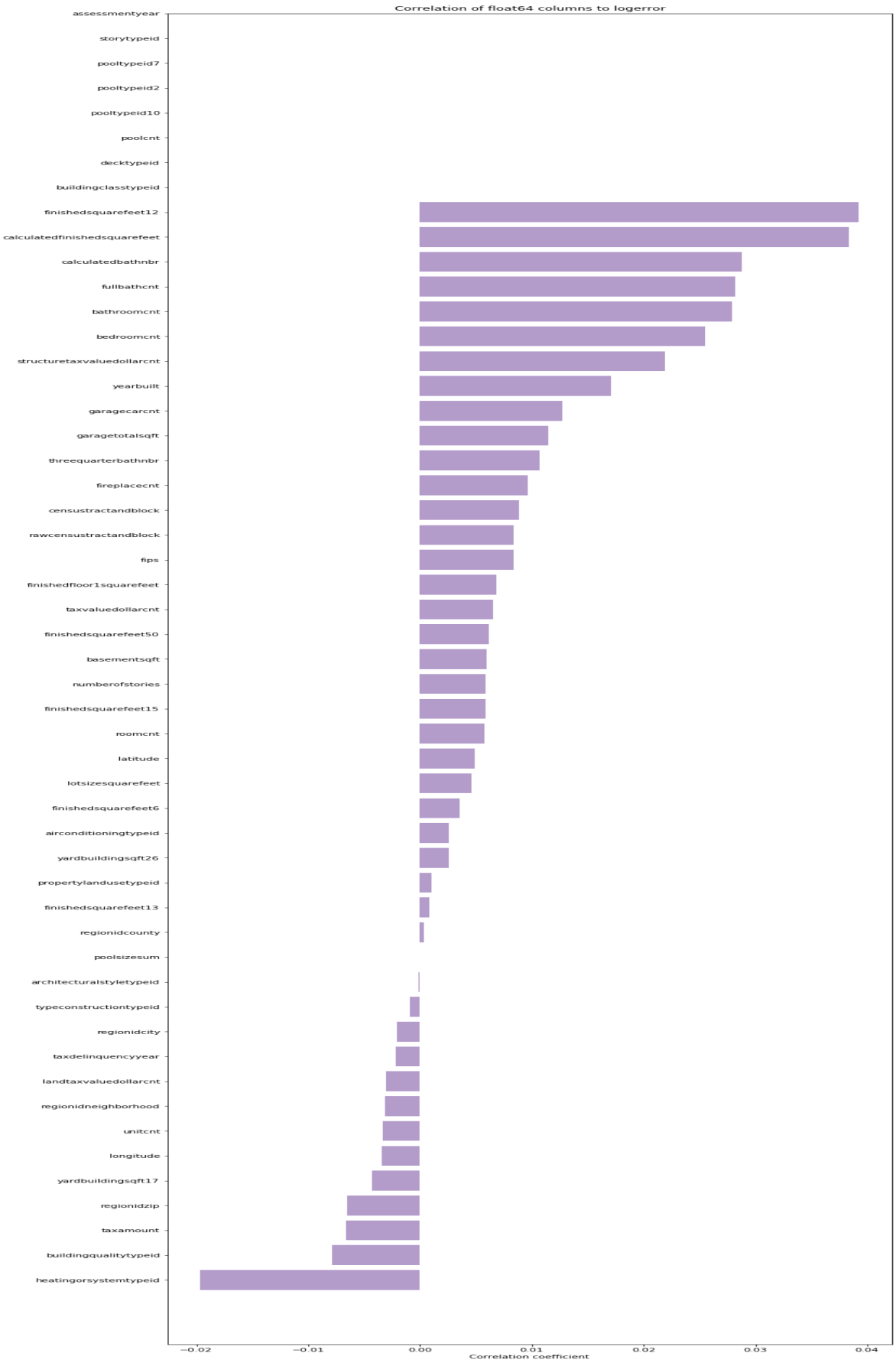

Figure 4 Correlation of variables with respect to target

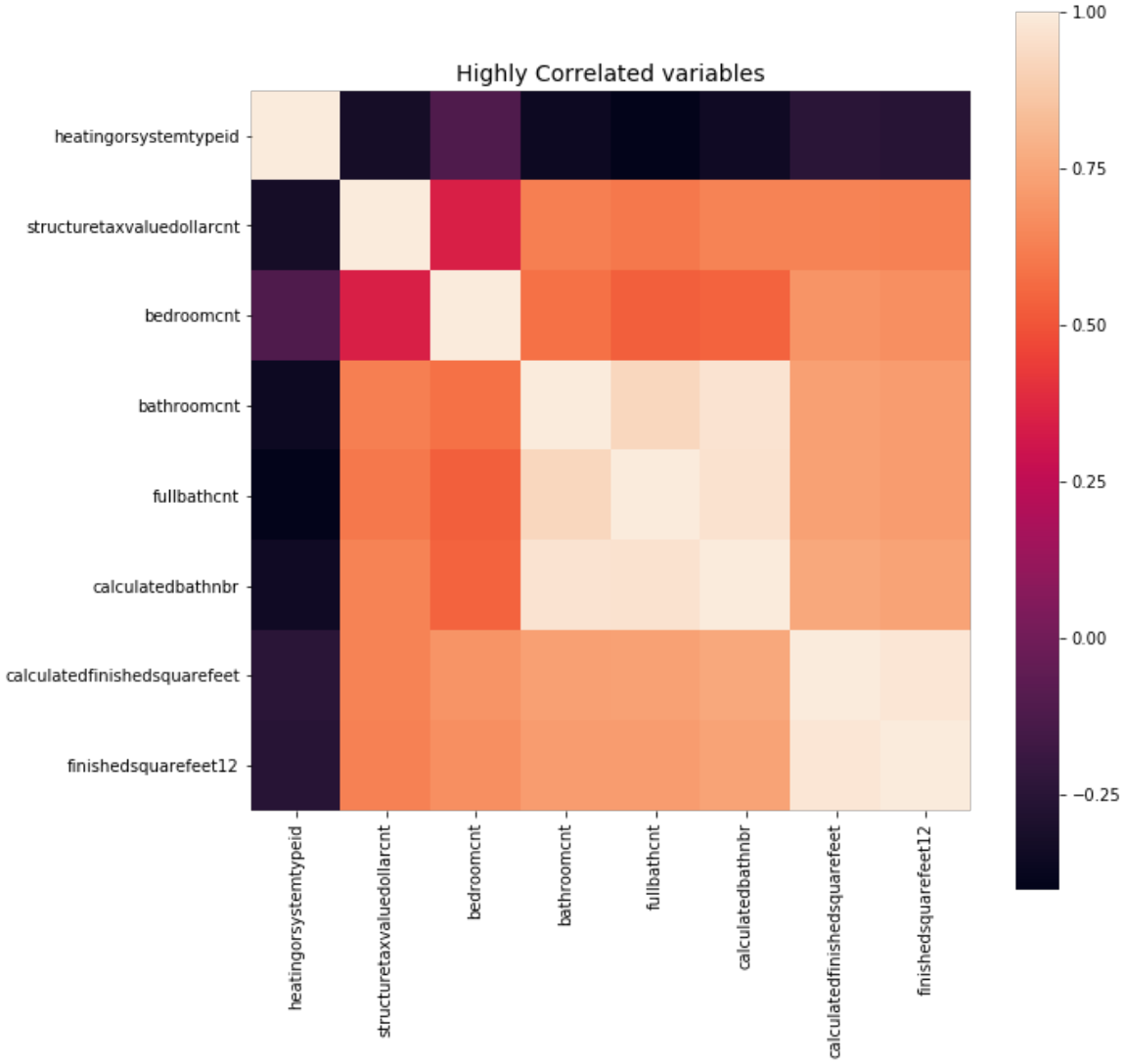

Figure 5 Most correlated features heat map with r/target

Feature Importance

Now, based on the correlation plots and its results it is desired to compare it with a most robust features importance algorithm based on decision trees. Decision trees use entropy as its voting mechanism to determine on which variable to split. Entropy obtains the best split based on the which split conserves the most information. Which can be defined as follows, where H denotes the entropy and p for the probability of occurring the event x where x is training feature and then the sum of all features it is taken.

$$H(X) = \sum_{i}^{x} -p_i(x)\, log_2(p_i(x))$$

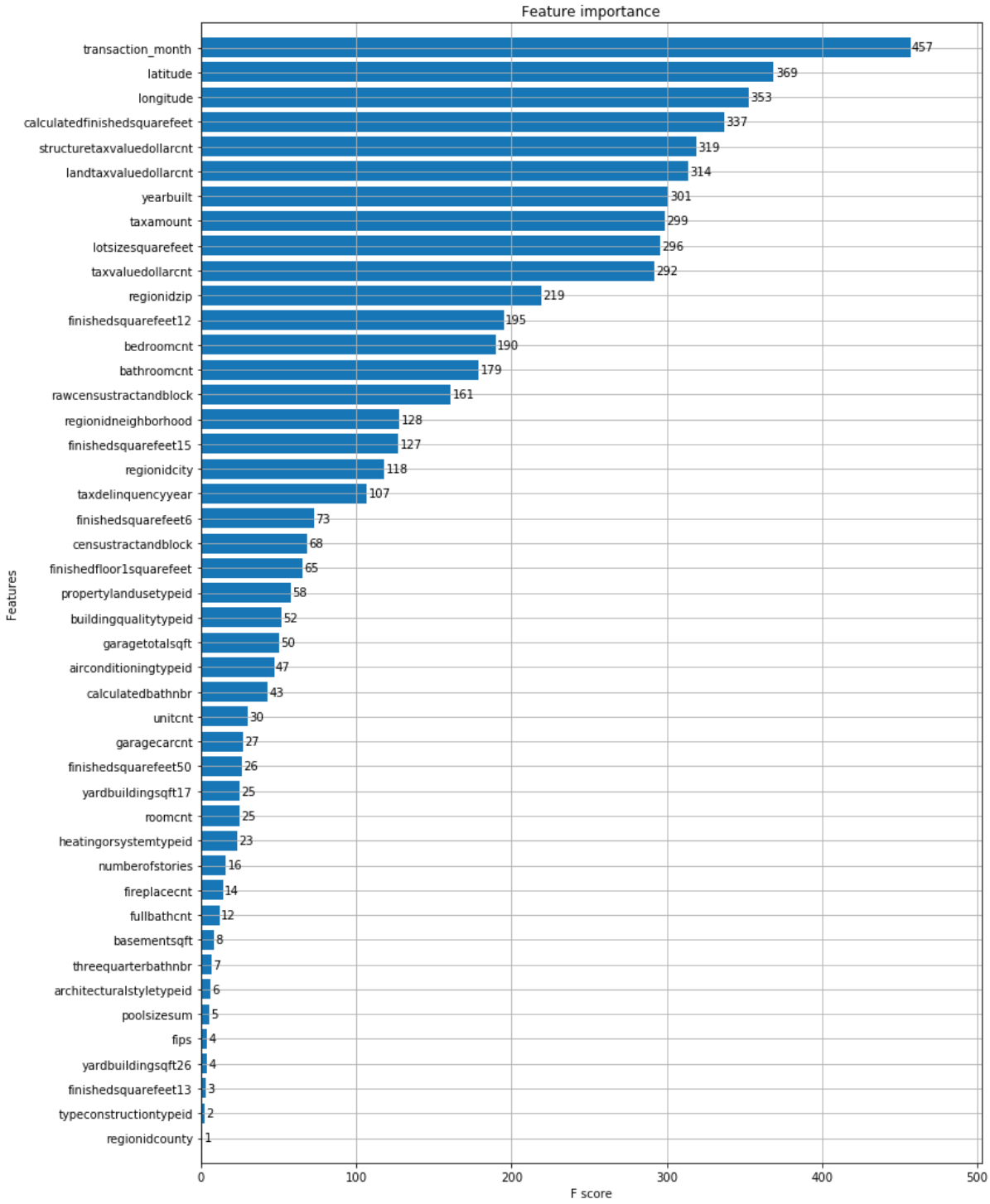

Figure 6 XGB feature importance

Feature Construction

As there were some variables discarded to high count of null values, new variables were constructed. The constructed features are:

Table 1 Engineered features

N-life	Amount of years since property was built to today
N-LivingAreaError	Error between real to theoretical area
N-LivingAreaProp	Ratio of constructed area
N-LivingAreaProp1	Ratio between permiter and area
N-ExtraSpace	Non constructed area
N-ExtraSpace-2	Difference between perimeter and area
N-TotalRooms	Bathrooms multiplied by bedrooms
N-AvRoomSize	The estimated size of each room in the house
N-ExtraRooms	Rooms that are bedrooms
N-ValueProp	Ratio of built home cost to land cost
N-GarPoolAC	Property has garage, pool or hottub and AC
N-location	Sum of coordinates
N-location-2	Multiply coordinates
N-location-2round	Round location-2
N-latitude-round	Round latitude to 4 decimals
N-longitude-round	Round longitude to 4 decimals

One important thing to notice is that most important features are used to produce new feature variables.

Algorithm and Techniques

Gradient Boosting Trees

Boosting works by sequentially applying a classification algorithm to reweighted versions for the training data and then taking majority vote of the sequence of classifiers thus provided. (Jerome Friedman, 2000)

Gradient boosting is base on this same principle. We'll start over a simplification of the using decision trees to solve a regularization problem. Let suppose we want to fit a decision tree to a dataset which is defined as follows:

$$F_1(x) = y$$

where F_1 represents the model and x is matrix that contains the training features. Then based on additive models, in this case an additive regression model.

$$F(x) = \sum_{j=1}^{p} f_f(x_j)$$

Where f_j is a separate function for each of the p feature variables x_j, technically its said that f_j belongs to a small, prespecified subset of the feature variables. In this case, we can say that p decision trees are fitted to dataset x. Now, lets explain what are this functions f_j over which x is fitted. Various algorithms have been used for additive models and one of them that is illustrative of them is a backfitting iterative algorithm where the result of fitting the model is subtracted to the real value. So for example, for the first fitted model F_1 , we have the following:

$$h_1(x) = y - F_1(x)$$

where h_1 is called the residual. The next model will be trained on this residual value and then added back to the first trained model. This continues iteratively and we can formally express this update rule as follows:

$$f_j(x_j) \leftarrow y - \sum_{k \neq j} f_k(x_k)$$

Remembering that when we defined the additive regression model we started from j = 1 until p. So this sort of approach takes weak learners (algorithms that has the probability to have better performance than chance). This is what gradient descent boosting knows as partial residuals.

This is a simplified cost function where gradient descent is being applied.

This composes the theoretical framework of XGBoost and Lightgbm which uses optimization beyond the scope of this paper. One more important thing to mention is the whole picture with respect to lost function as it not single-handedly depends on the model but also handling bias. This is solved by adding regularization terms and this boosting libraries have a well thought way of handling this:

$$\mathcal{L}(\phi) = \sum_i l(\hat{y}_i, y_i) + \sum_k \Omega(f_k)$$

$$\text{where } \Omega(f) = \gamma T + \frac{1}{2}\lambda \|w\|^2$$

In this equation taken from the original XGBoost paper (Tianqui Chen), here L represents the loss function and l is the model, the new part is Ω where is represent the regularization term. Regularization is a way of artificially prevent overfitting by discouraging complex predictions. On most reading, L1 and L2 methods are used (Nagpal, 2017) but this libraries uses the number of leaves (T) and weights that need to be learned to have an adaptative way of preventing overfitting on each iteration. Another important features, is the way it selects split candidates. On most academic introduction to decision trees entropy or gini measures are taught but the way XGBoost does is it depends on the the current and previous trees. Also, they try to optimize resoures such as cache and core usage.

Neural Network

A neural network was also used to train the dataset. Neural networks simulate the learning process of human neurological system and it tries to model the activation process of neurons when transmitting data. On figure X, we can see a representation of an artificial neural network where it receives an input (in orange), in our case, the dataset. The neural net is composed of different layers. There are three types of common layers, input, output and hidden layer. Each circle represents a neuron which is interconnected to other neurons, what is called as fully connected. This neurons have an initial weight assigned (normally initialized from a random normal distribution centered with zero mean). The blue neurons belong to hidden layers. The output of each hidden layers takes each value from previous layers and process this values through activation functions. This activation functions are what allow neural networks to model more complex functions, not only linear.

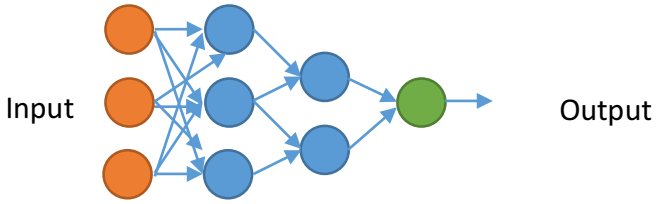

Figure 7 Neural Network Layers

In our case, we use rectified linear units (R). The algorithms that takes care of calculating the forward pass of values from the input to the output is called **feedforward propagation**. If we called X the input and W the initial weights of each neuron. The output of the first layer will be:

$$h_1 = R(Wx)$$

The output of this hidden layer then becomes on the output of the next hidden layers until the output is calculated. The following step, is to calculate the error or loss function which is the target to be optimized. Optimization has to deal with calculating derivatives and is what **backpropagation algorithm** is about. It calculates the derivative of the loss function (L) with respect to the weights that we want to optimize it values as it is the only independent parameter. This involves calculating the intermediate derivatives of this hidden layers, so it will look as follows:

$$\frac{\partial L}{\partial W} = \frac{\partial L}{\partial R}\frac{\partial R}{\partial h_n}\frac{\partial h_n}{\partial W}$$

Where h_n represents the feedforward values of the n hidden layers. After the error or cost is calculated, then the weights have to be updated accordingly.

$$W \rightarrow W - c * \frac{\partial L}{\partial W}$$

where c represents a constant which is a ratio of the learning rate and the number of features. Notice the simplification on the operations, which are considered to be tensors.

In our case we used Keras, which is a framework that constructs a computational graph which enables the easy calculation of derivatives of any neural network configuration. Keras is built on top of a backend which can be Tensorflow or Theano.

Hyperparameter optimization

K fold cross validation was used to optimize hyperparameters in the case of gradient boosting (Jain, 2016). Sklearn provides GridSearchCV functions that implements k-fold over the training data over a grid of parameters. This process takes days on regular CPUs which made us to use cloud providers. Specifically, we used hetzner.com machine SX131 which has 64GB RAM, 8 cores and Xeon CPU. On a regular, macOS device with 16RAM it can take over 3 days to run just one grid search for a couple of parameters. A code example is left on the final notebook.

Meta Ensembling

This term refers to the combination of several models to obtain a single prediction. Primarily, this is based on the assumption that no model is perfect but also finds inspirations in boosting which takes several weak learners to obtain strong learner. In practice, specially in Kaggle competitions there have been several implementations of stacking or meta ensembling. In our case, we trained four models. The models are

- Lightgbm (Lightgbm Github)
- Xgboost ((Xgboost Github)
- Neural Network
- Ordinary Least Squares

The final solution consists in a weighted average of the output of all this models. The winning submission had a score of 0.0740861253. Our final submission had 0.0754311. When the winning solution was published it used a similar stacking method to combine several models and retraining the output of other models to have a more robust model which is something very similar as boosting methods do that they train more models based on previous outputs.

Conclusion

Participating in a very diverse community as Kaggle. I had the opportunity to discover practical approaches towards solving machine learning problems. Also, collaboration was a key factor to have finish the competition with a satisfying result for being the first time participating. The kernels that other competitors share was a baseline form learning new ways to answer questions about data and learning the practical explanation for many algorithms used. Being exposed to boosting methods has been a great exposure to professional machine learning solutions.

Bibliography

Jain, A. (2016). *Xgboost Hyperparameter Optimization Article*. Obtenido de Analytics Vidhdya: https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/

Jerome Friedman, T. H. (2000). Additive logistic regression: a statistical view of boosting. *Annals of Statistics* .

Lightgbm Github. (s.f.). Obtenido de https://github.com/Microsoft/LightGBM

Nagpal, A. (13 de October de 2017). *Towards Data Science*. Obtenido de L1 and L2 Regularization Methods: https://towardsdatascience.com/l1-and-l2-regularization-methods-ce25e7fc831c

Tianqui Chen, C. G. *XGBoost: A Scalable Tree Boosting System*. University of Washington. University of Washington.

Xgboost Github. (s.f.). Obtenido de https://github.com/dmlc/xgboost

Zillow. (2018). *Kaggle*. Obtenido de Zillow's Home Value Prediction: https://www.kaggle.com/c/zillow-prize-1