

**CZ4041 Machine Learning**

**Project Report**

**Group 22**

**SCHOOL OF COMPUTER SCIENCE AND ENGINEERING**

**NANYANG TECHNOLOGICAL UNIVERSITY**

1. **Introduction**

This project aims to expose students to real-life application of machine learning techniques.

* 1. **Team members**

The team consists of five people and each member is in charge of exploring various machine learning techniques that could be used to tackle the problem.

| **Name** | **Matriculation No.** | **Role & Responsibilities** |
| --- | --- | --- |
| Renice Loh | U1822247D | Coding & Report |
| Lim Shi Hao | U1823110B | Coding & Report |
| Xing Wanting | U1721577A | Video |
| Elroy Liang | U1720486H | Coding & Video |
| Chen Gangzhe | U1822840E | Video |

* 1. **Problem Statement**

A house is a big-ticket item and often the most expensive purchase a person makes in his or her lifetime. Hence, making an informed choice is critical to them. Zestimate was created to provide consumers an estimate of property prices using the available information, regardless of whether you want to sell a house, buy a house or rent a house.

* 1. **Competition Details**

This competition, Zillow Price: Zillow’s Home Value Prediction (Zestimate), was carried out in 2017. The contest was structured into 2 rounds, the qualifying round and the private round. In our case, we will be doing a late submission for the qualifying round. In the qualifying round, participants are required to build a model to improve the Zestimate residual error.

**logerror = log(Zestimate) - log(SalePrice)**

The Zestimate residual error is also known as the log error, with its formula shown above. Submissions are evaluated using the Mean Absolute Error (MAE) between the predicted log error and the actual log error. Participants will be predicting the log error for the properties in 6 timepoints (201610, 201611, 201612, 201710, 201711, 201712). The team with the lowest MAE for these predictions will win. A total of 3770 teams participated officially in the competition.

* 1. **Objective**

In this competition, the team is required to come up with a machine learning model, Zestimate (a home valuation model for Zillow), to estimate the market value of a house more accurately. This is done by incorporating public and user-submitting data, taking into account home facts, location as well as market conditions.

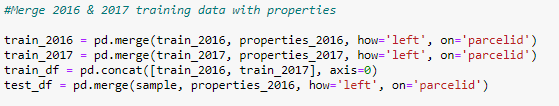
* 1. **Data**

In total, 6 comma-separated-value (.csv) files are provided - a training data set: *train\_2016\_v2*, *train\_2017*, *properties\_2016.csv*, *properties\_2017.csv* and a testing set: *sample\_submission.csv* as well a data dictionary *zillow\_data\_dictionary.csv.*

* 1. **Methodology**

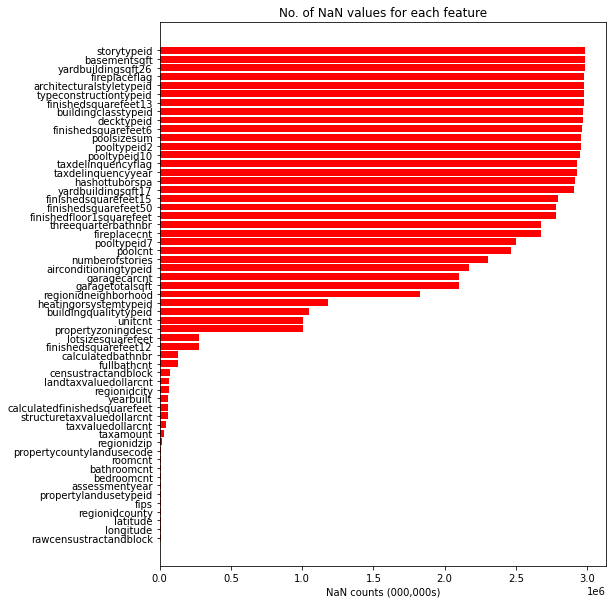
1. Perform data exploration to manually determined data patterns
2. Perform data visualisation to further determined data patterns
3. Perform data cleaning
4. Conduct feature engineering to prepare dataset for the models
5. Implement models and train models using given data
6. Use the models to predict value of houses
7. **Data Preprocessing**

We first merge *train\_2016\_v6.csv* and *train\_2017.csv* with its respective properties on ‘parcelid’ and then concatenated the two to get the training dataset. To obtain the test data, we merge the *sample\_submission.csv* and *properties\_2016.csv*.

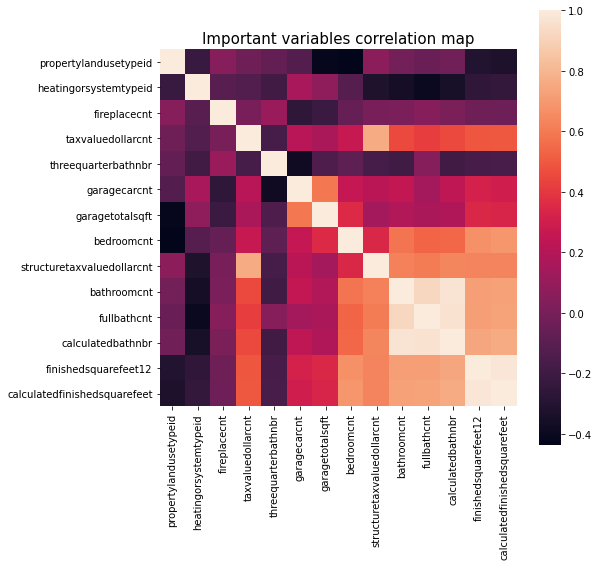


* 1. **Data Exploration**

To better understand the dataset we are working with, we started off by looking for missing data.

****

As shown in the figure above, there exists an overwhelming number of missing values in the dataset. Out of the 60 features in the dataset, 29 of the features had 50% or more “NaN” values. It was clear that the biggest challenge of this project will be to deal with the missing data appropriately before any training.



To understand the features better, correlation of each feature with the target variable (logerror) was computed. The features with the highest correlation with logerror was then plotted in the figure above to illustrate the correlation of each feature with each other.

As can be observed, some features have very high correlation with each other, thus, can be reduced during pre-processing. For example, ‘calculatedfinishedsquarefeet’ and ‘finishedsquarefeet12’ have very high correlation. It makes sense since both features seem to represent the area of a property.

* 1. **Data Cleaning**

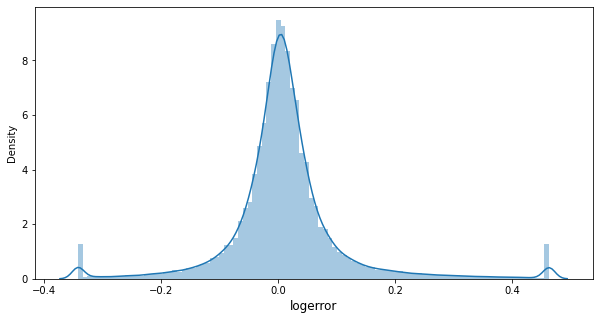
In this step, we clean the data by removing unnecessary data that will not help training and handling missing data.

* + 1. **Outliers**

When doing data exploratory using data visualization, we realise that log errors with value less than -0.4 and more than 0.42 are rare. Thus we remove as it reduces accuracy on the training of the models.

****

After removing the outliers, a gaussian distribution can be observed for our target variable.

****

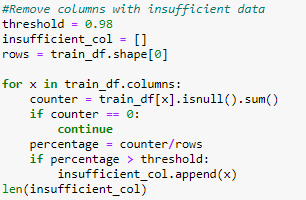
* + 1. **Fill in blanks**

Certain algorithms deal with missing data internally - for example, lightGBM ignores missing values during a split and allocates them to whichever side reduces the loss the most. However, we try to impute as much information as possible. We do this by filling in blanks with the value -999.



* + 1. **Insufficient data**

We remove insufficient data as it will do more harm to accuracy by injecting the noise. In this case, the threshold is 0.98. We do this by calculating the missing percentage and removing those columns with missing data more than threshold. A total of 13 features were dropped due to insufficient values.



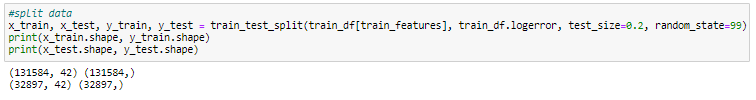
* + 1. **Unnecessary data**

Some other variables include ‘parcelid’, ‘logerror’, ‘propertyzoningdesc’ and ‘propertycountylandusecode’ which does not help the training of models are removed as well.

****

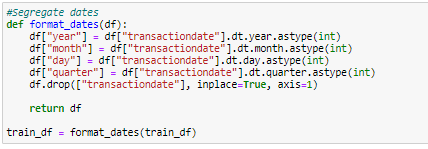
* + 1. **Split data**

Finally, we split the data into training and test data sets. The training to test ratio is 80:20. A total of 42 features are remaining.



* 1. **Feature Engineering**

To handle the date feature, manual encoding was performed. By encoding the ‘transactiondate’ into its 4 sub features, ‘year’, ‘month’, ‘day, and ‘quarter,’ the features are easier to train on as we casted each of them as an Integer type instead of datetime format the original feature.

****

1. **Solutions Proposed**

Regression predictive modeling is a task for continuous output variables. As our goal is to predict logerror, which is a continuous variable, we implemented 3 gradient boosting models for regression.

Gradient boosting on Decision Trees (GDBT) is a machine learning technique used to train complex models. In our case, our dataset is large and contains many categorical and numerical features, hence GDBT is a suitable technique to use.

To model the regression, we initially used the CatBoost model. CatBoost is a relatively new method based on gradient boosted decision trees. Since it was designed to take advantage of categorical data, it should perform very well with our dataset. Next, we implemented the 2 most commonly used boosting machines, LightGBM and XGBoost, and compared each model with our original CatBoost model. At last, we combined the predictions of each model in an attempt to improve our final prediction accuracy. The 3 models will be discussed extensively in the following sections.

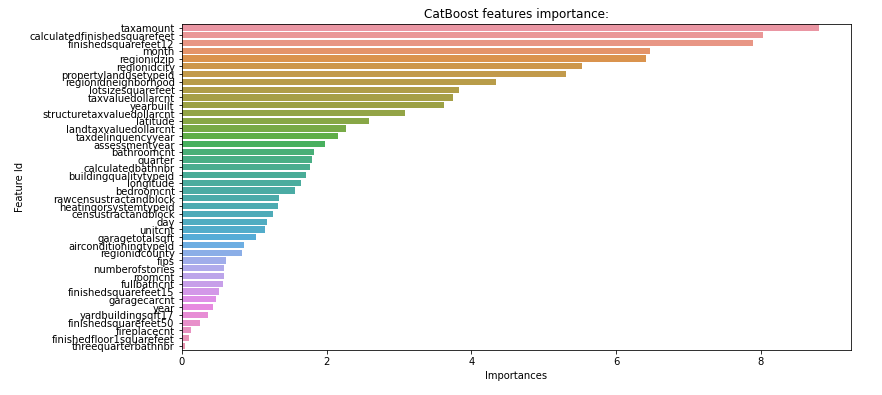
* 1. **CatBoost Model**

CatBoost is a gradient boosting library on gradient boosted decision trees that supports sophisticated categorical features. Similar to XGBoost and LightGBM, CatBoost builds a set of decision trees consecutively. Each successive tree is built to reduce loss as compared to the previous trees. The algorithm stops once overfitting has been detected. We chose this as it is a robust model and has low chances of overfitting, which reduces the need for extensive hyper-parameter tuning.

* + 1. **Implementation**

We used CatBoost - an open-source software library developed by Yandex’ to implement the CatBoost model, along with supporting libraries such as Pandas and Numpy.

* + 1. **Feature Importance**

****

**Figure: CatBoost Model Feature Importance**

From the graph plotted using the CatBoost model, the top five factors that affect the house pricing most significantly are ‘taxamount’, ‘calculatedfinishedsquarefeet’, ‘finishedsquarefeet12’, ‘month’ and ‘regionidzip’.

* + 1. **Experiments**

We adjust the parameters learning\_rate and depth to prevent overfitting.

| **learning\_rate** | **depth** | **l2\_leaf\_reg** | **Private score** | **Public score** |
| --- | --- | --- | --- | --- |
| **0.04** | **6** | **3** | **0.07538** | **0.06449** |
| 0.03 | 10 | 3 | 0.07539 | 0.06456 |
| 0.04 | 3 | 3 | 0.07545 | 0.06458 |

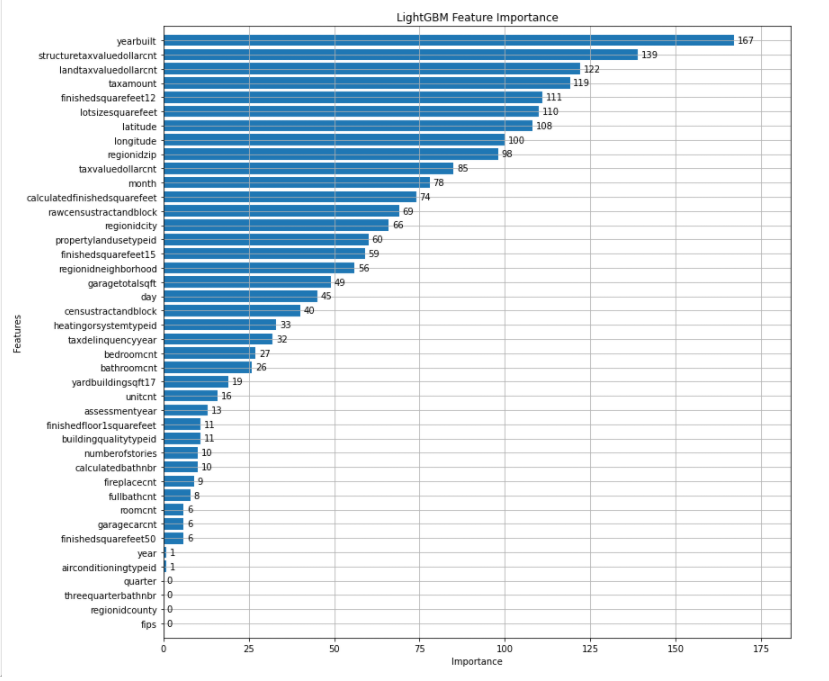
* 1. **LightGBM**

LightGBM is a fast, distributed, high-performance gradient boosting framework based on a decision tree algorithm. Unlike other algorithms, it grows trees vertically (leaf-wise). We choose to use this model as it is suitable for big datasets as it handles big datasets well with low memory usage and gives accurate results.

* + 1. **Implementation**

We use the open source lightGBM model developed by Microsoft to implement the lightGBM model, along with supporting libraries such as Pandas and Numpy.

* + 1. **Feature Importance**



**Figure: LightGBM Feature Importance**

From the graph plotted using the LightGBM model, the top five factors that affect the house pricing most significantly are ‘yearbuilt’, ‘structuretaxvaluedollarcnt’,’landtaxvaluedollarcnt’, ‘taxamount’ and ‘finishedsquarefeet12’.

* + 1. **Experiment**

In order to avoid overfitting, the parameters ‘num\_leaves’ and ‘max\_depth’ are adjusted. On the other hand, we adjust the parameter ‘learning\_rate’ to avoid overstepping. The default values for num\_leaves, max\_depth and learning\_rate are 31, -1 and 0.1 respectively.

| **num\_leaves** | **max\_depth** | **learning\_rate** | **Private score** | **Public score** |
| --- | --- | --- | --- | --- |
| 31 | -1 | 0.1 | 0.07654 | 0.06536 |
| 15 | -1 | 0.1 | 0.07592 | 0.06489 |
| 5 | -1 | 0.1 | 0.07560 | 0.06463 |
| 5 | 6 | 0.1 | 0.07560 | 0.06463 |
| **50** | **12** | **0.01** | **0.07547** | **0.06460** |

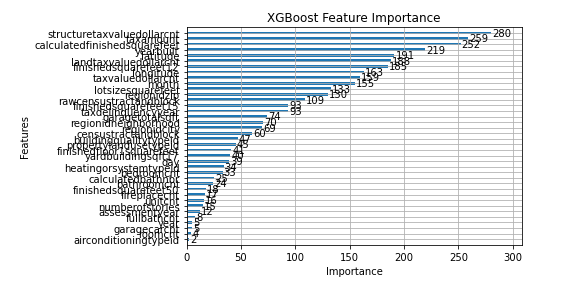
* 1. **XGBoost Model**

XGBoost is an extreme gradient boosting framework that implements gradient boosted decision trees. It has algorithm features which includes sparse aware implementation, block structure and continued training. We choose this model as it is an improved version of the boosted tree algorithm designed to focus on model performance and computational speed.

* + 1. **Implementation**

The preprocessed training data is used to construct a training DMatrix to train the XGBoost model. DMatrix is an internal data structure used by XGBoost which is optimised for both memory efficiency and training speed. The implementation of XGBoost model is done using the XGBoost open-source software library, along with supporting libraries such as Pandas and Numpy.

* + 1. **Feature Importance**

****

**Figure: XGBoost Feature Importance**

From the graph plotted using the XGBoost model, the top five factors that affect the house pricing most significantly are ‘structuretaxvaluedollarcnt’,’taxamount’, ‘calculatedfinishedsquarefeet’, ‘yearbuilt’ and ‘latitude’.

* + 1. **Experiments**

The learning rate represented by eta has a default value of 0.3 and shrinking the value makes boosting more conservative. The maximum depth of the tree represented by max\_depth has a default value of 6 and increasing the value makes the model more complex and prone to overfitting. Subsample ratio of the training instances represented by ‘subsample’ has a default value of 1 and its value helps prevent overfitting.

| **eta** | **max\_depth** | **subsample** | **Private score** | **Public score** |
| --- | --- | --- | --- | --- |
| 0.3 | 6 | 1 | 0.06865 | 0.07998 |
| 0.3 | 6 | 0.8 | 0.07013 | 0.08150 |
| 0.3 | 3 | 1 | 0.06535 | 0.07636 |
| 0.03 | 3 | 1 | 0.07561 | 0.06464 |
| **0.037** | **5** | **0.8** | **0.07557** | **0.06463** |

* 1. **Ensemble**

Each model has its strengths and limitations. For example, CatBoost is designed for categorical data and is known to have the best performance on it. Therefore, ensemble learning might be another approach to improve machine learning results. Ensemble methods are meta-algorithms that combine several machine learning techniques into one predictive model in order to decrease variance, bias or improve predictions.

* + 1. **Implementation**

In this project, we do a weighted average of the three models namely, LightGBM, CatBoost and XGBoost. The final weights are 0.4, 0.4 and 0.2 respectively. We determined the values through a trial-and-error method.

* + 1. **Experiments**

| **xgb\_weight** | **cat\_weight** | **lgb\_weight** | **Private score** | **Public score** |
| --- | --- | --- | --- | --- |
| 0.2 | 0.4 | 0.4 | 0.07531 | 0.06442 |
| 0.4 | 0.2 | 0.4 | 0.07539 | 0.06448 |
| **0.4** | **0.4** | **0.2** | **0.07530** | **0.06441** |
| 0.5 | 0.4 | 0.1 | 0.07530 | 0.06443 |
| 0.5 | 0.3 | 0.2 | 0.07532 | 0.06444 |

1. **Results**

We consolidate the best results we get from each model according to Kaggle as shown in a table below. From our results, we can tell that the most accurate single model is CatBoost, followed by XGBoost and lastly LightGVM model for this dataset. This could be due to the hyperparameters tuning especially since CatBoost requires little fine-tuning of the parameters which reduces the chances of overfitting automatically.

By using the combined predictions of all models, we were able to obtain an improved private score of **0.07530**, which is the **top 14%** on the private leaderboard. In the public leaderboard, we managed to score **0.06441** which is the **top 31%** in the public rankings. The collation of all the scores and ranking of each model and the ensemble model is shown in the table below. The screenshot of the best scoring submission has been attached in the appendix.

| **Model** | **Private Score** | **Private Rank** | **Public Score** | **Public Rank** |
| --- | --- | --- | --- | --- |
| CatBoost | 0.07538 | 692nd / 3770 | 0.06449 | 1419th / 3770 |
| LightGBM | 0.07547 | 1352nd / 3770 | 0.06460 | 1629th / 3770 |
| XGBoost | 0.07561 | 1806th / 3770 | 0.06464 | 1178th / 3770 |
| **Ensemble -  Cat + LGB + XGB** | **0.07530** | **530th / 3770**  **(Top 14%)** | **0.06441** | **1184th / 3770**  **(Top 31%)** |

Another ensemble learning method that might further improve our result is through model stacking. Model stacking is done by using LightGBM, CatBoost and XGBoost as three base learners and then combining the three via a GLM or a neural network to output a new model. However this method is highly unpredictable thus we cannot say for sure that it will improve our ranking on the leaderboard.

1. **Challenges**

There are multiple challenges we faced during the span of this project.

* 1. **Missing data**

There was a lot of missing data. Some columns also had large amounts of NaN rows. This means that data preprocessing was necessary and data is not complete for training.

* 1. **Time Constraint**

There are many algorithms and techniques in the realm of machine learning. However due to the time constraint, we did not manage to explore all the different possibilities and approaches.

* 1. **Hyperparameters tuning**

There are many parameters that could be set for each model in order to improve the accuracy of the models. For example, using an early stoppage parameter minimises the chance of over fitting. However, It is difficult to determine what numbers are best for this data set as it requires high level mathematical knowledge and background. Thus we can only tune it to our best via test-and-trial method.

1. **Conclusion**

Overall, this project has been insightful for all of us as we get to explore different models and techniques. We realised that predicting the house pricing accurately is a feat due to the number of variables involved as well as the number of missing data. We learnt that data preprocessing and feature engineering are vital steps in Machine Learning and can greatly determine the accuracy of the model. We managed to attain decent scores using an ensemble of boosting machines.

It has been a great experience for us to apply the knowledge we have learnt from the lectures to this project and see tangible results of our application. This course has been beneficial to us and will be a great stepping stone for our future in the field of Machine Learning.

1. **Appendix**

