

CZ4032 Data Analytics and Mining

Group 27 Project Report

AY19/20 Sem 1

|  |  |  |
| --- | --- | --- |
| No | Name | Matriculation Number |
| 1 | Kyle Huang Junyuan | U1721717G |
| 2 | Russell Chua Kui Hon | U1720526F |
| 3 | Nelson Ko Mingwei | U1721410B |
| 4 | Ngoh Guang Wei | U1722281K |
| 5 | Wilson Neo Yuan Wei | U1721538L |
| 6 | Khin Yamin Thu | U1721925F |

**Table of Contents**

[**Abstract**](#_yhruja6i5xi3) **3**

[**Problem Description**](#_dmwoyt7i00tf) **3**

[Motivation](#_6mvufayqn3of) 3

[Problem Definition](#_bz3mi2ylkty) 3

[Related Work](#_3wtghhneqpr2) 3

[**Approach**](#_fcq02sb56uo) **4**

[Methodology](#_wmfwnh6zdlcn) 4

[Dataset Selection](#_6xit0ivbv9tp) 4

[Data Preprocessing](#_uyzjplt5cfgl) 4

[Algorithms](#_s4nw92etl4ff) 4

[Linear Regression](#_9n368dwrb9nk) 5

[Support Vector Regressor (SVR)](#_v0s60f3bprs7) 5

[Ridge Regression](#_stpfshzgr6ds) 5

[Random Forest Regression](#_u6iawgqr3f9q) 6

[Extreme Gradient Boosting (XGBoost)](#_qx1up66cctyr) 6

[Artificial Neural Network](#_o3597ov4ugcf) 7

[**Implementation**](#_saxxc27b97sw) **10**

[Understanding the Dataset](#_lvwldoentj6g) 10

[Input Features](#_uiuv21d9m8n) 10

[Output Target](#_25lw3bs5oebd) 10

[Data Cleaning and Preprocessing](#_ejpeo91qe0i4) 10

[Replacement of null values](#_vsw806hk857s) 10

[One-hot Encoding](#_t3hrnpzf7wrw) 11

[Feature Discovery](#_8hbtyzjdtxql) 11

[Removing of outliers](#_npxzthvzdlvy) 12

[Standardization](#_kd2a0l7tnavu) 13

[**Experimental Results and Analysis**](#_prttcvapgequ) **14**

[Experimental Setup](#_cyuy2v2i5sqy) 15

[Comparison Schemes](#_s5pg2nhtwtdp) 15

[Results and Analysis](#_77wpfvddwq4f) 22

[**Discussion of Pros and Cons**](#_59n4r1tpmoxt) **22**

[**Conclusion**](#_b37skvs1qwuk) **24**

[**Summary of Project Achievements**](#_662bdsocb7se) **24**

[Directions for Improvements](#_d6e4v0kmcggh) 24

[**References**](#_1819fr8w91ly) **25**

[**Appendix**](#_1819fr8w91ly) **26**

[Datasets’ Features](#_886e8j38uuan) 26

[Features with missing values (NaN or NULL)](#_5uq8xo8zomfe) 29

[Accuracy comparison between most and least correlated features](#_fuo9q8yz7gw4) 29

# Abstract

For this project, our team has decided to take on a Kaggle competition to predict the final price of homes based on provided features using regression techniques. A dataset of 1459 samples containing 79 variables describing the aspect of the residential home is provided on the Kaggle. Techniques undertaken to predict the house prices include Linear Regression, Support Vector Regressor (SVR), Ridge Regression, Random Forest Regression, Extreme Gradient Boosting (XGBoost) and Artificial Neural Network. This report aims to provide detailed documentation of the team’s approach to solving this challenge.

# Problem Description

## Motivation

Everyone would want to live in a home that they feel comfortable in. If one is asked to describe their dream home, they would most likely have a huge variety of variables to choose, such as the room’s size, number of rooms, garden’s size, proximity to the nearest train station, etc. With each choice of the house’s features by the client, the final price of the home will vary.

## Problem Definition

However, it is difficult to understand how much influence would each choice of the features would affect the price negotiations of the house. A client may desire to have certain features for his home, but due to his budget considerations, he is unable to confirm whether he could afford such a house. On the other hand, a property owner may intend to sell his house, but is unable to confirm how much his house is worth based on his house’s features.

## Related Work

The Ames Housing dataset that our team will be using was compiled by Dean De Cock from Truman State University for data science education. It is an amazing alternative for data scientists looking for an expanded and more modernized version of the usual and common Boston Housing dataset to be used for house property’s price predictions.

There is also a similar attempt to predict property prices in Singapore where it predicts a property’s price based on distance to major roads and point of interests like restaurants and hotels. Property investors want to invest by placing their properties near major roads or point of interests to increase the demand of the properties. However investing in a multimillion-dollar property based on the investors’ intuition is risky. Hence investors decide to base their decisions on the prediction data. For further detailed explanation, refer to the following website:

<https://www.techinasia.com/machine-learning-estimate-singapore-property-value>

# Approach

## Methodology

### Dataset Selection

The datasets obtained are from [Kaggle](https://www.kaggle.com/c/house-prices-advanced-regression-techniques/data) and consists of training data (train.csv) and test data (test.csv). Detailed description and explanation of the provided datasets can be found under the [Implementation](#_saxxc27b97sw) section.

### Data Preprocessing

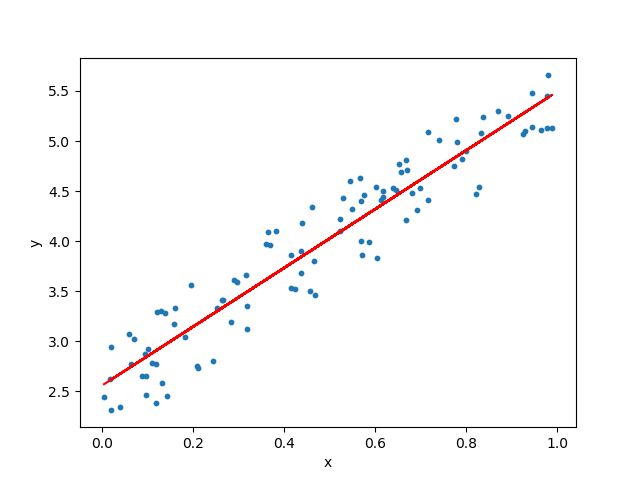
Once we obtained the datasets, we proceeded with data cleaning and preprocessing as the obtained real-world data could be dirty. Dirty data refers to the inconsistency in data values, inaccurate or incomplete data which could affect the accuracy of the prediction models. Further detailed explanation of the steps taken during data cleaning and preprocessing can be found under the [Data Cleaning and Preprocessing](#_ejpeo91qe0i4) section.

## Algorithms

Since the provided data is in the form of continuous data, we used different methods of regression as listed and described below:

* Linear Regression
* Support Vector Regression (SVR)
* Ridge Regression
* Random Forest Regression
* Extreme Gradient Boosting (XGBoost)
* Neural Network

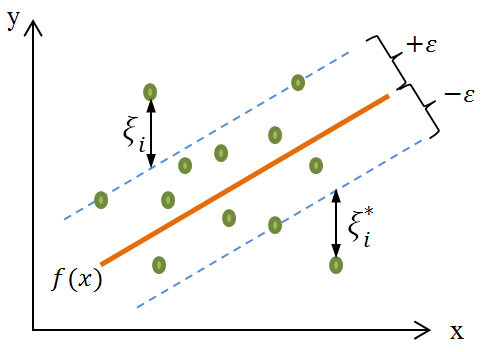
### Linear Regression



Linear Regression is one the most basic method of regression and it is a linear method to model the relationship between a scalar dependent variable and an independent variable. This method compares the various data points of the predicted and actual values and plots a linear line to represent the trend as shown in the graph diagram above.

### Support Vector Regressor (SVR)

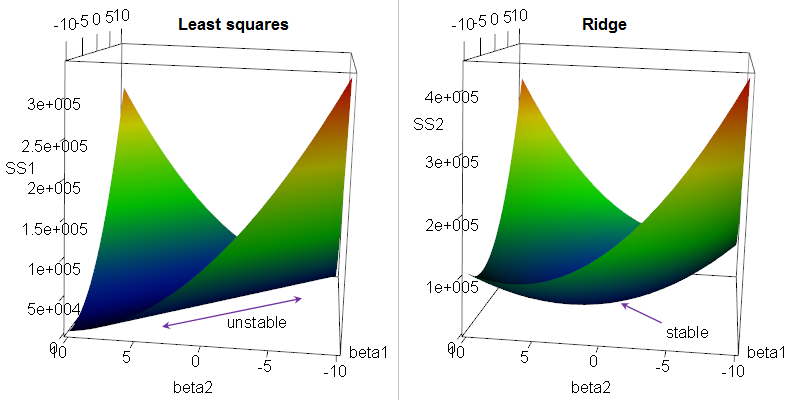
The support vector regressor is slightly different from support vector machine (SVM). SVR is used for a regression problem and working with continuous values whereas SVM is used to handle Classification problems. SVR and basic linear regression works differently. In basic linear regression approaches, a regression line is drawn to minimize the errors between the prediction and actual data, while for SVR, error will be fitted within a specific threshold as represented by the area between the 2 blue dotted lines in the diagram below.



### Ridge Regression

Multicollinearity refers to the existence of a high intercorrelated relationship between independent variables. This relationship causes disturbance in the dataset, which in turn results in unreliable statistical inferences. When multicollinearity occurs, the least squares estimates will be unbiased, but their variances may be far from the true value as the estimates’ values are large.

Utilizing ridge regression will allow us to reduce the standard errors by adding a degree of bias to the regression estimates. By including this degree of bias, we will ideally obtain a result with more reliable estimates.



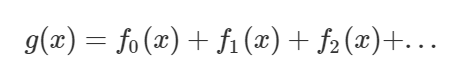
[Figure 1](#figur_least_squares_vs_ridge): Caption here  
(Adapted from medium.com/@rrfd)

Referring to the figure above, when multicollinearity occurs, we obtain a "ridge" in the likelihood function. This in turn results in a long "valley" in the Residual Sum of Squares (RSS). Ridge regression fixes this by adding a penalty that transforms the ridge into a nice peak in the likelihood space, reducing the effects of multicollinearity.

### Random Forest Regression

A random forest is a supervised ensemble machine learning technique that uses multiple decision trees and a technique called Bootstrap Aggregation or also known as “Bagging”.

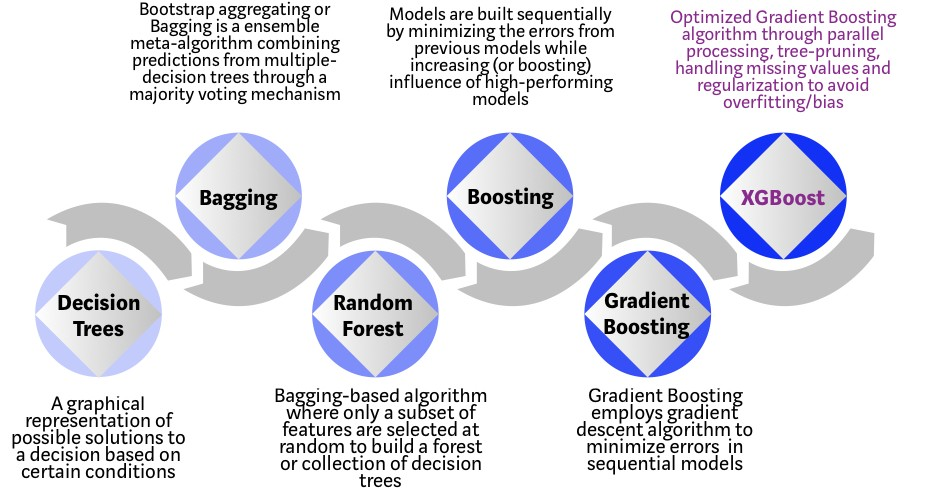
Bootstrap aggregation combines multiple decision trees by training each tree on a different data sample where sampling is done with replacement which means that some samples will be used multiple times in a single tree. This helps to get a more accurate and stable prediction, prevent overfitting and overall lower variance without increasing bias. During test time, predictions are made by averaging the prediction of each decision tree. The formula is as shown below:



The final model *g* is the summation of the simple base models *fi* while each base classifier is a simple decision tree. This technique of using multiple models to acquire better predictive performance is called model ensembling.

### Extreme Gradient Boosting (XGBoost)

The next technique explored in this project is Extreme Gradient Boosting (XGBoost or XGBRegressor). It is a decision-tree-based ensemble Machine Learning algorithm built upon the principles of gradient boosting frameworks as shown below in [Figure 2](#fig_evolutionofxgboost).



[Figure 2](#figur_evolutionofxgboost): Evolution of XGBoost Algorithm from Decision Trees  
(Adapted from towardsdatascience.com)

Since its introduction, XGBoost has caught the attention of the data science community. The algorithm is not only used to solve regression problems but can also be used in a wide range of applications as such classification problems, ranking and user-defined prediction problems.

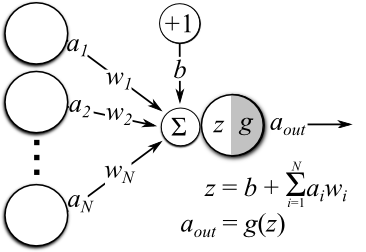
It implements machine learning algorithms under the Gradient Boosting framework and is designed to go to the extreme of the machines’ computation limits to provide a highly efficient, flexible, portable and accurate distributed gradient boosting library.

However, unlike Gradient Boosting, it is unique as it uses a more regularised model formalisation to control and reduce overfitting. It improves on the regular Gradient Boosting machines through system optimization as well as algorithmic enhancements. This results in a better performance than the base Gradient Boosting machines.

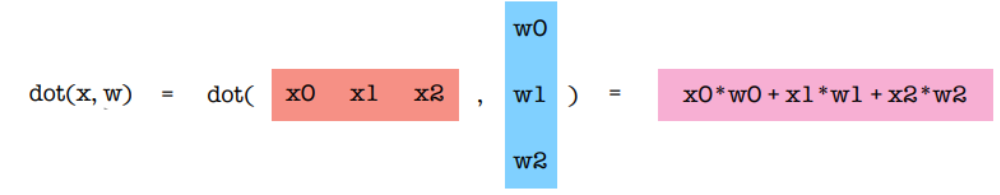
### Artificial Neural Network

The last technique explored in our project is using Artificial Neural Network with Tensorflow. The idea of Artificial Neural Network is by using biomimicry, that mimics how the human brain’s neurons work. By applying the architecture of the brain into the Artificial Neural Network that recreate the structure of a human neurons to process information resulting in more accurate result.

The architecture of a neural network consists of Neuron(Node), Connections, Bias, Weights and Layers. A Node is the basic unit of a neural network. It receives input values together with weights and biases. When an input data value arrives, weights are applied to the input which by doing the dot product of the matrix of weight and matrix of input data. If a neuron has 3 inputs, that means it has 3 weight values which can be adjusted during training time.

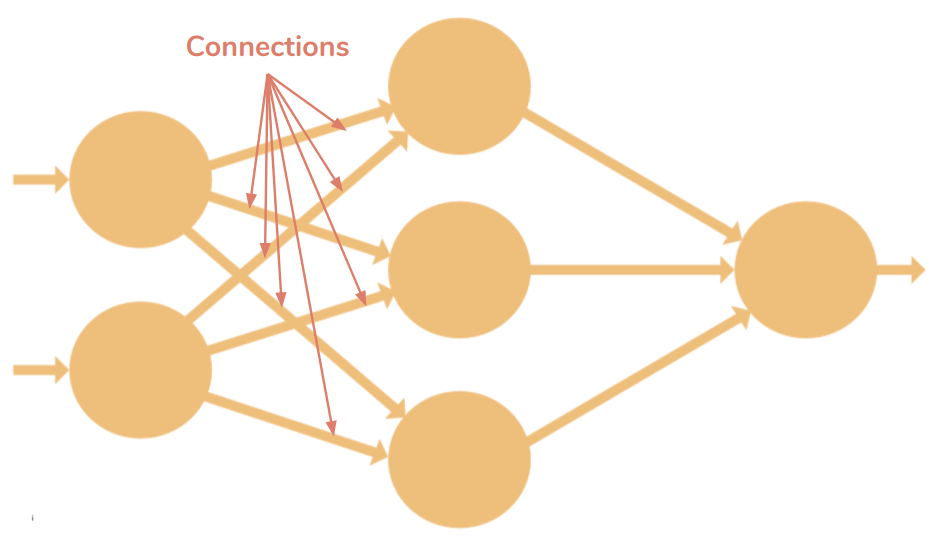


[Figure 3](#figur_single_neuron_operation): A single neuron operation with multiple inputs

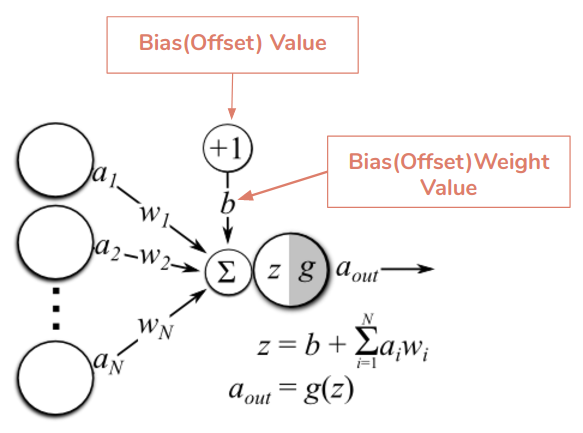


[Figure 4](#figur_dot_product_matrix_neuron): Dot product matrix in a neuron

The connection means to connect from one neuron to another neuron of a different layer or the same layer. A weight value is always associated with a connection, this is because during training the weight value are updated to decrease error output.

[Figure 5](#figur_connection_neurons): The connections to neurons

A bias is an extra input to neurons and the value of the bias can be set to 1 or other values depending how you want to structure. Bias also has it own connection weight to the neurons. Having bias is to ensure that even when all inputs are zeros (0) there will be an activation in the neuron.



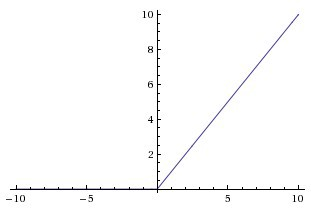
[Figure 6](#figur_bias_introduce_neuron): Bias introduce to a neuron

Activation function are used in neuron to decides whether a neuron should be activated. The decision is made by calculating weighted sum and further adding bias with it. The purpose of having activation function is to introduce non-linearity into the output of a neuron. If a neuron network without an activation function is basically just a linear regression model. Therefore activation function does the non-linear transformation to the input making it capable to learn and perform more complex tasks.

There are different types of activation function, for our model we will be using ReLu (Rectified Linear Unit) activation function for the hidden layer and a Linear activation function for the output layer.

ReLu activation function: **A(x) = max(0,x)**

It gives an output x if x is positive and 0 otherwise.



[Figure 7](#figur_relu_func): ReLu Function

ReLu give the benefit of making the activations sparse and efficient. For example a network with random initialized weights or normalised data input and almost 50% of the network yields 0, which also output 0 for negative values of x activation due to the characteristic of ReLu. Hence, fewer neurons are firing making a sparse activation and the network is lighter. ReLu is less computationally expensive which involves simpler mathematical operations.

Depends on the design of an artificial neural network, data are fed into different layers. These layers are also known as Input layer, Hidden layer and Output layer.

**Input Layer**: The first layer in the neural network which does not apply any operation on the input but it takes the input values and passes it onto the next layer.

**Hidden Layer**: The hidden layer which is the layer between the input layer and output layer. In this layer where the neurons which apply different transformation to the input data.

**Output Layer:** The output layer is the last layer which the network receives input from the last hidden layer and produce the desired output result.

When training the neural network, a process called forward propagation where feeding the input values to the neural network and getting an output which is called the predicted value. The learning process starts with back propagation after a forward propagation which output a predicted value that are compared with the actual output value. By using a loss function to calculate the error value, a optimizers are used to update the weights of the neural network.

# Implementation

## Understanding the Dataset

There are a total of 1459 samples in the provided datasets. Each data sample represents a housing property and consists of its features and its selling price.

Some of the data samples does not have a specific value for certain features as it may be due to a lack of data on the particular feature or possibly that the data sample does not have such a feature. This lack of value appears in the form of black spaces in the datasets for the specific features.

### Input Features

For each of the 1459 samples, there are a total of 79 input features excluding the “ID” column, which is just the unique identifier to represent a data sample out of the 1459 samples.

Input features refer to a residential home or property’s features and their values. According to the provided datasets, there are 79 features of a residential home or property that will affect the selling price of the property.

A more detailed description of the input features can be found in Appendix, under [Datasets’ Features](#_886e8j38uuan).

### Output Target

The output target refers to the target variable which is the result of considering the 79 input features.

For our project of predicting the selling price of a housing property, this output target is labelled as “SalePrice” in the provided datasets and it refers to the property’s actual selling price in dollars.

## Data Cleaning and Preprocessing

Data cleaning and preprocessing is an essential part of data mining as it ensures that consistent data gets fed accurately into the various models. As such, the results obtained will be accurate and reliable.

The following are the steps taken:

### Replacement of null values

* 1. Data cleaning involved first considering the removal of features that contains a majority of null values. From post processing results we were able to derive that features should be removed if they contain 70 to 80 percent of null values (60% and below results in higher error).
  2. After the removal of redundant features we will split the dataset into 2 feature types, namely numerical and non-numerical data types
  3. For numerical features, 3 data types are considered; continuous, categorical and optional. We would first identify optional data sets by looking for optional property attributes like Garages, Basements etc, things that do not have to be part of a house, null values of these data would be replaced with zeros since it does not make sense to provide a modal or average value. After consideration of optional features, null continuous data types will be replaced with a mean score while null categorical data type will be replaced with a mode score
  4. For non-numerical values, 2 data types will be considered; categorical and optional values. Same procedure applied to numerical values will be applied to non-numerical values; where categorical values are replaced with modal score optional values will be replaced with a ‘None’ string.

### One-hot Encoding

One-hot encoding is performed on columns that contain non-numerical values. This is done to ensure that our correlation matrix takes non-numerical columns into account. It also allows us to perform normalization on these columns. We performed one-hot encoding using the pandas *get\_dummies()* function, which splits the non-numerical columns into multiple columns, depending on the number of classes it contains. An example of non-numerical columns before and after performing one-hot encoding is shown below in [Figure 8](#fig_bef_onehot_encoding) and [Figure 9](#fig_aft_onehot_encoding).

|  |  |  |
| --- | --- | --- |
| **RoofStyle** | **RoofMatl** | **Exterior1st** |
| Gable | CompShg | Plywood |
| Gable | CompShg | VinylSd |
| Hip | CompShg | Wd Sdng |

[Figure 8](#figur_bef_onehot_encoding): Non-numerical columns before one-hot encoding

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **RoofStyle\_**  **Gable** | **RoofStyle\_**  **Hip** | **RoofMatl\_**  **CompShg** | **Exterior1st\_Plywood** | **Exterior1st\_VinylSd** | **Exterior1st\_Wd Wdng** |
| 1 | 0 | 1 | 1 | 0 | 0 |
| 1 | 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 | 1 |

[Figure 9](#figur_aft_onehot_encoding): Non-numerical columns after one-hot encoding

### Feature Discovery

Before we can generate the proper feature sets for training, we have to consider collinear features. For a decent and unique spread of features to accurately train our models, we have to make sure that each feature is only strongly correlated to the Sale Price and itself.

A correlation heatmap matrix is constructed to identify the correlation coefficient between the output, ‘SalePrice’, and the other input features. Based on these coefficients, we then identify the top **k** most correlated features.

We also experimented with the **k** least correlated features and compared their accuracies. It is observed that using the top **k** most correlated features results in a much higher accuracy, hence proving the importance of identifying these correlation coefficients while performing data preprocessing. The graphs comparing these accuracies can be found in the Appendix under the [Accuracy comparison between most and least correlated features](#_fuo9q8yz7gw4) section.

The top features identified will then be used as the selected input features to be analysed for possible collinear features. For illustration and readability, below are top 15 features extracted.

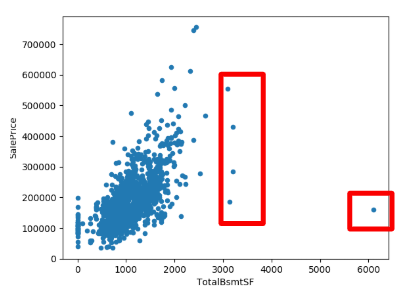
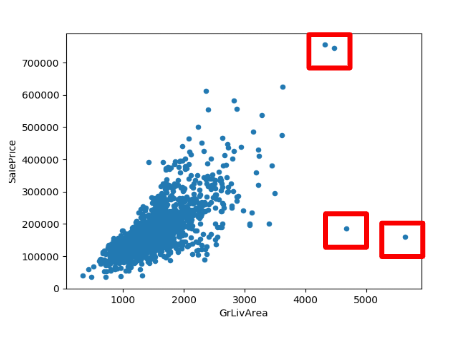
|  |  |
| --- | --- |
| Correlation matrix of top **15** input features | |
|  |  |
| Before Preprocessing | After Preprocessing |

[Figure 10](#figur_comparison_corr_matrix): Comparison of correlation matrix

From the heat map in [Figure 10](#fig_comparison_corr_matrix), before preprocessing we can see that the top 15 features contains some collinear features such as GarageCar and GarageArea which computes a correlation coefficient of 0.83, we could also see that a large garage area can fit many cars which makes garage car a redundant feature. Hence, for each of the top feature extracted we will identify correlation with other features that has a correlation coefficient of 0.8, every feature that satisfies the 0.8 coefficient will be removed while retaining the highest ranked feature. This will result in the diagram shown in the After Preprocessing where collinearlarity is reduced which makes space for consideration of more unique feature types.

### Removing of outliers

After identifying the features we needed, the next layer of pre-processing would be to remove possible outliers, outliers will refer to extreme values that tend to deviate from other observations on the dataset. Some examples extracted from 2 features are shown below.



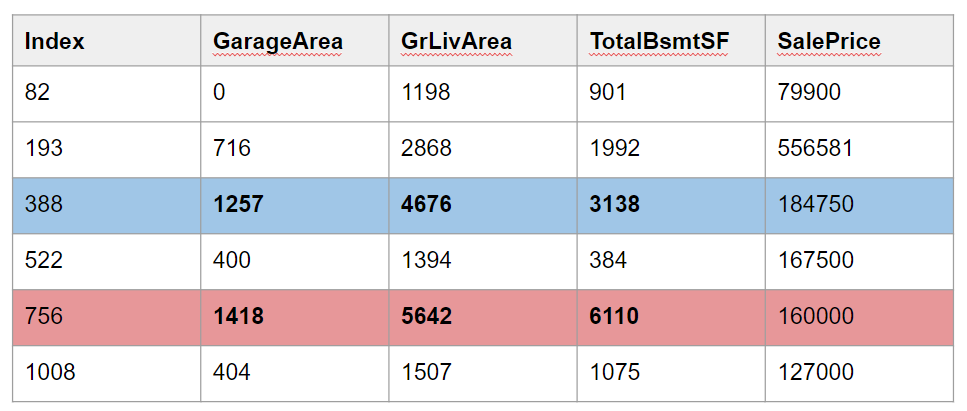
The graphs above are scatter plots of individual features against the Sale Price. Data points bounded within the rectangle are values that values diverges from an overall pattern. This could possibly indicate a variability in measurement, experimental or human errors and hence should be removed from training.

Since variability of data can only be found in continuous data, all selected features with continuous data types are assessed. For every feature, all indexes of outlier data is extracted with the following code:

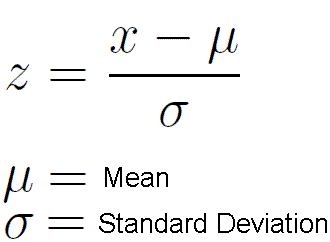


[Figure 11](#figur_indexes_boundary): Code returns list of indexes based on specified boundary

Below is an example of an outlier feature data with their respective indexes. We will only consider indexes that are most common among all feature being analysed, this is to make sure that we do not remove critical rows. Detecting outliers is important. Like in any quantitative discipline, quality of data is as important as the quality of prediction model.



### Standardization



After cleaning the data, the final step is to standardize the data. The objective of standardization is to rescale all feature inputs to achieve a standard normal distribution. This critical process will bring data to a more common format that allow

for larger scale analytics and definitely crucial for regression problems that we will be using.

# Experimental Results and Analysis

We use the following various error metrics to evaluate the performance of a model based on the residual. Residual refers to the difference between the actual value and the model’s estimated value.

|  |  |
| --- | --- |
| Error Metrics Used | Explanation |
| R-Squared | - A statistical measure of difference between the data and the fitted regression line  - R-squared value will be between 0 and 1,whereby 0 represents a model that does not explain any of the variation in the response variable and 1 represents a model that explains all of the variation in the response variable around its mean |
| Mean Absolute Error (MAE) | - MAE takes the absolute value of residual for every data point |
| Root Mean Square Error (RMSE) | - A type of measure that compares how wide-spread the differences between predicted values by a model and the actual values observed. (Measure of how widespread the residuals are)  - The measurement is then converted into percentage and is referred to as Root Mean Square Percentage Error (RMSPE) |
| Root Mean Square Log Error  [RMSLE]  (Metric that is used in Kaggle) | - A variant of RMSE, but unlike RMSE, RMSLE will only focus on the relative difference between the predicted value and the actual value. Hence, the RMSLE for small residuals between smaller predicted and actual values will be approximately the same as the RMSLE for the large residuals between larger predicted and actual values  - RMSLE will penalise underestimates more than overestimates, resulting in an asymmetry in the error curve  - Therefore, RMSLE is a good way to predict future house prices as prices are continuous values |

## Experimental Setup

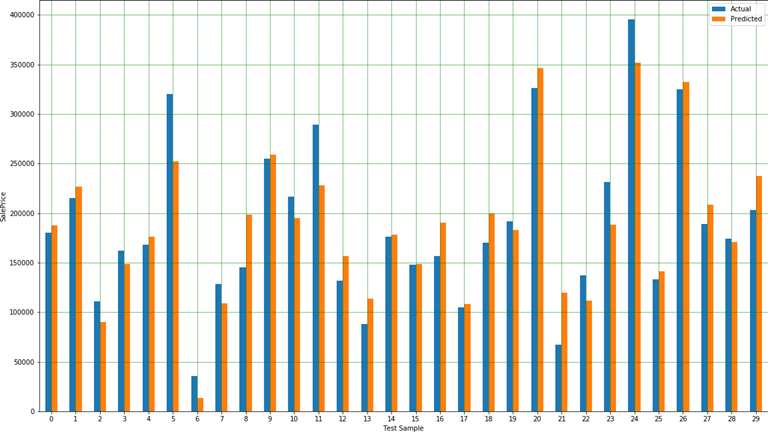
To prepare our models for the prediction, we first split the provided training dataset into training data and testing data in the ratio of 70:30. We will then train the model with the training data and predict the sale price using the test data.

To ensure consistency, accuracy and fairness between the models’ results, all the models will be using the same features to evaluate and use the same amount of training and testing data for evaluation.

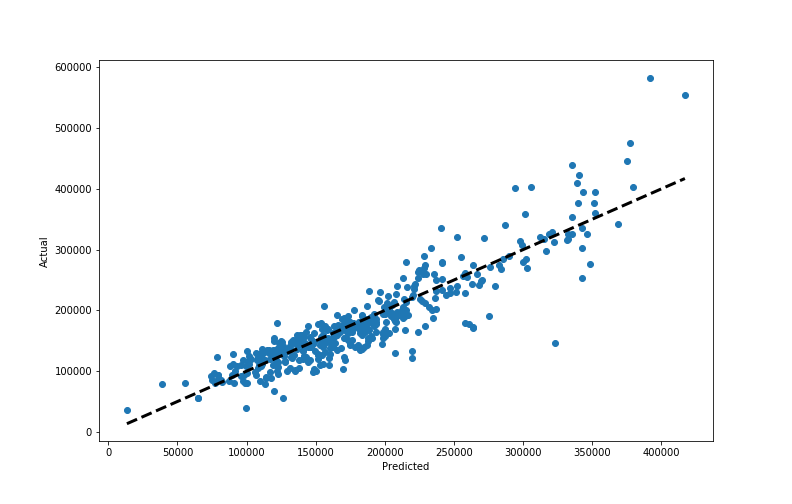
## Comparison Schemes

The following graphs and charts in the following pages will illustrate the prediction and regression results of the various models used in our project.

Linear Regression:



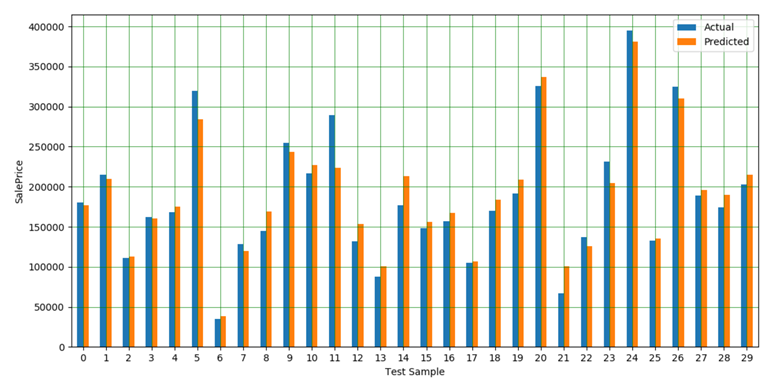
[Figure 12](#figur_linear_regression_bar): Predicted vs Actual price of first 30 test samples



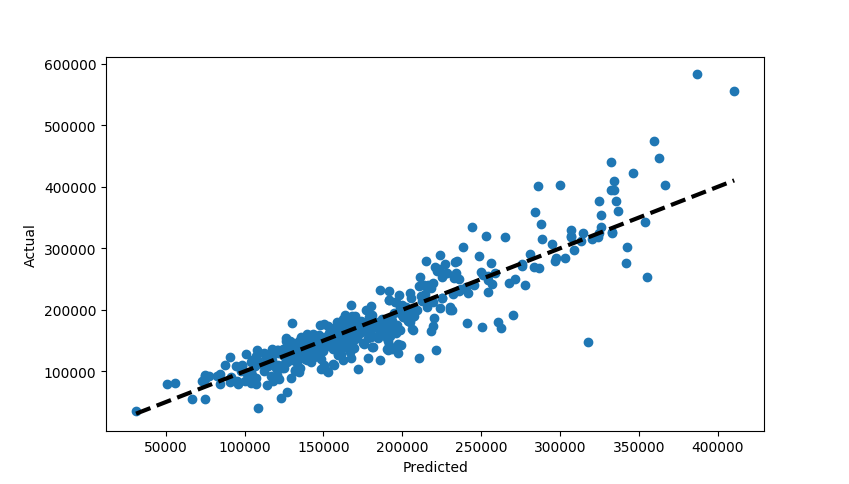
[Figure 13](#figur_linear_regression_scatter): Scatter plot of Predicted against Actual price

|  |  |
| --- | --- |
| **Measure Used** | **Value obtained** |
| R-Squared Value | 0.837 |
| Mean Absolute Error (Thousands) | 22.516 |
| Mean Absolute Percentage Error (%) | 13.98 |
| Root Mean Square Error (Thousands) | 32.488 |
| Root Mean Square Percentage Error (%) | 21.13 |
| Root Mean Square Log Error (Kaggle Score) | 0.154 |

Support Vector Regressor (SVR):



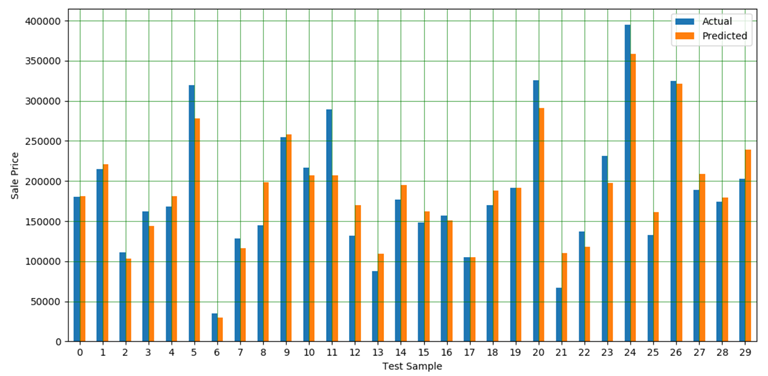
[Figure 14](#figur_SVR_bar): Predicted vs Actual price of first 30 test samples

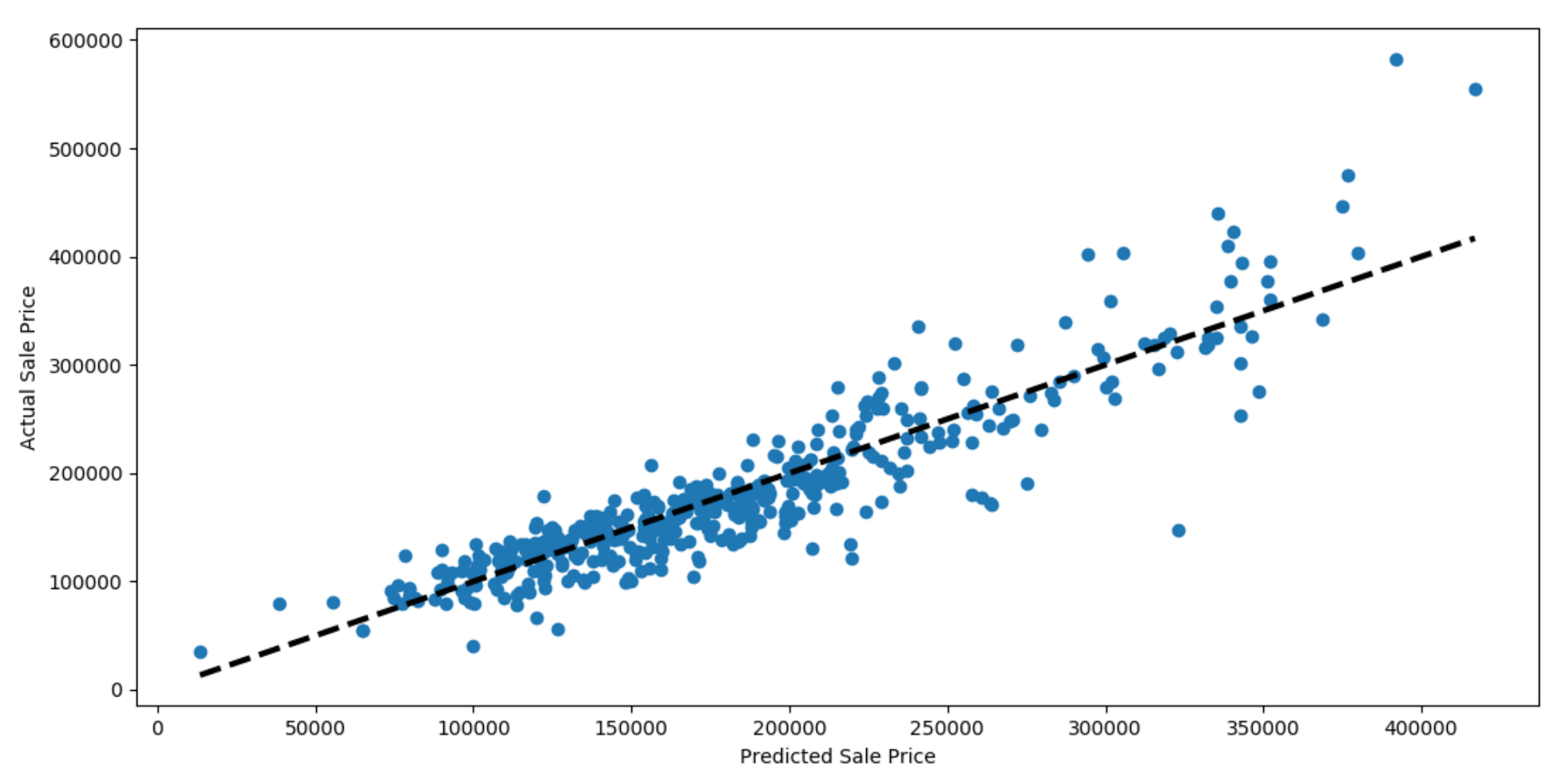


[Figure 15](#figur_SVR_scatter): Scatter plot of Predicted Sale Price against Actual price

|  |  |
| --- | --- |
| **Measure Used** | **Value obtained** |
| R-Squared Value | 0.833 |
| Mean Absolute Error (Thousands) | 21.077 |
| Mean Absolute Percentage Error (%) | 12.53 |
| Root Mean Square Error (Thousands) | 31.546 |
| Root Mean Square Percentage Error (%) | 19.14 |
| Root Mean Square Log Error (Kaggle Score) | 0.143 |

Ridge Regression:

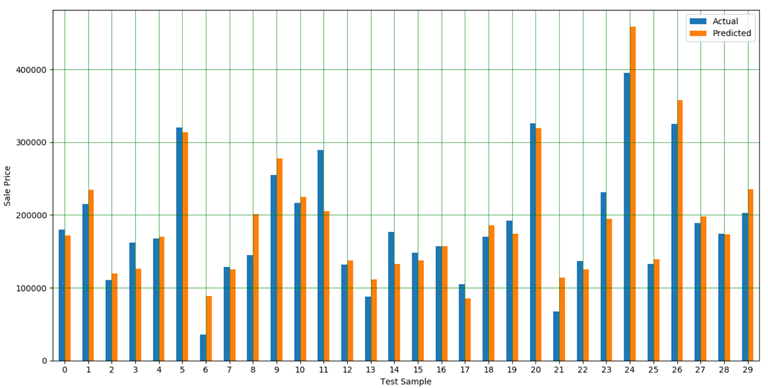
  
[Figure 16](#figur_ridge_bar): Predicted vs Actual price of first 30 test samples



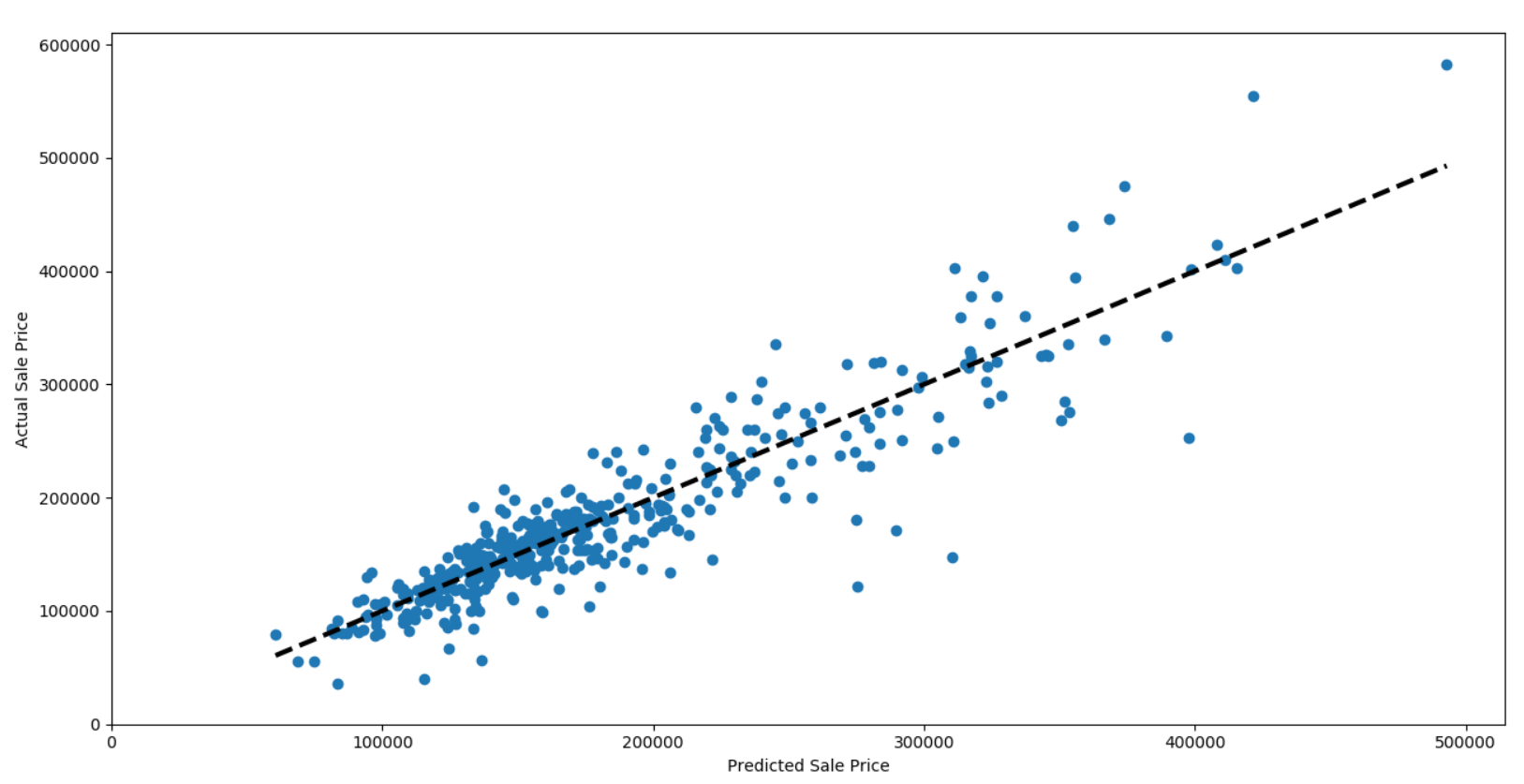
[Figure 17](#figur_scatter): Scatter plot of Predicted against Actual price

|  |  |
| --- | --- |
| **Measure Used** | **Value obtained** |
| R-Squared Value | 0.837 |
| Mean Absolute Error (Thousands) | 21.926 |
| Mean Absolute Percentage Error (%) | 13.20 |
| Root Mean Square Error (Thousands) | 31.163 |
| Root Mean Square Percentage Error (%) | 19.31 |
| Root Mean Square Log Error (Kaggle Score) | 0.160 |

Random Forest Regression



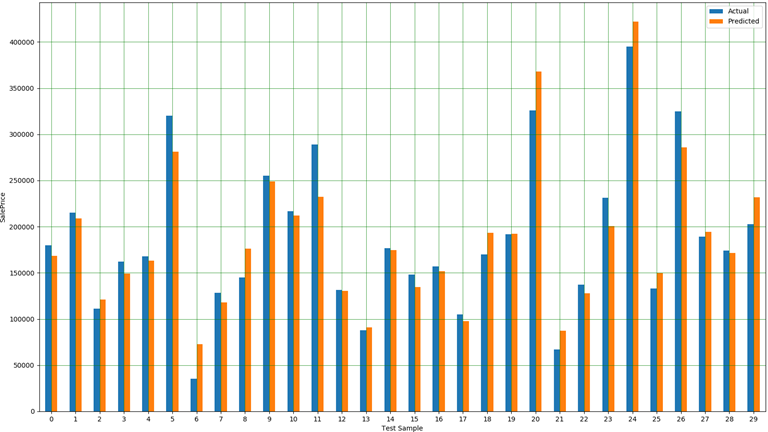
[Figure 18](#figur_random_forest_bar): Predicted vs Actual price of first 30 test samples



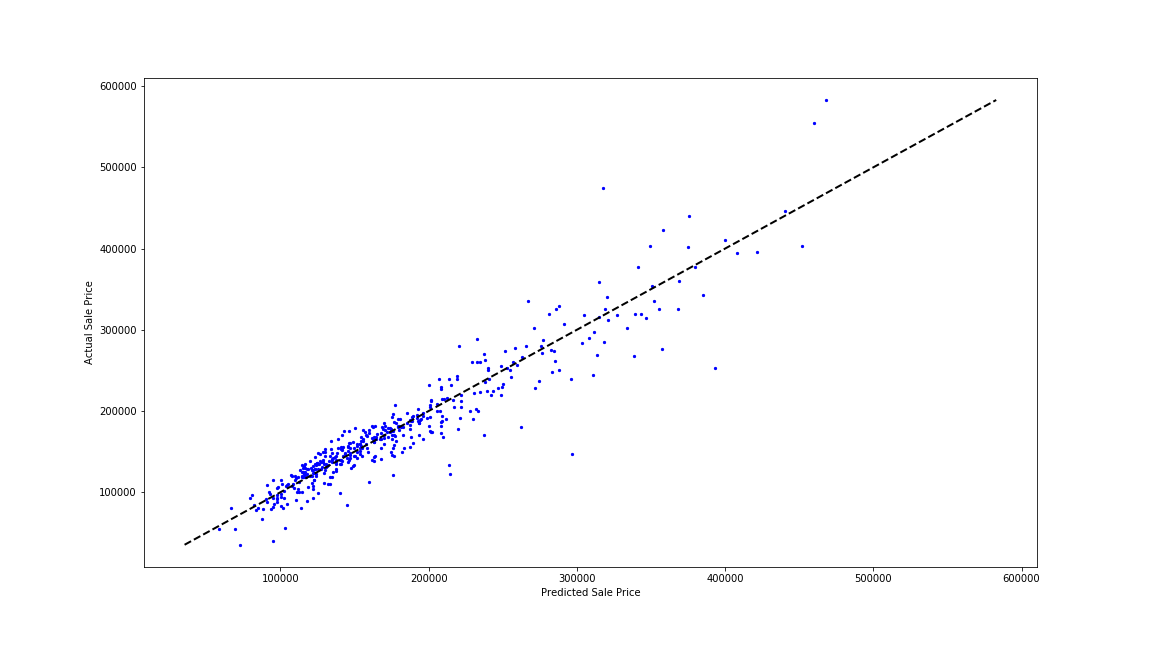
[Figure 19](#figur_random_forest_scatter): Scatter plot of Predicted against Actual price

|  |  |
| --- | --- |
| **Measure Used** | **Value obtained** |
| R-Squared Value | 0.827 |
| Mean Absolute Error (Thousands) | 21.764 |
| Mean Absolute Percentage Error (%) | 13.34 |
| Root Mean Square Error (Thousands) | 32.186 |
| Root Mean Square Percentage Error (%) | 22.40 |
| Root Mean Square Log Error (Kaggle Score) | 0.158 |

Extreme Gradient Boosting (XGBoost):



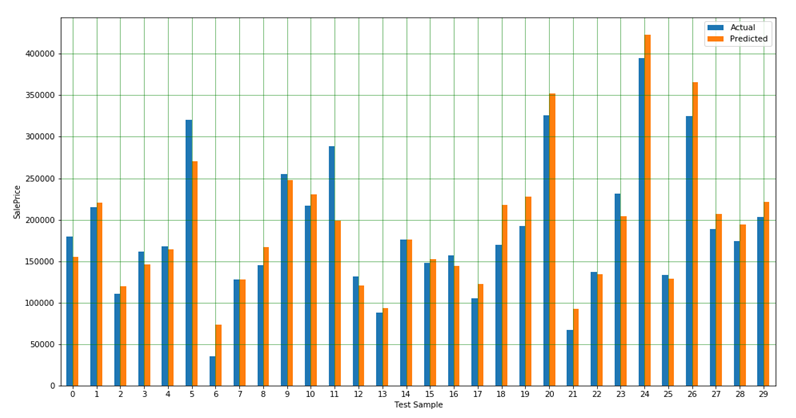
[Figure 20](#figur_xgboost_bar): Predicted vs Actual price of first 30 test samples



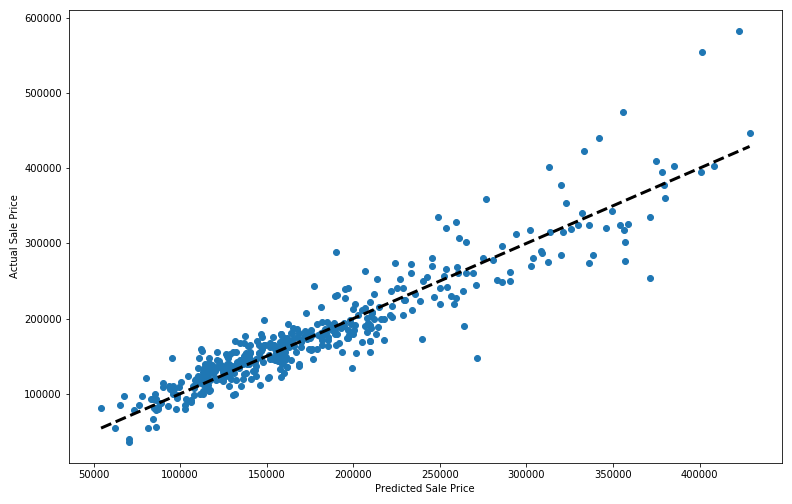
[Figure 21](#figur_xgboost_scatter): Scatter plot of Predicted against Actual price

|  |  |
| --- | --- |
| **Measure Used** | **Value obtained** |
| R-Squared Value | 0.871 |
| Mean Absolute Error (Thousands) | 16.337 |
| Mean Absolute Percentage Error (%) | 9.86 |
| Root Mean Square Error (Thousands) | 25.488 |
| Root Mean Square Percentage Error (%) | 16.60 |
| Root Mean Square Log Error (Kaggle Score) | 0.143 |

Neural Network:



[Figure 22](#figur_nn_bar): Predicted vs Actual price of first 30 test samples



[Figure 23](#figur_nn_scatter): Scatter plot of Predicted against Actual price

|  |  |
| --- | --- |
| **Measure Used** | **Value obtained** |
| Mean Absolute Error (Thousands) | 18.944 |
| Mean Absolute Percentage Error (%) | 10.97 |
| Root Mean Square Error (Thousands) | 28.333 |
| Root Mean Square Percentage Error (%) | 15.90 |
| Root Mean Square Log Error (Kaggle Score) | 0.152 |

## Results and Analysis

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Measure Used** | **Linear Regression** | **SVR** | **Ridge** | **Random Forest** | **XGBoost** | **Neural Network** |
| R-Squared Value | 0.837 | 0.833 | 0.837 | 0.827 | 0.871 | - |
| Mean Absolute Error (Thousands) | 22.516 | 21.077 | 21.926 | 21.764 | 16.337 | 18.944 |
| Mean Absolute Percentage Error (%) | 13.98 | 12.53 | 13.20 | 13.34 | 9.86 | 10.97 |
| Root Mean Square Error (Thousands) | 32.488 | 31.546 | 31.163 | 32.186 | 25.488 | 28.333 |
| Root Mean Square Percentage Error (%) | 21.13 | 19.14 | 19.31 | 22.40 | 16.60 | 15.90 |
| Root Mean Square Log Error  [Kaggle Score] | 0.154 | 0.143 | 0.160 | 0.158 | 0.143 | 0.152 |

# Discussion of Pros and Cons

|  |  |  |
| --- | --- | --- |
| **Algorithm** | **Pros** | **Cons** |
| Linear Regression | * Low space complexity * Good interpretability * Simple to implement | * Prone to outliers * Susceptible to multicollinearity |
| Support Vector Regression | * Effective in higher dimensions * Less susceptible to outliers | * Long processing time for large datasets * Does not perform work in the case of overlapped classes |
| Ridge Regression | * Counters multicollinearity and overfitting * Performs well even with large multivariate data | * Does not perform feature selections * Model shrinks the coefficients towards zero |
| Random Forest Regression | * Good results even when a large part of data are missing * Low computation cost * Helps to overcome overfitting | * High complexity and time-consuming * Less interpretable |
| Extreme Gradient Boosting | * Very effective for large datasets * XGBoost includes a built in L1 (Lasso Regression) and L2 (Ridge Regression) regularization which can prevent the model from overfitting | * Only works with numerical features * Hyperparameters need to be tuned to avoid overfitting |
| Artificial Neural Network | * Neural networks can be trained with any number of inputs and layers. * Neural networks work best with more data points | * Neural networks are black boxes, which cannot know how much each independent variable is influencing the dependent variables. |

# Conclusion

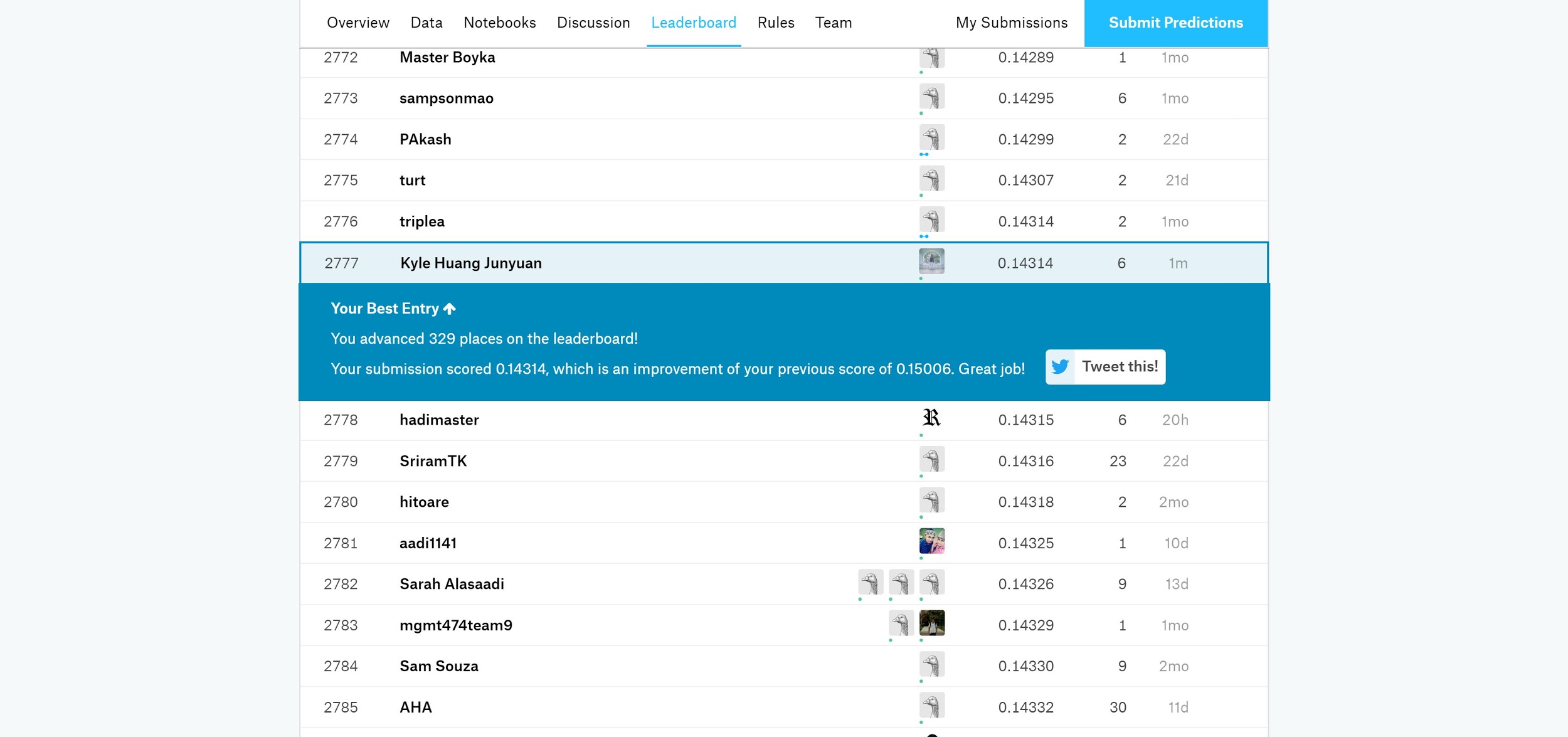
In conclusion, the team has chosen to take part in a Kaggle challenge to predict house prices from a given training dataset. Participants are also given a testing dataset (without targets) to fairly evaluate their model’s performance against a public leaderboard that can be found at the following link: <https://www.kaggle.com/c/house-prices-advanced-regression-techniques/leaderboard>

Since the dataset contains 79 features in total, appropriate preprocessing and cleaning of the input features was paramount before any training can be performed on the chosen model. This was evident in our experiments where a lower error rate was achieved when a ‘clean’ dataset is presented to a model versus the original ‘unclean’ dataset.

After the preprocessing, careful steps were taken to ensure that every model gets presented with the same set of training and testing dataset. That is, the training dataset was first split into 70% for training and 30% for testing and subsequently applied to all models in that particular arrangement.

The models the team has evaluated consisted of Linear Regression, Support Vector Regressor (SVR) Ridge Regression, Random Forest Regression, Extreme Gradient Boosting (XGBoost) and Artificial Neural Network with the best performing models being SVR and XGBoost.

# Summary of Project Achievements



Our team managed to achieve a Root Mean Squared Logarithmic Error (RMSLE) of 0.143 in our SVR and XGBoost models and attained the position of 2777th out of over 5000 participants as of 13th November 2019.

## Directions for Improvements

* Apply more types of data cleaning methods to improve data consistency and accuracy
* Create automated code (recursion) to determine features with multicollinearity
* Search and source for more training data sets to improve prediction’s accuracy
* Further optimise all models by experimenting with different hyper parameters

# References

<https://www.dataquest.io/blog/understanding-regression-error-metrics/>

<https://www.kaggle.com/c/house-prices-advanced-regression-techniques>

<https://towardsdatascience.com/machine-learning-project-predicting-boston-house-prices-with-regression-b4e47493633d>

<https://www.techinasia.com/machine-learning-estimate-singapore-property-value>

<https://towardsdatascience.com/linear-regression-using-python-b136c91bf0a2>

<https://www.researchgate.net/publication/323588842_Runoff_prediction_with_a_combined_artificial_neural_network_and_support_vector_regression#pf2>

<https://medium.com/@rrfd/what-is-ridge-regression-applications-in-python-6ed3acbb2aaf>

<https://turi.com/learn/userguide/supervised-learning/random_forest_regression.html>

<https://blog.exploratory.io/introduction-to-extreme-gradient-boosting-in-exploratory-7bbec554ac7>

<https://machinelearningmastery.com/gentle-introduction-xgboost-applied-machine-learning/>

<https://hackernoon.com/everything-you-need-to-know-about-neural-networks-8988c3ee4491>

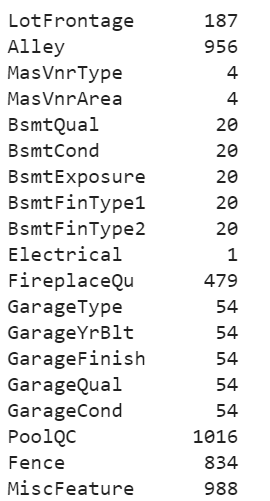
<https://medium.com/datadriveninvestor/neural-network-fundamentals-1956a3000c24>

# Appendix

## Datasets’ Features

|  |  |
| --- | --- |
| **Feature** | **Description** |
| MSSubClass | Building Class |
| MSZoning | General Zoning Classification |
| LotFrontage | Linear feet of street connected to property |
| LotArea | Area of lot size |
| Street | Type of road access |
| Alley | Type of alley access |
| LotShape | General shape of property |
| LandContour | Flatness of property |
| Utilities | Types of available utilities |
| LotConfig | Lot configuration |
| LandSlope | Slope of property |
| Neighbourhood | Physical location within city limits |
| Condition1 | Proximity to main road / railroad |
| Condition2 | Proximity to main road / railroad (If available) |
| BldgType | Type of dwelling |
| HouseStyle | Style of dwelling |
| OverallQual | Overall material and finish quality |
| OverallCond | Overall condition rating |
| YearBuilt | Original construction date |
| YearRemodAdd | Remodel date |
| RoofStyle | Type of roof |
| RoofMatl | Material of roof |
| Exterior1st | Exterior covering on house |
| Exterior2nd | Exterior covering on house (If there are more than 1 material used) |
| MasVnrType | Masonry veneer type |
| MasVnrArea | Masonry veneer area |
| ExterQual | Quality of exterior material |
| ExterCond | Present condition of exterior material |
| Foundation | Type of foundation |
| BsmtQual | Height of basement |
| BsmtCond | General condition of basement |
| BsmtExposure | Walkout or Garden level basement walls |
| BsmtFinType1 | Quality of basement finished area |
| BsmtFinSF1 | Type 1 finished’s area |
| BsmtFinType2 | Quality of basement finished area (If available) |
| BsmtFinSF2 | Type 2 finished’s area |
| BsmtUnfSF | Unfinished basement area |
| TotalBsmtSF | Total area of basement |
| Heating | Type of heating |
| HeatingQC | Heating quality and condition |
| CentralAir | Central air conditioning |
| Electrical | Electrical system |
| 1stFlrSF | 1st Floor’s area |
| 2ndFlrSF | 2nd Floor’s area |
| LowQualFinSF | Low quality finished’s area (All floors) |
| GrLivArea | Above ground living space’s area |
| BsmtFullBath | Basement full bathrooms |
| BsmtHalfBath | Basement half bathrooms |
| FullBath | Full bathrooms (Above ground) |
| HalfBath | Half bathrooms (Above ground) |
| Bedroom | Number of bedrooms above ground |
| Kitchen | Number of kitchen |
| KitchenQual | Quality of kitchen |
| TotRmsAbvGrd | Total rooms (Above ground excluding bathrooms) |
| Functional | Home functionality rating |
| Fireplaces | Number of fireplaces |
| FireplaceQu | Quality of fireplace |
| GarageType | Location of garage |
| GarageYrBlt | Year of garage’s construction |
| GarageFinish | Interior finish of garage |
| GarageCars | Garage’s car capacity |
| GarageArea | Garage’s area |
| GarageQual | Quality of garage |
| GarageCond | Condition of garage |
| PavedDrive | Paved driveway |
| WoodDeckSF | Wood deck’s area |
| OpenPorchSF | Open porch’s area |
| EnclosedPorch | Enclosed porch’s area |
| 3SsnPorch | 3 Seasons porch’s area |
| ScreenPorch | Screen porch’s area |
| PoolArea | Pool’s area |
| PoolQC | Quality of pool |
| Fence | Quality of fence |
| MiscFeature | Miscellaneous features |
| MiscVal | Value of miscellaneous feature |
| MoSold | Month Sold |
| YrSold | Year Sold |
| SaleType | Type of sale |
| SaleCondition | Condition of sale |

## Features with missing values (NaN or NULL)



## Accuracy comparison between most and least correlated features

|  |
| --- |
| Predicted vs Actual Sale Price based on most correlated features |
| Predicted vs Actual Sale Price based on least correlated features |