IS424 – Data Mining and Business Analytics

AY 2020/21 Term 1

G2T3: Project Final Report

Prepared for

Professor Wang Zhaoxia

Done by:

Name	**Student ID**
Sia Yan Rui (Jayden)	01351538
Tan Yan Sheng	01337459
Neo Tee Yong	01325982
Yieh Yuheng (Donavan)	01338241
Yohana Meiliana Lee	01350370

Table of Contents

1. Introduction/ Problem Statement

Accurately estimating the value of resale houses is an important problem for many Singaporeans [1]. Based on an article by PropertyGuru, 88% of Singaporeans especially millennials are unhappy over high property prices, and at least 69% of them face difficulty in funding their housing purchase due to the inability to gauge housing prices [2].

Though it is common knowledge that factors such as size, number of rooms, and location affect the price, there are many other considerations at play. Certain local factors such as distance to amenities may affect the demand of houses in a particular area due to Government policies, such as how distance to a school may affect chances of a child getting into that school, we believe that taking into account these factors may provide an additional layer of accuracy in the models.

In this project, we compare different data mining methods performance using regression techniques in predicting the prices of resale houses based on features such as flat type, geographical location, storey and lease commence date. We aim to explore the breadth of machine learning models, their applications, what makes them good or bad in different situations, and explore the depth of optimizing these models.

2. Motivation

Most Singaporean millennials, as quoted by the PropertyGuru article [1], such as ourselves, are burdened with the task of having to save up for our future house without the knowledge or insight of how much our future house would cost. This in turn, affects our early financial planning, and is both time and effort costly. This is especially crucial given that Singapore has one of the highest costs of living in the world.

Hence, it is of utmost importance for us to be concerned about housing prices, and the factor(s) that contributes to the increase of housing prices needs to be determined.

3. Literature Review

Our literature review consisted of three articles. The dataset for the first and last articles are based in Singapore while the other article is based on a dataset downloaded from Kaggle. The aim of all three studies is to predict the prices of housing flats.

<u>Predicting the prices of HDB flats in SG by Zhi Liang, Chen[3]</u>
The literature study endeavours to construct a robust basis on regression techniques such as Random Forest, Single, Decision Tree and K-Nearest Neighbours followed by feature selection using correlation matrix and Variance Inflation Factor and feature extraction using Principal Component Analysis (PCA). The dataset used for this study is taken from Data.Gov and only dataset from 2017 were used.

The author shared that one should always transform the test data using PCA fitted from training data. This is to reduce the dimension of features in the dataset. The decision on the final number dimensions of PCA can be decided manually or via a variance threshold, which the author set the variance threshold to 90% as he wants the number of final features to explain 90% of variance in data.

Comprehensive Data Exploration with Python by Pedro Marcelino [4]
The author for this literature study aims to find insights through basic Exploratory Data Analysis(EDA) and visualizations. He conducted an univariable study to focus on the dependent variable (‘SalePrice’) and multivariate study to understand the relationship between dependent variables and independent variables. The author shared that the most important takeaway from this study is to avoid drawing conclusions from bivariate analysis to prevent Simson paradox.

Singapore Flat Price Predictor by Sie Huai, Gan [5]
The author conducted his project based on Ensemble learning, Stacking by combining four base learners namely ElasticNet, Support Vector Machine(SVM), Lasso Regression and Gradient Boosting. He used data backdated from 2012 to 2014 to train his model. Besides the features in the dataset, he has also created additional features such as Region (region the HDB flat is located in SG), No. of rooms and toilet(s), Storey (storey level of HDB flat) and Sales Date (Sales Year, Sales Month). He emphasizes that feature creation can help models to perform better.

4. Datasets

4.1 Data Integration

In order to train a reliable machine learning model as accurately as possible, our team compiled a dataset from January 2015 to December 2016 and datasets from January 2017 onwards to have a 5 year dataframe. These datasets are taken from Data.Gov. The entire dataset consists of 112,626 entries for houses in Singapore [6]. We then integrated these datasets together to increase the number of records we have that may go into training the models, overall improving accuracy.

4.2 Overview of Columns

There are 7 nominal, 1 ordinal, 0 interval and 2 ratio features.

Source of dataset: **https://data.gov.sg/dataset/resale-flat-prices**.		
Column Name	**Data Type**	**Description**
month	Datetime (Month) “YYYY-MM”	The date when the resale flat was sold, eg. 2020-01
town	Text (General)	The name of the town the resale flat is located at, eg. ANG MO KIO

flat_type	Text (General)	The type of the resale flat eg. 4 ROOM, 3 ROOM
block	Text (General)	The block number of the resale flat eg. 182, 181
street_name	Text (General)	The name of the street the resale flat is located at, eg. ‘JELAPANG RD’
storey_range	Text (General)	The range of storey that the resale unit is located in eg. ‘04 TO 06’
floor_area_sqm	Numeric (General)	The size of the resale unit in Sqm eg. ‘100’
flat_model	Text (General)	The model type of the flat eg. ‘Model A’, ‘Improved’
lease_commence_date	Datetime (Year) “YYYY”	The year that the lease of the flat started eg. ‘1997’
remaining_lease	Text (General)	The amount of time that the lease has remaining in years and months eg. ‘76 years 08 months’
resale_price	Numeric (General)	The price of the resale flat in $, also the **target variable** in our analysis. Eg. “350,000”

5. Methodology

5.1 Overview

The main goal of this project is to use data mining techniques to predict the price of a HDB resale unit, as well as to understand the factors affecting the price of a HDB resale unit so that young adults can make a better informed decision when deciding to purchase a HDB resale flat.

Firstly, we did exploratory data analysis to understand our dataset better and perform univariate and bivariate analysis. This will give us an idea of what needs to be done during pre-processing. Next, we did feature engineering and created features that we believe are relevant in a Singaporean context and can add accuracy and insight to our models. For example, calculating the various distances from a resale flat to amenities based on existing domain knowledge, to improve the performance of our model.

Next, we moved on to pre-processing, where we performed label encoding to convert our categorical (ordinal) to numerical data to use for regression analysis. We also performed dimensionality reduction, where we dropped columns that carry high collinearity and are noise.

We compared a total of 11 models, ranging from linear regression to our own stacking ensemble model, before choosing the best model based on MSE, to conduct our analysis on. We then did hyperparameters tuning to further improve the performance of our model. In this report, we will present the findings of our analysis and prediction model.

5.2 Technologies, Tools and Resources

5.2.1 Libraries

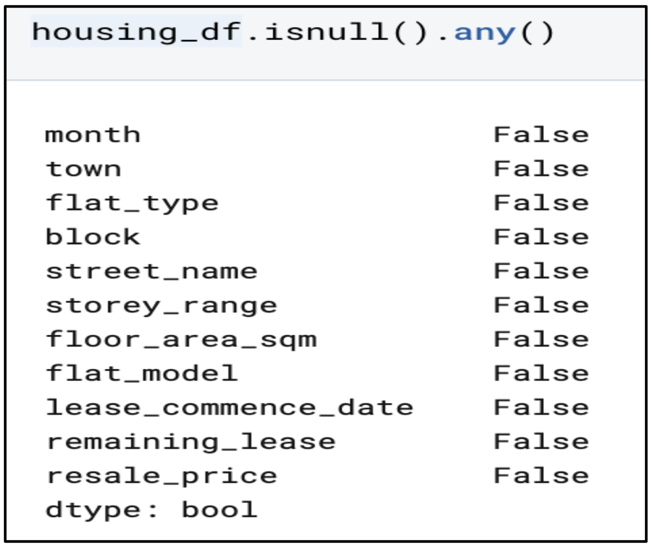
For this project, we have used Python Anaconda libraries such as Pandas, Matplotlib, GeoPandas, Shapley, Seaborn and Scikit-Learn.

5.2.2 APIs

We have used OneMapAPI to retrieve latitude and longitude in order to calculate the distance to the nearest amenities such as shopping malls, train stations and schools. The reasoning behind calculating these features will be elaborated further in section 5.4.

5.3 Exploratory Data Analysis

5.3.1 Missing Values



```
housing_df.isnull().any()
```

```
month                   False
town                    False
flat_type               False
block                   False
street_name             False
storey_range            False
floor_area_sqm          False
flat_model              False
lease_commence_date     False
remaining_lease         False
resale_price            False
dtype: bool
```

Fig 01: Code and output to check for missing values

There were no missing values throughout any attributes.The dataset is relatively “cleaned” as sourced from Data.gov, and do not hold any missing values.

5.3.2 Check for Multicollinearity

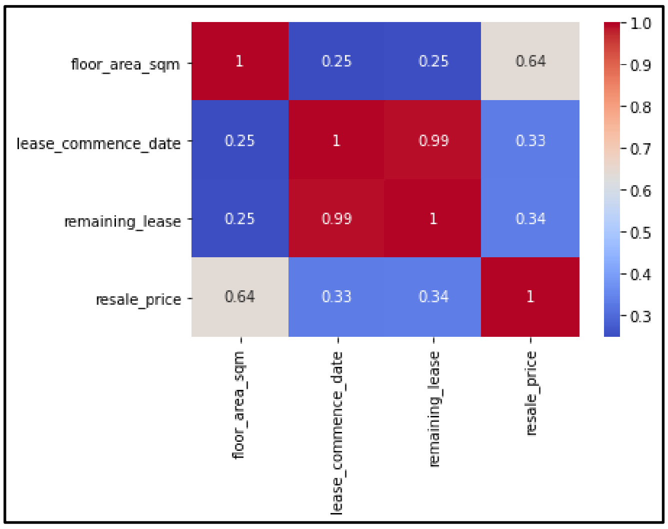


Fig 02: Correlation heatmap of each feature

lease_commence_date and remaining_lease were found to have a 0.99 Pearson correlation coefficient value. There is strong multicollinearity between these 2 features, hence we should drop one of these features. It is also worth noting that floor_area_sqm has a strong correlation to resale_price.

5.3.3 Numerical Distribution

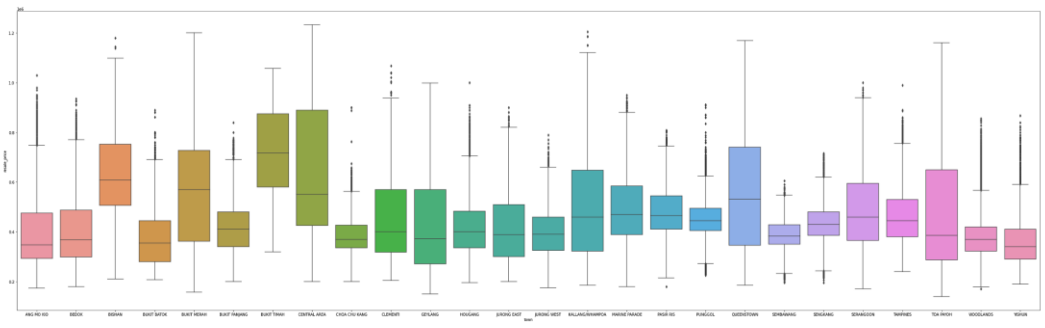


Fig 03: Boxplots of the independent variables 'town', 'flat_type' and 'flat_model' to the target variable 'resale_price'

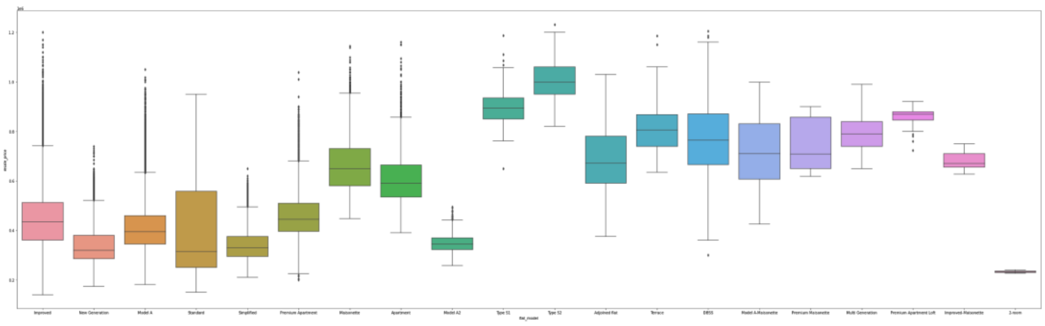


Fig 04: Boxplots of the independent variables 'town', 'flat_type' and 'flat_model' to the target variable 'resale_price'

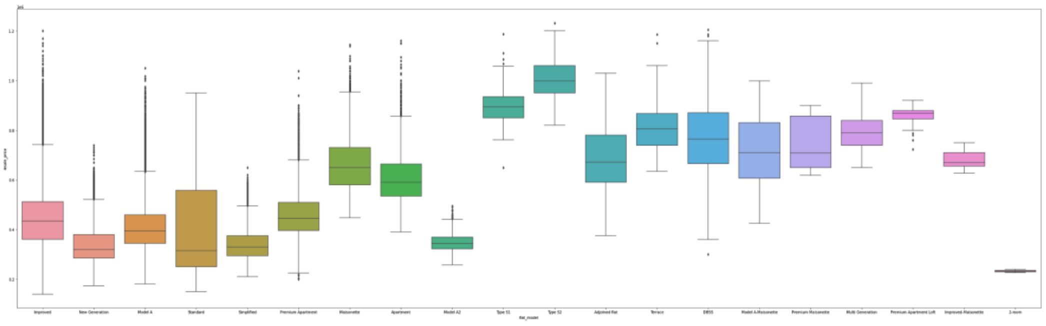
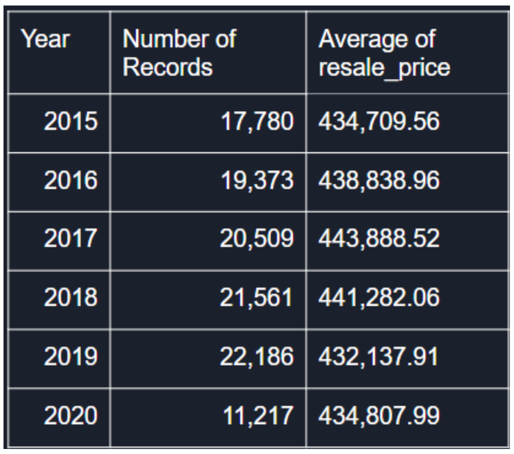


Fig 05: Boxplots of the independent variables 'town', 'flat_type' and 'flat_model' to the target variable 'resale_price'

We look at box plot diagrams to compare the range of resale price in relation to different independent variables to visualize the outliers and the differences in median. For example, we can see that multi generation flats have the highest median resale price, followed by Executive and 5 room flats. This makes sense as we identified in our EDA, floor area is highly correlated to resale price.

5.3.4 Contextual Analysis

Our team has also performed additional analysis on HDB Resale prices from 2015-2020 to determine trends in HDB Resale prices over the years.



Year	Number of Records	Average of resale_price
2015	17,780	434,709.56
2016	19,373	438,838.96
2017	20,509	443,888.52
2018	21,561	441,282.06
2019	22,186	432,137.91
2020	11,217	434,807.99

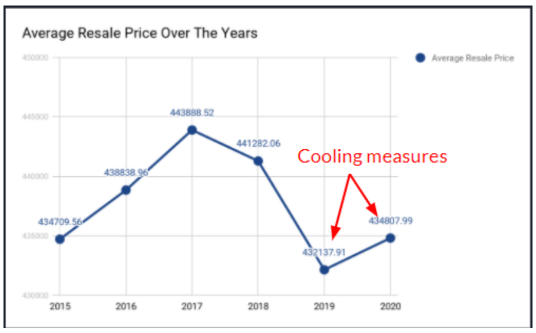


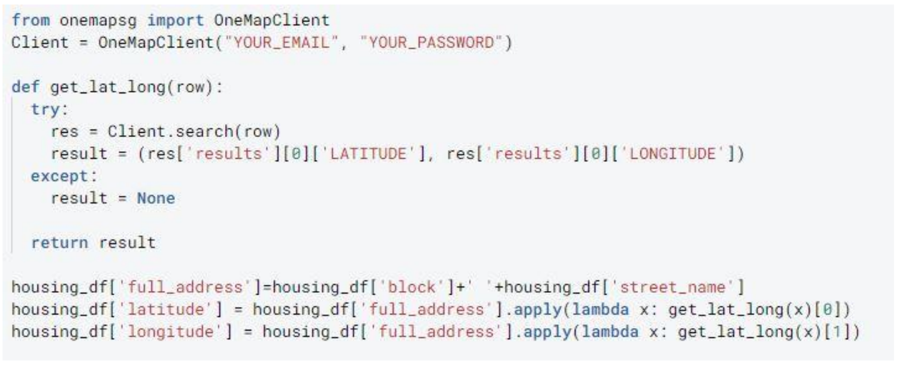
Fig 06: Average price over the years

From our analysis above, it can be observed that there was an upwards trend in HDB Resale prices from 2015 to 2018, showing that HDB Resale prices generally tend to increase over the years.

However, our team also noted that there was a significant drop in HDB Resale Prices in years 2019 and 2020, after conducting research, our team deduced that the significant drop in HDB resale prices in years 2019 and 2020 are attributed to the cooling measure in place by the government to reduce HDB Resale prices and make such prices more affordable [7].

5.4 Feature Engineering

5.4.1 Retrieval of Longitude and Latitude



```python
from onemapsg import OneMapClient
Client = OneMapClient("YOUR_EMAIL", "YOUR_PASSWORD")

def get_lat_long(row):
  try:
    res = Client.search(row)
    result = (res['results'][0]['LATITUDE'], res['results'][0]['LONGITUDE'])
  except:
    result = None

  return result

housing_df['full_address']=housing_df['block']+' '+housing_df['street_name']
housing_df['latitude'] = housing_df['full_address'].apply(lambda x: get_lat_long(x)[0])
housing_df['longitude'] = housing_df['full_address'].apply(lambda x: get_lat_long(x)[1])
```

Fig 07: Code snippet of OneMap API

As mentioned previously, we aim to account for local context to improve the accuracy of our models. Using OneMapAPI, we retrieve longitude and latitude for each building.

5.4.2 Minimum Distance to Amenities

One of the important factors that Singaporeans will take into consideration when purchasing a resale flat includes travelling time and the proximity to schools [8], as it will affect the chances of getting their children into the school.

Hence, we retrieved the longitude and latitude of all the MRT, School and Mall locations and calculated the minimum distance between the building and amenities using the euclidean distance formula multiplied by the scale of 100,000 (map scaling).

4.4.3 Maturity of estate

Maturity of estate plays a role in decision making when Singaporean purchase a flat as non-mature estates are cheaper than mature estates [9]. Hence, our team added additional attributes for the maturity nature of the town. This categorization is provided by HDB Singapore, with the criteria of mature estates being over 20 years old.

5.5 Data Preprocessing

5.5.1 Encoding Categorical Data

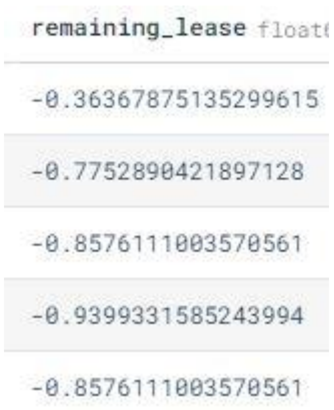
Variables such as flat_type, flat_model and month are categorical values and take on a limited set of values. For example, *flat_type* which represents the type of flats takes on the values such as “3 Room”, “4 Room” and “5 Room”.

Such categorical values cannot be interpreted by conventional machine learning algorithms without converting them to a numerical format.

We thus applied label encoding to the following categorical data (ordinal) to convert it to numerical format for regression:

- *town*
- *flat_type*
- *flat_model*
- *month*
- *storey_range*

5.5.2 Data Standardization



```
remaining_lease float
-0.36367875135299615
-0.7752890421897128
-0.8576111003570561
-0.9399331585243994
-0.8576111003570561
```

Fig 08: scaled values of *remaining_lease* attribute

To prevent inaccuracy and bias, data scaling and transformation has been applied to numerical columns such as *floor_area_sqm*, *resale_price, remaining_lease and minDistanceFromMall* . This is to prevent attributes with a larger range from dominating the prediction of our dependent variable. We do so by using sklearn’s StandardScaler(), which standardizes the data within each column by applying the formula z = (x - u) / s.

5.5.3 Dimensionality Reduction

To reduce the number of time and memory required by the data mining algorithms, we have performed dimensionality reduction on columns such as *lease_commence_date* and *floor_area_sqm* that are high in multicollinearity with regards to remaining_lease and flat_type_encode respectively. This is to eliminate irrelevant features and reduce noise.

5.6 Error metrics

Our team has used Mean Squared Error (MSE) as an indicator to measure how good a prediction the model makes. MSE measures the prediction error by taking the mean of all squared absolute values of all errors.

We felt that this may be a better measure of the accuracy of our predictions as any deviation from the prediction may result in cost incurred by the users. Another metric we considered was the R-squared error. However, the R-squared error does not measure predictive error. In other words, a good model can have a low R-squared error.

Note that we compared the MSE values of the scaled target variables to derive the best model before hyperparameter tuning.

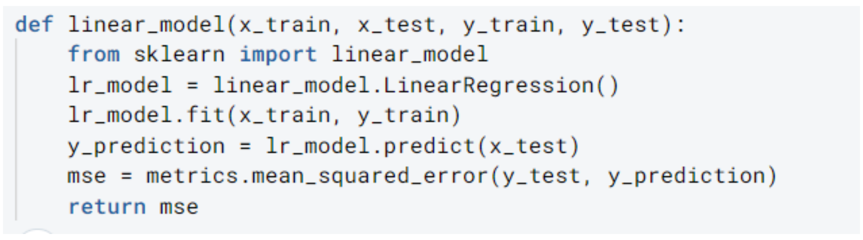
5.7 Regression Techniques

5.7.1 Train Test Splitting

In order to compare the machine learning models, we have split our data into training data (80%) and testing data (20%) and ran several regression models. (dependent variable and independent variable)

5.7.2 Linear Regression

Ordinary least squares Linear Regression model that aims to minimize the residual sum of squares between the observed and predicted targets. This method gives a MSE score of: 0.265

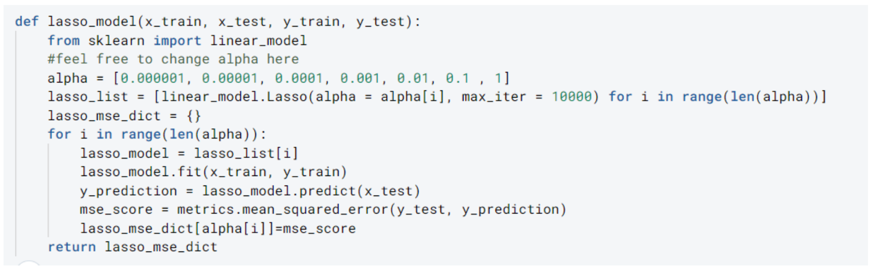


```python
def linear_model(x_train, x_test, y_train, y_test):
    from sklearn import linear_model
    lr_model = linear_model.LinearRegression()
    lr_model.fit(x_train, y_train)
    y_prediction = lr_model.predict(x_test)
    mse = metrics.mean_squared_error(y_test, y_prediction)
    return mse
```

Fig 09: code snippet of Linear Regression

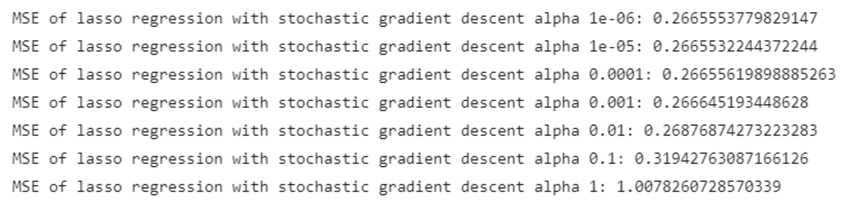
5.7.3 Lasso Regression

Lasso regression is a regression analysis method that uses shrinkage, where data values are shrunk towards a central point, in order to enhance the prediction accuracy and interpretability of the statistical model



```
def lasso_model(x_train, x_test, y_train, y_test):
    from sklearn import linear_model
    #feel free to change alpha here
    alpha = [0.000001, 0.00001, 0.0001, 0.001, 0.01, 0.1 , 1]
    lasso_list = [linear_model.Lasso(alpha = alpha[i], max_iter = 10000) for i in range(len(alpha))]
    lasso_mse_dict = {}
    for i in range(len(alpha)):
        lasso_model = lasso_list[i]
        lasso_model.fit(x_train, y_train)
        y_prediction = lasso_model.predict(x_test)
        mse_score = metrics.mean_squared_error(y_test, y_prediction)
        lasso_mse_dict[alpha[i]]=mse_score
    return lasso_mse_dict
```

Fig 10: code snippet of lasso model code



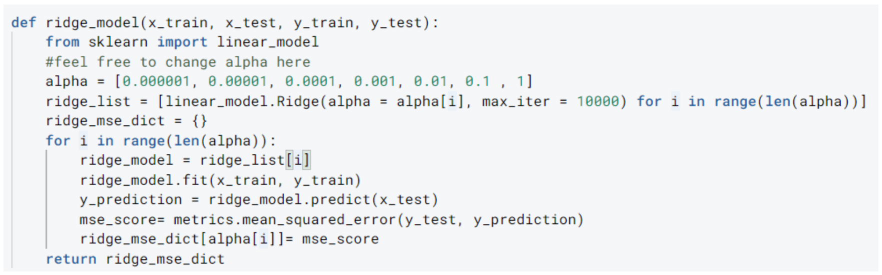
```
MSE of lasso regression with stochastic gradient descent alpha 1e-06: 0.2665553779829147
MSE of lasso regression with stochastic gradient descent alpha 1e-05: 0.2665532244372244
MSE of lasso regression with stochastic gradient descent alpha 0.0001: 0.26655619898885263
MSE of lasso regression with stochastic gradient descent alpha 0.001: 0.266645193448628
MSE of lasso regression with stochastic gradient descent alpha 0.01: 0.26876874273223283
MSE of lasso regression with stochastic gradient descent alpha 0.1: 0.31942763087166126
MSE of lasso regression with stochastic gradient descent alpha 1: 1.0078260728570339
```

Fig 11: Lasso regression results

According to our modelling, this model has a MSE of 0.267 for the best alpha.

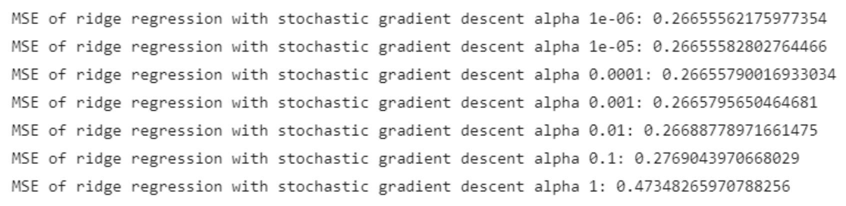
5.7.4 Ridge Regression

Ridge regression is a technique for analysing multiple regression data that suffer from multicollinearity. When multicollinearity occurs, it adds a penalty equivalent to square of the magnitude of the coefficients.



```
def ridge_model(x_train, x_test, y_train, y_test):
    from sklearn import linear_model
    #feel free to change alpha here
    alpha = [0.000001, 0.00001, 0.0001, 0.001, 0.01, 0.1 , 1]
    ridge_list = [linear_model.Ridge(alpha = alpha[i], max_iter = 10000) for i in range(len(alpha))]
    ridge_mse_dict = {}
    for i in range(len(alpha)):
        ridge_model = ridge_list[i]
        ridge_model.fit(x_train, y_train)
        y_prediction = ridge_model.predict(x_test)
        mse_score= metrics.mean_squared_error(y_test, y_prediction)
        ridge_mse_dict[alpha[i]]= mse_score
    return ridge_mse_dict
```

Fig 12: code snippet of ridge model code



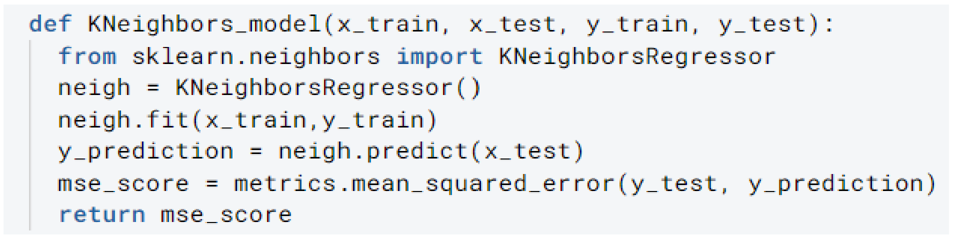
```
MSE of ridge regression with stochastic gradient descent alpha 1e-06: 0.26655562175977354
MSE of ridge regression with stochastic gradient descent alpha 1e-05: 0.26655582802764466
MSE of ridge regression with stochastic gradient descent alpha 0.0001: 0.26655790016933034
MSE of ridge regression with stochastic gradient descent alpha 0.001: 0.2665795650464681
MSE of ridge regression with stochastic gradient descent alpha 0.01: 0.26688778971661475
MSE of ridge regression with stochastic gradient descent alpha 0.1: 0.2769043970668029
MSE of ridge regression with stochastic gradient descent alpha 1: 0.47348265970788256
```

Fig 13: Ridge regression results

According to the result of the lasso regression, it has a MSE of 0.267 from the best alpha.

5.7.5 K-Nearest Neighbour

K-Nearest Neighbours Regressor predicts the dependent variable by interpolating the target associated with the nearest neighbours in the training set. In other words, the model takes the nearest 5 (default value if unspecified) and assigns weights (uniform by default) to these neighbours. The model then predicts based on the aggregation of the weights on the k-nearest neighbours. The MSE of this regression technique is 0.05991

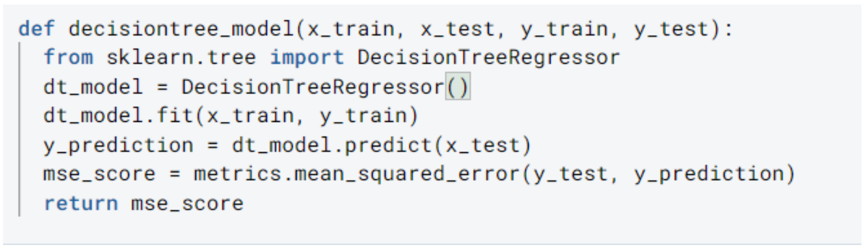


```python
def KNeighbors_model(x_train, x_test, y_train, y_test):
    from sklearn.neighbors import KNeighborsRegressor
    neigh = KNeighborsRegressor()
    neigh.fit(x_train,y_train)
    y_prediction = neigh.predict(x_test)
    mse_score = metrics.mean_squared_error(y_test, y_prediction)
    return mse_score
```

Fig 14: code snippet of Kneighbors regression model

5.7.6 Decision Tree Regressor

Decision tree regression observes features of an object and trains a model in the structure of a tree to predict data in the future to produce meaningful continuous output. The MSE of Decision Tree regression gives 0.0635.

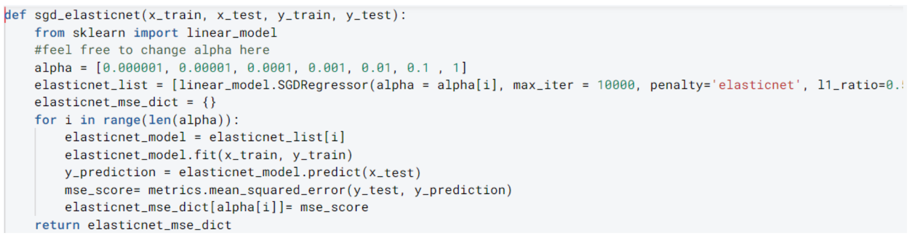


```python
def decisiontree_model(x_train, x_test, y_train, y_test):
    from sklearn.tree import DecisionTreeRegressor
    dt_model = DecisionTreeRegressor()
    dt_model.fit(x_train, y_train)
    y_prediction = dt_model.predict(x_test)
    mse_score = metrics.mean_squared_error(y_test, y_prediction)
    return mse_score
```

Fig 15: code snippet of Decision Tree regression model

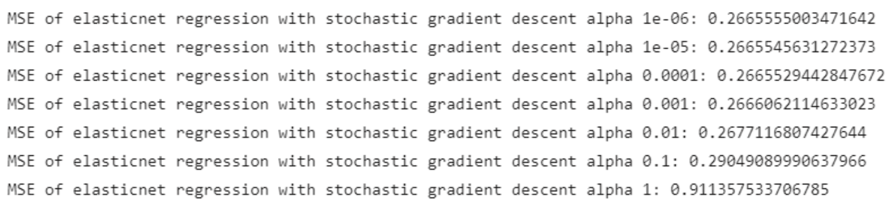
5.7.7 ElasticNet

ElasticNet is essentially a hybrid of ridge regression and lasso regression in its penalty [10]. It combines the aspect of feature elimination from Lasso Regression and the feature coefficient reduction aspect of Ridge Regression. This balance of penalties sometimes result in a better performance than Lasso or Ridge.



```python
def sgd_elasticnet(x_train, x_test, y_train, y_test):
    from sklearn import linear_model
    #feel free to change alpha here
    alpha = [0.000001, 0.00001, 0.0001, 0.001, 0.01, 0.1 , 1]
    elasticnet_list = [linear_model.SGDRegressor(alpha = alpha[i], max_iter = 10000, penalty='elasticnet', l1_ratio=0.
    elasticnet_mse_dict = {}
    for i in range(len(alpha)):
        elasticnet_model = elasticnet_list[i]
        elasticnet_model.fit(x_train, y_train)
        y_prediction = elasticnet_model.predict(x_test)
        mse_score= metrics.mean_squared_error(y_test, y_prediction)
        elasticnet_mse_dict[alpha[i]]= mse_score
    return elasticnet_mse_dict
```

Fig 16: code snippet of ElasticNet regression model



```
MSE of elasticnet regression with stochastic gradient descent alpha 1e-06: 0.2665555003471642
MSE of elasticnet regression with stochastic gradient descent alpha 1e-05: 0.2665545631272373
MSE of elasticnet regression with stochastic gradient descent alpha 0.0001: 0.2665529442847672
MSE of elasticnet regression with stochastic gradient descent alpha 0.001: 0.2666062114633023
MSE of elasticnet regression with stochastic gradient descent alpha 0.01: 0.2677116807427644
MSE of elasticnet regression with stochastic gradient descent alpha 0.1: 0.29049089990637966
MSE of elasticnet regression with stochastic gradient descent alpha 1: 0.911357533706785
```

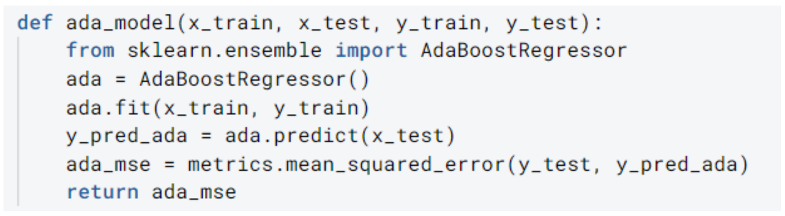
Fig 17: ElasticNet regression results

According to the result of the lasso regression, it has a MSE of 0.266 for the best alpha.

5.8 Ensemble Techniques

5.8.1 AdaBoost

AdaBoost is an ensemble technique that uses an adaptive boosting algorithm to train and predict the model [11]. This is achieved by weighing the training dataset to put more focus on training examples on which prior models made prediction errors. Performing AdaBoost ensemble on our dataset gives an MSE of 0.486.

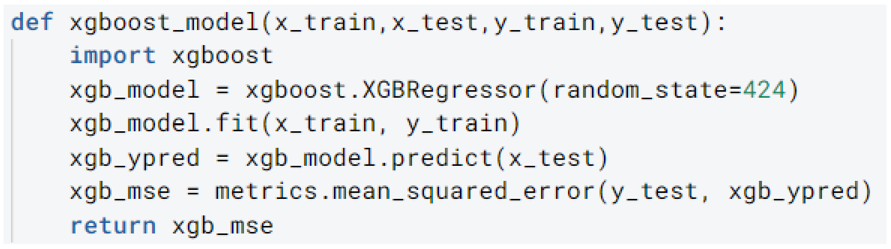


```
def ada_model(x_train, x_test, y_train, y_test):
    from sklearn.ensemble import AdaBoostRegressor
    ada = AdaBoostRegressor()
    ada.fit(x_train, y_train)
    y_pred_ada = ada.predict(x_test)
    ada_mse = metrics.mean_squared_error(y_test, y_pred_ada)
    return ada_mse
```

Fig 18: code snippet of AdaBoost

5.8.2 XGBoost

XGBoost is a decision-tree-based ensemble Machine Learning algorithm that uses a gradient boosting framework to minimize the loss when adding new models while avoid overfitting and bias [12] [13]. It is also best used for structured, tabular dataset, which is our dataset [14]. The MSE of XGBoost ensemble is 0.0438.

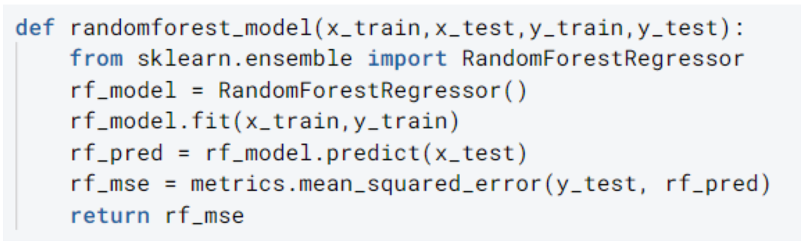


```
def xgboost_model(x_train,x_test,y_train,y_test):
    import xgboost
    xgb_model = xgboost.XGBRegressor(random_state=424)
    xgb_model.fit(x_train, y_train)
    xgb_ypred = xgb_model.predict(x_test)
    xgb_mse = metrics.mean_squared_error(y_test, xgb_ypred)
    return xgb_mse
```

Fig 19: code snippet of XGBoost

5.8.3 Random Forest Ensemble

Random forest models are constructed by using a collection of decision trees. It uses a bagging ensemble to reduce bias as each tree is built on different parts of the input at random. After, it makes a prediction on the average prediction of a collection of trees. The method of averaging the predictions of decision trees reduces the overfitting that can occur when using single decision trees. Running Random Forest ensemble gives us an MSE of 0.0345.



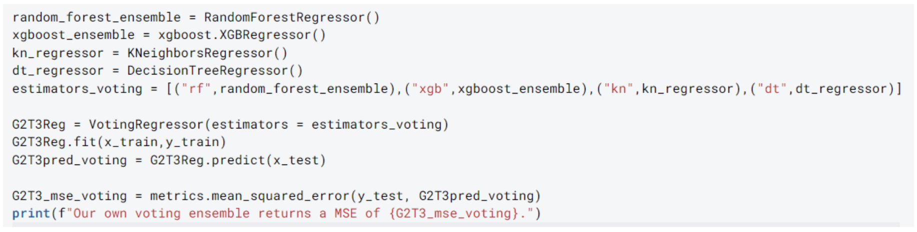
```python
def randomforest_model(x_train,x_test,y_train,y_test):
    from sklearn.ensemble import RandomForestRegressor
    rf_model = RandomForestRegressor()
    rf_model.fit(x_train,y_train)
    rf_pred = rf_model.predict(x_test)
    rf_mse = metrics.mean_squared_error(y_test, rf_pred)
    return rf_mse
```

Fig 20: code snippet of random forest ensemble

5.8.4 G2T3Voting

A voting regressor works by fitting several regression models (also referred to as *base estimators*), each on the whole train dataset, and aggregates each base estimator's individual prediction[15]. By default, the voting regressor aggregates the predictions from these base estimators using a uniform weight. However, you can change this by providing a list of integers/floats into the *weights* parameter when initializing the ensemble.

As random forest, XGBoost, k neighbours and decision tree are our top performing models with under 0.1, we used these models as our base estimators and trained the voting ensemble with them.



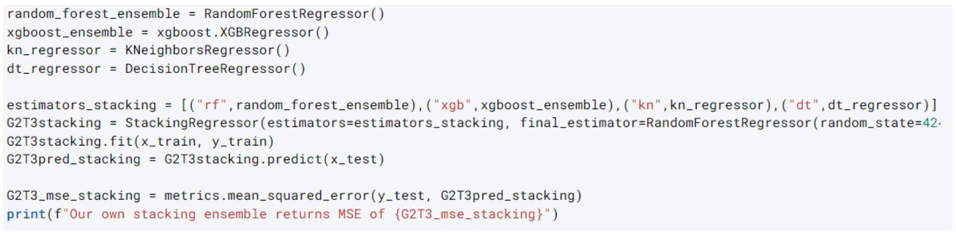
```python
random_forest_ensemble = RandomForestRegressor()
xgboost_ensemble = xgboost.XGBRegressor()
kn_regressor = KNeighborsRegressor()
dt_regressor = DecisionTreeRegressor()
estimators_voting = [("rf",random_forest_ensemble),("xgb",xgboost_ensemble),("kn",kn_regressor),("dt",dt_regressor)]

G2T3Reg = VotingRegressor(estimators = estimators_voting)
G2T3Reg.fit(x_train,y_train)
G2T3pred_voting = G2T3Reg.predict(x_test)

G2T3_mse_voting = metrics.mean_squared_error(y_test, G2T3pred_voting)
print(f"Our own voting ensemble returns a MSE of {G2T3_mse_voting}.")
```

Fig 21: code snippet of our voting ensemble

MSE of G2T3Voting ensemble:0.0361

5.8.5 G2T3Stacking

Stacking regressor works in a similar way as a voting ensemble. However, there is an additional layer of estimator above the base estimators, called the *meta-estimator*. The meta-estimator is then trained using the cross-validated predictions of the base estimators [16]. Similar to the voting ensemble above, we use the four best performing models with under 0.1 as our base and meta-estimators.

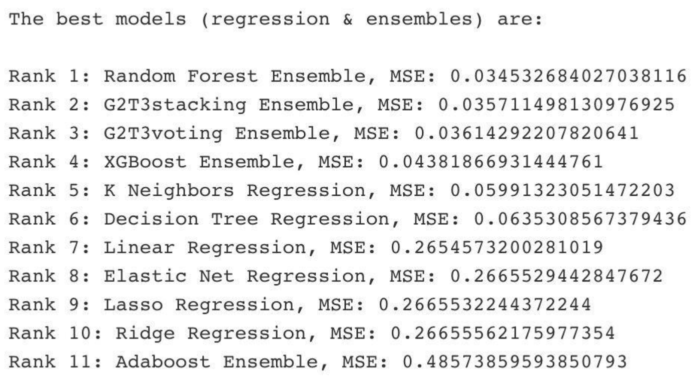


```
random_forest_ensemble = RandomForestRegressor()
xgboost_ensemble = xgboost.XGBRegressor()
kn_regressor = KNeighborsRegressor()
dt_regressor = DecisionTreeRegressor()

estimators_stacking = [("rf",random_forest_ensemble),("xgb",xgboost_ensemble),("kn",kn_regressor),("dt",dt_regressor)]
G2T3stacking = StackingRegressor(estimators=estimators_stacking, final_estimator=RandomForestRegressor(random_state=42
G2T3stacking.fit(x_train, y_train)
G2T3pred_stacking = G2T3stacking.predict(x_test)

G2T3_mse_stacking = metrics.mean_squared_error(y_test, G2T3pred_stacking)
print(f"Our own stacking ensemble returns MSE of {G2T3_mse_stacking}")
```

Fig 22: code snippet of our stacking ensemble

MSE of G2T3Stacking ensemble: 0.0357

5.9 Machine Learning Models Comparison



```
The best models (regression & ensembles) are:

Rank 1: Random Forest Ensemble, MSE: 0.034532684027038116
Rank 2: G2T3stacking Ensemble, MSE: 0.035711498130976925
Rank 3: G2T3voting Ensemble, MSE: 0.03614292207820641
Rank 4: XGBoost Ensemble, MSE: 0.04381866931444761
Rank 5: K Neighbors Regression, MSE: 0.05991323051472203
Rank 6: Decision Tree Regression, MSE: 0.0635308567379436
Rank 7: Linear Regression, MSE: 0.2654573200281019
Rank 8: Elastic Net Regression, MSE: 0.2665529442847672
Rank 9: Lasso Regression, MSE: 0.2665532244372244
Rank 10: Ridge Regression, MSE: 0.26655562175977354
Rank 11: Adaboost Ensemble, MSE: 0.48573859593850793
```

Fig 23: Comparison of regression and ensemble techniques

According to the result, Random forest ensemble was the best model with a MSE of 0.0345, followed by G2T3stacking Ensemble with a MSE of 0.0357, followed by G2T3Voting Ensemble with a MSE of 0.0361. Based on the ranking, it shows that stacking and voting are able to provide a competitive result even among more conventional ensemble techniques such as XGBoost.

We will talk more about the difference between voting and stacking in our results and discussion section below.

5.10 Selecting Regression Model

From the analysis above, random forest ensemble was the best performing model with a MSE of 0.0347. Hence, we will be proceeding with random forest ensemble method for the rest of our regression analysis.

5.11 Results and Discussion of Models

5.11.1 Stochastic Gradient Descent

We incorporated Stochastic Gradient Descent through utilizing the SKlearn SGDRegressor into some of our regression models, namely for Ridge, Lasso and ElasticNet. Stochastic Gradient descent consists of looking at the error that our weight currently gives us, using the derivative of the cost function to find the gradient, and then changing the weight to move in the direction opposite of the gradient so as to have a decreased error.

This process is then iterated through a specified number of iterations to obtain the most "locally optimal gradient" which results in the lowest MSE for each specified regression model.

However, we realised that the least_squares loss function of the standard multilinear regression is better at reducing error than Stochastic Gradient Descent for our use case, hence, our linear regression model has lower MSE than our SGDRegressor models.

5.11.2 Alpha Value

The initial result without altering the alpha value of certain models such as ridge and lasso, might cause an overfitting problem as models are trained to the optimal without factoring the error. By testing out the model with different alpha values is known as regularization technique. It penalised the coefficients of the model to prevent the model from overfit, and allowed us to find the optimal MSE.

However, we realised that regularization did not have a huge impact in reducing our MSE. This could be due to the fact that we did not face a huge problem of overfitting.

5.11.3 Stacking vs Voting

The Random Forest regression was found to perform better than the other algorithm in terms of smaller errors and be better suited as a prediction model for the house price problem.

In our modelling, stacking gave a better accuracy (lower MSE) than voting, despite given similar conditions. The base estimators and meta-estimators (where applicable) are the same models. This is largely due to the nature of the stacking algorithm as we see from the graph.

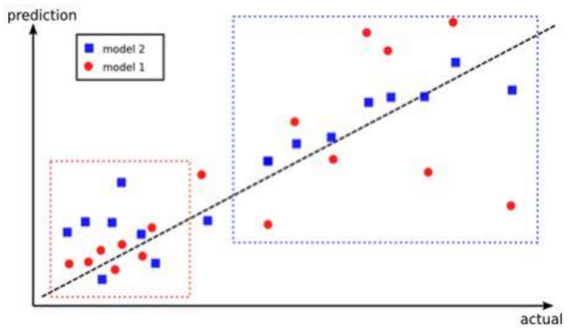


Fig 24: Stacking graph

Stacking works by selectively combining the models where it performs best, whereas voting just aggregates them. Hence, voting is more sensitive to poor predictions from its base estimators, and intuitively, stacking performs better [17].

Deep diving into the technical aspects of how this works, stacking ensemble works by splitting the training data into K number of folds.

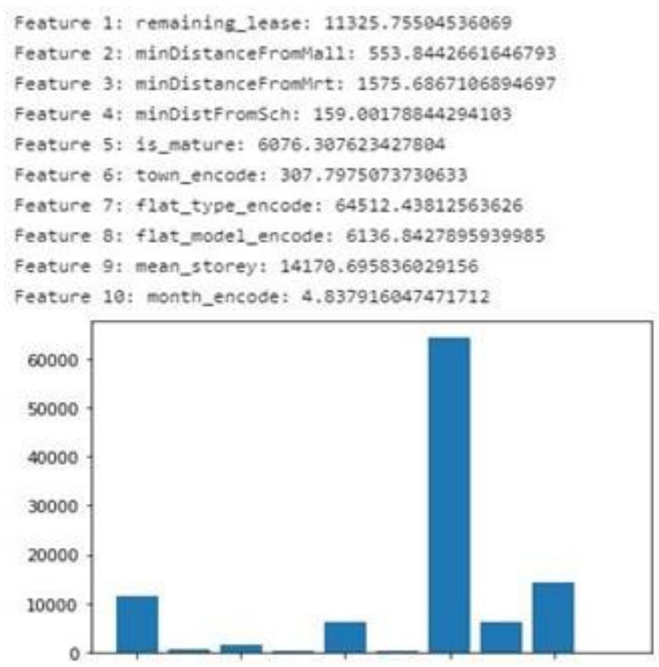
For each base model, the base model is fitted on unique K-1 parts and then tested on the remaining K parts. This is repeated for all parts of the training data. This base model is then fitted on the whole train data to calculate performance on the test set. The meta estimator is trained on these performances on the test set by the base estimators to give its final output [18].

5.12 Feature Selection

5.12.1 f_regression scores

Given the number of features we have, the team decided to perform feature selection to identify which of these features (if any) are noise. We proceeded with looking at the f_regression score of each independent variable weighted on the dependent variable.

To do so, we used the SelectKBest class from the feature selection library from sklearn. This tests the individual effect of each independent variable to the dependent variable to give the following weights

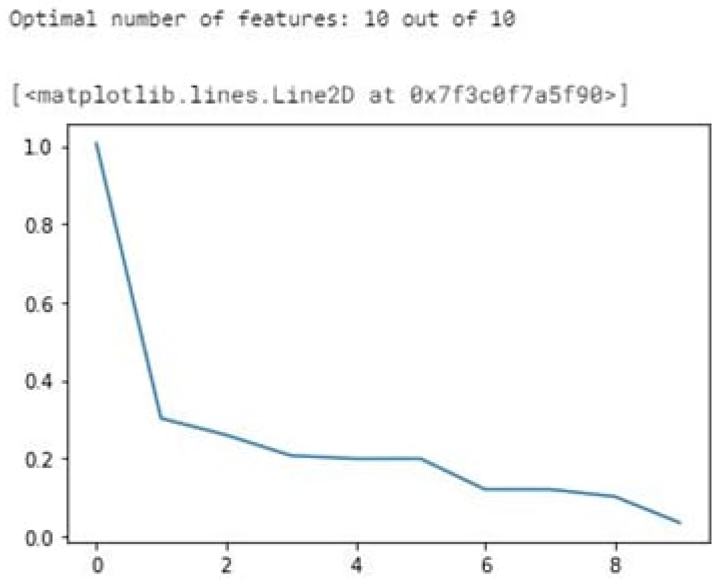


```
Feature 1: remaining_lease: 11325.75504536069
Feature 2: minDistanceFromMall: 553.8442661646793
Feature 3: minDistanceFromMrt: 1575.6867106894697
Feature 4: minDistFromSch: 159.00178844294103
Feature 5: is_mature: 6076.307623427804
Feature 6: town_encode: 307.7975073730633
Feature 7: flat_type_encode: 64512.43812563626
Feature 8: flat_model_encode: 6136.8427895939985
Feature 9: mean_storey: 14170.695836029156
Feature 10: month_encode: 4.837916047471712
```

Fig 25: Summary of f regression results

5.12.2 Greedy Selection of Features

According to the weights assigned to each feature above, we ran our selected model in a greedy fashion, where we ran the model with the top feature, followed by the top 2 features, until all features were ran. The graph below shows the MSE on the y-axis, where the x-axis shows the number of top features ran with the model.



```
Optimal number of features: 10 out of 10

[<matplotlib.lines.Line2D at 0x7f3c0f7a5f90>]
```

Fig 26: Graph of MSE to number of features used in model

5.12.3 Selected Features

As seen in 5.12.2, the MSE continuously decreases as more features are added, suggesting that all features are noise. Given our dataset with ~112 thousand rows and 10 features, we do not require removing any columns or performing Principal Component Analysis to reduce dimensionality further. Hence, we will be proceeding with our current dataset.

5.13 Model Optimization

5.13.1 Stratified Sampling

As there is a large amount of data in the dataset, to optimize the model by changing each parameter to find out which state is better requires a large amount of computational power that we do not have access to.

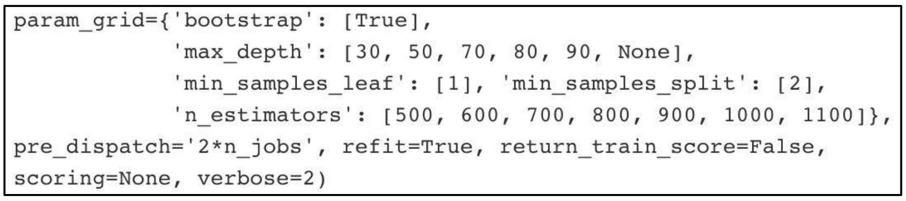
Stratified sampling is performed as a percentage of the data to retrieve a more representative sample of our dataset as compared to simple random sampling. With the sample data, the model can be trained at a faster rate while performing GridSearchCV and RandomizedSearchCV.

5.13.2 Hyperparameter tuning with GridSearchCV()

The grid search algorithm was used to find the best set of values for the selected hyperparameter for Random Forest Regression.

This method exhaustively tests all combinations of hyperparameter values for a set of specified parameters and values. When running grid search and testing different values for the hyperparameters, 5-fold cross-validation is used on the data set on which the machine learning algorithms runs with the given hyperparameters.
The errors of the predictions by the algorithms are then compared for different hyperparameter values and the set of hyperparameter values that results in the lowest prediction error is the best set of hyperparameters [19].

For this project, we did a Grid search of parameters with 5-fold cross validation and fitted 210 different combinations of hyperparameters to get the lowest MSE of 0.03419. The parameters we explored with are as shown below:



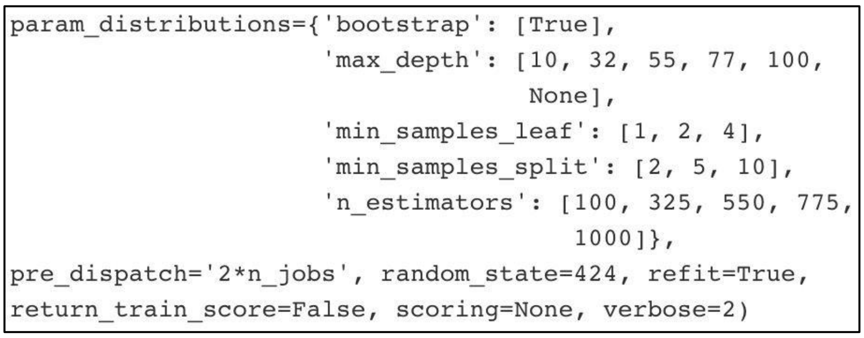
```
param_grid={'bootstrap': [True],
            'max_depth': [30, 50, 70, 80, 90, None],
            'min_samples_leaf': [1], 'min_samples_split': [2],
            'n_estimators': [500, 600, 700, 800, 900, 1000, 1100]},
pre_dispatch='2*n_jobs', refit=True, return_train_score=False,
scoring=None, verbose=2)
```

Fig 27: Parameters explored with GridSearchCV()

5.13.3 Hyperparameter Tuning with RandomizedSearchCV()

RandomizedSearchCV can sample a given number of candidates from a parameter space with a specified distribution. The parameters of the estimator used to apply these methods are optimized by cross-validated search over parameter settings.

In contrast to GridSearchCV, not all parameter values are tried out but rather a fixed number of parameter settings is sampled from the specified distributions. We defined parameters such as n_estimators (number of trees in random forest), max_depth (maximum number of levels in tree), min_samples_split, min_samples_leaf using the Numpy Linspace library [20]. The parameters generated by the library are as shown below:



```
param_distributions={'bootstrap': [True],
                     'max_depth': [10, 32, 55, 77, 100,
                                   None],
                     'min_samples_leaf': [1, 2, 4],
                     'min_samples_split': [2, 5, 10],
                     'n_estimators': [100, 325, 550, 775,
                                      1000]},
pre_dispatch='2*n_jobs', random_state=424, refit=True,
return_train_score=False, scoring=None, verbose=2)
```

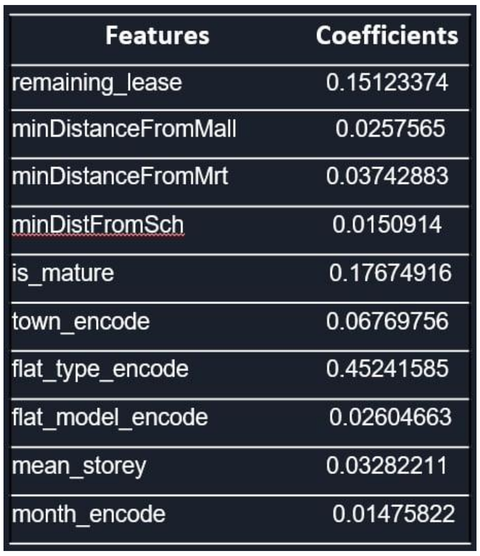
Fig 28: Parameters explored with RandomizedSearchCV()

The MSE of the RandomizedSearchCV() is 0.03422.

5.13.4 Result from Optimization

After performing optimization, GridSearchCV was found to be the best optimization technique to help reduce the MSE of the model. It helps to improve the initial Random Forest Ensemble's MSE by 0.00025 (0.75%) from 0.03444 to 0.03419.

5.14. Insights on Important Features



Features	Coefficients
remaining_lease	0.15123374
minDistanceFromMall	0.0257565
minDistanceFromMrt	0.03742883
minDistFromSch	0.0150914
is_mature	0.17674916
town_encode	0.06769756
flat_type_encode	0.45241585
flat_model_encode	0.02604663
mean_storey	0.03282211
month_encode	0.01475822

Fig 29: Coefficient for each independent variable

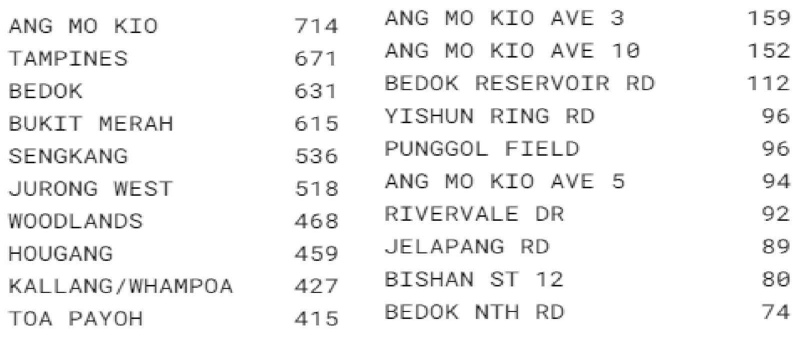
After knowing the best model for prediction and optimizing the best model, the coefficient for each independent variable is generated for each feature to create the regression line. The coefficient represents the accurate weightage of each feature related to housing price.

5.15 Insights on Undervalued Properties

Our team also conducted additional analysis to find out which are the undervalued properties in Singapore which gives more 'bang for the buck' when purchased. Such a scenario happens when predicted resale HDB flat price is above actual resale price.

Our team applied the original dataset used in our analysis onto the best Machine Learning model determined, Optimized Random Forest Ensemble. After which, we compared the Resale Flat prices between the prices predicted by our model to the actual resale prices obtained from the dataset. We categorized resale HDB Flats to be undervalued if their predicted prices are more than their actual resale HDB Price, and overvalued if otherwise.

From this, our team further broke down the Undervalued properties according to their respective town locations and street name as follows:



```
ANG MO KIO          714    ANG MO KIO AVE 3        159
TAMPINES            671    ANG MO KIO AVE 10       152
BEDOK               631    BEDOK RESERVOIR RD      112
BUKIT MERAH         615    YISHUN RING RD           96
SENGKANG            536    PUNGGOL FIELD            96
JURONG WEST         518    ANG MO KIO AVE 5         94
WOODLANDS           468    RIVERVALE DR             92
HOUGANG             459    JELAPANG RD              89
KALLANG/WHAMPOA     427    BISHAN ST 12             80
TOA PAYOH           415    BEDOK NTH RD             74
```

Fig 30: undervalued properties by town and address

From this analysis, we are able to gain insights into the location of the properties which are undervalued and can capitalize on these insights, perhaps by purchasing HDB Resale flats from these specific towns and streets.

6. Comparison to Literature Reviews

	Literature Review	Evaluation Metrics
1	Predicting the prices of HDB flats in SG	RMSE: 48.88
2	Comprehensive Data Exploration with Python	NIL
3	Singapore Flat Price Predictor	RMSLE: 0.049

Our model's RMSE is 15259.67 (Unscaled). The model has a high RMSE value as the magnitude of the target variable is relatively high. However, our model's RMSLE has a low value of 0.0343 (unscaled).

For 1, even though they had a lower RMSE, they used a different approach as they trained several models based on resale units in the same town. However, we built a model that generalises all HDB resale flats, as it would help us compare different flats in different towns and derive the undervalued properties as described in our predictive analytics.

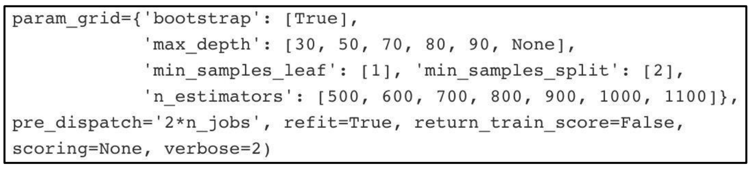
For 3, Our model's RMSLE is 0.0343 as compared to their RMSLE as compared to theirs, an estimated 40% reduction in error. they used the same approach as us, however, they trained the model on lesser features compared to our model, as they did not do an extensive feature engineering as compared to our team.

7. Effectiveness of proposed solution

We believe that our proposed solution solves the problem statement effectively. The two aims of our problem statement are as follows:

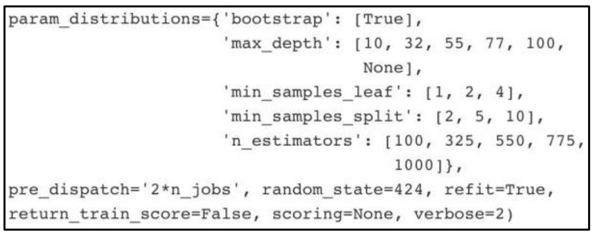
7.1 Find out key features affecting HDB resale prices

After conducting our analysis on various Machine Learning models, we have determined the optimal model to be that of the Random Forest Ensemble, which provides the lowest MSE. Afterwhich, we performed hyperparameter tuning with RandomizedSearchCV() and GridSearchCV() with 5-fold cross validation on the Random Forest Ensemble to optimize the hyperparameters.



```
param_grid={'bootstrap': [True],
            'max_depth': [30, 50, 70, 80, 90, None],
            'min_samples_leaf': [1], 'min_samples_split': [2],
            'n_estimators': [500, 600, 700, 800, 900, 1000, 1100]},
pre_dispatch='2*n_jobs', refit=True, return_train_score=False,
scoring=None, verbose=2)
```

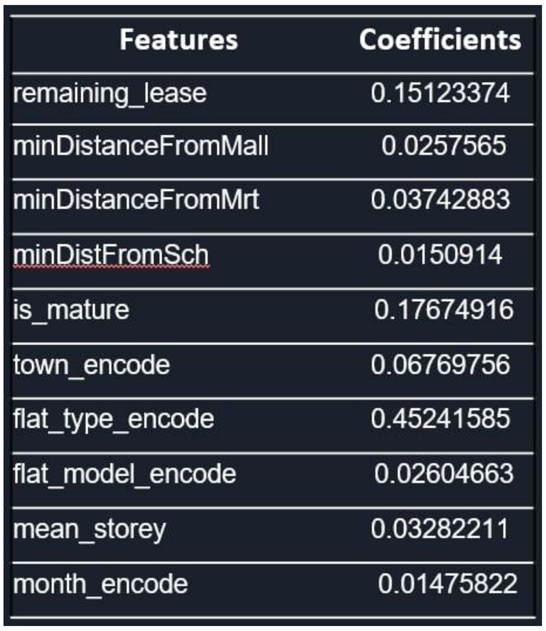
Figure 31: GridSearchCV() results



```
param_distributions={'bootstrap': [True],
                     'max_depth': [10, 32, 55, 77, 100,
                                   None],
                     'min_samples_leaf': [1, 2, 4],
                     'min_samples_split': [2, 5, 10],
                     'n_estimators': [100, 325, 550, 775,
                                      1000]},
pre_dispatch='2*n_jobs', random_state=424, refit=True,
return_train_score=False, scoring=None, verbose=2)
```

Figure 32 :RandomizedSearchCV() results

The coefficients of the features in the optimized Random Forest Ensemble captures the accurate weights of each feature related to housing price as shown below.



Features	Coefficients
remaining_lease	0.15123374
minDistanceFromMall	0.0257565
minDistanceFromMrt	0.03742883
minDistFromSch	0.0150914
is_mature	0.17674916
town_encode	0.06769756
flat_type_encode	0.45241585
flat_model_encode	0.02604663
mean_storey	0.03282211
month_encode	0.01475822

Figure 33: Features and Coefficients

As shown from the figure above, the key features affecting housing prices are features such as flat_type_encode (eg: 4-Room, 3-Room), is_mature (eg: if resale flats are located in mature/non-mature estates.) and remaining_lease left. The higher the coefficient, the greater the importance of the feature affecting resale flat price.

7.2 Predict HDB Resale prices

After optimizing our best model, the Random Forest Ensemble and obtaining the various weights for each of the respective features. We have obtained the most optimal regression equation for predicting HDB Resale prices based on the coefficients of the features.

The MSE of the optimized Random Forest Ensemble is 0.0342, which is considered to be minimal in the case of predicting resale prices. Thus, we can say with utmost confidence that our regression equation is accurate in predicting HDB resale prices.
Overall, we are very confident that our proposed solution solves the problem statements very effectively.

8. Future Work and Conclusion

8.1 Future Work

8.1.1 Increased Features

For future work, the model trained can take into account other important factors such as inflation rate and regulation. As we can tell from our Exploratory Data Analysis, there is a significant price difference when regulations are put in place by the government. Taking into account these factors can help to further enhance the accuracy of the model.

Additional features such as distance to the business district can be taken in consideration as the labour force participation rate in Singapore is at 68% as of 2019 [21]. This shows that there are a large proportion of Singaporeans who are working and may prefer buying a resale flat that is closer to their place of work.

8.1.2 Further Parameter Tuning

We could optimize all our Machine Learning models used, through hyperparameter tuning by GridSearchCV() and RandomizedSearchCV(), instead of just optimizing the best model, Random Forest Ensemble.

This is so we can obtain a better result from other optimized models, which can be used to obtain a regression equation for HDB resale price with a lower MSE than that of the current optimized Random Forest Ensemble.

8.1.3 Accessibility of Information

We could share our analysis online and findings online with others who may be interested in our project topic as well. Through such sharing, it can help others in their financial planning to determine the HDB resale prices and also help them to understand how our analysis is conducted.

Furthermore, we may even be able to gain valuable insights from others through the discussions and sharing, on what could be done to improve our current models or analysis for future improvements.

8.2 Conclusion

In conclusion, the research question for this study is to determine how well resale flat prices in Singapore can be predicted by using the different machine learning algorithms studied. From our analysis, we determined that Random Forest Ensemble was the best model to predict Resale HDB prices.

In tree-based models, we can reduce bias and prevent overfitting by using the random subspace method. Additionally, scaling of the variables does not matter as any monotonic transformation of a single variable is implicitly captured by the tree.

Overall, our model is proven effective in determining key features that affect Resale HDB prices, as well as predicting Resale HDB Housing prices.

9. References

[1] R. Navaratnarajah, “88% Of Singaporeans Unhappy Over High Property Prices: PropertyGuru,” 3 8 2018. [Online]. Available: https://www.propertyguru.com.sg/property-management-news/2018/8/173701/88-of-singaporeans-unhappy-over-high-property-prices-propertyguru. [Accessed 28 9 2020].

[2] J. Muller, “Population living in public
housing in Singapore 2010-2018 Published by J. Müller, Jan 14, 2020 This
statistic shows the share of the resident population in Singapore living in
public housing under the Housing and Development Board (HDB), from 2010 to 20,”
14 1 2020. [Online]. Available: https://www.statista.com/statistics/966747/population-living-in-public-housing-singapore/.

[Accessed 28 9 2020].

[3] C. Z. Liang, “Predicting the prices of HDB
flats in SG,” [Online]. Available:
https://www.kaggle.com/chenzhiliang/predicting-the-prices-of-hdb-flats-in-sg .

[Accessed 27 9 2020].

[4] P. Marcelino, “Comprehensive data
exploration with Python,” 2 2017. [Online]. Available:
https://www.kaggle.com/pmarcelino/comprehensive-data-exploration-with-python.

[Accessed 27 9 2020].

[5] S. H. Gan, “Singapore Flat Price Predictor,”
3 6 2019. [Online]. Available:
https://towardsdatascience.com/singapore-flat-price-predictor-6f74ed8da311.

[Accessed 28 9 2020].

[6] Housing and Development Board, “Resale Flat
Prices,” [Online]. Available: https://data.gov.sg/dataset/resale-flat-prices.

[Accessed 29 9 2020].

[7] StraitsTimes, “Property sentiment hit by cooling measures: Report,” 23 10 2018. [Online]. Available: https://www.straitstimes.com/business/property/property-sentiment-hit-by-cooling-measures-report.

[8] R. Navaratnarajah, “The MRT Effect: How It Will Affect Your Property's Value,” 25 2 2015. [Online]. Available: https://www.propertyguru.com.sg/property-management-news/2015/2/85301/the-mrt-effect-how-it-will-affect-your-propertys-value#:~:text=Distance%20to%20train%20station,those%20which%20are%20further%20away.. [Accessed 4 11 2020].

[9] J. T. A. LIN, “Make new towns more
appealing to home buyers Read more at
https://www.todayonline.com/voices/make-new-towns-more-appealing-home-buyers,”
22 6 2014. [Online]. Available:
https://www.todayonline.com/voices/make-new-towns-more-appealing-home-buyers.

[Accessed 4 11 2020].

[10] Scikit Learn, “Scikit Learn - Elastic-Net,” 5 5 2020. [Online]. Available: https://www.tutorialspoint.com/scikit_learn/scikit_learn_elastic_net.htm. [Accessed 4 11 2020].

[11] A. Navlani, “AdaBoost Classifier in Python,” 21 11 2018. [Online]. Available: https://www.datacamp.com/community/tutorials/adaboost-classifier-python. [Accessed 4 11 2020].

[12] ReadTheDocs, “XGBoost Documentation,” 2020. [Online]. Available: https://xgboost.readthedocs.io/en/latest/. [Accessed 4 11 2020].

[13] V. Morde, 4 8 2019. [Online]. Available:
https://towardsdatascience.com/https-medium-com-vishalmorde-xgboost-algorithm-long-she-may-rein-edd9f99be63d.

[Accessed 4 11 2020].

[14] T. Chen, “XGBoost: A Scalable Tree Boosting System,” 2016. [Online]. Available: https://www.kdd.org/kdd2016/papers/files/rfp0697-chenAemb.pdf. [Accessed 4 11 2020].

[15] Scikit Learn, “sklearn.ensemble.VotingRegressor,” 11 5 2020. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.VotingRegressor.html. [Accessed 4 11 2020].

[16] Scikit Learn,
“sklearn.ensemble.StackingRegressor,” 11 5 2020. [Online]. Available:
https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.StackingRegressor.html.

[Accessed 4 11 2020].

[17] U. S. B. Burak Himmetoglu, “Stacking
Models for Improved Predictions,” 2 2017. [Online]. Available:
https://www.kdnuggets.com/2017/02/stacking-models-imropved-predictions.html.

[Accessed 4 11 2020].

[18] GeeksforGeeks, “Stacking in Machine
Learning,” 20 5 2019. [Online]. Available:
https://www.geeksforgeeks.org/stacking-in-machine-learning/. [Accessed 4 11

2020].

[19] Scikitl-learn, “sklearn.model_selection.GridSearchCV,” 8 8 2020. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html.

[20] scikit-learn, “sklearn.model_selection.RandomizedSearchCV,” 8 8 2020. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.RandomizedSearchCV.html.

[21] MOM, “Summary Table: Labour Force,” 30 1 2020. [Online]. Available: https://stats.mom.gov.sg/Pages/Labour-Force-Summary-Table.aspx