



DSA211 Statistical Learning with R – G1

Group Project Part 2

Chan Yeu Herng (01391785)

Chong Le Kai (01397863)

Gayle Lee Xin Ni (01407799)

Sarah Tan Jia En (01429167)

Prithviraj Mukhopadhyay (01395279)

Cover Page	
<b>1 Project Scope</b>	<b>3</b>
1.1 Project Objective	3
1.2 Model Selection Criteria (Mean Squared Error)	3
<b>2 Model Selection Methods</b>	<b>3</b>
2.1 Multiple Linear Regression Model	4
2.2 Best Subset Selection	4
2.2 Regularization Techniques	5
2.3.1 Ridge Regression Approach	5
2.3.2 Lasso Approach	7
2.4 Tree-Based Methods	8
2.4.1 Bagging (mtry = 5)	8
2.4.2 Random Forest Approach (mtry = 4)	9
<b>3 Best Predictive Model</b>	<b>11</b>
<b>4. Predictive Power of Model on Test set Rm5HDB2023testP.cv</b>	<b>12</b>
4.1 Test Error Derivation	12
4.2 Comparison between Predicted & Observed values	12
4.3 Residual Assumptions	12
<b>5 Annex</b>	<b>13</b>
5.1 Linear Model	14
5.2 Best Subset Selection	15
5.3 Ridge Regression	16
5.4 Lasso	17
5.5 Bagging Approach	18
5.6 Random Forest	18
5.7 Final Model (Random Forests mtry=4)	19
5.8 Comparison between Predicted and Observed Values	20
5.9 Residual Assumptions	20

# 1 Project Scope

## 1.1 Project Objective

The objective of the report is to explore and construct a predictive model for resale *Price* of 5 room HDB flats using the following parameters: *Town*, *Story*, *Area*, *Model* and *LeaseRem*.

To delve deeper into this study, we explored the training data set, Rm5HDB2023P.csv to derive the best recommended model before testing out its predictive power in the Rm5HDB2023testP.csv dataset.

**Figure 1:** Best Model selected from all approaches

```
finmodel<- randomForest(Price~., data=hdb, mtry=4, importance=TRUE)

call:
randomForest(formula = Price ~ ., data = hdb, mtry = 4, importance = TRUE)
  Type of random forest: regression
  Number of trees: 500
No. of variables tried at each split: 4

  Mean of squared residuals: 2153262009
  % Var explained: 88.68
```

MSE on Test Rm5HDB2023testP.csv: **2339643383**.

## 1.2 Model Selection Criteria (Mean Squared Error)

To generalise the model beyond the training data, it is essential that the prediction model does not have high variance or high bias indicating that the training data is either overfitted or underfitted in the model.

To ensure that the model performs well on the test data set, it is necessary to achieve reasonably low variance and low bias due to the variance-bias tradeoff. Therefore, the model with lowest mean-squared error will be selected.

# 2 Model Selection Methods

Prior to running any models, our team checked the model for NA values which yield 0 fields.

In order to test the significance of the variables and to find the count of all the independent variables (36), our team first decided to run a linear regression of *Price* against all independent variables. All the independent variables were significant at 95% significance level.

To ensure uniformity in testing and enhance the predictability within each model, we have decided to apply cross validation to all models.

The models that we will be comparing are:

1. Multiple Linear Regression Model
2. Best Subset Selection
3. Ridge Regression Approach
4. Lasso Approach
5. Bagging Procedure
6. Random Forest Approach

## 2.1 Multiple Linear Regression Model

Our team has split our training data into two equal parts: one as training data ("train"), and the other as test data ("test").

Once the Linear Model was populated on our training set, we used the model to predict the values of Price with our test data. This gave us a Test MSE of **32674663975**.

## 2.2 Best Subset Selection

To enhance precision accuracy and to control the variance and improve interpretability of our model, we decided to run the Best Subset selection to identify a subset of p predictors.

In order to run the Best Subset Selection, our team first random seed to (6789), followed by calling on the leaps library in order to run subset selection functions. The `leaps::regsubsets()` function was then run to populate a subset selection on the whole training data set.

Once the subset selection was done, we decided to use cross-validation to make less assumptions about the true underlying model. Cross-Validation was completed in the following steps:

1. Creating a `predict.regsubsets` function that allowed us to predict subset selection outcomes based on varying data sets and objects.
2. After that, we assigned  $k = 10$  to dictate our number of folds.
3. Created a training set based of the training data to determine the number of folds
4. Created a matrix to store the cross validation error values for each of the 10-fold CV.
5. Once all the vectors were assigned, a nested for loop was run in order to run a best subset selection via 10-fold Cross validation. Each loop will then input a cross validation error value into our `cv.errors` matrix.
6. Once the `cv.errors` matrix was filled, the mean cross validation error for each model was calculated using an `apply()` function on the columns. This provided us with a `mean.cv = 3446954895`.
7. Once the means of the cross validation errors was calculated, we had to select the model that provided us with the minimum cross validation error. Based on this particular model ("Best Subset Selection Model"), we assigned the number of regressors within its subset to "bb". This provided us with a `bb = 36`, which indicated that the Best model included 36 regressors.
8. Finally, we regressed the Best Model with the 36 variables against the total training set (`regfit.all`) to find the coefficient values. This provided us with the model of:

**Figure 2:** Coefficients of all variables in the Best Model

> coef(hdb.all, aa)			
(Intercept)	TownBEDOK	TownBISHAN	TownBUKIT BATOK
-358109.466	-69783.631	92520.388	-149768.093
TownBUKIT MERAH	TownBUKIT PANJANG	TownBUKIT TIMAH	TownCHOA CHU KANG
85147.607	-202439.718	183842.856	-251485.220
TownCLEMENTI	TownGEYLANG	TownHOUgang	TownJURONG EAST
25079.051	59262.830	-155379.588	-138868.042
TownJURONG WEST	TownKALLANG/WHAMPOA	TownMARINE PARADE	TownPASIR RIS
-225543.722	28847.425	180995.195	-175969.515
TownPUNGGOL	TownQUEENSTOWN	TownSEMBAWANG	TownSENGKANG
-212138.480	123751.140	-225825.085	-218438.116
TownSERANGOON	TownTAMPINES	TownTOA PAYOH	TownWOODLANDS
-53649.826	-132530.428	19508.205	-239516.045
TownYISHUN	Story04 TO 06	Story07 TO 09	Story10 TO 12
-195853.513	23564.063	41975.236	50199.181
Story13 TO 15	Story16 TO 18	Story19Above	Area
58666.420	77111.692	126983.921	6114.591
ModelImproved	ModelModel A	ModelPremium Apartment	ModelStandard
-164817.607	-175690.256	-154994.930	-164757.939
LeaseRem			
7266.997			

### Evaluation

However, since  $p = 36$  is large, the model may be overfitted and have high variance of the coefficient estimates due to enormous search space.

## 2.2 Regularization Techniques

Besides using Subset Selection approaches like Best Subset Selection, our team checked Regularization techniques such as Ridge Regression where the objective was to shrink the regressors coefficient instead of conducting variable selection. Hence, our team decided to run a model using the ridge regression and lasso method.

Prior to running Regularization techniques, our team has decided to define our  $x$  and  $y$  variables, while keeping our grid to the default R grid. In order to ease computational inputs, we have also assigned vectors for both  $x$  and  $y$  in the training and test data.

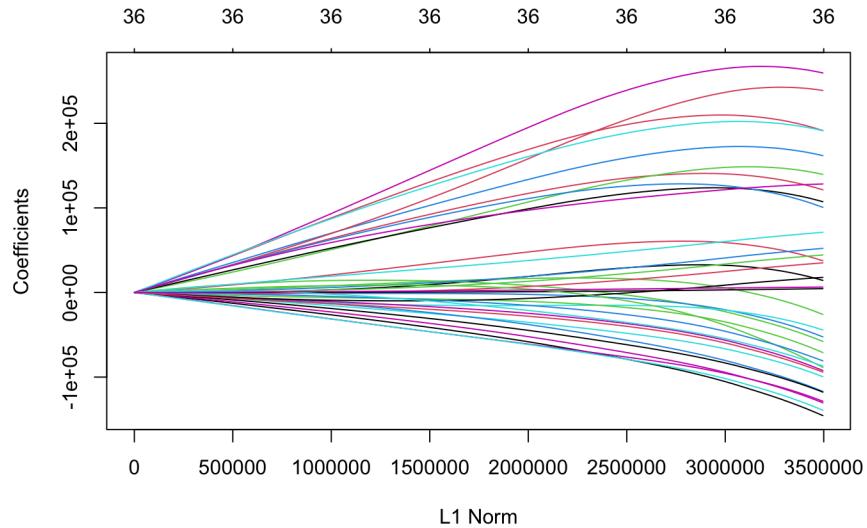
### 2.3.1 Ridge Regression Approach

As the ridge regression imposes a tuning parameter to control the relative impact of the coefficient regressors in the model by identifying the best lambda as a standardised parameter, allowing the model to have a lowest mse with the most optimal bias-variance tradeoff.

Similar to Best Subset Selection, we first set random seed to (6789), followed by calling the “glmnet” library in order for us to conduct our Regularization techniques. Once all the vectors were assigned properly, we conducted Ridge Regression based on the following steps:

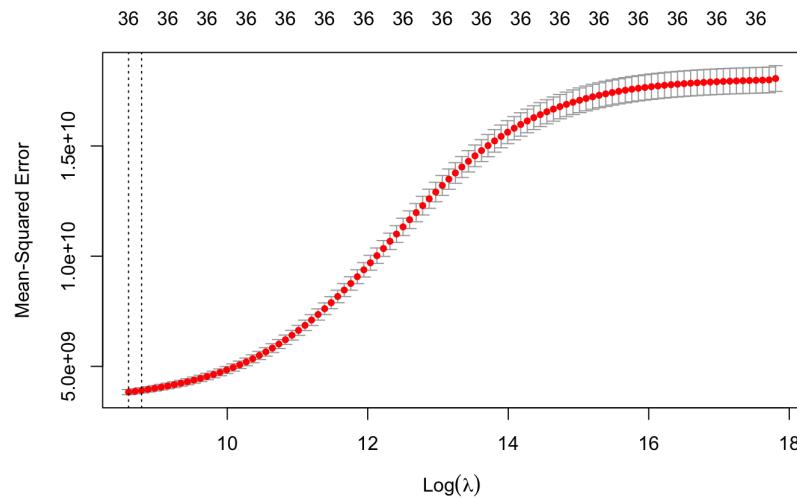
1. We first created our ridge regression model based on the training data.

**Figure 3:** Plot of Coefficients against L1 Norm for Ridge Regression



2. Next, we conducted cross validation using `leaps::cv.glmnet()` and identified the optimal tuning parameter (`lambda`)  
This gave us a value of **5407.103** for our optimal tuning parameter.

**Figure 4:** Plot of Mean-Squared Error against Log of Lambda for Ridge Regression



3. Next, in order to find out the test MSE, we used our optimal training model to predict the outputs based on the test data set. This provided us with a MSE of **3975723793**.
4. Finally, with our optimal lambda, we predicted the coefficients of the optimal model using our overall training data. This provided us with the model of:

**Figure 5:** Summary statistics of coefficients in the model incorporating the optimal lambda value

(Intercept)	-253592.624	TownQUEENSTOWN	204296.709
(Intercept)	.	TownSEMAWANG	-124281.280
TownBEDOK	22630.373	TownSENGKANG	-117606.423
TownBISHAN	189995.702	TownSERANGOON	41782.231
TownBUKIT BATOK	-49483.637	TownTAMPINES	-25563.537
TownBUKIT MERAH	167877.180	TownTOA PAYOH	121005.538
TownBUKIT PANJANG	-102804.410	TownWOODLANDS	-133734.873
TownBUKIT TIMAH	267288.677	TownYISHUN	-91396.551
TownCHOA CHU KANG	-144136.321	Story04 TO 06	16613.252
TownCLEMENTI	118202.320	Story07 TO 09	33293.019
TownGEYLANG	142049.059	Story10 TO 12	41976.477
TownHOUgang	-51114.398	Story13 TO 15	48941.650
TownJURONG EAST	-45011.945	Story16 TO 18	70599.915
TownJURONG WEST	-124673.728	Story19above	131855.340
TownKALLANG/WHAMPOA	121497.793	Area	4395.065
TownMARINE PARADE	256030.552	ModelImproved	-92938.882
TownPASIR RIS	-66909.483	ModelModel A	-82852.488
TownPUNGGOL	-110079.637	ModelPremium Apartment	-83143.371
TownQUEENSTOWN	204296.709	ModelStandard	-95588.600
TownSEMAWANG	-124281.280	LeaseRem	6407.155

## Evaluations

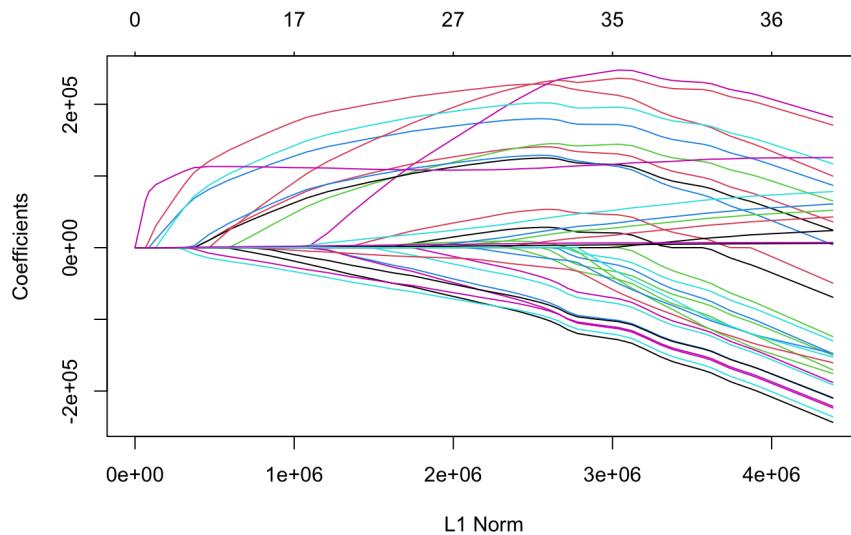
However, we evaluated the limitations of the ridge regression model where the model adds in all p predictors in the final model. Hence, we decided to do a Lasso approach to overcome this disadvantage by minimising the quantity whilst penalising the coefficients of the regressors.

### 2.3.2 Lasso Approach

Similar to Lasso Approach, we first set random seed to (6789), followed by calling the “glmnet” library in order for us to conduct our Regularization techniques. Once all the vectors were assigned properly, we conducted Lasso based on the following steps:

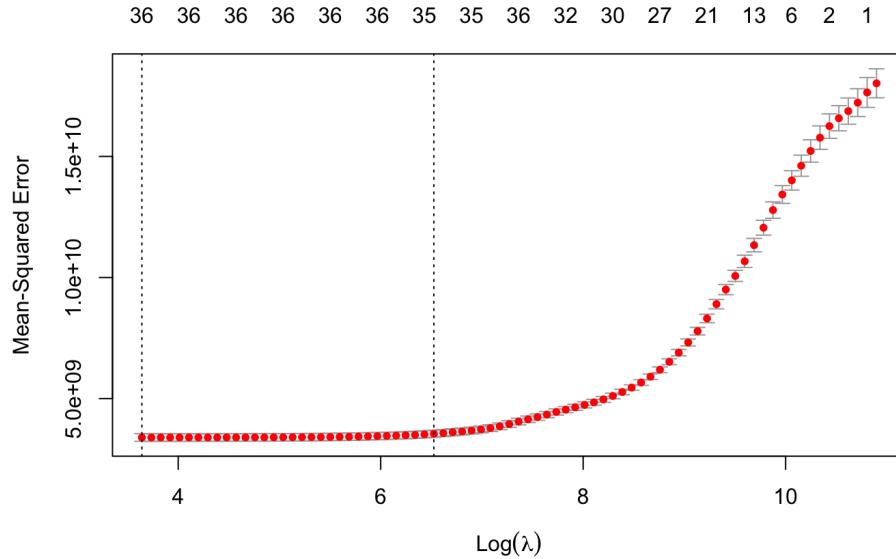
1. We first created our lasso model based on the training data.

**Figure 6:** Plot of Coefficients against L1 Norm for Lasso



2. Next, we conducted cross validation using `leaps::cv.glmnet()` and identified the optimal tuning parameter (`lambda`). This gave us a value of **38.14604** for our optimal tuning parameter.

**Figure 7:** Plot of Mean-Squared Error against Log of Lambda for Lasso



3. Next, in order to find out the test MSE, we used our optimal training model to predict the outputs based on the test data set. This provided us with a MSE of **3547183929**.
4. Finally, with our optimal lambda, we predicted the coefficients of the optimal model using our overall training data. Additionally, for better visualisation, we decided to exclude all the coefficients that equalled 0. This provided us with the output of:

**Figure 8:** Summary statistics of coefficients in the model incorporating the optimal lambda value (excluding coefficients equating to 0)

```
[1] -358794.605 -63594.420  98643.905 -143541.601  90656.224 -196286.258  188996.472
[8] -245220.045  30746.511  64699.516 -148914.972 -132654.323 -219423.424   34544.615
[15] 186201.396 -169401.944 -206030.164 129006.614 -219716.310 -212411.384  -47208.264
[22] -126018.023  25472.612 -233320.298 -189626.283  23058.975  41416.654  49663.217
[29]  58041.888  76656.411 127190.395   6059.751 -162079.896 -172168.048 -152382.105
[36] -161825.553  7251.847
```

## Evaluation

Lasso and Ridge regression tend to struggle when it comes to capturing non-linear relationships between predictors and response variables. Therefore, if we are uncertain about the linearity of the relationship between these variables, these regression techniques may not be the optimal models for accurately explaining the training set or making predictions for the final test set.

## 2.4 Tree-Based Methods

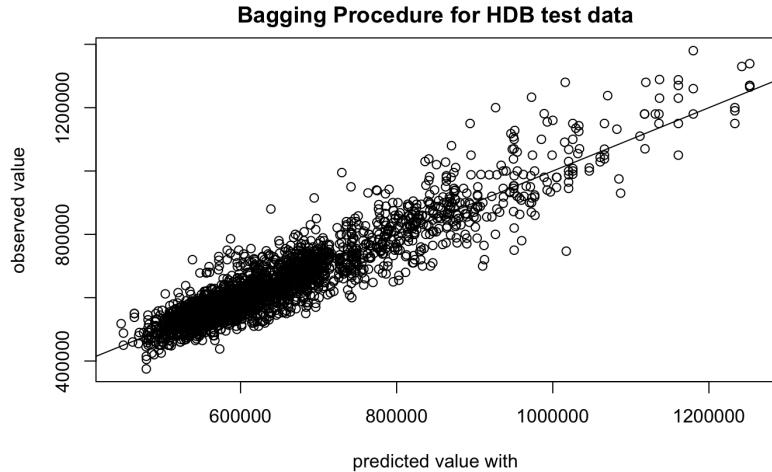
Finally, our group also considered testing for Bagging and Random Forest Approaches as the bagging could solve the issue of having non-linear predictor-response relationships.

### 2.4.1 Bagging ( $mtry = 5$ )

As usual, we set random seed to (6789) to ensure uniformity of results, followed by calling the “randomForest” library. Once it was done, we conducted bagging based on the following steps:

- With all 5 independent variables, we created a random forest using `randomForest::randomForest()`.
- We then ran our training model on the test model to find the predicted `yhat` values with our bagging approach

**Figure 9:** Plot of observed value against predicted value (Bagging procedure)

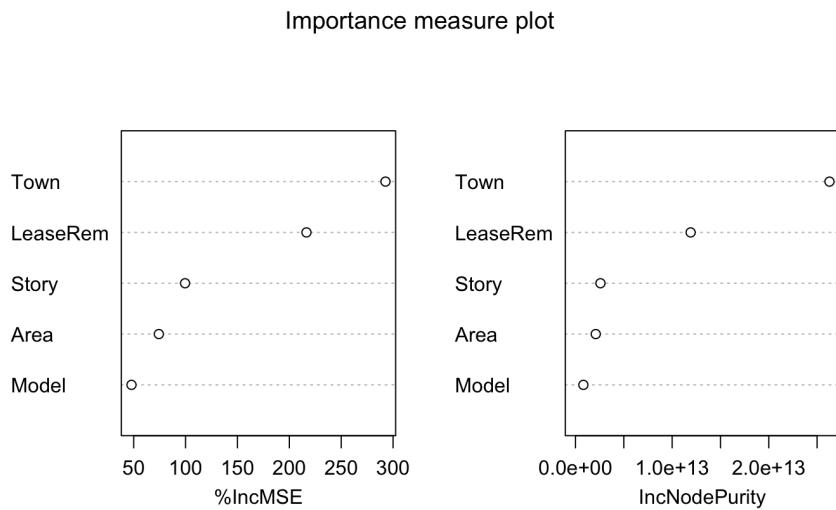


- Finally, we found the test MSE for the bagging approach. Our test MSE was found to be **2641712577**.

### Important Variables

Using `randomForest::Importance()`, we have identified that the two most important variables are Town (**2.628247e+13**) and LeaseRem (**1.191510e+13**).

**Figure 10:** Importance measure plot for Bagging Procedure:

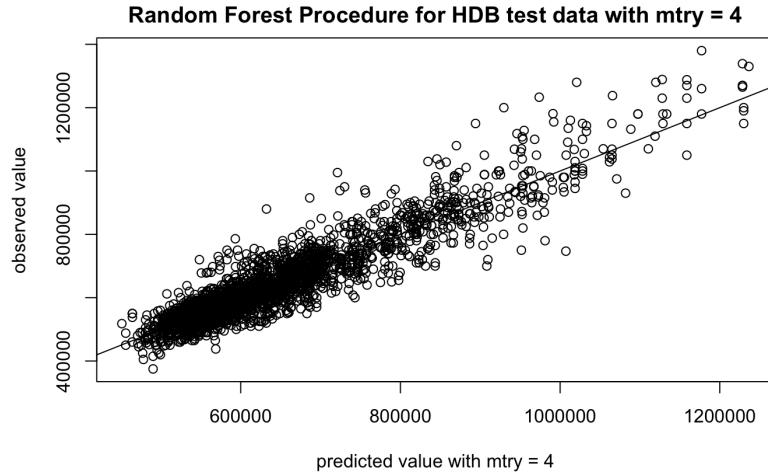


### 2.4.2 Random Forest Approach (mtry = 4)

Similarly, we set the random seed to (6789) to ensure uniformity of results. Once that was done, we conducted `randomForest` based on the following steps:

- With all 5 independent variables, we created a random forest using `randomForest::randomForest()`, with 4 variables first.
- We then ran our training model on the test model to find the predicted `yhat1` values with our random forest approach

**Figure 11:** Plot of observed value against predicted value for Random Forest approach

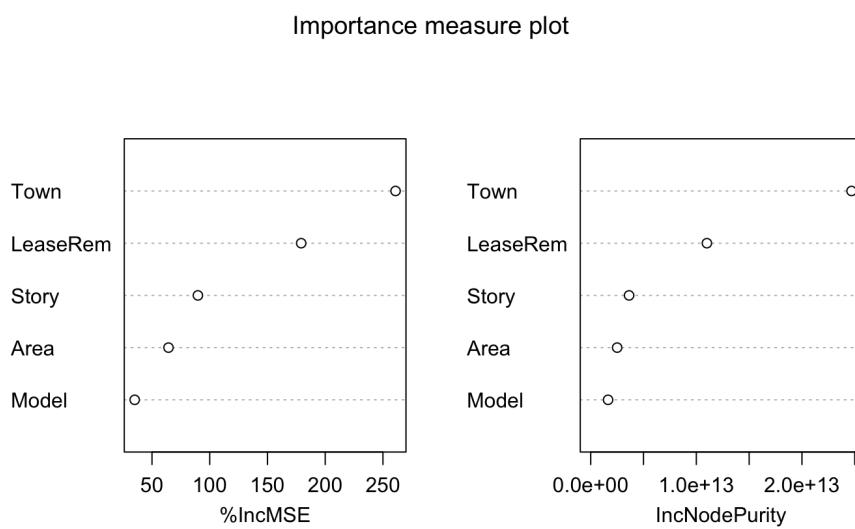


- Finally, we found the test MSE for the random tree with `mtry = 4`
- We then repeat steps 1 to 3 with different quantity of variables, `mtry = 1, 2, 3`
- Comparing the different test MSE, our best model will be based on the smallest test MSE. Our smallest test MSE was found to be **2628577302**.

### Important Variables

Using `randomForest::Importance()`, we have identified that the two most important variables are Town (**2.469564e+13**) and LeaseRem (**1.099981e+13**).

**Figure 12:** Importance measure plot for Random Forest approach



## Random Forest (mtry = 4) VS Bagging procedure

Figure 13: Table displaying values of TestMSE associated with increasing the mtry

mtry	Test MSE
5	2641712577
4	2628577302
3	2639320289
2	2736301069
1	3738783894

We can see that mtry = 4 has the lowest test MSE, which implies that within the bagging approach and random forest approaches, mtry = 4 will give the optimal predictive model.

Additionally, the training error of mtry = 4 (**86.47%**) is relatively the same as that of bagging (86.48%). Hence, this means that within the training set itself, mtry = 4 is validated to explain the same amount of variance in the training data as when all independent variables are used.

## 3 Best Predictive Model

Figure 14: Table displaying values of TestMSE associated with various models

Model	MSE
Multiple Linear Regression Model	32674663975
Best Subset Selection	3446954895
Ridge Regression Approach	3975723793
Lasso Approach	3547183929
Bagging	2641712577
Random Forest Approach (mtry = 4)	2628577302

Based on the training data provided with *Rm5HDB2023P.cv*, we concluded that our best predictive model would be Random Forest (mtry = 4) as it has the lowest MSE. Hence, it would have the strongest predictive power of all models.

## 4. Predictive Power of Model on Test set

### Rm5HDB2023testP.csv

#### 4.1 Test Error Derivation

Figure 15: Summary statistics of the final model

```
finmodel<- randomForest(Price~., data=hdb, mtry=4, importance=TRUE)

call:
randomForest(formula = Price ~ ., data = hdb, mtry = 4, importance = TRUE)
Type of random forest: regression
Number of trees: 500
No. of variables tried at each split: 4

Mean of squared residuals: 2153262009
% Var explained: 88.68
```

Using the best predictive model, we predicted the test dataset and concluded the test error of **2339643383**.

#### 4.2 Comparison between Predicted & Observed values

Figure 16: Comparison between the predicted and observed values

	y.1.5. <dbl>	pred <dbl>
1	780000	825482.7
2	630000	672742.1
3	838000	810250.2
4	750000	825482.7
5	800000	873453.0

Upon comparing the Price values predicted by our model against the observed prices in the test dataset, we can observe that the predicted values consistently overestimate the values but not by a very large margin. Furthermore, the differences between both values seem to be consistent throughout the different rows. It is a reasonable indication of the strong predictive power of our model.

#### 4.3 Residual Assumptions

Based on the residual plots that we have indicated in the Annex below, we can observe a recurring trend: the residuals when plotted against the independent variables show a constant variance and no distinct deviation.

This can also be observed in the q-q and p-p plots below, where no distinct deviation can be spotted. We can conclude that our recommended model does not violate any residual assumption, further justifying the predictive power of our model

Hence, **our best predictive model is Random Forest (mtry = 4)** to predict resale prices.

## 5 Annex

```
> hdb <- read.csv("RmSHDB2023P.csv", stringsAsFactors = TRUE)
> attach(hdb)
The following objects are masked from hdb (pos = 3):
  Area, LeaseRem, Model, Price, Story, Town
The following objects are masked from hdb (pos = 4):
  Area, LeaseRem, Model, Price, Story, Town
The following objects are masked from hdb (pos = 5):
  Area, LeaseRem, Model, Price, Story, Town
The following objects are masked from hdb (pos = 6):
  Area, LeaseRem, Model, Price, Story, Town
The following objects are masked from hdb (pos = 7):
  Area, LeaseRem, Model, Price, Story, Town
The following objects are masked from hdb (pos = 9):
  Area, LeaseRem, Model, Price, Story, Town
The following objects are masked from hdb (pos = 13):
  Area, LeaseRem, Model, Price, Story, Town
> View(hdb)
>
> # for checking
> sum(is.na(hdb)) # 0
[1] 0
```

```

> summary(lm(Price~, data = hdb))

Call:
lm(formula = Price ~ ., data = hdb)

Residuals:
    Min      1Q  Median      3Q     Max 
-254770 -36757 -4285   31985  280003 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)    
(Intercept) -358109.5   30675.8 -11.674 < 2e-16 ***
TownBEDOK      -69783.6    7126.7 -9.792 < 2e-16 ***
TownBISHAN       92520.4   8056.6 11.484 < 2e-16 ***
TownBUKIT BATOK -149768.1   7936.7 -18.870 < 2e-16 ***
TownBUKIT MERAH   85147.6   7188.1 11.846 < 2e-16 ***
TownBUKIT PANJANG -202439.7   6880.6 -29.422 < 2e-16 ***
TownBUKIT TIMAH    183842.9   19314.9 9.518 < 2e-16 ***
TownCHOA CHU KANG -251485.2   6441.5 -39.041 < 2e-16 ***
TownCLEMENTI      25079.1    9155.4  2.739 0.006180 **  
TownGEYLANG        59262.8   9723.6  6.095 1.18e-09 *** 
TownHOUANG        -155379.6   7094.7 -21.901 < 2e-16 ***
TownJURONG EAST    -138868.0   8464.6 -16.406 < 2e-16 ***
TownJURONG WEST     -225543.7   6379.5 -35.355 < 2e-16 ***
TownKALLANG/WHAMPOA -28847.4   8369.8  3.447 0.000572 *** 
TownMARINE PARADE    180995.2   11121.5 16.274 < 2e-16 ***
TownPASIR RIS       -175969.5   7073.6 -24.877 < 2e-16 ***
TownPUNGOL         -212138.5   6450.3 -32.888 < 2e-16 ***
TownQUEENSTOWN      123751.1   8682.2 14.253 < 2e-16 ***
TownSEMBAWANG      -225825.1   7179.5 -31.454 < 2e-16 ***
TownSENGKANG        -218438.1   6231.2 -35.056 < 2e-16 ***
TownSERANGOON       -53649.8    9231.7 -5.811 6.58e-09 *** 
TownTAMPINES        -132530.4   6502.3 -20.382 < 2e-16 ***
TownTOA PAYOH        19508.2   8235.4  2.369 0.017882 *  
TownWOODLANDS       -239516.0   6312.4 -37.944 < 2e-16 ***
TownYISHUN          -195853.5   6707.9 -29.197 < 2e-16 ***

Story04 T0 06      23564.1   2805.2  8.400 < 2e-16 ***
Story07 T0 09      41975.2   2772.8 15.138 < 2e-16 ***
Story10 T0 12      50199.2   2898.4 17.320 < 2e-16 ***
Story13 T0 15      58666.4   3196.8 18.351 < 2e-16 ***
Story16 T0 18      77111.7   3978.4 19.383 < 2e-16 ***
Story19Above       126983.9   4147.6 30.616 < 2e-16 ***
Area                6114.6    212.2 28.811 < 2e-16 ***
ModelImproved     -164817.6   6003.1 -27.455 < 2e-16 ***
ModelModel A      -175690.3   8035.0 -21.866 < 2e-16 ***
ModelPremium Apartment -154994.9   6361.1 -24.366 < 2e-16 ***
ModelStandard      -164757.9   8005.6 -20.580 < 2e-16 ***
LeaseRem           7267.0    111.8 65.003 < 2e-16 ***

---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 58330 on 4958 degrees of freedom
Multiple R-squared:  0.8224, Adjusted R-squared:  0.8211 
F-statistic: 637.8 on 36 and 4958 DF, p-value: < 2.2e-16

```

## 5.1 Linear Model

```

> RNGkind(sample.kind = "Rounding")
Warning: non-uniform 'Rounding' sampler used
> set.seed(6789)
>
> train <- sample(1:nrow(hdb), nrow(hdb)/2) # half sample
> test <- -train
>
> lm1 <- lm(Price ~., data = hdb[train,])
> lm.pred <- predict(lm1, hdb[test,])
> mean((hdb$Price-lm.pred)^2) # test MSE 32674663975
Warning: longer object length is not a multiple of shorter object length[1] 32674663975

```

## 5.2 Best Subset Selection

```

> set.seed(6789)
>
> library(leaps)
>
> # function to predict
> predict.regsubsets <- function(object,newdata,id) {
+   form <- as.formula(object$call[[2]])
+   mat <- model.matrix(form, newdata)
+
+   coefi <- coef(object, id = id)
+   xvars <- names(coefi)
+   mat[,xvars]%%coefi
+ }
>
> k <- 10 # assign the k-fold CV
> folds <- sample(1:k, nrow(hdb), replace = TRUE) # training set
> cv.errors <- matrix(NA, k, 36, dimnames = list(NULL, paste(1:36))) # 36 independent variables
>
> for (j in 1:k) {
+   best.fit <- regsubsets(Price~, data = hdb[folds!=j],nvmax = 36) # running the BSS
+   for (i in 1:36) {
+     pred <- predict.regsubsets(best.fit, hdb[folds == j], id = i) #extracting coefficients and then
+     multiplying them to form the prediction
+     cv.errors[j,i] <- mean((hdb$Price[folds == j]-pred)^2) # calculate test MSE
+   }
+ }
>
> mean.cv <- apply(cv.errors, 2, mean) #average over the column of the matrix to error for each model
> mean.cv
      1        2        3        4        5        6        7        8
15611514414 14916030391 13875880836 12369899127 11474470565 10974061099 10169597851 9599279962
         9       10       11       12       13       14       15       16
8647693860  8159461064  7503225273  7019126871  6662548326  6475968539  5929992350  5516437205
        17       18       19       20       21       22       23       24
4929894572  4729360036  4579348822  4516674121  4542593359  4239005796  4127929711  4031496772
        25       26       27       28       29       30       31       32
3943354527  3920207125  3753072368  3677861554  3696412079  3581142167  3528348667  3463968305
        33       34       35       36
3450742519  3452416458  3451870783  3446954895
>
> min(mean.cv) # 3446954895
[1] 3446954895
>

> # finding optimal model
> aa <- which.min(mean.cv)
> aa # 36
36
36
>
> hdb.all <- regsubsets(Price~, data = hdb, nvmax = 36)
> coef(hdb.all, aa)
    (Intercept) TownBEDOK TownBISHAN TownBUKIT BATOK
-358109.466 -69783.631 92520.388 -149768.093
TownBUKIT MERAH TownBUKIT PANJANG TownBUKIT TIMAH TownCHOA CHU KANG
  85147.607 -202439.718 183842.856 -251485.220
TownCLEMENTI TownGEYLANG TownHOUgang TownJURONG EAST
  25079.051  59262.830 -155379.588 -138868.042
TownJURONG WEST TownKALLANG/WHAMPOA TownMARINE PARADE TownPASIR RIS
 -225543.722  28847.425 180995.195 -175969.515
TownPUNGGOL TownQUEENSTOWN TownSEMBAWANG TownSENGKANG
 -212138.480  123751.140 -225825.085 -218438.116
TownSERANGOON TownTAMPINES TownTOA PAYOH TownWOODLANDS
 -53649.826 -132530.428 19508.205 -239516.045
TownYISHUN Story04 T0 06 Story07 T0 09 Story10 T0 12
 -195853.513  23564.063 41975.236 50199.181
Story13 T0 15 Story16 T0 18 Story19Above Area
               58666.420  77111.692 126983.921 6114.591
ModelImproved ModelModel A ModelPremium Apartment ModelStandard
 -164817.607 -175690.256 -154994.930 -164757.939
LeaseRem
 7266.997

>
> # plot
> plot(1:36, mean.cv, main = "Relationship between Mean.cv and Number of variables", xlab = "number of variable s", ylab = "Mean Cross Validation error", type = "b")

```

### 5.3 Ridge Regression

```
> x <- model.matrix(Price~.,hdb)[,-1]
> y <- hdb$Price
> # grid use default
>
> RNGkind(sample.kind = "Rounding")
Warning: non-uniform 'Rounding' sampler used
> set.seed(6789)
>
> library(glmnet)
>
> train <- sample(1:nrow(hdb),nrow(hdb)/2) # half sample
> test <- -train
>
> x.train <- x[train,]
> y.train <- y[train]
> x.test <- x[test,]
> y.test <- y[test]
>
> ridge.mod <- glmnet(x.train,y.train, alpha = 0) # training model; default grid
>
> # plot
> plot(ridge.mod)
>
> cvrr.out <- cv.glmnet(x.train,y.train, alpha = 0)
> bestlam <- cvrr.out$lambda.min # min lambda
> bestlam # 5407.103
[1] 5407.103
>
> # plot
> plot(cvrr.out)
>
> # for test model to test for MSE
> ridge.pred <- predict(ridge.mod, s = bestlam, newx = x.test)
> mean((ridge.pred - y.test)^2) # 3975723793
[1] 3975723793
```

```
> # for general model
> out.rr <- glmnet(x,y, alpha = 0) # general model; default grid
> predict(out.rr, type = "coefficients", s = bestlam)
37 x 1 sparse Matrix of class "dgCMatrix"
           s1
(Intercept) -253592.624
TownBEDOK      22630.373
TownBISHAN     189995.702
TownBUKIT BATOK -49483.637
TownBUKIT MERAH 167877.180
TownBUKIT PANJANG -102804.410
TownBUKIT TIMAH 267288.677
TownCHOA CHU KANG -144136.321
TownCLEMENTI    118202.320
TownGEYLANG     142049.059
TownHOUgang     -51114.398
TownJURONG EAST -45011.945
TownJURONG WEST -124673.728
TownKALLANG/WHAMPOA 121497.793
TownMARINE PARADE 256030.552
TownPASIR RIS   -66909.483
TownPUNGGOL    -110079.637
TownQUEENSTOWN   204296.709
TownSEMBAWANG   -124281.280
TownSENGKANG     -117606.423
TownSERANGOON    41782.231
TownTAMPINES     -25563.537
TownTOA PAYOH     121005.538
TownWOODLANDS    -133734.873
TownYISHUN       -91396.551
Story04 TO 06     16613.252
Story07 TO 09     33293.019
Story10 TO 12     41976.477
Story13 TO 15     48941.650
Story16 TO 18     70599.915
Story19Above      131855.340
Area             4395.065
ModelImproved    -92938.882
ModelModel A      -82852.488
ModelPremium Apartment -83143.371
ModelStandard     -95588.600
LeaseRem          6407.155
```

## 5.4 Lasso

```
> RNGkind(sample.kind = "Rounding")
Warning: non-uniform 'Rounding' sampler used
> set.seed(6789)
>
> lasso.mod <- glmnet(x.train,y.train, alpha = 1) # training model; default grid
>
> plot(lasso.mod)
>
> cvlasso.out <- cv.glmnet(x.train,y.train, alpha = 1)
> bestlam1 <- cvlasso.out$lambda.min # min lambda
> bestlam1 # 38.14604
[1] 38.14604
>
> plot(cvlasso.out)
>
> # for test model to test for MSE
> lasso.pred <- predict(lasso.mod, s = bestlam1, newx = x.test)
> mean((lasso.pred - y.test)^2) # 3547183929
[1] 3547183929
>
> # for general model
> out.lasso <- glmnet(x,y, alpha = 1) # general model; default grid
> lasso.coef <- predict(out.lasso, type = "coefficients", s = bestlam1)
> lasso.coef[lasso.coef!=0]
<sparse>[ <logical> ] : .M.sub.i.logical() maybe inefficient
[1] -358844.573 -63882.473  98365.140 -143835.991   90412.100 -196575.688  188785.434
[8] -245514.772  30493.296  64461.528 -149222.404 -132947.905 -219709.815  34290.619
[15] 185984.729 -169715.961 -206316.228  128776.824 -220004.788 -212694.172 -47520.188
[22] -126329.359  25200.544 -233614.554 -189924.200  23094.059   41454.443  49700.865
[29]  58084.042  76690.724  127188.365   6063.013 -162231.080 -172370.968 -152528.994
[36] -162001.595   7252.751
```

## 5.5 Bagging Approach

```
> RNGkind(sample.kind = "Rounding")
Warning: non-uniform 'Rounding' sampler used
> set.seed(6789)
> library(randomForest)
>
> bag.hdb <- randomForest(Price~, data = hdb, subset = train, mtry = 5, importance = TRUE) # mtry is the number
of predictors
> bag.hdb

Call:
randomForest(formula = Price ~ ., data = hdb, mtry = 5, importance = TRUE,      subset = train)
Type of random forest: regression
Number of trees: 500
No. of variables tried at each split: 5

Mean of squared residuals: 2441168359
% Var explained: 86.48
>
> yhat.bag <- predict(bag.hdb, newdata = hdb[test,])
> mean((yhat.bag - hdb[test,]$Price)^2) # 2641712577; % Var explained: 86.48
[1] 2641712577
>
> plot(yhat.bag, hdb[test,]$Price, main = "Bagging Procedure for HDB test data", xlab = "predicted value with",
ylab = "observed value") + abline(0,1)
integer(0)
>
> importance(bag.hdb)
  %IncMSE IncNodePurity
Town    292.68174 2.628247e+13
Story   99.48829 2.582049e+12
Area    74.23896 2.094600e+12
Model   47.87751 8.158154e+11
LeaseRem 216.52058 1.191510e+13
>
> varImpPlot(bag.hdb, main = "Importance measure plot")
>
```

## 5.6 Random Forest

```
> RNGkind(sample.kind = "Rounding")
Warning: non-uniform 'Rounding' sampler used
> set.seed(6789)
> library(randomForest)
>
> rf.hdb <- randomForest(Price~, data = hdb, subset = train, mtry = 4, importance = TRUE) # mtry is the number
of predictors
> rf.hdb

Call:
randomForest(formula = Price ~ ., data = hdb, mtry = 4, importance = TRUE,      subset = train)
Type of random forest: regression
Number of trees: 500
No. of variables tried at each split: 4

Mean of squared residuals: 2442691923
% Var explained: 86.47
>
> yhat.rf <- predict(rf.hdb, newdata = hdb[test,])
> mean((yhat.rf - hdb[test,]$Price)^2) # 2628577302, % Var explained: 86.47
[1] 2628577302
>
> plot(yhat.rf, hdb[test,]$Price, main = "Random Forest Procedure for HDB test data with mtry = 4", xlab = "predicted
value with mtry = 4", ylab = "observed value") + abline(0,1)
integer(0)
>
> importance(rf.hdb)
  %IncMSE IncNodePurity
Town    260.91813 2.469564e+13
Story   89.75755 3.632937e+12
Area    64.27566 2.503133e+12
Model   35.02059 1.642799e+12
LeaseRem 179.20508 1.099981e+13
>
> varImpPlot(rf.hdb, main = "Importance measure plot")
```

## 5.7 Final Model (Random Forests mtry=4)

```
> finmodel<- randomForest(Price~.,data=hdb, mtry=4, importance=TRUE)
> finmodel

Call:
randomForest(formula = Price ~ ., data = hdb, mtry = 4, importance = TRUE)
      Type of random forest: regression
                  Number of trees: 500
No. of variables tried at each split: 4

      Mean of squared residuals: 2160505873
          % Var explained: 88.64
>
```

### Testing final Model on Test set

```
> #testing set
> testset <- read.csv("Rm5HDB2023testP.csv", stringsAsFactors = TRUE)
> testset[,1:5]
> finmodel

Call:
randomForest(formula = Price ~ ., data = hdb, mtry = 4, importance = TRUE)
      Type of random forest: regression
                  Number of trees: 500
No. of variables tried at each split: 4

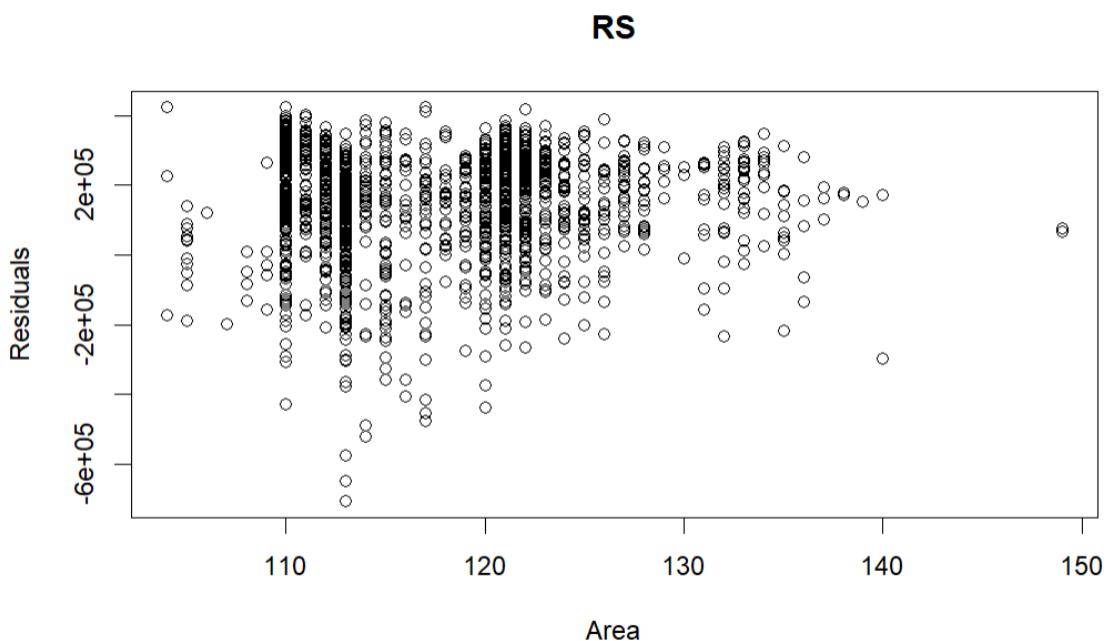
      Mean of squared residuals: 2153262009
          % Var explained: 88.68
> test.pred <- predict(finmodel, newdata=testset[,1:5])
> mean((test.pred-testset$Price)^2)
[1] 2339643383
.
```

## 5.8 Comparison between Predicted and Observed Values

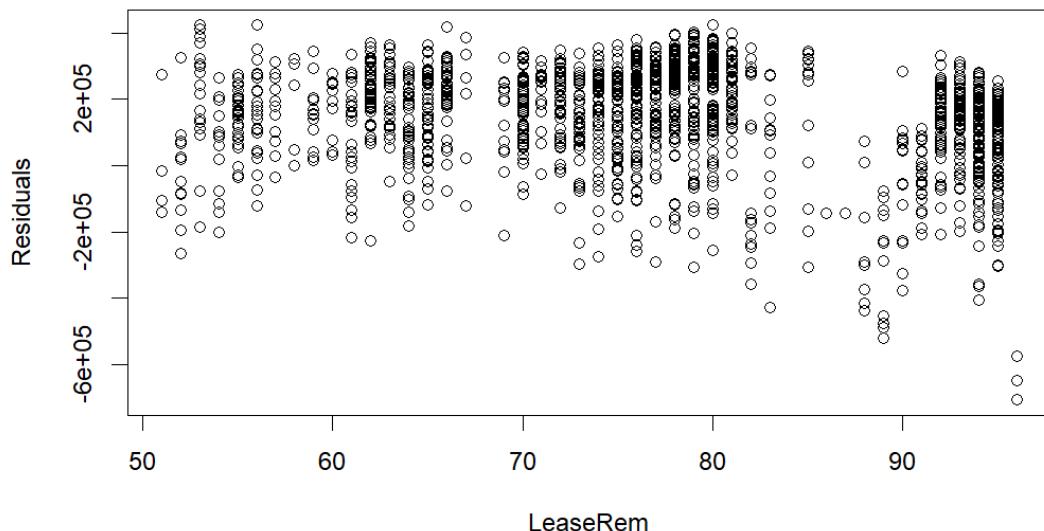
```
> y<-testhdb$Price  
>  
> pred<-predict(finalmodel,newdata=testhdb[1:5,1:5])  
> pred  
1 2 3 4 5  
825482.7 672742.1 810250.2 825482.7 873453.0  
>  
> compare<-data.frame(y[1:5],pred)  
> compare  
>  
>
```

## 5.9 Residual Assumptions

```
> plot(testhdb$Area,pred-y,main="RS",xlab="Area",ylab="Residuals")  
Warning: longer object length is not a multiple of shorter object length  
> plot(testhdb$LeaseRem,pred-y,main="RS",xlab="LeaseRem",ylab="Residuals")  
Warning: longer object length is not a multiple of shorter object length
```



## RS



```

> library(fitdistrplus)
> fnorm<-fitdist(pred-y, "norm")
Warning: longer object length is not a multiple of shorter object length
Warning: NaNs produced
Warning: NaNs produced
>
> summary(fnorm)
Fitting of the distribution ' norm ' by maximum likelihood
Parameters :
Loglikelihood: -21494.63   AIC: 42993.26   BIC: 43004.03
Correlation matrix:
[1] NA
> plot(fnorm)
> 

```

