DSA3361_Team Project

Group 18

Contents

Executive Summary

Discussion

Using data on HDB resale prices in Singapore, we developed a predictive regression model to estimate future resale prices, as well as determine significant factors that affect resale prices. Similar hedonic models have been conducted on the Singaporean housing market, where structural attributes including age and floor area were significant predictors of price (Cao et al., 2019). Our main finding is that by utilising multi-linear, ridge, and LASSO regressions, we derived statistically significant coefficients that show geographical locations (as indicated by categorical variables of towns) have the largest impact on resale prices, as compared to other variables. Our second finding is that there there is a logarithmic relationship between resale price and floor area sqm.

We extended on the existing discussion by introducing various methodologies beyond the scope of this course.As we used a logarithmic response variable, a multivariate adaptive regression splines (MARS) was employed to capture complex, non-linear relationships. MARS model gave us a similar results as LASSO regression.We also utilised a random forest to check the most important variables, which reaffirmed our findings that locations were one of the most important factors, as certain towns like Bukit Merah and Queenstown feature highly in the list. Random Forest gave us the highest R squared.

Limitations

Despite the high adjusted R-squared value and low MSE for training and test data, our models performed poorly on the unseen data, with an low R squared and high MSE. Including new variables such as proximity to MRT stations and other amenities could provide better accuracy in the models. Further analyses could include investigation into potential interaction terms, and testing on past data to check the accuracy of our model. It can also include time sensitivity analysis, for example, the month that the hdb lease commenced. Furthermore, we could also include further spatial data like what amenities are in the vicinity, since these will also significantly affect the hdb prices. Inflation and interest rates over the years are also important data to take into consideration, especially if the model is predicting over a range of years, and this could help to standardise the true value of resale prices, potentially improving the accuracy of the model.

First we start by loading the dataset and filtering for years 2022 onwards. By focusing on data from 2022 onwards, we capture the recovery period, allowing for an analysis that reflects the current, more stabilized market conditions rather than the volatile period of 2020–2021.

```
# Reading the dataset
dfall <- read.csv("ResaleflatpricesbasedonregistrationdatefromJan2017onwards.csv",
                  stringsAsFactors = TRUE)

# Filter for years 2022-2024
dfall$month <- as.Date(paste0(dfall$month, "-01"), format = "%Y-%m-%d")
df <- subset(dfall, month >= as.Date("2022-01-01"))
head(df,n = 1)
```

```
           month       town flat_type block      street_name storey_range
116677 2022-01-01 ANG MO KIO    2 ROOM   323 ANG MO KIO AVE 3     07 TO 09
       floor_area_sqm flat_model lease_commence_date    remaining_lease
116677             44   Improved               1977 54 years 05 months
       resale_price
116677       245000
```

Data Preparation

We check for missing values to assess the quality of the data. Then we add variables that might be included in regression. First we wanted to convert remaining lease into rounded_years for it to become numerical for easier analysis.

```
# Checking for missing values
sum(is.na(df))
```

```
[1] 0
```

```
df1 <- df %>%
  mutate(years = as.numeric(gsub(" years.*", "", remaining_lease)),
         resale_price = resale_price,
         months = ifelse(grepl("month", remaining_lease),
                         as.numeric(gsub(".*years | month.*| months", "", remaining_lease)), 0),
         rounded_years = ifelse(months >= 6, years + 1, years))
```

```
# Selecting relevant variables for analysis
df2 <- df1 %>% select(!c(month, block, street_name, lease_commence_date,
                         remaining_lease, years, months)) %>% mutate(
    flat_type = as.numeric(factor(flat_type)),
    storey_range = sapply(storey_range, function(x) {
      levels <- as.numeric(unlist(strsplit(gsub(" TO ", "-", x), "-")))
      # Calculate the average of the levels
      mean(levels, na.rm = TRUE)
    })
  )

# Grouping by NSEW
north_towns <- c( "YISHUN", "WOODLANDS", "SEMBAWANG")
northeast_towns <- c("ANG MO KIO","HOUGANG","PUNGGOL","SENGKANG","SERANGOON")
central_towns <- c("BUKIT MERAH", "QUEENSTOWN", "MARINE PARADE","BISHAN","CENTRAL
                   AREA","GEYLANG","KALLANG/WHAMPOA","MARINE PARADE","QUEENSTOWN","TOH PAYOH")
east_towns <- c("BEDOK", "TAMPINES", "PASIR RIS")
west_towns <- c("JURONG EAST", "JURONG WEST", "CLEMENTI", "BUKIT BATOK","BUKIT PANJANG",
                "CHOA CHU KANG","CLEMENTI","TENGAH")

df2location <- df2 %>%
  mutate(location_var = case_when(
    town %in% north_towns ~ "North",
    town %in% northeast_towns ~ "Northeast",
    town %in% central_towns ~ "Central",
    town %in% east_towns ~ "East",
    town %in% west_towns ~ "West",
    TRUE ~ "Other"  ))
```

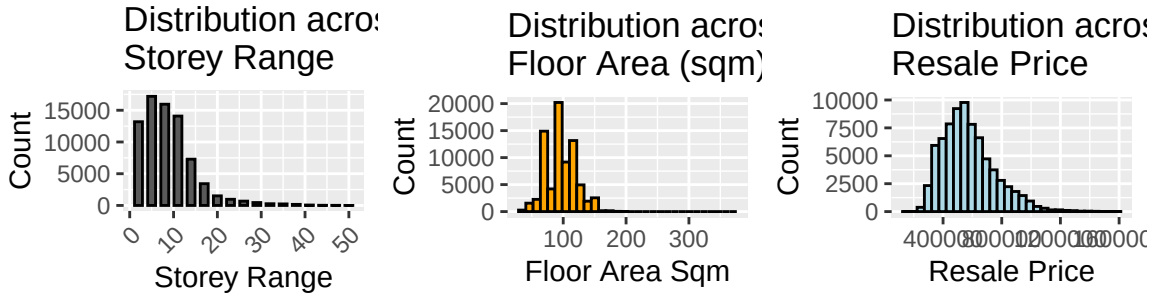
In selecting the predictors for our analysis, we excluded variables such as month, block, and street_name, as they are unlikely to have a meaningful relationship with resale prices. These factors do not typically reflect buyer preferences, suggesting that they won't contribute to a model predicting price. Additionally, we removed remaining_lease, years, and months because we already created the variable rounded_years, that effectively captures the relevant information about lease duration. By focusing on more the chosen

predictors, we aim to enhance the model's predictive accuracy and interpretability. Furthermore we replaced storey_range with its average level as it provides a more realistic numeric representation.

In order to facilitate a more comprehensive analysis of resale prices, we decided to group the towns into broader locations (North, Northeast, Central, East and West). This dataframe df2location, will be used when building one of the MLR models, in order to examine the impact of location on resale prices at different level, not just by individual town but also across the different areas of Singapore

EDA

```
library(stringr)
# univariate plots
# storey_range
plot.storeyrange <- ggplot(df2, aes(x = storey_range)) +
  geom_bar(color = "black",aes(fill = storey_range)) +
  labs(title = str_wrap('Distribution across Storey Range', width = 20),
       x = 'Storey Range', y = 'Count') +
  theme(legend.position = "none",
        axis.text.x = element_text(angle = 45, vjust = 1, hjust=1),
        aspect.ratio = 0.5)

# floor_area_sqm
plot.floorareasqm <- ggplot(df2, aes(x = floor_area_sqm)) +
  labs(title = str_wrap('Distribution across Floor Area (sqm)',width = 20),
       x = 'Floor Area Sqm', y = 'Count') +
  geom_histogram(color = "black", fill = "orange") + theme(aspect.ratio = 0.5)

# resale_price
plot.resaleprice <- ggplot(df2, aes(x = resale_price)) +
  geom_histogram(color = "black", fill = "lightblue") +
  labs(title = str_wrap('Distribution across Resale Price',width = 20),
       x = 'Resale Price', y = 'Count') + theme(aspect.ratio = 0.5)

grid.arrange(plot.storeyrange, plot.floorareasqm,plot.resaleprice,
             ncol = 3)
```

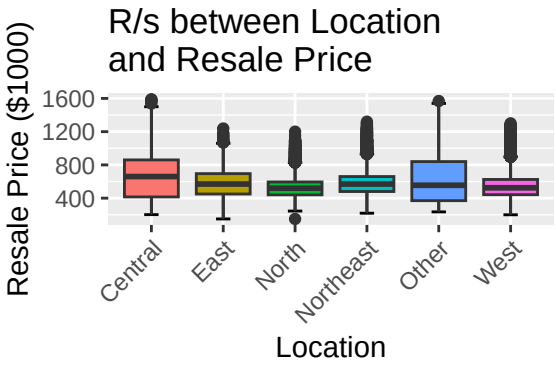


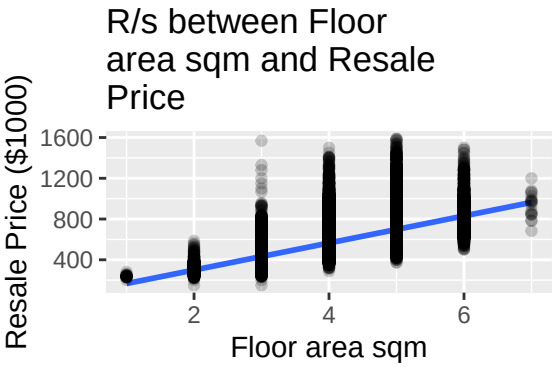

```
# skewness
data.frame(cat = c('skew_storey', 'skew_floorarea', 'skew_price'),
           skew = c(skewness(df2$storey_range),
                    skewness(df2$floor_area_sqm),
                    skewness(df2$resale_price)))
```

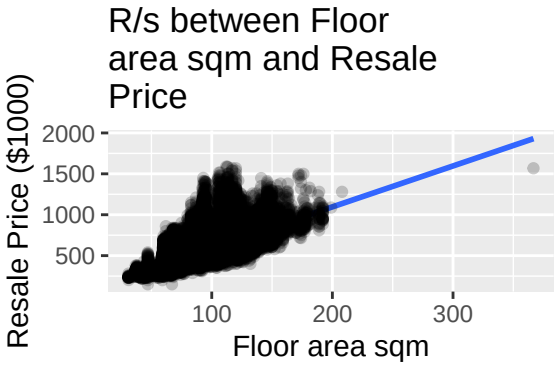
```
             cat      skew
1    skew_storey 1.6127840
2 skew_floorarea 0.2750865
3     skew_price 0.9096645
```

Based on the summary, we expect the distribution of storey range to be highly skewed, similarly for floor area and resale price. The skew results show an extremely skewed distribution of storey range, but a low skew for floor area and a moderate skew for resale price. We tried binning for storey_range by putting floors 1-5 as low rise, 6-20 as mid rise, 21 and above as high rise but it did not improve our model.

```
# bivariate plots
# location vs resale_price
plot.location.price <- ggplot(df2location,
                              aes(x = location_var, y = resale_price/1000)) +
  stat_boxplot(geom = "errorbar", width = 0.2) +
  geom_boxplot(aes(fill = location_var)) +
  labs(title = str_wrap('R/s between Location and Resale Price',width = 20),
       x = 'Location', y = 'Resale Price ($1000)') +
  theme(legend.position = "none",
        axis.text.x = element_text(angle = 45, vjust = 1, hjust=1))

# flat_type vs resale_price
plot.flattype.price <- ggplot(df2, aes(x = flat_type, y = resale_price/1000)) +
  geom_smooth(method = "lm") +
  labs(title = str_wrap('R/s between Floor area sqm and Resale Price',width=20),
       x = 'Floor area sqm', y = 'Resale Price ($1000)') +
  geom_point(alpha = 0.2)

# floor_area_sqm vs resale_price
plot.floorareasqm.price <- ggplot(df2, aes(x = floor_area_sqm, y = resale_price/1000)) +
  geom_smooth(method = "lm") +
  labs(title = str_wrap('R/s between Floor area sqm and Resale Price',width =20),
       x = 'Floor area sqm', y = 'Resale Price ($1000)') +
  geom_point(alpha = 0.2)

grid.arrange(plot.location.price, plot.flattype.price, plot.floorareasqm.price, ncol = 2)
```







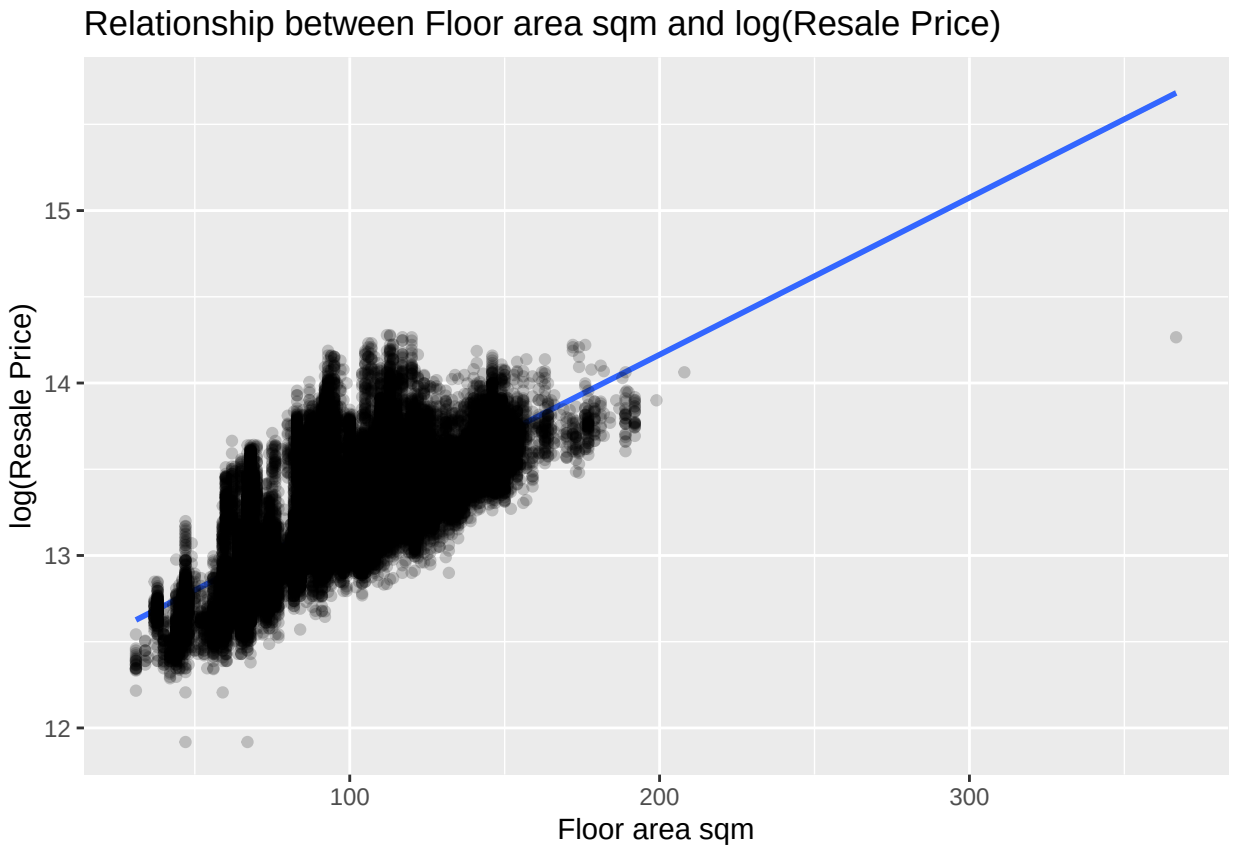
Based on the locations versus resale_price plot, central has the highest median resale price as it is the most accessible location, and north has the lowest median resale price.

Observing the plot with Floor area sqm and price, the plot does not seem to follow a linear relationship. This suggests that there might be a non-linear relationship between price and the other variables. We decided to try log transformation of Resale price to see if that could give a better fit.

```
# Log(resale_price)
df2 <- df2 %>% mutate(resale_price = log(resale_price))
df2location <- df2location %>% mutate(resale_price = log(resale_price))
```

Plotting floor area sqm and log(resale price) gave a closer linear fit.

```
# Plot floor area sqm with log(price)
(plot.floorareasqm.logprice <- ggplot(df2,
                                      aes(x = floor_area_sqm, y = resale_price)) +
  geom_smooth(method = "lm") +
  labs(title = 'Relationship between Floor area sqm and log(Resale Price)',
       x = 'Floor area sqm', y = 'log(Resale Price)')
 + geom_point(alpha = 0.2))
```




```r
dfnotown <- df2 %>% select(-town,-flat_model)
correlation_matrix <- cor(dfnotown, use = "complete.obs")

# Print the correlation matrix
print(correlation_matrix)
```

```
                flat_type storey_range floor_area_sqm resale_price
flat_type      1.00000000   0.01531077     0.95538975    0.7278002
storey_range   0.01531077   1.00000000    -0.02293772    0.3356122
floor_area_sqm 0.95538975  -0.02293772     1.00000000    0.7180706
resale_price   0.72780016   0.33561221     0.71807057    1.0000000
rounded_years  0.12632038   0.27354509     0.06435117    0.4077344
               rounded_years
flat_type         0.12632038
storey_range      0.27354509
floor_area_sqm    0.06435117
resale_price      0.40773439
rounded_years     1.00000000
```

There is a very strong positive correlation between floor area sqm and flat type, which might indicate redundancy . There is also strong positive correlation between resale_price and floor_area_sqm, and resale price and flat_type. There is moderate correlation between resale_price and storey_range, and resale_price and rounded_years.

Model Fitting

Standardisation

```
set.seed(123)
df3location <- df2location %>% select(!c(town, flat_type, flat_model,location_var))
scaler <- apply(df3location, 2, sd)
df4location <- as.data.frame(apply(df3location, 2, function(x) x/sd(x)))
df2location <- df2location %>% mutate(floor_area_sqm = df4location$floor_area_sqm,
                      resale_price = df4location$resale_price,
                      rounded_years = df4location$rounded_years,
                      storey_range = df4location$storey_range)
```

Standardization ensures all predictors contribute equally to the model. Standardizes each column in df3 by dividing each value by its respective column's standard deviation. This step creates df4, which is a new data frame where each column has been standardized to have a standard deviation of 1.This process helps improve the distribution of contribution that each predictor has in the models later

Train-Test Split We first split our dataset into train and test via a 80-20 ratio. This will allow us to test our model later on.

```
# Splitting data into train-test
rate <- 0.2
train.size <- round(nrow(df2location) * rate)
sampleloc <- sample(nrow(df2location), train.size)
trainingloc <- df2location[sampleloc,]
testloc <- df2location[-sampleloc,]
```

Multiple Linear Regression (MLR) model

In our data preparation, we created a variable: location. This represents all the towns compartmentalized into the different zones of Singapore (north, northeast, south, west, east). We wanted to determine if the particular towns in Singapore had a significant effect on resale_price, or zones in Singapore sufficiently capture the effect of location on the resale_price. Since town and location were very closely related, we wanted to see which one to use as predictor in our model. We began building our model by fitting a multiple linear regression model. Thus, we created 2 baseline models: 1. baselineLOC, which excludes towns, and 2. baselinetown, which excludes location.

Rule of thumb: GVIF^(1/(2*Df)) $<$ 2 typically considered acceptable

From these models, we checked for multicollinearity using vif function. The GVIF^(1/(2*Df)) *values of flat_type and floor_area_sqm are both above 4, indicating multicollinearity. Thus, we created a revised model for each, excluding flat_type. This lowered the GVIF^(1/(2*Df)) value of floor_area_sqm, indicating the issue of multicollinearity was resolved.

```
# Baseline model with location
baselineLOC <- lm(resale_price ~. -town , data = trainingloc)
vif(baselineLOC)
```

```
                  GVIF Df GVIF^(1/(2*Df))
flat_type      19.449151  1        4.410119
storey_range    1.199659  1        1.095290
```

```
floor_area_sqm 17.501747  1        4.183509
flat_model      7.194234 20        1.050569
rounded_years   2.592994  1        1.610278
location_var    1.698565  5        1.054407
```

```
revise_LOC1 <- lm(resale_price ~ . - town - flat_type, data = trainingloc)
vif(revise_LOC1)
```

```
                  GVIF Df GVIF^(1/(2*Df))
storey_range   1.194456  1        1.092912
floor_area_sqm 1.820166  1        1.349135
flat_model     4.210723 20        1.036595
rounded_years  2.130591  1        1.459654
location_var   1.684462  5        1.053528
```

```
# Baseline model with individual towns
baselinetown <- lm(resale_price ~. -location_var , data = trainingloc)
vif(baselinetown)
```

```
                   GVIF Df GVIF^(1/(2*Df))
town            5.185708 25        1.033466
flat_type      20.316635  1        4.507398
storey_range    1.248503  1        1.117364
floor_area_sqm 18.770243  1        4.332464
flat_model     14.603310 20        1.069329
rounded_years   2.796124  1        1.672161
```

```
revise_baselinetown1 <- lm(resale_price ~. -location_var -flat_type, data = trainingloc)
vif(revise_baselinetown1)
```

```
                  GVIF Df GVIF^(1/(2*Df))
town           4.923069 25        1.032392
storey_range   1.244411  1        1.115531
floor_area_sqm 1.947069  1        1.395374
flat_model     8.583187 20        1.055216
rounded_years  2.352831  1        1.533894
```

Next, we compared the AIC values of the models, and the model with town has a lower AIC indicating a better fit. In addition, the R squared of revise_baselinetown1 is higher than revise_LOC1.

```
AIC(baselineLOC,baselinetown)
```

```
             df      AIC
baselineLOC  31 15055.04
baselinetown 51 11241.38
```

```
AIC(revise_LOC1,revise_baselinetown1)
```

```
                     df      AIC
revise_LOC1          30 15342.97
revise_baselinetown1 50 11647.87
```

Ridge and LASSO Models

We decided to utilize Ridge and LASSO models as alternative ways to resolve the multicollinearity issue observed. We first split train-test set for df2, which does not contain the location_var. We also converted them into matrices for ridge and LASSO regression.

```
set.seed(123)
rate <- 0.2
train.size <- round(nrow(df2) * rate)
sample <- sample(nrow(df2), train.size)
training <- df2[sample,]
test <- df2[-sample,]

train.x <- model.matrix(resale_price ~ ., data = training)[, -1]
train.y <- training$resale_price
test.x  <- model.matrix(resale_price ~ ., data = test)[, -1]
test.y  <- test$resale_price
```

```
# Ridge model
set.seed(123)
cvridge <- cv.glmnet(train.x, train.y, alpha = 0, type.measure = "mse")
glmridge <- glmnet(train.x, train.y, alpha = 0, lambda = cvridge$lambda.min)
print(cvridge$lambda.min)
```

```
[1] 0.02183911
```

Using the cv.glmnet function, we obtained lambda.min = 0.0218 as the optimal lambda value for our ridge model. We proceeded to build our ridge model with this lambda value. This will be evaluated with a common set of evaluation metrics later on.

```
# LASSO model
set.seed(123)
cvlasso <- cv.glmnet(train.x, train.y, alpha = 1, type.measure = "mse")
glmlasso <- glmnet(train.x, train.y, alpha = 1, lambda = cvlasso$lambda.min)
cvlasso
```

```
Call:  cv.glmnet(x = train.x, y = train.y, type.measure = "mse", alpha = 1)

Measure: Mean-Squared Error

       Lambda Index Measure        SE Nonzero
min 0.0001404    80 0.01137 0.0002027      49
1se 0.0014369    55 0.01157 0.0001791      43
```

Next, we obtained the optimal lambda value for the LASSO model, where lambda.min = 0.0001284. From the summary of cvlasso, we notice that 48 predictors will remain in the model when this lambda value is used. Plotting the coefficients vs log lambda graph confirms that variable selection was carried out.

Evaluation metrics for MLR, Ridge and LASSO models

After building our models, we decided to use the evaluation metrics, including MSE, MAE, RMSE, MAPE and R^2, to compare the models. This will allow us to determine which model best predicts resale_price on the training and test data.

```
eval_results <- function(fit, true) {
  actual <- data.matrix(true)
  SSE <- sum((actual - fit)^2)
  SST <- sum((actual - mean(actual))^2)
  R_square <- 1 - SSE / SST
  data.frame(
    MSE = MSE(fit, true),
    MAE = MAE(fit, true),
    RMSE = RMSE(fit, true),
    MAPE = MAPE(fit, true),
    R2 = R_square
  )
}
# Evaluate MLR locations
fit <- revise_LOC1$fitted.values
true <- trainingloc$resale_price
summary_mlrLOC_train <- eval_results(fit, true)
fit <- predict(revise_LOC1, newdata = testloc)
true <- testloc$resale_price
summary_mlrLOC_test <- eval_results(fit, true)

# Evaluate MLR towns (trainingloc and testloc have both town and location_var)
fit <- revise_baselinetown1$fitted.values
true <- trainingloc$resale_price
summary_mlrtown_train <- eval_results(fit, true)
fit <- predict(revise_baselinetown1, newdata = testloc)
true <- testloc$resale_price
summary_mlrtown_test <- eval_results(fit, true)

# Evaluate Ridge Model
fit <- predict(glmridge, train.x)
true <- train.y
summary_ridge_train <- eval_results(fit, true)
fit <- predict(glmridge, test.x)
true <- test.y
summary_ridge_test <- eval_results(fit, true)

# Evaluate LASSO  models
fit <- predict(glmlasso, train.x)
true <- train.y
summary_LASSO_train <- eval_results(fit, true)
fit <- predict(glmlasso, test.x)
true <- test.y
summary_LASSO_test <- eval_results(fit, true)
```

Comparing evaluation metrics for MLR, Ridge and LASSO models

```
# Summary train
summary_train <- rbind(summary_mlrtown_train,
                       summary_mlrLOC_train,
                       summary_ridge_train,
```

```
                        summary_LASSO_train)
rownames(summary_train) <- c("MLRtown_train",
                             "MLRLOC_train",
                             "Ridge_train",
                             "LASSO_train")
colnames(summary_train)[5] <- "R^2"
knitr::kable(summary_train, digits = 3)
```

	MSE	MAE	RMSE	MAPE	R^2
MLRtown_train	0.126	0.276	0.355	0.006	0.874
MLRLOC_train	0.161	0.309	0.401	0.007	0.838
Ridge_train	0.012	0.085	0.109	0.006	0.869
LASSO_train	0.011	0.083	0.106	0.006	0.877

Comparing the evaluation metrics of each model, MLR using location instead of town incurs a higher error than the MLR using town. This is further justified by a smaller R^2 value of MLR using location compared to town. This indicates that using town instead of location in the model lowers the error during prediction. However, there are advantages of a more interpretable model (MLR using location). As such, this trade-off is considered when choosing the final model.

```
#summary test
summary_test <- rbind(summary_mlrtown_test,
                      summary_mlrLOC_test,
                       summary_ridge_test,
                       summary_LASSO_test)
rownames(summary_test) <- c("MLRtown_test",
                            "MLRLOC_test",
                             "Ridge_test",
                             "LASSO_test")
colnames(summary_test)[5] <- "R^2"
knitr::kable(summary_test, digits = 3)
```

	MSE	MAE	RMSE	MAPE	R^2
MLRtown_test	0.121	0.273	0.348	0.006	0.879
MLRLOC_test	0.155	0.306	0.394	0.007	0.845
Ridge_test	0.012	0.084	0.108	0.006	0.872
LASSO_test	0.011	0.082	0.105	0.006	0.881

Evaluating the metrics on our test dataset, all values are comparable to those of the train dataset. This tells us that our models are not overfitted or underfitted.The LASSO regression model exhibited the best overall performance across both training and testing datasets, with the lowest MSE, MAE, and RMSE values, as well as a high R^2 value of 0.881 on the test set. This suggests that LASSO is particularly effective in handling data sparsity and variable selection for this dataset. As such, we decided to retain the MLR town and LASSO model for analysis of the unseen test data later on.

Extended Learning

We decided to use Random Forest model and MARS model for our extended learning portion.

MARS model

Multivariate Adaptive Regression Splines(MARS) resulted in a R squared of 0.757. During the exploratory analysis, we applied a log transformation to resale_price to better understand the relationship between the response variable and the predictors. The log transformation resulted in a better fit, suggesting the presence of non-linear relationships. Thus, we decided to use a MARS model. This model is capable of capturing complex, non-linear relationships that a standard linear model might miss. It uses a set of piece wise linear functions that characterize the data and uses them in aggregate to make a prediction. The function is of 2 types a right function where f(x) = x if x > value, else 0 or a left function where f(x) = x if x < value, else 0. These are called hinge functions and they are used to to allow the model to change slopes at certain values of the predictor variables. The algorithm generates many of these functions and this allows the model to capture more complex variables.

```
# Fit the MARS model
mars_model <- earth(
  resale_price ~ .,
data = df2)

mars_predict_train <- predict(mars_model,newdata = training)
true <- train.y
summary_mars_train <- eval_results(mars_predict_train, true)
mars_predict_test <- predict(mars_model,newdata = test)
true <- test.y
summary_mars_test <- eval_results(mars_predict_test, true)


summary_mars <- rbind(summary_mars_train,
                      summary_mars_test)
rownames(summary_mars) <- c("mars_train",
                            "mars_test"
                             )
colnames(summary_mars)[5] <- "R^2"
knitr::kable(summary_mars, digits = 3)
```

	MSE	MAE	RMSE	MAPE	R^2
mars_train	0.011	0.082	0.106	0.006	0.876
mars_test	0.011	0.081	0.104	0.006	0.881

The MARS model resulted in the error and R^2 values as the LASSO model earlier. This supports our findings from the LASSO model, indicative that appropriate variable selection was carried out and the model is a good fit.

Random Forest

We also tried using random forest for get more insights on variable selection. The model included hinge functions for variables like floor area and years, showing nuanced impacts on pricing beyond linear trends.

```
set.seed(4991)
rf_model <- randomForest(x = train.x, y = train.y, ntree = 500,
                         mtry = floor(sqrt(ncol(train.x))))
```

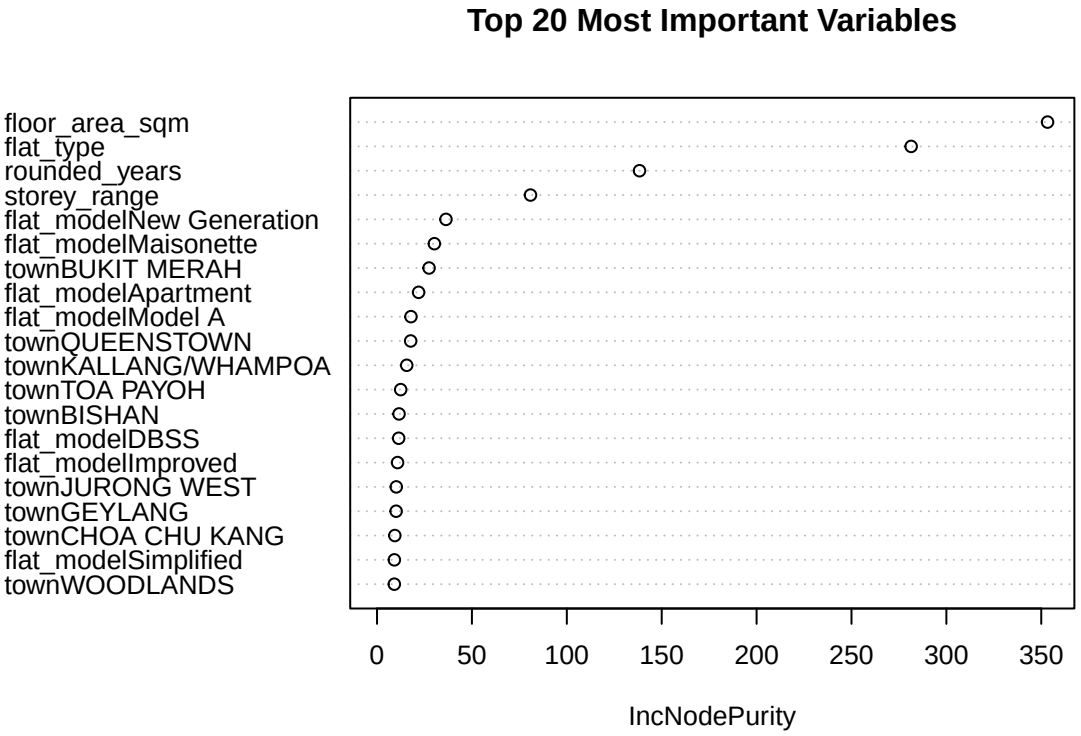
```r
# Predictions on train and test sets
train_preds <- predict(rf_model, train.x)
test_preds <- predict(rf_model, test.x)
trainsummary_random_forest <- eval_results(train_preds,train.y)
testsummary_random_forest <- eval_results(test_preds,test.y)
summary_random_forest <- rbind(trainsummary_random_forest, testsummary_random_forest)
rownames(summary_random_forest) <- c("Random Forest_Train", "Random Forest_Test")
colnames(summary_random_forest)[5] <- "R^2"
knitr::kable(summary_random_forest, digits = 3)
```

	MSE	MAE	RMSE	MAPE	R^2
Random Forest_Train	0.008	0.068	0.087	0.005	0.917
Random Forest_Test	0.009	0.072	0.095	0.005	0.902

Evaluating the metrics for random forest model, we obtained the lowest error and highest R^2 compared to all our model. This suggests random forest gives us the highest accuracy in prediction resale price.

We then plotted a graph of the top 20 most important variables according to the random forest model. Apart from the expected factors such as floor_area_sqm, flat_type and number of years remaining in lease, it has identified more nuanced aspects. Location north seems be an importance factor that determines price and by looking at the resale price across locations, North has the lowest median resale price so it might signify that there is lesser demand for houses in the north compared to other locations. This could prompt further analysis into the area's amenities, accessibility etc to address any shortcoming in order to make it more attractive in the resale market. Towns such as Bukit Merah and Queenstown also feature highly in the list, and both were in the top 5 highest median for resale price so the random forest reaffirms this finding. This suggests that properties in these areas are in high demand or are considered more desirable, possibly due to location, amenities, or other factors.Furthermore policy makers could explore how more resource allocation could support affordability in these areas.

```r
# Importance scores for each feature
varImpPlot(rf_model,
           n.var = 20,
           main = "Top 20 Most Important Variables",
           cex = 0.8)
```



Here we get the names of the top 20 most important variables to train new models to see if they would provide a better fit as suggested by the random forest

```
importance_df <- as.data.frame(importance(rf_model))
importance_df$variable <- rownames(importance(rf_model))
importance_df <- importance_df %>% arrange(desc(IncNodePurity))
top_20_vars <- rownames(importance_df)[1:20]
top_20_vars
```

```
 [1] "floor_area_sqm"           "flat_type"
 [3] "rounded_years"            "storey_range"
 [5] "flat_modelNew Generation" "flat_modelMaisonette"
 [7] "townBUKIT MERAH"          "flat_modelApartment"
 [9] "flat_modelModel A"        "townQUEENSTOWN"
[11] "townKALLANG/WHAMPOA"      "townTOA PAYOH"
[13] "townBISHAN"               "flat_modelDBSS"
[15] "flat_modelImproved"       "townJURONG WEST"
[17] "townGEYLANG"              "townCHOA CHU KANG"
[19] "flat_modelSimplified"     "townWOODLANDS"
```

Using these 20 variables, we decided to retrain our ridge and LASSO models, attempting to get a better fit. First, we made a new test.x and train.x matrix.

```
newtrain.x <- train.x[, top_20_vars, drop = FALSE]
newtrain.y <- training$resale_price
newtest.x  <- test.x[, top_20_vars, drop = FALSE]
newtest.y  <- test$resale_price
```

Then, we built the new Ridge and LASSO Models using the top 20 predictors from random forest.

```r
# Ridge model
set.seed(123)
newcvridge <- cv.glmnet(newtrain.x, newtrain.y, alpha = 0, type.measure = "mse")
newglmridge <- glmnet(newtrain.x, newtrain.y, alpha = 0, lambda = newcvridge$lambda.min)
```

```r
# LASSO model
set.seed(123)
newcvlasso <- cv.glmnet(newtrain.x, newtrain.y, alpha = 1, type.measure = "mse")
newglmlasso <- glmnet(newtrain.x, newtrain.y, alpha = 1, lambda = newcvlasso$lambda.min)
```

We then evaluated the models using the same evaluation metrics and made comparisons between the test and train dataset.

```r
# Evaluate Ridge Model
fit <- predict(newglmridge, newtrain.x)
true <- newtrain.y
new_summary_ridge_train <- eval_results(fit, true)
fit <- predict(newglmridge, newtest.x)
true <- newtest.y
newsummary_ridge_test <- eval_results(fit, true)

# Evaluate LASSO  models
fit <- predict(newglmlasso, newtrain.x)
true <- newtrain.y
newsummary_LASSO_train <- eval_results(fit, true)
fit <- predict(newglmlasso, newtest.x)
true <- newtest.y
newsummary_LASSO_test <- eval_results(fit, true)
```

```r
# Summary train
newsummary_train <- rbind(
                      new_summary_ridge_train,
                      newsummary_LASSO_train)
rownames(newsummary_train) <- c(
                             "Ridge_train",
                             "LASSO_train")
colnames(newsummary_train)[5] <- "R^2"
knitr::kable(newsummary_train, digits = 3)
```

	MSE	MAE	RMSE	MAPE	R^2
Ridge_train	0.017	0.099	0.129	0.007	0.816
LASSO_train	0.016	0.097	0.128	0.007	0.820

```r
#summary test
newsummary_test <- rbind(
                      newsummary_ridge_test,
                      newsummary_LASSO_test)
rownames(newsummary_test) <- c(
                             "Ridge_test",
                             "LASSO_test")
colnames(newsummary_test)[5] <- "R^2"
knitr::kable(newsummary_test, digits = 3)
```

	MSE	MAE	RMSE	MAPE	R^2
Ridge_test	0.017	0.098	0.129	0.007	0.819
LASSO_test	0.016	0.097	0.127	0.007	0.825

Comparing the evaluation metrics of the new models (with the top 20 predictors), the MSE is greater and R^2 value is lower than our previous ridge and LASSO models. This may be because Random Forest is a non-linear ensemble model, while LASSO regression is a linear model. The features deemed important by Random Forest might capture non-linear interactions that are less relevant or not as effective when applied to a linear model like LASSO. Random Forest ranks features by importance, but importance does not always correlate with predictive power. Some of the less important features in the Random Forest model could still have value when combined with other features or when regularized by Lasso or Ridge.

Testing Unseen Data

Predicting with MLR town model

```
set.seed(123)
unseen_test <- read.csv("project_testdata.csv")

# Adding variables that might be included in regression
dfb <- unseen_test %>%
  mutate(years = as.numeric(gsub(" years.*", "", remaining_lease)),
         months = ifelse(grepl("month", remaining_lease),
                         as.numeric(gsub(".*years | month.*| months", "", remaining_lease)), 0),
         rounded_years = ifelse(months >= 6, years + 1, years),
         resale_price = log(resale_price))

# Selecting relevant variables for analysis
dfc <- dfb %>% select(!c(month, block, street_name, lease_commence_date, remaining_lease, years)) %>% mu
    flat_type = as.numeric(factor(flat_type)),
    storey_range = sapply(storey_range, function(x) {
      levels <- as.numeric(unlist(strsplit(gsub(" TO ", "-", x), "-")))
      # Calculate the average of the levels
      mean(levels, na.rm = TRUE)
    })
  )

north_towns <- c( "YISHUN", "WOODLANDS", "SEMBAWANG")
northeast_towns <- c("ANG MO KIO","HOUGANG","PUNGGOL","SENGKANG","SERANGOON")
central_towns <- c("BUKIT MERAH", "QUEENSTOWN", "MARINE PARADE","BISHAN","CENTRAL AREA","GEYLANG","KALL
east_towns <- c("BEDOK", "TAMPINES", "PASIR RIS")
west_towns <- c("JURONG EAST", "JURONG WEST", "CLEMENTI", "BUKIT BATOK","BUKIT PANJANG","CHOA CHU KANG"

dfc <- dfc %>%
  mutate(location_var = case_when(
    town %in% north_towns ~ "North",
    town %in% northeast_towns ~ "Northeast",
    town %in% central_towns ~ "Central",
    town %in% east_towns ~ "East",
    town %in% west_towns ~ "West",
```

```
    TRUE ~ "Other"  ))

dfd <- dfc %>% select(-flat_type) %>%
  mutate(
    storey_range = storey_range / scaler[1],
    floor_area_sqm = floor_area_sqm / scaler[2],
    rounded_years = rounded_years / scaler[4]
  )
dfd <- dfd %>% mutate(flat_type = dfc$flat_type)

# predicting test data
predictions <- predict(revise_baselinetown1, newdata = dfd)
predictions_mlr_town <- exp(predictions*scaler[3])

# Evaluate model performance
true <- unseen_test$resale_price
MLRtownsummary_unseen <- eval_results(predictions_mlr_town, true)
```

Predicting with MARS model

```
unseen_test <- read.csv("project_testdata.csv")


# Adding variables that might be included in regression
dfb <- unseen_test %>%
  mutate(years = as.numeric(gsub(" years.*", "", remaining_lease)),

         months = ifelse(grepl("month", remaining_lease),
                         as.numeric(gsub(".*years | month.*| months", "", remaining_lease)), 0),
         rounded_years = ifelse(months >= 6, years + 1, years))

# Selecting relevant variables for analysis
dfc <- dfb %>% select(!c(months, block, street_name, lease_commence_date, remaining_lease, years)) %>% m
    flat_type = as.numeric(factor(flat_type)),
    storey_range = sapply(storey_range, function(x) {
      levels <- as.numeric(unlist(strsplit(gsub(" TO ", "-", x), "-")))
      # Calculate the average of the levels
      mean(levels, na.rm = TRUE)
    })
  )
dfd <- dfc %>% select(-month)
```

```
predictions <- predict(mars_model, newdata = dfd)
predictions_mars <- exp(predictions)
true <- unseen_test$resale_price
marssummary_unseen <- eval_results(predictions_mars, true)
```

Predicting with LASSO model

```
# Assuming 'test.x' already has the full set of columns including dummies
# One-hot encode 'dfc' to match 'test.x' structure
dummy_model <- dummyVars(~., data = dfd)
dfd_transformed <- predict(dummy_model, newdata = dfd)

# Convert to a data frame
dfd_transformed <- as.data.frame(dfd_transformed)

# Ensure all columns in 'test.x' are in 'dfc_transformed'
missing_cols <- setdiff(colnames(test.x), colnames(dfd_transformed))

# Add missing columns with zeros
for (col in missing_cols) {
  dfd_transformed[[col]] <- 0
}
# Reorder columns in 'dfc_transformed' to match 'test.x'
dfd_transformed <- dfd_transformed[, colnames(test.x)]

# Convert to matrix if needed for prediction
dfd_matrix <- as.matrix(dfd_transformed)
```

```
predictionslasso <- predict(glmlasso, dfd_matrix)
predictions_lasso <- exp(predictionslasso)
true <- unseen_test$resale_price
LASSO_unseen <- eval_results(predictions_lasso, true)
```

Comparing Evaluation Metrics on Unseen Data

```
# Comparing evaluation metrics for each model
summary_unseen <- rbind(MLRtownsummary_unseen, marssummary_unseen, LASSO_unseen)
rownames(summary_unseen) <- c("MLR_Town", "MARS", "LASSO")
colnames(summary_unseen)[5] <- "R^2"
knitr::kable(summary_unseen, digits = 3)
```

	MSE	MAE	RMSE	MAPE	R^2
MLR_Town	6105453822	60435.40	78137.4	0.094	0.826
MARS	12188164767	93112.07	110400.0	0.146	0.652
LASSO	14102514534	103176.48	118754.0	0.162	0.597

Conclusion

Based on our models, we propose the MLR model with towns as our final model for predicting HDB resale prices as it performed the best with the lowest MSE and highest R^2 for the unseen test data. Furthermore, it does not include external data so new data can be fitted in easily with minimal data manipulation.

```
coef(revise_baselinetown1)
```

```
                    (Intercept)                        townBEDOK
                     38.04387007                      -0.10182084
                      townBISHAN                   townBUKIT BATOK
                      0.41841438                      -0.47638463
                   townBUKIT MERAH                townBUKIT PANJANG
                      0.51076487                      -0.69024109
                   townBUKIT TIMAH                townCENTRAL AREA
                      0.95451466                       0.71319289
                 townCHOA CHU KANG                    townCLEMENTI
                     -0.92563582                       0.09373967
                     townGEYLANG                      townHOUGANG
                      0.19735616                      -0.41215256
                   townJURONG EAST                  townJURONG WEST
                     -0.54793497                      -0.76363571
              townKALLANG/WHAMPOA                townMARINE PARADE
                      0.30904730                       0.82930918
                    townPASIR RIS                      townPUNGGOL
                     -0.51720917                      -0.63024317
                   townQUEENSTOWN                    townSEMBAWANG
                      0.54107097                      -0.80073509
                    townSENGKANG                     townSERANGOON
                     -0.67146494                      -0.01782986
                    townTAMPINES                     townTOA PAYOH
                     -0.21602793                       0.25724114
                   townWOODLANDS                      townYISHUN
                     -0.82845879                      -0.62152475
                    storey_range                    floor_area_sqm
                      0.15573340                       0.79948020
                 flat_model3Gen          flat_modelAdjoined flat
                      0.01041418                       0.30710229
            flat_modelApartment                   flat_modelDBSS
                      0.16986233                       0.83166975
             flat_modelImproved    flat_modelImproved-Maisonette
                      0.18300324                       1.02319409
          flat_modelMaisonette                flat_modelModel A
                      0.36632411                       0.27091042
  flat_modelModel A-Maisonette               flat_modelModel A2
                      0.45464161                       0.37888433
    flat_modelMulti Generation         flat_modelNew Generation
                      0.21578521                       0.37173177
 flat_modelPremium Apartment flat_modelPremium Apartment Loft
                      0.33056966                       0.86046550
  flat_modelPremium Maisonette             flat_modelSimplified
                      0.11663674                       0.50597995
            flat_modelStandard                flat_modelTerrace
                      0.10324895                       0.79996609
             flat_modelType S1                flat_modelType S2
                      0.85156726                       1.00112097
                  rounded_years
                      0.46834531
```