DSE3101: HDB Resale Prices (Backend)

Presented by Brandon and Jessica

Overview of the presentation

- Motivation of our project
- Data Pre-Processing
- Exploratory Data Analysis
- Machine Learning Models: Algorithms, Tuning, Performance

Motivation

Am I paying too much
for my HDB resale flat?

How do I know if the location of my HDB flat is right for me?

Motivation

Create a tool that allows users to choose various characteristics of their HDB flat and predict the price.

Allow users to visualise the map of their HDB block selected to see nearby amenities

Data Pre-Processing

Gathering required data from various data sources

Obtaining Geospatial information for our HDB flats and amenities

Cleaning of HDB Resale Transactions Dataset

Obtaining Geospatial Data

1. Geospatial data was lacking for our HDB flats and amenities so we had to gather geo-coordinates
2. We made API calls to OneMap API leveraging on their "Search" API to gather the geospatial data

```
GET /api/common/elastic/search?searchVal=200640&returnGeom=Y&getAddrDetails=Y&pageNum=1
```


"SEARCHVAL": "640 ROWELL ROAD SINGAPORE 200640",
"BLK_NO": "640",
"ROAD_NAME": "ROWELL ROAD",
"BUILDING": "NIL",
"ADDRESS": "640 ROWELL ROAD SINGAPORE 200640",
"POSTAL": "200640",
"X": "30381.1007417506",
"Y": "32195.1006872542",
"LATITUDE": "1.30743547948389",
"LONGITUDE": "103.854713903431",
"LONGTITUDE": "103.854713903431"

Nearest Amenities

Using the geospatial data obtained, for each HDB block in our dataset, we calculated the distance to nearest amenities and number of such amenities within a 1km radius. We also mapped the distance to Central Business District (Downtown Core).






Data Cleaning

We made several key changes to the variables given in the dataset:

- Converted remaining lease from “X years Y months” to a continuous variable. For example, converting “61 years 06 months” to 61.5.
- Separated the date of transaction into month and year variables to create time dummies. This helps to control seasonality and time fixed effects.
- Converted storey_range from categorical to continuous. For example, we will take the average of "01 TO 03" as 2 for the storey range.

Data Cleaning

- Performed a left join of HDB resale transactions with the geospatial data we collected earlier
- Using the towns and the first two digits of postal code, we sought to control spatial heterogeneity for hedonic analysis
- Performed one hot encoding for categorical variables in order to fit the data for OLS and ML models

```
hdb_data                          175359 obs. of 137 variables
  $ resale_price                 : num  232000 250000 262000 265000 265…
  $ year                         : num  2017 2017 2017 2017 2017 ...
  $ floor_area_sqm               : num  44 67 67 68 67 68 68 67 68 67 ...
  $ remaining_lease              : num  61.3 60.6 62.4 62.1 62.4 ...
  $ ave_storey                   : num  22 4 4 10 4 4 10 10 10 4 ...
  $ dist_to_nearest_mrt          : num  1.005 0.19 0.536 0.947 0.502 ...
  $ mrt_1km                      : num  0 1 1 1 1 2 1 1 1 1 ...
  $ dist_to_nearest_supermarket  : num  0.39 0.392 0.854 0.48 0.899 ...
  $ supermarket_1km              : num  2 6 1 4 1 3 4 7 4 5 ...
  $ dist_to_nearest_hawkers      : num  0.182 0.357 0.586 0.246 0.612 ...
  $ hawkers_1km                  : num  3 5 3 3 3 4 4 4 4 4 ...
```

Exploratory Data Analysis

Testing for Multicollinearity

Metric:

Variation Inflation Factor

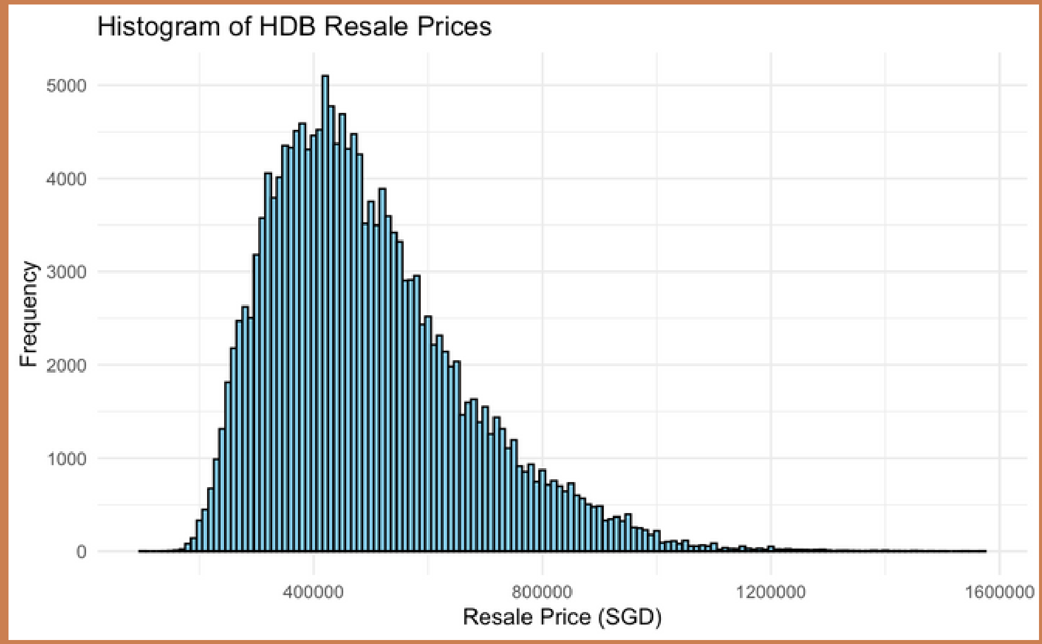
```
vif(model)
                      year              floor_area_sqm             remaining_lease
                  1.015082                    1.125389                    1.648845
                ave_storey         dist_to_nearest_mrt                     mrt_1km
                  1.199088                    1.676409                    2.927767
dist_to_nearest_supermarket            supermarket_1km      dist_to_nearest_hawkers
                  1.263692                    1.466485                    3.484218
               hawkers_1km dist_to_nearest_primary_schools         primary_schools_1km
                  2.718937                    1.352345                    2.040507
  dist_to_nearest_hospital                hospitals_1km                    dist_cbd
                  2.124141                    1.383144                    3.359596
```

VIF < 5 for all continuous variables, hence there is no severe multicollinearity within our data set.

Testing for Skewness

Metric:

Histogram of resale_prices



HDB Resale Prices follow a right skewed distribution.

Testing for Influential Points

Metric:

Cook's Distance

```
> # Testing for influential points
> C = cooks.distance(lm(log_resale_price~., data = hdb_resale))
> which(C>1)
named integer(0)
> # No influential points found.
```

Even though our dataset contained outliers, but based on Cook's Distance, we found no highly influential points.

Exploratory Data Analysis

Testing for Multicollinearity

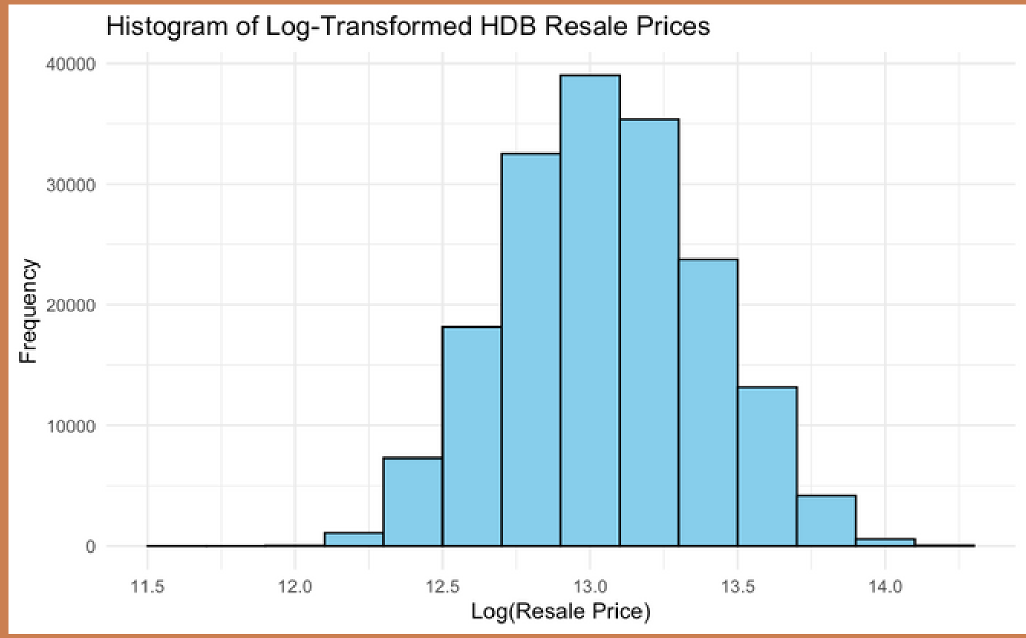
Metric:
Variation Inflation Factor

```
vif(model)
                       year                 floor_area_sqm                remaining_lease
                   1.015082                       1.125389                       1.648845
                 ave_storey            dist_to_nearest_mrt                        mrt_1km
                   1.199088                       1.676409                       2.927767
dist_to_nearest_supermarket                 supermarket_1km        dist_to_nearest_hawkers
                   1.263692                       1.466485                       3.484218
                hawkers_1km dist_to_nearest_primary_schools             primary_schools_1km
                   2.718937                       1.352345                       2.040507
   dist_to_nearest_hospital                   hospitals_1km                       dist_cbd
                   2.124141                       1.383144                       3.359596
```

VIF < 5 for all continuous variables, hence there is no severe multicollinearity within our data set.

Testing for Skewness

Metric:
Histogram of resale_prices



A log transformation of HDB resale prices gives a more symmetric distribution.

Testing for Influential Points

Metric:
Cook's Distance

```
> # Testing for influential points
> C = cooks.distance(lm(log_resale_price~., data = hdb_resale))
> which(C>1)
named integer(0)
> # No influential points found.
```

Even though our dataset contained outliers, but based on Cook's Distance, we found no highly influential points.

Train-Test Split

HDB resale transaction data from Jan 2017 to Mar 2024
(~175k observations)

Performed stratified Sampling by "Year" to ensure that our train / test dataset has equal proportions of observations belonging to each year.

Training Set
(~35k observations)

Test Set
(~140k observations)

Machine Learning Model Training Flow

Calculate RMSE based on predictions made via 10 fold cross validation on hyper parameters

OR

Fit the model to train data using different hyperparameter values.

Choose the hyperparameter values which give the lowest RMSE.

Fit the optimal model using tuned hyperparameters (using train data), generate final predictions using test data, calculate RMSE.

Root Mean Squared Error (RMSE)

Model	RMSE
Extreme Gradient Boosting Machine (XGBoost)	0.057909164860733
Random Forest	0.0631247324693827
Bagging	0.0670001339301782
Traditional Gradient Boosting Machine (GBM)	0.0919899399523445

Root Mean Squared Error (RMSE)

Model	RMSE
Post - LASSO Regression	0.104452066137839
OLS using important features selected from XGBoost	0.128174207150794
Benchmark OLS based on domain knowledge	0.1349587838384
Regression Trees	0.185634606041432

Benchmark OLS Model

```
Call:
lm(formula = log_price ~ year + remaining_lease + floor_area_sqm +
    ave_storey + +dist_to_nearest_mrt + dist_to_nearest_primary_schools +
    dist_cbd, data = train_ml)

Residuals:
     Min       1Q   Median       3Q      Max 
-0.72423 -0.09144 -0.00177  0.08885  0.79698 

Coefficients:
                                  Estimate Std. Error t value Pr(>|t|)    
(Intercept)                     -1.028e+02  7.100e-01 -144.83   <2e-16 ***
year                             5.671e-02  3.514e-04  161.38   <2e-16 ***
remaining_lease                  8.773e-03  5.863e-05  149.63   <2e-16 ***
floor_area_sqm                   1.009e-02  3.095e-05  326.13   <2e-16 ***
ave_storey                       4.455e-03  6.560e-05   67.91   <2e-16 ***
dist_to_nearest_mrt             -2.853e-02  1.708e-03  -16.70   <2e-16 ***
dist_to_nearest_primary_schools  3.914e-02  2.942e-03   13.31   <2e-16 ***
dist_cbd                        -3.259e-02  1.867e-04 -174.53   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.135 on 35064 degrees of freedom
Multiple R-squared:  0.8388,	Adjusted R-squared:  0.8388 
F-statistic: 2.607e+04 on 7 and 35064 DF,  p-value: < 2.2e-16
```

- RMSE: 0.1349587838384

Tree-Based Models

- Regression Trees
- Bootstrap Aggregation (Bagging)
- Random Forest
- Traditional Gradient Boosting
- Extreme Gradient Boosting

Types of Ensemble Methods

Sequential

Learners are generated sequentially. These methods use the dependency between base learners. Each learner influences the next one, likewise, a general paternal behavior can be deduced.

- GBM
- XGBoost

Parallel

Learners are generated in parallel. The base learners are created independently to study and exploit the effects related to their independence and reduce error by averaging the results.

- Bagging
- Random Forest

Algorithms: Regression Trees

Large trees are grown

The tree algorithm recursively does binary splits by selecting a predictor Xi and the cut-off point n.

It then splits the predictor space of Xi into two regions {X|Xj < s} and {X|Xj ≥ s} such that this leads to the biggest reduction in the Sum of Squared Residuals.

The algorithm continues to perform binary splits within each of the partitioned region until the stopping condition is met (i.e., each region contains no more than t = 5 observations)

Pruning of large trees to obtain a more parsimonious split, e.g., selecting the subtree with the lowest cross-validated error.

<u>Limitation:</u> performs poorly due to high variance, hence are often used in ensemble schemes which aim to reduce variance <u>(bagging and boosting)</u>

Fitting: Regression Trees

```r
# Create decision tree using regression
fit <- rpart(log_price ~ .,
             method = "anova", data = train_ml)

# Plot
plot(fit, uniform = TRUE,
          main = "log_price prediction using decision trees")
text(fit, use.n = TRUE, cex = .7)

# method anova is used for regression
predictions_rt <- predict(fit, test_ml, method = "anova")

# Calculate RMSE
rmse_rt <- sqrt(mean((test_ml_y - predictions_rt)^2))

# Print RMSE
print(paste("RMSE for Regression Trees:", rmse_rt))

# RMSE for Regression Trees: 0.185634606041432
```

- RMSE: 0.185634606041432 (highest RMSE)
- No tuning required (no manual selection of hyperparameters)

Algorithms: Bagging

fits low-bias, but high-variance trees

→

Takes the average of predictions (using test data) from multiple tree models using bootstrap samples of training data

→

final prediction: average of all predictions from multiple tree models

<u>Limitation:</u> Trees may be correlated with each other, which limits the benefits of averaging

Tuning: Bagging

```
ntreev = c(500, 1000)

nset = length(ntreev) #number of cases for tree numbers - will determine number of iterations in the loop

for(i in 1:nset) {
  rffit = randomForest(log_price~.,data=train_ml,ntree=ntreev[i], mtry= 136)
  predictions_rf <- predict(rffit, test_ml, method = "anova")
  # Calculate RMSE
  rmse_rt <- sqrt(mean((test_ml_y - predictions_rf)^2))
  print(paste("RMSE for Random Forest",c, rmse_rt))
}

# RMSE for Bagging w 500 trees: 0.0670008408203258
# RMSE for Bagging w 1000 trees: 0.0670001339301782

# We pick ntree = 1000 due to lower RMSE.
```

- Tune the number of bootstrap iterations: ntree
- Optimal ntree: 1000
- Optimal mtry (no. of variables per level): P (136)
- Optimal RMSE: 0.0670001339301782

Algorithms: Random Forest

fits decorrelated tree models (using train data) with only a subset of predictors for each tree

→

Takes the average or weighted of predictions (using test data) from multiple tree models using bootstrap samples of training data

→

final prediction: average of all predictions / weighted predictions from multiple tree models

<u>Benefit over bagging</u>: decorrelating trees improve the benefit of averaging to generate more accurate predictions

Tuning: Random Forest

```
### Random Forest
### Tuning
```{r}
set.seed(42)
# Now try the Random forest. To go from bagging to proper random forest, we
# need to add the option mtry - number of predictors randomly sampled for each tree:
# Here we set mtry= P/3 to reflect the default choice of P/3 for regression problems.
ntreev = c(500, 1000)

nset = length(ntreev) #number of cases for tree numbers - will determine number of iterations in the loop

for(i in 1:nset) {
  rffit = randomForest(log_price~.,data=train_ml,ntree=ntreev[i], mtry= floor(136/3))
  predictions_rf <- predict(rffit, test_ml, method = "anova")
  # Calculate RMSE
  rmse_rt <- sqrt(mean((test_ml_y - predictions_rf)^2))
  print(paste("RMSE for Random Forest","(trees:", ntreev[i],"):", rmse_rt))
}

# RMSE for Random Forest (trees: 500): 0.0641815677018226

# RMSE for Random Forest (trees: 1000): 0.0631247324693827

# We pick ntree = 500 due to both faster runtime and lower RMSE.
```

- Tune the number of bootstrap iterations: ntree
- Optimal ntree: 1000
- Optimal mtry (no. of variables per level): P/3 (136/3)
- Optimal RMSE: 0.0631247324693827
```

Algorithms: GBM Gradient Boosting Decision Trees Algorithm

Aim: fit initial small tree models with low variance and high bias, where predictions are iteratively added to reduce the bias

Use of Sequential learning techniques: Identify and correct the errors of previously learnt patterns/algorithms via gradient descent.

prevents overfitting

Starts off from a small tree model

As predictions are iteratively added, the tree learns more about the data and algorithms step-by-step.

<u>Key idea</u>: Every node of each tree takes a different subset of features / covariates to select the best split.

<u>Limitations:</u>
a. lacks regularisation techniques, unable to handle correlated covariates
b. less sophisticated tree pruning methods
c. no built-in feature seelction methods (to select themost significant covariates for prediction)

Tuning: GBM

```r
# Define the parameter grid
ntree_values <- c(500, 1000)  # Number of trees
interaction_depth_values <- c(2, 5)  # Interaction depth

# Initialize variables to store best parameters and performance
best_ntree <- NULL
best_interaction_depth <- NULL
best_performance <- Inf  # Initialize with a large value for minimization problems
```

```r
# Perform grid search
for (ntree in ntree_values) {
  for (depth in interaction_depth_values) {
    # Train GBM model on training data with current hyperparameters
    gbm_model <- gbm(log_price ~ ., data = train_ml, distribution = 'gaussian',
                     interaction.depth = depth, n.trees = ntree, shrinkage = 0.01, cv.folds = 10)

    # Evaluate performance (you can use different metrics here)
    # For example, you might want to use mean squared error from cross-validation

    gbm_perf_plot <- gbm.perf(gbm_model, method = "cv")

    pdf("gbm_perf_plot.pdf")
    print(gbm_perf_plot)
    dev.off()

    performance <- gbm.perf(gbm_model, method = "cv")

    print(gbm_perf_plot)


    # Manually calculate cross-validation error
    best_tree_index <- which.min(gbm_model$cv.error)

    # Get MSE for best tree
    mse_cv <- gbm_model$cv.error[best_tree_index]

    # Update best parameters if performance is improved
    if (mse_cv < best_performance) {
      best_ntree <- ntree
      best_interaction_depth <- depth
      best_performance <- mse_cv
```

- Tune n_tree and interaction_depth using 10-fold CV
- Optimal n_tree: 500 vs 1000
- Optimal interaction_depth: 2 vs 5
- Optimal RMSE: 0.0919899399523445

Algorithms: XGBoost Gradient Boosting Decision Trees Algorithm

Aim: fit initial small tree models with low variance and high bias, where predictions are iteratively added to reduce the bias

Use of Sequential learning techniques: Identify and correct the errors of previously learnt patterns/algorithms via gradient descent.

Starts off from a small tree model

As predictions are iteratively added, the tree learns more about the data and algorithms step-by-step.

<u>Advantages over GBM:</u>

a. Handles multicollinearity using L1 and L2 regularisation techniques, prevents overfitting
b. Nonlinearity detection
c. Built-in feature selection methods (to select the most significant covariates for prediction)

Limitation: Prone to overfitting in large datasets

Tuning: Extreme Gradient Boosting Machine

```
### Tuning XGB
```{r}
set.seed(42)

nroundv = c(500, 1000)

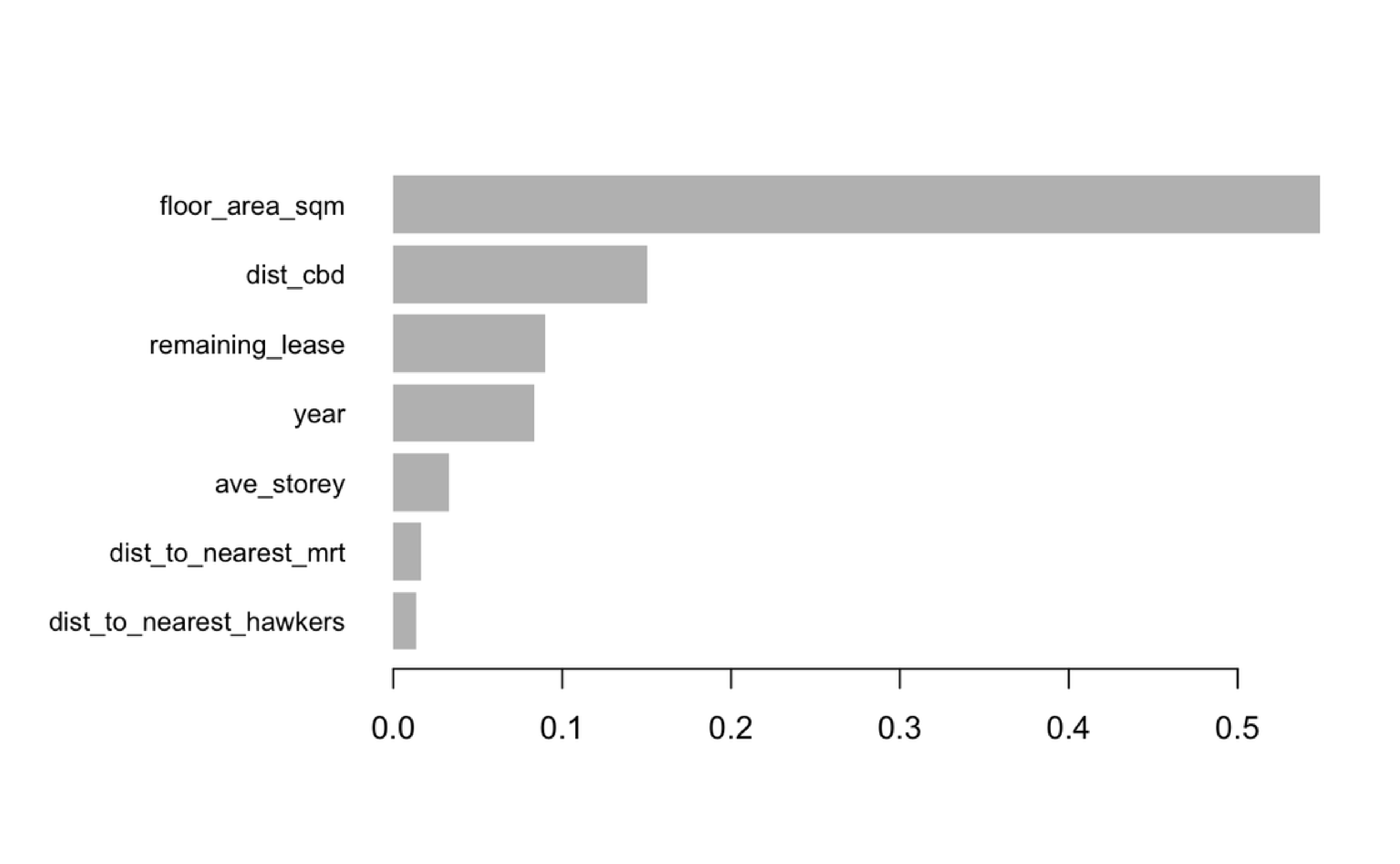
nset = length(nroundv) #number of cases for tree numbers - will determine number of iterations in the loop

for(i in 1:nset) {
  xgbfit = xgboost(
  data = X_train,
  label = Y_train,
  nrounds = nroundv[i],
  objective = "reg:squarederror"
)
  predictions_xg <- predict(xgb.fit, X_test)
  # Calculate RMSE
  rmse_xg <- sqrt(mean((test_ml_y - predictions_xg)^2))
  print(paste("RMSE for XG Boost:", "(rounds:", nroundv[i],"):", rmse_xg))
}
```

- Tune nrounds: Number of sequential trees that are trained to correct errors of previously trained trees.
- Optimal nrounds: 500
- Optimal RMSE: 0.0579091648607333
```

OLS based on feature selection by XGB

Variable Importance Plot




```
Call:
lm(formula = formula, data = train_ml)

Residuals:
     Min       1Q   Median       3Q      Max 
-0.72558 -0.08608 -0.00188  0.08306  0.84638 

Coefficients:
                         Estimate Std. Error t value Pr(>|t|)    
(Intercept)            -1.055e+02  6.742e-01 -156.53   <2e-16 ***
floor_area_sqm          1.025e-02  2.942e-05  348.58   <2e-16 ***
dist_cbd               -2.832e-02  1.896e-04 -149.39   <2e-16 ***
remaining_lease         1.049e-02  6.187e-05  169.50   <2e-16 ***
year                    5.800e-02  3.336e-04  173.87   <2e-16 ***
ave_storey              4.167e-03  6.239e-05   66.80   <2e-16 ***
dist_to_nearest_mrt    -5.152e-02  1.650e-03  -31.22   <2e-16 ***
dist_to_nearest_hawkers -3.434e-02  5.356e-04  -64.12   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.128 on 35064 degrees of freedom
Multiple R-squared:  0.855,	Adjusted R-squared:  0.855 
F-statistic: 2.954e+04 on 7 and 35064 DF,  p-value: < 2.2e-16
```

- Optimal RMSE: 0.128174207150794

Benchmark OLS vs XGB Feature Selected OLS

<table>
<tr><th>Benchmark OLS Model</th><th>XGB Feature Selected OLS Model</th></tr>
<tr><td colspan="2">floor_area_sqm</td></tr>
<tr><td colspan="2">dist_cbd</td></tr>
<tr><td colspan="2">remaining_lease</td></tr>
<tr><td colspan="2">year</td></tr>
<tr><td colspan="2">ave_storey</td></tr>
<tr><td colspan="2">dist_to_nearest_mrt</td></tr>
<tr><td>dist_to_nearest_primary_schools</td><td>dist_to_nearest_hawkers</td></tr>
</table>

- RMSE: 0.1349587838384

- RMSE: 0.128174207150794

Conclusion: Why XGBoost?

Best Performance

- Lowest RMSE for prediction using test-data: overfitting is not a concern

Most suited to our dataset characteristics

- Handles possible issues from having many correlated covariates

Boosting over bagging

- Sequential learning allows for more comprehensive learning processes of the dataset characteristics than bootstrap aggregation. The use of weights over simple averaging of the predictions from multiple tree models further minimises loss/error, generating more accurate predictions.

Our Model Evaluation - What could we have done better?

- <u>Complete control in overfitting prevention</u>

  1. train with more samples
  2. reduce the number of features (compare the importances)
  3. reduce the maximum depth
  4. increase the minimum samples at the leaves

- <u>Finding the most optimal model</u>

  Conduct hyperparameter tuning for each hyperparameter, via 10-fold CV

Thank You

References

Ensemble methods:
https://neptune.ai/blog/xgboost-everything-you-need-to-know

Regression trees:
LAM, N. Z. D. (2021). UNDERSTANDING HDB RESALE PRICES IN SINGAPORE: A MACHINE LEARNING & ECONOMETRICS APPROACH (thesis).

Gradient Boosting:
https://medium.com/all-things-ai/in-depth-parameter-tuning-for-gradient-boosting-3363992e9bae
https://bradleyboehmke.github.io/HOML/gbm.html
https://machinelearningmastery.com/configure-gradient-boosting-algorithm/
https://neptune.ai/blog/xgboost-everything-you-need-to-know
https://medium.com/@gabrieltseng/gradient-boosting-and-xgboost-c306c1bcfaf5

Images:
Mrt: https://ireus.nus.edu.sg/mrt-and-property-value/
Primary School: https://en.wikipedia.org/wiki/Tao_Nan_School
Hospitals: https://www.ntfgh.com.sg/About-NTFGH/Pages/Overview.aspx
Supermarkets: https://expatliving.sg/supermarkets-in-singapore-grocery-stores-and-groceries-online/
Hawker Centers: https://www.visitsingapore.com/editorials/the-street-food-of-singapore/
CBD: https://en.wikipedia.org/wiki/Central_Area,_Singapore
Stock Images from FreePik