Predicting Apartment Prices

Team 6
Paula Demacker, Peter Mankiewich, (Vince)Chenzhi Pan, (Sylar)Jiajian Guo

Short Recap

Problem: How to decide on a reasonable price for an apartment?

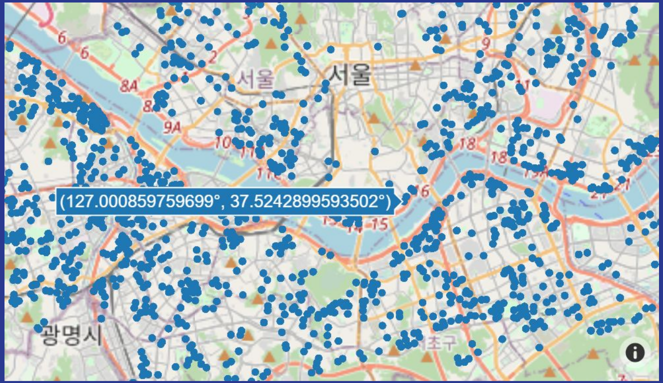
- CSV-File downloaded via Kaggle
- 1,6 Million rows
- 25 columns
- numeric, character and logical variables
- Target variable: Sales Price

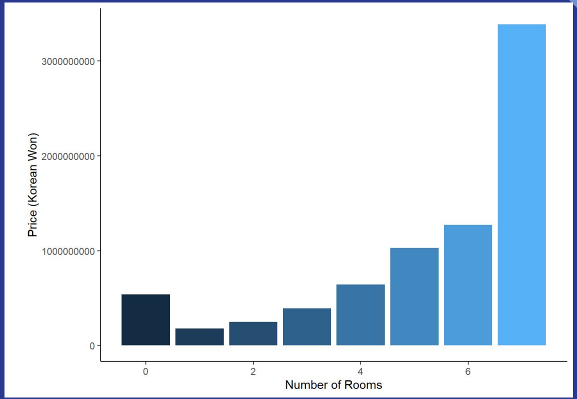
Real World Application:

- Apartment buyers can compare with this model's predicted prices to estimate if their offered prices by sellers were overvalued or undervalued

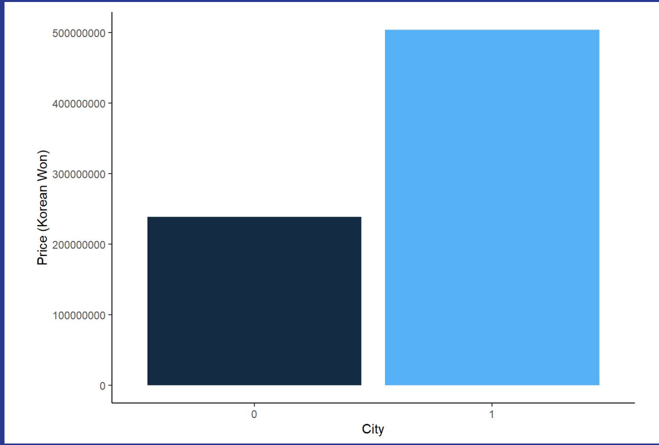
- Real estate developers can predict future sales and calculate ROI (Return on Investment) more accurately before the apartments are open to market yet

EDA and Cleaning



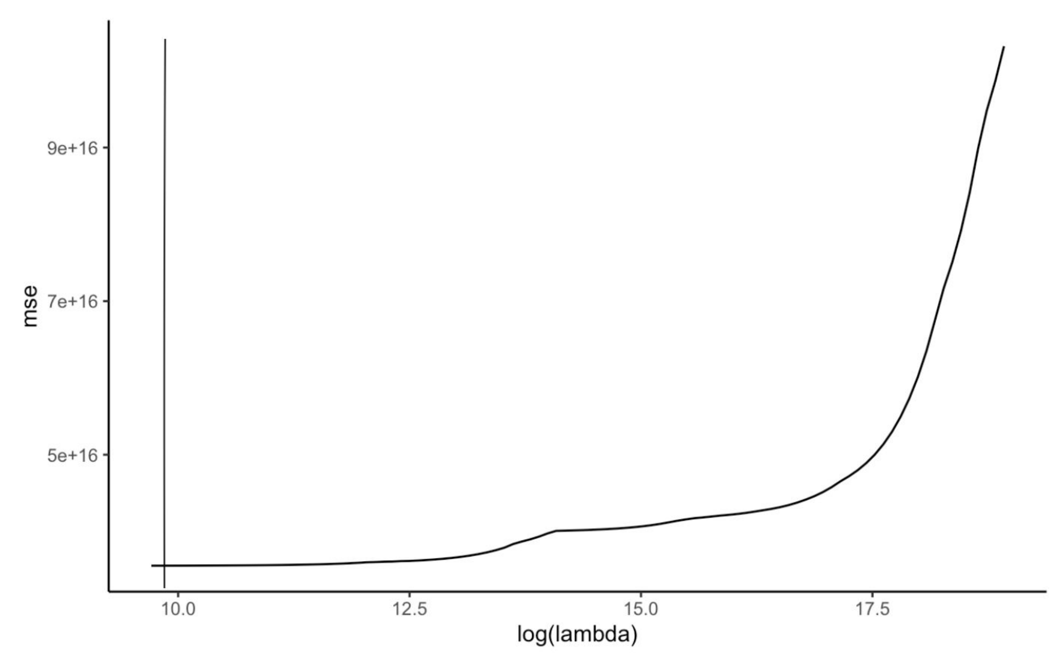


- Developed some charts to understand the distribution of variables and their relationships
- Verified location information
- Created dummy variables for later machine learning steps



Lasso regression

- Best Lambda = 16,491
- Test RMSE = 188,678,874
- No zero coefficients
- All variables are good predictors



Main Results

- Variables importances;
  Based on Tree, Boosting , consistent with each other

- OLS, Lasso, Ridge, Forward, Backward, Trees
  -> better interpretation
  Bagging, Boosting, RandomForest -> relatively more accurate

- Boosting

Method name	**Training RMSE**	**Test RMSE**
OLS	1.88332e+8	1.88545e+8
Forward	1.88324e+8	1.88550e+8
Backward	1.88322e+8	1.88550e+8
Ridge	2.01424e+8	2.01931e+8
Lasso	1.88455e+8	1.88679e+8
Tree	1.82321e+8	1.98231e+8
Bagging	3.6043e+7	9.13133e+7
Random Forest	4.3019e+7	9.33058e+7
Boosting	2.7564e+7	4.5081e+5
Boosting with FE	4.0315e+7	5.01138e+7

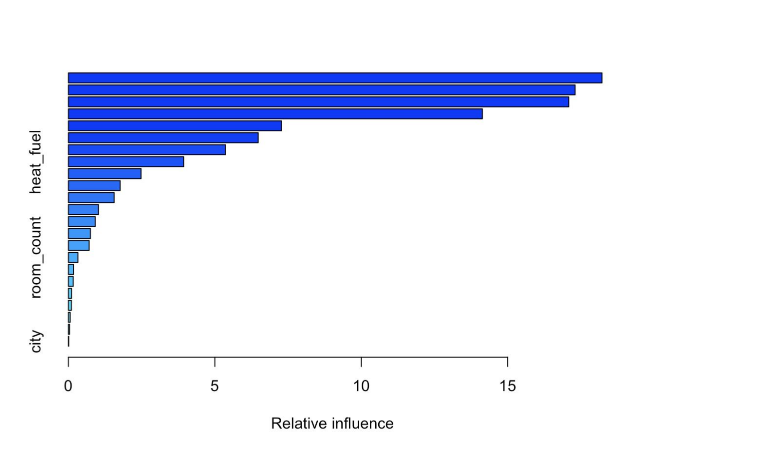
Choosing the Best Model

Boosting

- Randomly use 8000 rows
- 5000 Lowest CV error
- Use 4 as depth
- Lowest Test RMSE (8*10^5)
- Significantly better than other method
- RMSE varies a lot (Use different test sample test)

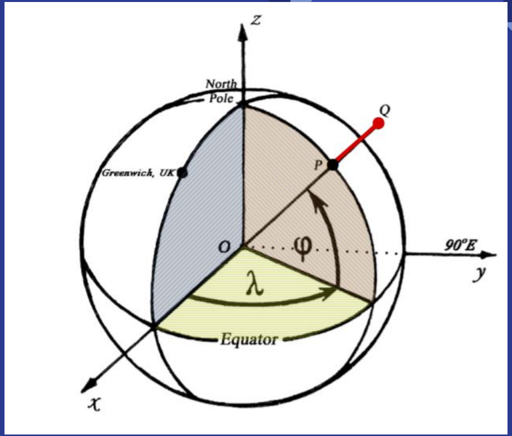
Relatively important predictors

- Supply_area
- Exclusive_use_area
- address _by_law



Feature engineering

- Converting latitude and longitude into a 3d space
  - x: cos(longitude) * cos(longitude)
  - y: cos(latitude) * sin(longitude)
  - z: sin(latitude)
- Supply area: area of entire apartment complex
- Exclusive use area: total floor area of building
- Living area = supply area - exclusive use area
- Area ratio = exclusive_use_area / supply_area



	Training RMSE	Test RMSE
Boosting	2.7564e+7	7.17116e+5
Boosting with FE	4.0315e+7	5.01138e+7

Challenges

- Language & Industry knowledge:
    - *Challenge*: Korean auto-translated English columns names without explanations
    - *Solution*: Google satellite images with coordinates and learned Korean apartment structures
- Data Cleaning:
    - *Challenge*: Miscellaneous character type qualitative variable inputs data
    - *Solution*: Hard coded into dummy variables or numerics
- Modeling
    - *Challenge*: Little RMSE improvement in linear models
    - *Solution*: Performed a wide range of different models with flexibilities
- Collaboration:
    - *Challenge*: Combing everyone’s code into one final file without conflicts
    - *Solution*: Used Git linking RStudio and avoided accidental overwrites

Conclusion

- What We Learned:
  - Needless to mess with millions rows of data when similar results can be generated by a 1/10 size
  - Thought the seemingly intuitional dataset should perform well on simple models but totally the opposite
  - Code collaborations suck without cloud environments
  - Industry Applications:
    - Help predict prices based off of the characteristics of apartments and their immediate environment
    - Speed up real estate agents price prediction labelling work by automation

- What to Put into Practice in Future:
  - Spend more time on models which perform better
  - Use cloud environments for team projects
  - Collect unstructured data to improve prediction accuracy if possible: satellite images for computer vision

Q&A

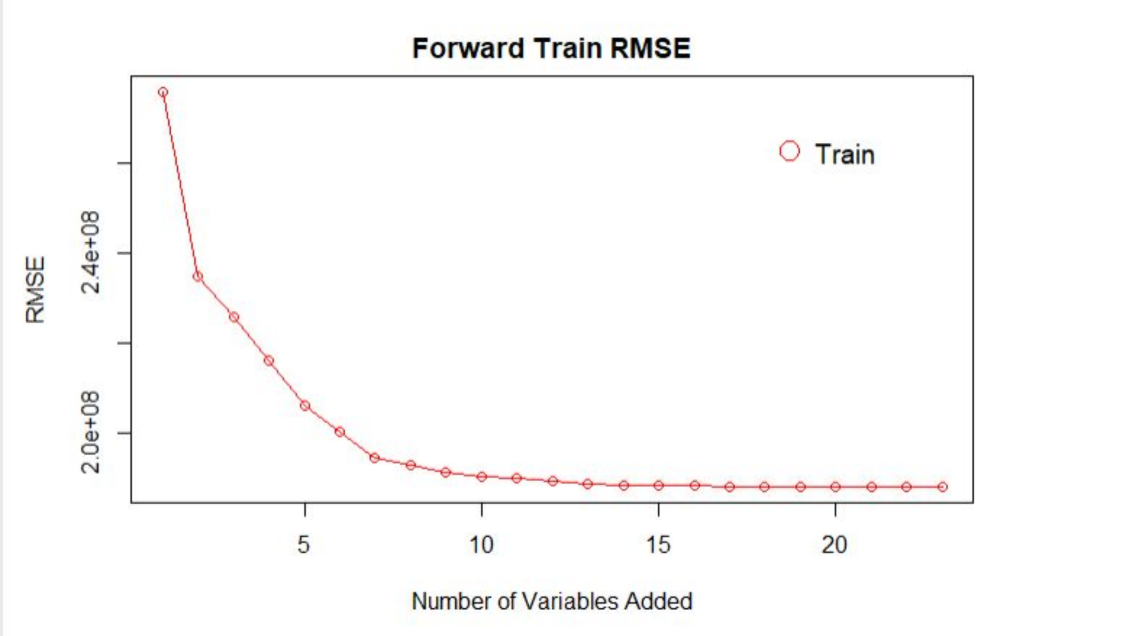
OLS Linear Regression

```
Call:
lm(formula = transaction_real_price ~ ., data = dd_train)

Residuals:
       Min         1Q     Median         3Q        Max 
-1.118e+09 -9.743e+07 -1.692e+07  7.074e+07  6.097e+09 

Coefficients:
                                  Estimate Std. Error t value Pr(>|t|)    
(Intercept)                     -5.436e+10  9.421e+08 -57.707  < 2e-16 ***
city                             5.765e+09  5.642e+07 102.180  < 2e-16 ***
transaction_year_month           1.668e+05  1.428e+03 116.823  < 2e-16 ***
transaction_date                 6.465e+05  6.468e+05   1.000 0.317529    
year_of_completion              -2.637e+06  9.746e+04 -27.053  < 2e-16 ***
exclusive_use_area               1.553e+06  1.346e+05  11.538  < 2e-16 ***
floor                            1.869e+06  8.684e+04  21.525  < 2e-16 ***
latitude                        -1.026e+09  1.106e+07 -92.799  < 2e-16 ***
longitude                        4.498e+08  7.047e+06  63.830  < 2e-16 ***
address_by_law                   1.417e+00  3.967e-02  35.713  < 2e-16 ***
total_parking_capacity_in_site   2.850e+04  1.047e+03  27.226  < 2e-16 ***
total_household_count_in_sites  -8.174e+04  1.679e+03 -48.684  < 2e-16 ***
apartment_building_count_in_sites  6.668e+06  8.495e+04  78.494  < 2e-16 ***
tallest_building_in_sites        2.406e+06  1.213e+05  19.832  < 2e-16 ***
lowest_building_in_sites         3.271e+06  1.199e+05  27.291  < 2e-16 ***
heat_fuel                        5.167e+07  3.737e+06  13.828  < 2e-16 ***
supply_area                      3.326e+06  1.148e+05  28.977  < 2e-16 ***
total_household_count_of_area_type -1.459e+04  1.929e+03  -7.565 3.90e-14 ***
room_count                      -6.117e+06  1.280e+06  -4.778 1.77e-06 ***
bathroom_count                  -6.561e+06  1.725e+06  -3.803 0.000143 ***
heat_type_central                7.847e+06  2.117e+06   3.707 0.000210 ***
heat_type_district               6.760e+07  3.779e+06  17.890  < 2e-16 ***
front_door_structure_corridor   -1.441e+07  1.760e+06  -8.184 2.77e-16 ***
front_door_structure_mixed      -2.789e+06  4.178e+06  -0.668 0.504395    
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 188300000 on 127976 degrees of freedom
Multiple R-squared:  0.6566,	Adjusted R-squared:  0.6565 
F-statistic: 1.064e+04 on 23 and 127976 DF,  p-value: < 2.2e-16
```

Simple Model

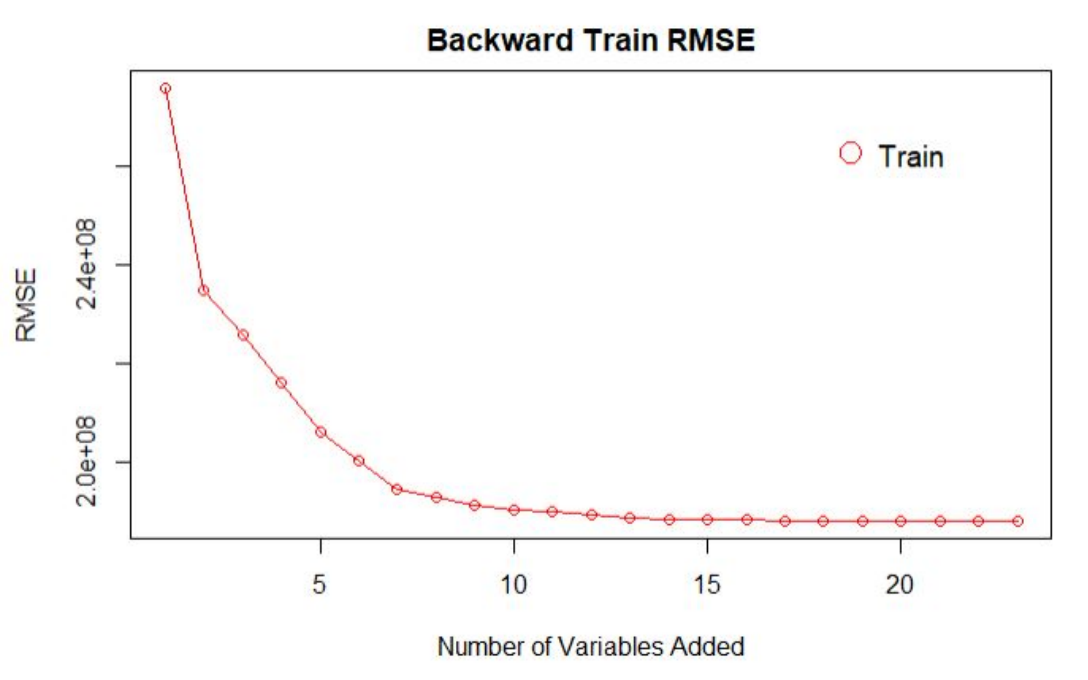
- Train RMSE: 188,332,311
- Test RMSE: 188,544,540

Forward Selection



- Only intercept: train RMSE is 275,820,794
- After adding 21 predictors one by one, Train MSE kept reducing to 188,321,979.
- Adding transaction_date & front_door_structure_mixed increases RMSE
- Test RMSE is 188,550,321

Backward Selection



- Only intercept: train RMSE is 275,820,794
- After adding 21 predictors one by one, Train RMSE kept reducing to 188,321,979.
- Adding transaction_date & front_door_structure_mixed increases RMSE
- Test RMSE is 188,550,321

Ridge Regression

```
Train RMSE:  201425915 Test RMSE:  201936147 Best Lambda:  16491016
```

- Train RMSE: 201425915
- Test RMSE: 201936147
- Best Lambda: 16491016

Bagging

```
[35]: set.seed(217)
      test.mse = c()

      for (i in seq(1,5)) {
       train = sample(1:nrow(Price),(nrow(Price)/100)*i)
        tree.testy = Price[-train,transaction_real_price]
        tree.test = Price[-train]
       bag.price = randomForest(transaction_real_price ~ ., data = Price, subset = train, mtry = 24, importance = TRUE)
        yhat.bag = predict(bag.price, newdata = tree.test)
       test.mse = c(test.mse,mean((tree.testy-yhat.bag)^2)))
      }

      test.mse
```

6538671859298821 · 4728267858107837 · 4151840578511839 · 3401919969449496 · 3153720272922860

MORE THAN 12 HOURS to train in GOOGLE AI NOTEBOOK with EXPENSIVE COMPUTER OPTION

Decision Tree