property_model.R

2024-09-19

Loading in libraries and data

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.0
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
unneeded_cols <- c('suburb', 'date_sold', 'suburb_lat','suburb_lng','suburb_elevation', 'suburb_sqkm',
df <- read.csv('master_dataset.csv') |> as_tibble() |> dplyr::select(-unneeded_cols)
```

```
## Warning: Using an external vector in selections was deprecated in tidyselect 1.1.0.
## i Please use `all_of()` or `any_of()` instead.
##   # Was:
##   data %>% select(unneeded_cols)
##
##   # Now:
##   data %>% select(all_of(unneeded_cols))
##
## See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
df
```

```
## # A tibble: 8,629 x 11
##      price num_bath num_bed num_parking property_size type  suburb_median_income
##      <int>    <int>   <int>       <int>         <int> <chr>                <int>
##  1  452000        1       3           1           344 House                32292
##  2  495000        1       3           2           582 House                32292
##  3  890000        2       4           3           715 House                40560
##  4  533000        3       4           2           695 House                24180
##  5 1120500        2       4           2           904 House                40092
##  6  830000        3       6           2          2109 House                38740
```

```
## 7  675000        3       3           2          263 Town~                  24388
## 8  473000        1       3           3          581 House                  32292
## 9  520000        1       3           1          651 House                  32292
## 10 510000        1       3           1          993 House                  38740
## # i 8,619 more rows
## # i 4 more variables: cash_rate <dbl>, property_inflation_index <dbl>,
## #   km_from_cbd <dbl>, suburb_ranking <dbl>
```

We will do stepwise selection to select relevant variables from master dataset

```r
null_model <- lm(price ~ 1, data = df)
full_model <- lm(price ~ ., data = df)

forward_model <- step(null_model,
                      scope = list(lower = null_model, upper = full_model),
                      direction = "both")
```

```
## Start:  AIC=243793.6
## price ~ 1
##
##                            Df  Sum of Sq        RSS    AIC
## + num_bath                  1 2.6182e+15 1.3448e+16 242261
## + suburb_median_income      1 2.0500e+15 1.4016e+16 242618
## + km_from_cbd               1 1.9377e+15 1.4128e+16 242687
## + suburb_ranking            1 1.6903e+15 1.4376e+16 242836
## + num_bed                   1 1.6746e+15 1.4391e+16 242846
## + num_parking               1 8.5613e+14 1.5210e+16 243323
## + cash_rate                 1 7.0024e+14 1.5366e+16 243411
## + property_inflation_index  1 6.9110e+14 1.5375e+16 243416
## + type                     13 5.4490e+14 1.5521e+16 243522
## + property_size             1 1.6087e+14 1.5905e+16 243709
## <none>                                   1.6066e+16 243794
##
## Step:  AIC=242260.6
## price ~ num_bath
##
##                            Df  Sum of Sq        RSS    AIC
## + km_from_cbd               1 2.0538e+15 1.1394e+16 240833
## + suburb_median_income      1 1.9108e+15 1.1537e+16 240940
## + suburb_ranking            1 1.7086e+15 1.1739e+16 241090
## + property_inflation_index  1 5.2186e+14 1.2926e+16 241921
## + cash_rate                 1 5.0587e+14 1.2942e+16 241932
## + type                     13 2.7959e+14 1.3168e+16 242105
## + property_size             1 4.0131e+13 1.3408e+16 242237
## + num_parking               1 9.8244e+12 1.3438e+16 242256
## + num_bed                   1 4.7413e+12 1.3443e+16 242260
## <none>                                   1.3448e+16 242261
## - num_bath                  1 2.6182e+15 1.6066e+16 243794
##
## Step:  AIC=240832.5
## price ~ num_bath + km_from_cbd
##
##                            Df  Sum of Sq        RSS    AIC
```

```
## + suburb_median_income      1 9.2557e+14 1.0468e+16 240103
## + type                     13 6.5097e+14 1.0743e+16 240351
## + property_inflation_index  1 4.3705e+14 1.0957e+16 240497
## + suburb_ranking            1 3.9730e+14 1.0997e+16 240528
## + cash_rate                 1 3.6134e+14 1.1033e+16 240556
## + property_size             1 2.7075e+14 1.1123e+16 240627
## + num_bed                   1 1.6371e+14 1.1230e+16 240710
## + num_parking               1 1.1369e+14 1.1280e+16 240748
## <none>                                 1.1394e+16 240833
## - km_from_cbd               1 2.0538e+15 1.3448e+16 242261
## - num_bath                  1 2.7343e+15 1.4128e+16 242687
##
## Step:  AIC=240103.4
## price ~ num_bath + km_from_cbd + suburb_median_income
##
##                            Df  Sum of Sq        RSS    AIC
## + type                     13 6.6860e+14 9.7997e+15 239560
## + property_inflation_index  1 4.7441e+14 9.9939e+15 239705
## + cash_rate                 1 3.6608e+14 1.0102e+16 239798
## + num_bed                   1 2.9576e+14 1.0173e+16 239858
## + property_size             1 2.8461e+14 1.0184e+16 239868
## + num_parking               1 2.3779e+14 1.0231e+16 239907
## + suburb_ranking            1 9.4017e+13 1.0374e+16 240028
## <none>                                 1.0468e+16 240103
## - suburb_median_income      1 9.2557e+14 1.1394e+16 240833
## - km_from_cbd               1 1.0686e+15 1.1537e+16 240940
## - num_bath                  1 2.5988e+15 1.3067e+16 242015
##
## Step:  AIC=239559.9
## price ~ num_bath + km_from_cbd + suburb_median_income + type
##
##                            Df  Sum of Sq        RSS    AIC
## + property_inflation_index  1 4.2284e+14 9.3769e+15 239181
## + cash_rate                 1 3.4198e+14 9.4578e+15 239255
## + property_size             1 1.8536e+14 9.6144e+15 239397
## + num_parking               1 1.6976e+14 9.6300e+15 239411
## + suburb_ranking            1 1.4663e+14 9.6531e+15 239432
## + num_bed                   1 1.0869e+14 9.6910e+15 239466
## <none>                                 9.7997e+15 239560
## - type                     13 6.6860e+14 1.0468e+16 240103
## - suburb_median_income      1 9.4320e+14 1.0743e+16 240351
## - km_from_cbd               1 1.3896e+15 1.1189e+16 240702
## - num_bath                  1 2.1635e+15 1.1963e+16 241279
##
## Step:  AIC=239181.3
## price ~ num_bath + km_from_cbd + suburb_median_income + type +
##     property_inflation_index
##
##                            Df  Sum of Sq        RSS    AIC
## + property_size             1 1.9472e+14 9.1822e+15 239002
## + num_parking               1 1.6238e+14 9.2145e+15 239033
## + suburb_ranking            1 1.3437e+14 9.2425e+15 239059
## + num_bed                   1 9.7917e+13 9.2790e+15 239093
## + cash_rate                 1 1.7357e+13 9.3595e+15 239167
```

```
## <none>                                 9.3769e+15 239181
## - property_inflation_index  1 4.2284e+14 9.7997e+15 239560
## - type                     13 6.1703e+14 9.9939e+15 239705
## - suburb_median_income      1 9.7837e+14 1.0355e+16 240036
## - km_from_cbd               1 1.2967e+15 1.0674e+16 240297
## - num_bath                  1 2.0289e+15 1.1406e+16 240870
##
## Step:  AIC=239002.2
## price ~ num_bath + km_from_cbd + suburb_median_income + type +
##     property_inflation_index + property_size
##
##                            Df  Sum of Sq        RSS    AIC
## + num_parking               1 1.2298e+14 9.0592e+15 238888
## + suburb_ranking            1 1.0874e+14 9.0734e+15 238901
## + num_bed                   1 9.0187e+13 9.0920e+15 238919
## + cash_rate                 1 1.4390e+13 9.1678e+15 238991
## <none>                                 9.1822e+15 239002
## - property_size             1 1.9472e+14 9.3769e+15 239181
## - property_inflation_index  1 4.3220e+14 9.6144e+15 239397
## - type                     13 5.1938e+14 9.7016e+15 239451
## - suburb_median_income      1 9.9008e+14 1.0172e+16 239884
## - km_from_cbd               1 1.4309e+15 1.0613e+16 240250
## - num_bath                  1 1.8698e+15 1.1052e+16 240600
##
## Step:  AIC=238887.9
## price ~ num_bath + km_from_cbd + suburb_median_income + type +
##     property_inflation_index + property_size + num_parking
##
##                            Df  Sum of Sq        RSS    AIC
## + suburb_ranking            1 1.1668e+14 8.9425e+15 238778
## + num_bed                   1 5.4594e+13 9.0046e+15 238838
## + cash_rate                 1 1.2739e+13 9.0465e+15 238878
## <none>                                 9.0592e+15 238888
## - num_parking               1 1.2298e+14 9.1822e+15 239002
## - property_size             1 1.5532e+14 9.2145e+15 239033
## - property_inflation_index  1 4.2467e+14 9.4839e+15 239281
## - type                     13 4.7622e+14 9.5354e+15 239304
## - num_bath                  1 1.0511e+15 1.0110e+16 239833
## - suburb_median_income      1 1.0692e+15 1.0128e+16 239849
## - km_from_cbd               1 1.4806e+15 1.0540e+16 240192
##
## Step:  AIC=238778
## price ~ num_bath + km_from_cbd + suburb_median_income + type +
##     property_inflation_index + property_size + num_parking +
##     suburb_ranking
##
##                            Df  Sum of Sq        RSS    AIC
## + num_bed                   1 5.7984e+13 8.8845e+15 238724
## + cash_rate                 1 1.1357e+13 8.9312e+15 238769
## <none>                                 8.9425e+15 238778
## - suburb_ranking            1 1.1668e+14 9.0592e+15 238888
## - num_parking               1 1.3091e+14 9.0734e+15 238901
## - property_size             1 1.3097e+14 9.0735e+15 238901
## - property_inflation_index  1 4.1225e+14 9.3548e+15 239165
```

```
## - type                      13 5.1912e+14 9.4616e+15 239239
## - suburb_median_income      1 7.2386e+14 9.6664e+15 239448
## - km_from_cbd               1 8.7651e+14 9.8190e+15 239583
## - num_bath                  1 1.0402e+15 9.9827e+15 239726
##
## Step:  AIC=238723.9
## price ~ num_bath + km_from_cbd + suburb_median_income + type +
##     property_inflation_index + property_size + num_parking +
##     suburb_ranking + num_bed
##
##                            Df  Sum of Sq        RSS    AIC
## + cash_rate                 1 1.0568e+13 8.8740e+15 238716
## <none>                                   8.8845e+15 238724
## - num_bed                   1 5.7984e+13 8.9425e+15 238778
## - num_parking               1 9.3196e+13 8.9777e+15 238812
## - suburb_ranking            1 1.2007e+14 9.0046e+15 238838
## - property_size             1 1.3008e+14 9.0146e+15 238847
## - num_bath                  1 3.1868e+14 9.2032e+15 239026
## - type                     13 4.0390e+14 9.2884e+15 239082
## - property_inflation_index  1 4.0456e+14 9.2891e+15 239106
## - suburb_median_income      1 7.5081e+14 9.6353e+15 239422
## - km_from_cbd               1 9.1191e+14 9.7964e+15 239565
##
## Step:  AIC=238715.6
## price ~ num_bath + km_from_cbd + suburb_median_income + type +
##     property_inflation_index + property_size + num_parking +
##     suburb_ranking + num_bed + cash_rate
##
##                            Df  Sum of Sq        RSS    AIC
## <none>                                   8.8740e+15 238716
## - cash_rate                 1 1.0568e+13 8.8845e+15 238724
## - num_bed                   1 5.7196e+13 8.9312e+15 238769
## - num_parking               1 9.2087e+13 8.9661e+15 238803
## - property_inflation_index  1 1.0681e+14 8.9808e+15 238817
## - suburb_ranking            1 1.1869e+14 8.9927e+15 238828
## - property_size             1 1.2837e+14 9.0023e+15 238838
## - num_bath                  1 3.1763e+14 9.1916e+15 239017
## - type                     13 4.0674e+14 9.2807e+15 239076
## - suburb_median_income      1 7.4711e+14 9.6211e+15 239411
## - km_from_cbd               1 9.0148e+14 9.7754e+15 239549
```

```r
summary(forward_model)
```

```
##
## Call:
## lm(formula = price ~ num_bath + km_from_cbd + suburb_median_income +
##     type + property_inflation_index + property_size + num_parking +
##     suburb_ranking + num_bed + cash_rate, data = df)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -15607636   -410120   -104746    237709  46563409
##
## Coefficients:
```

```
##                                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)                       -6.801e+05  6.129e+05  -1.110  0.26713
## num_bath                           2.777e+05  1.582e+04  17.551  < 2e-16 ***
## km_from_cbd                       -2.963e+04  1.002e+03 -29.568  < 2e-16 ***
## suburb_median_income               2.975e+01  1.105e+00  26.917  < 2e-16 ***
## typeApartment / Unit / Flat       -1.052e+06  5.918e+05  -1.777  0.07555 .
## typeBlock of Units                -1.299e+06  6.226e+05  -2.087  0.03693 *
## typeDevelopment Site               9.381e+05  7.030e+05   1.334  0.18211
## typeDuplex                        -6.942e+05  6.041e+05  -1.149  0.25049
## typeHouse                         -3.088e+05  5.895e+05  -0.524  0.60040
## typeNew Apartments / Off the Plan -6.320e+05  6.807e+05  -0.928  0.35318
## typeNew House & Land              -3.832e+05  7.207e+05  -0.532  0.59495
## typeNew land                      -1.086e+06  9.292e+05  -1.168  0.24265
## typeSemi-Detached                 -6.669e+05  5.956e+05  -1.120  0.26290
## typeStudio                        -1.355e+06  7.458e+05  -1.817  0.06920 .
## typeTerrace                       -7.556e+05  6.042e+05  -1.251  0.21112
## typeTownhouse                     -9.123e+05  5.944e+05  -1.535  0.12488
## typeVilla                         -6.879e+05  5.988e+05  -1.149  0.25068
## property_inflation_index           7.128e+03  7.004e+02  10.178  < 2e-16 ***
## property_size                      1.360e+02  1.219e+01  11.158  < 2e-16 ***
## num_parking                        9.180e+04  9.714e+03   9.450  < 2e-16 ***
## suburb_ranking                    -9.814e+02  9.147e+01 -10.729  < 2e-16 ***
## num_bed                            9.832e+04  1.320e+04   7.448 1.04e-13 ***
## cash_rate                         -8.507e+04  2.657e+04  -3.201  0.00137 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1015000 on 8606 degrees of freedom
## Multiple R-squared:  0.4477, Adjusted R-squared:  0.4462
## F-statistic:   317 on 22 and 8606 DF,  p-value: < 2.2e-16
```

So all variables are used in the model

Do a train / test split and test the out of sample mean absolute percentage error

```r
# Load necessary libraries
library(caret)  # For train-test split
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

```r
library(Metrics)  # For MAPE calculation
```

```
##
## Attaching package: 'Metrics'
```

```
## The following objects are masked from 'package:caret':
##
##     precision, recall
```

```r
# Set seed for reproducibility
set.seed(23)

# Create a train / test / validation split (70% / 15% / 15%)
train_index <- createDataPartition(df$price, p = 0.7, list = FALSE)
train_data <- df[train_index, ]
remaining_data <- df[-train_index, ]
validation_index <- createDataPartition(remaining_data$price, p = 0.5, list = FALSE)
validation_data <- remaining_data[validation_index, ]
test_data <- remaining_data[-validation_index, ]

# Fit a linear model on the training data
model <- lm(price ~ ., data = train_data)

# Predict on the test data
predictions <- predict(model, newdata = test_data)

# Calculate MAPE
mape_value <- mape(test_data$price, predictions)

# Print the MAPE
print(paste("Out-of-sample MAPE:", 100*round(mape_value, 4)))
```

```
## [1] "Out-of-sample MAPE: 29.48"
```

Decision Trees

```r
library(rpart)

# Fit the decision tree model
dt_model <- rpart(price ~ ., data = train_data)

# Predict on the test set
dt_predictions <- predict(dt_model, newdata = test_data)

# Calculate MAPE for Decision Tree
dt_mape <- mape(test_data$price, dt_predictions)
print(paste("Decision Tree MAPE:", 100*round(dt_mape, 4)))
```

```
## [1] "Decision Tree MAPE: 34.16"
```

```r
library(randomForest)
```

```
## randomForest 4.7-1.1
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'

## The following object is masked from 'package:dplyr':
##
##     combine

## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
# Fit the random forest model
rf_model <- randomForest(price ~ ., data = train_data)

# Predict on the test set
rf_predictions <- predict(rf_model, newdata = test_data)

# Calculate MAPE for Random Forest
rf_mape <- mape(test_data$price, rf_predictions)
print(paste("Random Forest MAPE:", 100*round(rf_mape, 4)))
```

```
## [1] "Random Forest MAPE: 16.93"
```

```
library(xgboost)  # For Gradient Boosting
```

```
## Warning: package 'xgboost' was built under R version 4.3.3

##
## Attaching package: 'xgboost'

## The following object is masked from 'package:dplyr':
##
##     slice
```

```
convert_to_factors <- function(data) {
  cat_vars <- sapply(data, is.character)
  data[cat_vars] <- lapply(data[cat_vars], as.factor)
  return(data)
}

train_data <- convert_to_factors(train_data)
test_data <- convert_to_factors(test_data)

one_hot_encode <- function(df) {
  # Identify character columns
  char_cols <- sapply(df, is.factor)

  # Apply one-hot encoding to character columns
  df_encoded <- df
  for (col in names(df)[char_cols]) {
    # Create one-hot encoded matrix for the column
```

```r
    one_hot <- model.matrix(~ get(col) - 1, data = df)
    # Remove the original character column
    df_encoded[[col]] <- NULL
    # Bind the one-hot encoded columns to the original data frame
    df_encoded <- cbind(df_encoded, one_hot)
  }
  names(df_encoded) <- gsub("get\\(col\\)", "", names(df_encoded))

  return(df_encoded)
}

n_train_rows <- nrow(train_data)
full_dataset <- rbind(train_data, test_data)
dataset_encoded <- one_hot_encode(full_dataset)
train_data_encoded <- dataset_encoded[1:n_train_rows, ] |> dplyr::select(-price)
test_data_encoded <- dataset_encoded[(n_train_rows + 1):nrow(dataset_encoded), ] |> dplyr::select(-pric
train_response <- dataset_encoded[1:n_train_rows, ]$price
test_response <- dataset_encoded[(n_train_rows + 1):nrow(dataset_encoded), ]$price

# Create DMatrix objects
train_matrix <- xgb.DMatrix(data = data.matrix(train_data_encoded), label = train_response)
test_matrix <- xgb.DMatrix(data = data.matrix(test_data_encoded))

# Train the XGBoost model
xgb_model <- xgboost(data = train_matrix,
                     objective = "reg:squarederror",
                     nrounds = 100,
                     verbose = 0)

# Make predictions
xgb_predictions <- predict(xgb_model, test_matrix)

# Calculate MAPE for XGBoost
xgb_mape <- mape(test_data$price, xgb_predictions)
print(paste("XGBoost MAPE:", 100*round(xgb_mape, 4)))
```

```
## [1] "XGBoost MAPE: 16.84"
```

```r
library(e1071)  # For SVM
```

```
## Warning: package 'e1071' was built under R version 4.3.3
```

```r
# Fit the SVM model
svm_model <- svm(price ~ ., data = train_data)

# Predict on the test set
svm_predictions <- predict(svm_model, newdata = test_data)

# Calculate MAPE for SVM
svm_mape <- mape(test_data$price, svm_predictions)
print(paste("SVM MAPE:", 100*round(svm_mape, 4)))
```

```
## [1] "SVM MAPE: 17.3"
```

The MAPE for XGBoost is the lowest among all models, so we will use it for further analysis. First get the datasets ready

```
# Get the train / test / validation sets ready for hyperparameter tuning
train_rows <- nrow(train_data)
test_rows <- nrow(test_data)
full_dataset <- rbind(train_data, test_data, validation_data)
dataset_encoded <- one_hot_encode(full_dataset)
train_data_encoded <- dataset_encoded[1:n_train_rows, ] |> dplyr::select(-price)
test_data_encoded <- dataset_encoded[(n_train_rows + 1):(n_train_rows + test_rows), ] |> dplyr::select(
validation_data_encoded <- dataset_encoded[(n_train_rows + test_rows + 1):nrow(dataset_encoded), ] |> dp
train_response <- dataset_encoded[1:n_train_rows, ]$price
test_response <- dataset_encoded[(n_train_rows + 1):(n_train_rows + test_rows), ]$price
validation_response <- dataset_encoded[(n_train_rows + test_rows + 1):nrow(dataset_encoded), ]$price
train_matrix <- xgb.DMatrix(data = data.matrix(train_data_encoded), label = train_response)
validation_matrix <- xgb.DMatrix(data = data.matrix(validation_data_encoded), label = validation_respons
```

Now we will do hyperparameter tuning for XGBoost

```
# Now that we have the best hyperparameters, we can train the final model and use k-fold cross-validati

set.seed(123)

# Define k-fold cross-validation
k <- 5
folds <- createFolds(train_data$price, k = k)

cv_results <- sapply(folds, function(fold) {
  train_fold <- train_data[-fold, ]
  test_fold <- train_data[fold, ]

  train_fold_matrix <- xgb.DMatrix(data = data.matrix(train_fold[,-which(names(train_fold) == "price")]
  test_fold_matrix <- xgb.DMatrix(data = data.matrix(test_fold[,-which(names(test_fold) == "price")]),

  # Train the model with the best hyperparameters
  final_model <- xgboost(data = train_fold_matrix,
                         nrounds = best_params$nrounds,
                         max_depth = best_params$max_depth,
                         eta = best_params$eta,
                         gamma = best_params$gamma,
                         colsample_bytree = best_params$colsample_bytree,
                         min_child_weight = best_params$min_child_weight,
                         objective = "reg:squarederror",
                         verbose = 0)

  # Predict on the test fold
  test_predictions <- predict(final_model, newdata = test_fold_matrix)

  # Calculate RMSE for the fold
  actual <- test_fold$price
  predicted <- test_predictions
```

```r
  rmse_fold <- sqrt(mean((actual - predicted)^2))
  mape_fold <- mean(abs((actual - predicted) / actual)) * 100
  return(list(rmse = rmse_fold, mape = mape_fold))
})

# Calculate average RMSE, MAPE across folds
cv_results_df <- data.frame(t(cv_results))
avg_rmse <- cv_results_df$rmse |> unlist() |> mean()
avg_mape <- cv_results_df$mape |> unlist() |> mean()

print(paste("Average RMSE across folds:", avg_rmse))
```

```
## [1] "Average RMSE across folds: 749941.718721463"
```

```r
print(paste("Average MAPE across folds:", avg_mape))
```

```
## [1] "Average MAPE across folds: 22.7147439693546"
```

Business Impact of the Analysis: This analysis aims to accurately predict property prices, providing significant value for stakeholders in real estate, including investors, developers, and buyers. An effective model could enhance decision-making by offering data-driven insights into pricing trends and potential property valuations. By integrating a range of predictive features such as property attributes and macroeconomic data, the model can better forecast future prices, thereby reducing uncertainty and helping businesses manage risks and seize investment opportunities more strategically.

Possible Improvements to the Model: Feature Selection: Although stepwise selection was used, exploring other feature selection methods (e.g., LASSO regression) could yield better results by reducing multicollinearity and improving generalization. Model Variety: Incorporating ensemble models like Random Forest or Gradient Boosting (e.g., XGBoost) might enhance performance compared to linear models. Validation Strategy: Implementing k-fold cross-validation instead of a single train/test split could provide a more robust evaluation of model performance and reduce variance in error estimates. Extra data ingestion: Extra data such as property age, proximity to amenities, and local school quality could further enhance the model's predictive power and accuracy.