KING COUNTY
HOUSING PRICE REGRESSION

WONG ZHAO WU

Introduction

The housing dataset contains home sales prices and characteristics for Seattle and King County, WA (May 2014 - 2015) and its corresponding shape file with the zip code zones for King County. Additional geoJSON file which contains the shape of each zip code for King County was retrieved from King county GIS Open Data to facilitate EDA.

Modelling Objectives

Perform EDA and Modelling to find the optimal solution in estimating the housing prices by minimizing **Root Mean Squared Error** as the primary metrics.

KING COUNTY

REAL ESTATE

EDA SUMMARIES

EXTREME OUTLIERS
Extreme outliers are observed for several numerical columns.

DATA SKEWNESS
Data distribution is positively skewed for most numerical columns including price.

MISSING VALUES
Missing Values are observed for yr_renovated (95.7%), sqft_renovated (60.73%).

GEOLOCATION FEATURES
Certain region has higher housing price by visualising the colored map.

WEAK STATIONARY TIME SERIES
Housing Price just fluctuating against selling date.

DATA PREPROCESSING

Dropping Columns

Drop the columns that does not review any relationship with the target variable: Id, Date, Zipcode.

Handling Missing Values

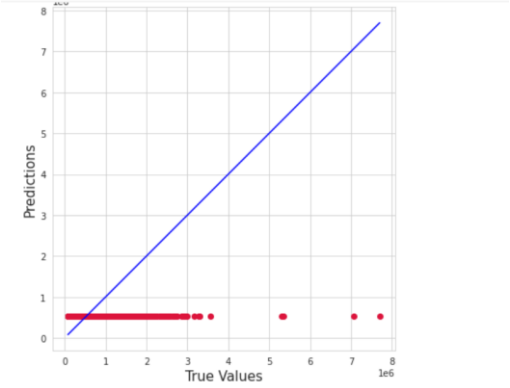
As there are more than around 95% and 60.73% of missing values from "yr_renovated", "sqft_basement", the safest approach is to drop both columns entirely.

Log Transformation

We perform log transformation to unskewed the feature columns that are highly skewed with extreme outliers.

BASELINE MODELLING & MODEL SELECTION

Baseline DummyRegressor



	train_rmse	test_rmse	train_mae	test_mae	train_r2	test_r2
DummyRegressor	360601.34	392133.02	232484.86	236998.42	0.0	-0.0

We make use of dummy regressor as baseline model by always predicting the mean of target variable.
The baseline predictor serves as reference point for model selection.

Baseline Model Selection

100%|██████████| 11/11 [03:30<00:00, 19.13s/it]

	train_rmse	test_rmse	train_mae	test_mae	train_r2	test_r2
LinearRegression	208609.48	238492.38	129734.92	132924.05	0.67	0.63
Lasso	208609.48	238492.01	129733.76	132922.32	0.67	0.63
Ridge	208610.14	238485.70	129725.16	132905.07	0.67	0.63
KNeighborsRegressor	170822.37	238171.55	103394.02	130667.94	0.78	0.63
SVR	356425.83	389582.89	209723.27	216474.33	0.02	0.01
SVR	371169.29	404216.71	220433.41	227348.05	-0.06	-0.06
SVR	371169.29	404216.72	220433.41	227348.04	-0.06	-0.06
SVR	371169.31	404216.73	220433.42	227348.06	-0.06	-0.06
DecisionTreeRegressor	9131.47	176654.86	831.13	98021.70	1.00	0.80
RandomForestRegressor	47679.24	135072.55	25987.06	71090.42	0.98	0.88
GradientBoostingRegressor	116535.90	146011.77	73628.25	81701.81	0.90	0.86

Conclusion

1. Tree-based models can produce promising results after reducing overfitting. Hence, further model tuning is required to reduce overfitting.
2. Linear models suffers from high biases. Hence, we can try to increase the model complexity to reduce overfitting.

Linear Models

- Does not seems to suffers from major overfitting
- Might be suffering from underfitting due to inductive biases.

KNeighborsRegressor

- Suffer from major overfitting.
- Might be suffering from slight underfitting.

SVMs (linear, rbf, poly, sigmoid)

- Changing of Kernels does not seems to improve the performance of SVMs
- SVMs suffers from high biases as the train and test set errors are relatively high

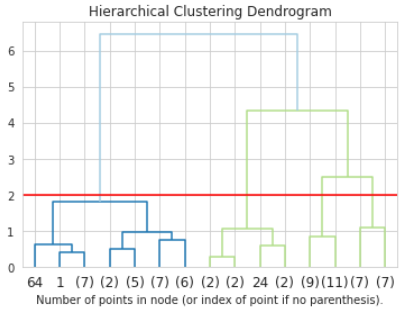
Decision Tree & Ensemble Models

- Suffer as much overfitting
- Low biases and able to obtain relatively lower testing error.

MODEL IMPROVEMENT

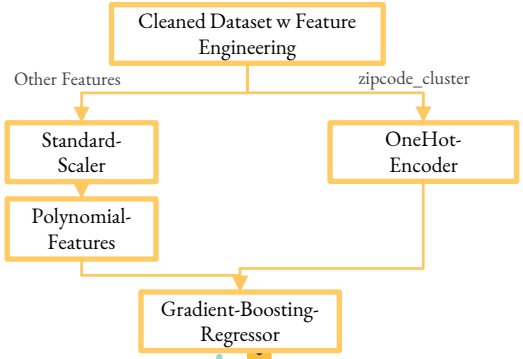
Geo-Spatial Data Feature Engineering

Perform hierarchical clustering based on common housing features grouped by zipcode.

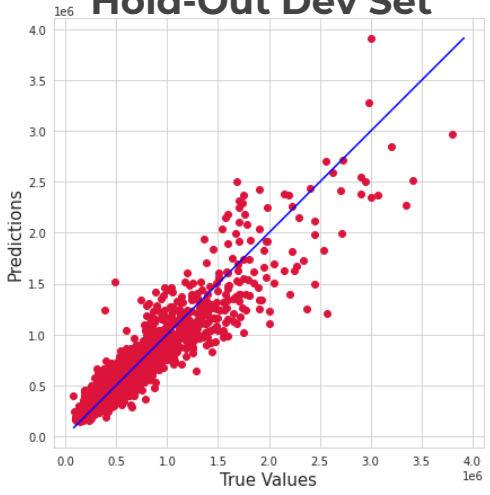


Preprocessing Pipeline

Using pipeline to preprocess the data before parsing it into the final model.



Hold-Out Dev Set



	train_rmse	test_rmse	train_mae	test_mae	train_r2	test_r2
Pipeline	101288.02	123656.67	66135.63	74465.56	0.92	0.87

	mean	std
Root Mean Square	123158.747507	9997.993595

5-Fold CV

Hyperpa-rams Tuning!

Hyperparameter Tuning & Final Model Evaluation

Model Selection w Pipeline

```
100%|██████████| 6/6 [03:53<00:00, 38.90s/it]
```

	train_rmse	test_rmse	train_mae	test_mae	train_r2	test_r2
DecisionTreeRegressor	6721.04	167194.19	523.77	98863.36	1.00	0.77
RandomForestRegressor	48663.73	123733.37	26045.14	70017.07	0.98	0.87
GradientBoostingRegressor	101288.02	123656.67	66135.63	74465.56	0.92	0.87
BaggingRegressor	58111.96	133378.91	30201.72	75157.08	0.97	0.85
AdaBoostRegressor	286550.68	294621.11	259179.82	263169.51	0.38	0.29
ExtraTreesRegressor	6721.04	120048.43	523.79	68150.41	1.00	0.88

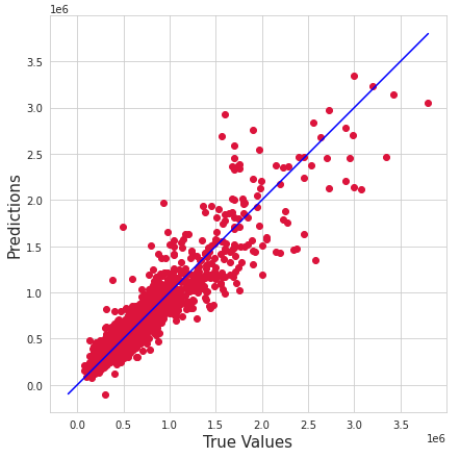
RandomSearchCV, 20 iters

```
pipe.steps.pop() # Remove estimator
pipe.steps.append(('estimator', GradientBoostingRegressor())) # Replace estimator with GradientBoostingRegr
params = {
    "estimator__n_estimators" : np.linspace(100,500,5, dtype = int),
    "estimator__subsample" : [0.2, 0.5, 0.8, 1.0],
    "estimator__min_samples_leaf" : [0.01],
    "estimator__random_state" : [12],
    "estimator__ccp_alpha" : [0, 0.1,0.2,0.3]
}
random_search_cv = RandomizedSearchCV(pipe, params,scoring = 'neg_root_mean_squared_error', n_iter = 20)
random_search_cv.fit(X_train,y_train)
```

```
random_search_cv.best_params_
```

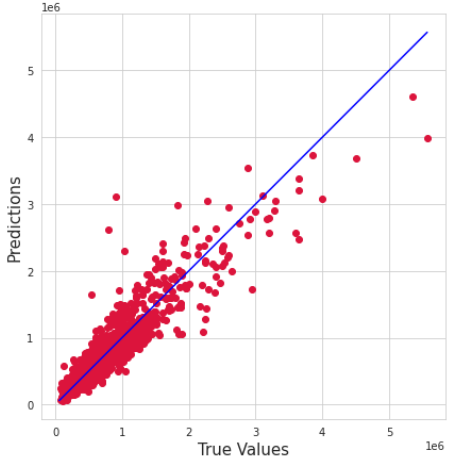
```
{'estimator__n_estimators': 500,
 'estimator__subsample': 1,
 'estimator__min_samples_leaf': 0.01,
 'estimator__random_state': 12,
 'estimator__ccp_alpha': 0.3}
```

Hold-Out Dev Set



	train_rmse	test_rmse	train_mae	test_mae	train_r2	test_r2
Pipeline	98402.78	122980.59	60758.43	71607.46	0.93	0.88

Hold-Out Test Set

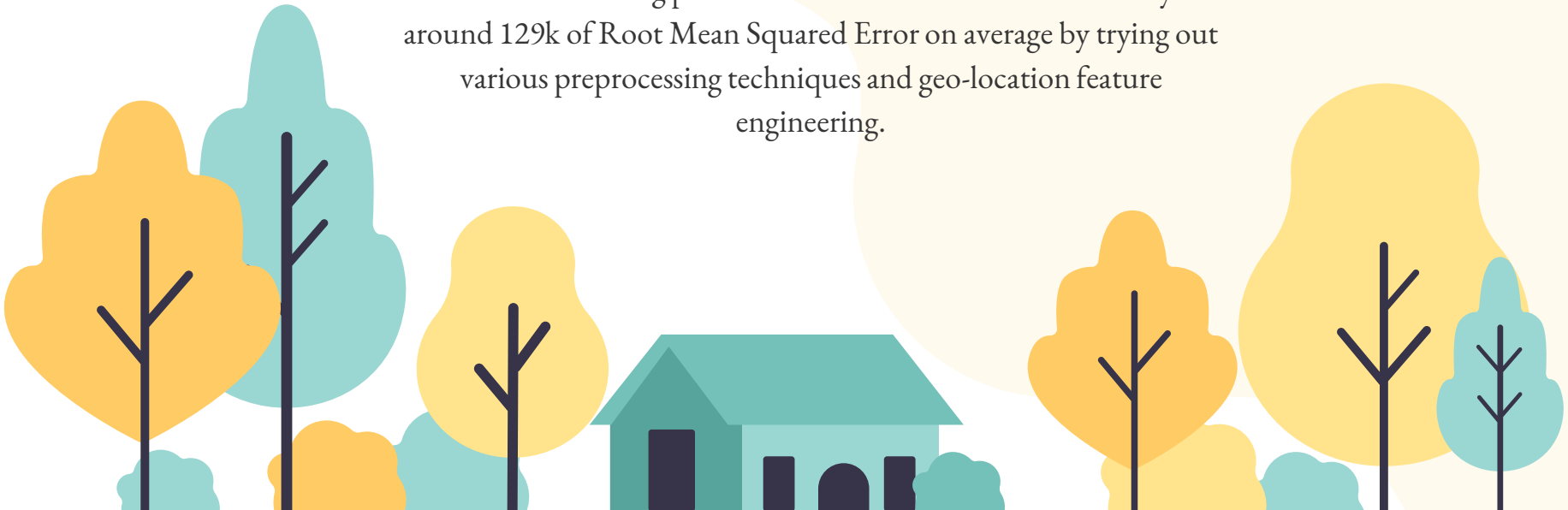


	train_rmse	test_rmse	train_mae	test_mae	train_r2	test_r2
Pipeline	98402.78	136190.03	60758.43	75508.14	0.93	0.88

After evaluating the model with hold-out dev set, we noticed that RandomizedCV hyperparameters tuning have reduced the dev set error of a small margin at the cost of slight increase in variance of our model which can be noticed by the 2% increase of gap between train and test for the final tuned pipeline.
Due to the time constraint, I did not pursue further to try reduce the high variance caused after hyperparameters tuning.(As this is an assignment with deadline)

CONCLUSION

By using GradientBoostingRegressor, we have managed to estimate the housing prices with its attribute down to accuracy of around 129k of Root Mean Squared Error on average by trying out various preprocessing techniques and geo-location feature engineering.



THANKS

Personal Learning Reflection

Through the seattle housing price prediction problem, I've learned more of **Geo-location Feature Engineering** without leaking the actual housing prices as well as grasp a better understanding of the **Bias-Variance trade-off** through countless iterations of redefining the params grid, hyperparameter tuning, model evaluation and again! Initially due to the poor design of parameter searching grid, which results in the resulting model being more overfitted than the default parameter, despite the drop in test error. By doing more research and read-up on the Gradient Boosting, I've identified several key hyperparameters that can improve the performance without increasing the variance as much. I've also decided to make use of **AWS Sagemaker** to host and run the entire experiment to speed up the experimenting iteration for this project.

Written By : Wong Zhao Wu
Last Modified : 26 May 2021