EXPLORATORY DATA ANALYSIS AND BUILDING PRICE PREDICTIVE MODEL

airbnb

Presented by :

Aaman Khan

Megha Unni

Pratik Datta

Soumyaneel Mandal

ABADS 8A (Group – 1)

https://github.com/pratikdatta90/PythonML

INTRODUCTION: REASON BEHIND THE WORK

- In today's fast-paced world, the way we travel and seek accommodations has undergone a remarkable transformation, thanks to platforms like Airbnb.
- This dynamic marketplace has empowered property owners and travelers, offering a diverse range of lodging options.
- However, one enduring challenge is setting the right price for a listing.
- Hosts aspire to optimize their earnings while ensuring competitive pricing, while guests seek value for their money.
- Balancing these interests can be intricate, and that's where the motivation for Airbnb price prediction comes in.

MOTIVATION: CONTRIBUTION OF THE WORK

- To harness the power of data science and machine learning to provide more accurate and data-driven pricing strategies for Airbnb hosts and guests.
- By developing predictive models that factor in myriad variables such as location, property type, and market dynamics.
- The objective is to help hosts maximize their income and guests find fair deals.
- In this exploration of Airbnb price prediction, we will delve into methodologies, data sources, and emerging trends.
- Shedding light on how technology is enhancing the overall Airbnb experience for both hosts and travelers.

OBJECTIVES: FINDING THE RIGHT PRICE

The task is to generate an ML based solution that can be used to suggest appropriate listing prices to the property owner when they try to list a property out for rent.

Steps undertaken to reach the final goal of the project:

- Data Understanding and feature creation
- Data Quality and checks
- Variable profiling and checking relationships between variables
- Modelling and insights

TOOLS USED: PYTHON(JUPYTER NOTEBOOK)

Basic packages used for data cleaning, merging and preliminary visualization:

- NumPy (For exploratory data analysis)
- Pandas
- Matplotlib (For Visualization)
- Seaborn

Packages used for building data model using the predictor variable in the data:

- scikit-learn (used for encoding, and performing linear and random forest regression)
- XGBoost (used for XGBoost Regression)

CHALLENGES: DATA PREPARATION WORKS

Basic Challenges faced with the raw data and measures taken to overcome those:

- Missing value inspection in the target variables and imputing appropriately
- Changing the date format to datetime in order to perform further analysis
- Renaming columns to reflect appropriate field name
- Removing field from different source files that may not be required building model
- Joining and merging different data sources to a single data frame for analysis
- Using the seaborn and matplotlib libraries basic exploratory data analysis (EDA)
- Segregating the numeric and categorical fields before performing the encoding
- Using the sci-kit learn package to perform encoding (Label Encoder from preprocessing)

CLEANING AND MERGING OF THE DATA

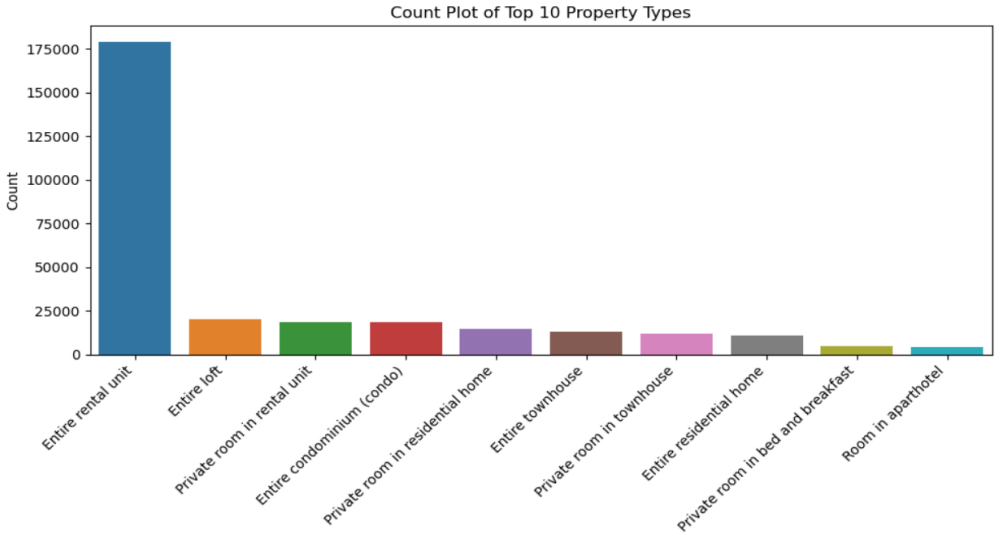
	listing_id	property_type	room_type	accomodates	bathrooms	bedrooms	beds	host_id	date	available	price	host_since
0	50904	Room in boutique hotel	Hotel room	2	1 private bath	1	1	234077	2022-06-23	1	165.0	2010-09-14
1	50904	Room in boutique hotel	Hotel room	2	1 private bath	1	1	234077	2022-01-16	1	150.0	2010-09-14
2	50904	Room in boutique hotel	Hotel room	2	1 private bath	1	1	234077	2022-05-17	1	165.0	2010-09-14
3	50904	Room in boutique hotel	Hotel room	2	1 private bath	1	1	234077	2022-05-31	1	165.0	2010-09-14
4	50904	Room in boutique hotel	Hotel room	2	1 private bath	1	1	234077	2022-10-20	0	165.0	2010-09-14
...	...	...	...	...	...	...	...	...	...	...	...	...
319187	53983318	Entire rental unit	Entire home/apt	3	1.5 baths	1	1	437309332	2022-09-09	1	150.0	2021-12-24
319188	53983318	Entire rental unit	Entire home/apt	3	1.5 baths	1	1	437309332	2022-08-27	1	150.0	2021-12-24
319189	53983318	Entire rental unit	Entire home/apt	3	1.5 baths	1	1	437309332	2022-03-22	1	150.0	2021-12-24
319190	53983318	Entire rental unit	Entire home/apt	3	1.5 baths	1	1	437309332	2022-07-30	1	150.0	2021-12-24
319191	53983318	Entire rental unit	Entire home/apt	3	1.5 baths	1	1	437309332	2022-02-27	1	150.0	2021-12-24

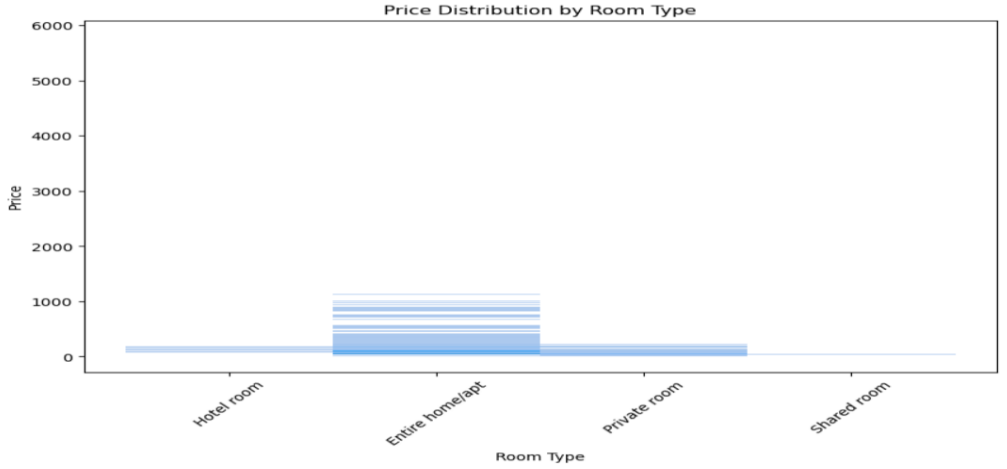
DATA QUALITY CHECKS AND OVERVIEW

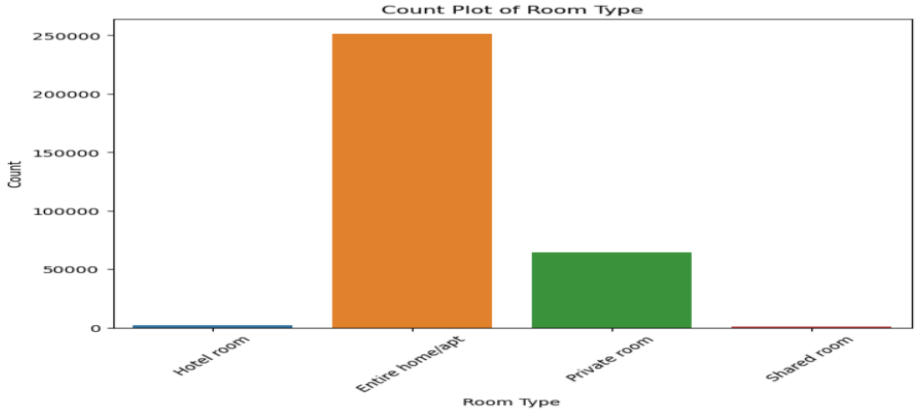
	listing_id	accomodates	bedrooms	beds	host_id	available	price
count	3.191920e+05	319192.000000	319192.000000	319192.000000	3.191920e+05	319192.000000	319192.000000
mean	3.488528e+07	3.762619	1.490285	2.217599	1.418691e+08	0.535192	109.917779
std	1.523257e+07	2.771459	1.065670	2.222932	1.287545e+08	0.498761	185.769339
min	5.090400e+04	1.000000	1.000000	1.000000	2.340770e+05	0.000000	13.000000
25%	2.338661e+07	2.000000	1.000000	1.000000	2.875771e+07	0.000000	59.000000
50%	3.891969e+07	3.000000	1.000000	2.000000	1.033579e+08	1.000000	79.000000
75%	4.839174e+07	4.000000	2.000000	3.000000	2.354916e+08	1.000000	115.000000
max	5.398332e+07	16.000000	20.000000	44.000000	4.373093e+08	1.000000	5800.000000

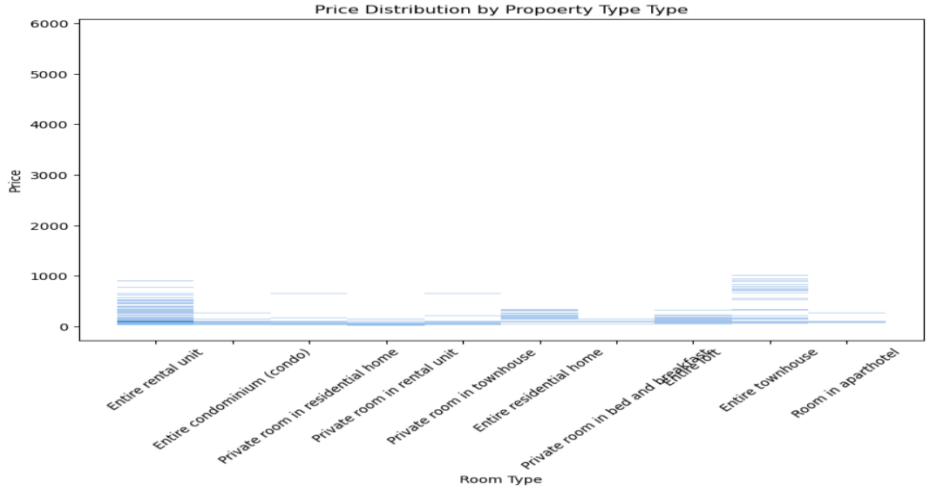
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 319192 entries, 0 to 319191
Data columns (total 12 columns):
 #   Column         Non-Null Count   Dtype
---  ------         --------------   -----
 0   listing_id     319192 non-null  int64
 1   property_type  319192 non-null  object
 2   room_type      319192 non-null  object
 3   accomodates    319192 non-null  int64
 4   bathrooms      319192 non-null  object
 5   bedrooms       319192 non-null  int32
 6   beds           319192 non-null  int32
 7   host_id        319192 non-null  int64
 8   date           319192 non-null  datetime64[ns]
 9   available      319192 non-null  int64
 10  price          319192 non-null  float64
 11  host_since     319192 non-null  datetime64[ns]
dtypes: datetime64[ns](2), float64(1), int32(2), int64(4), object(3)
memory usage: 29.2+ MB
```

EDA: PLOT. PRELIMINARY VISUALIZATION

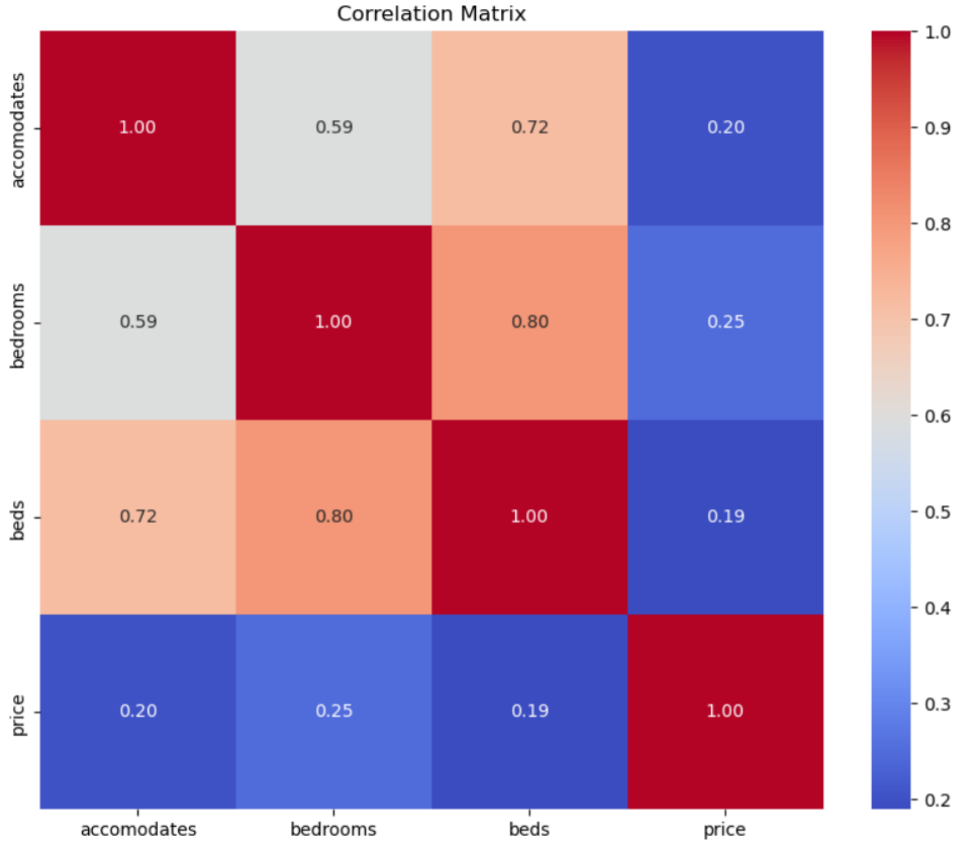


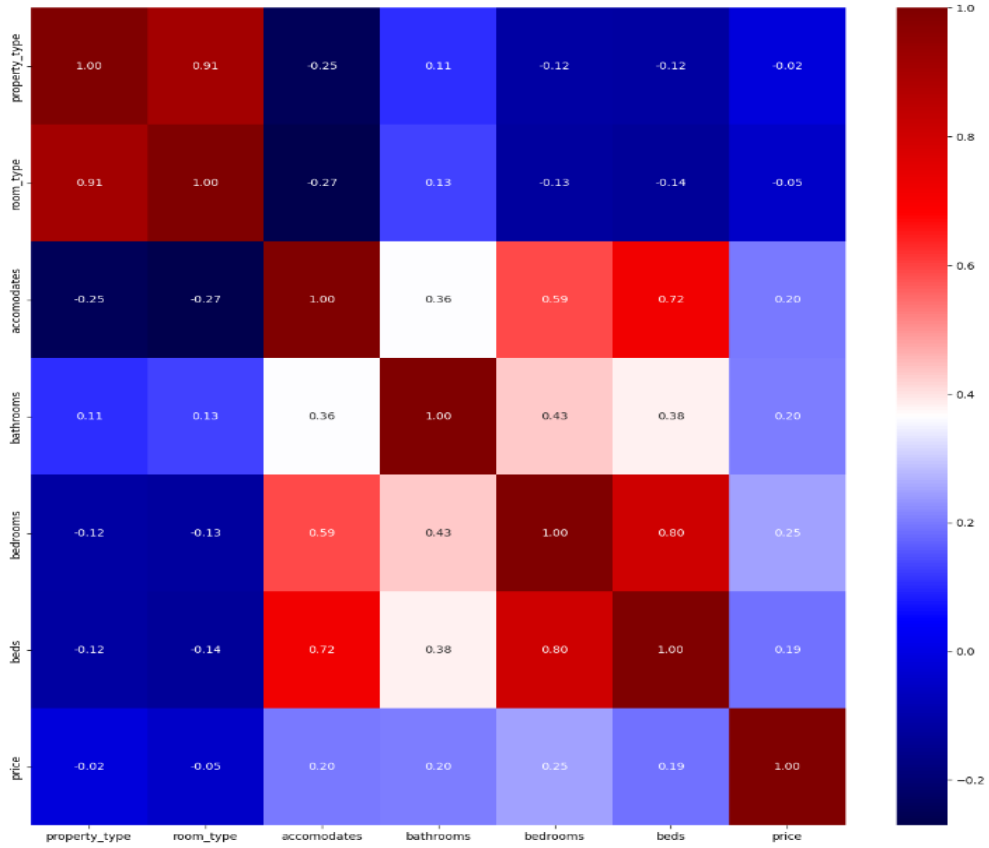




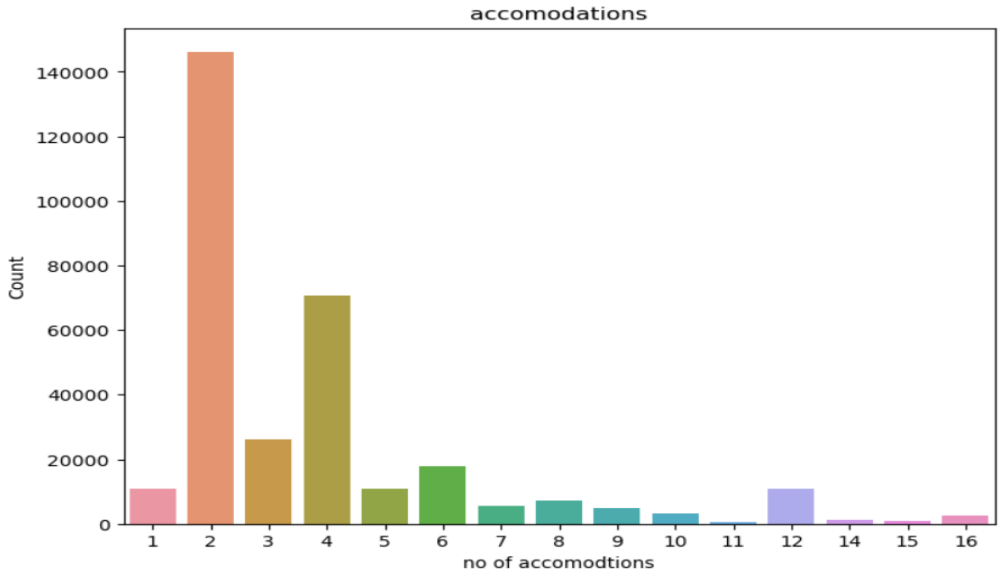


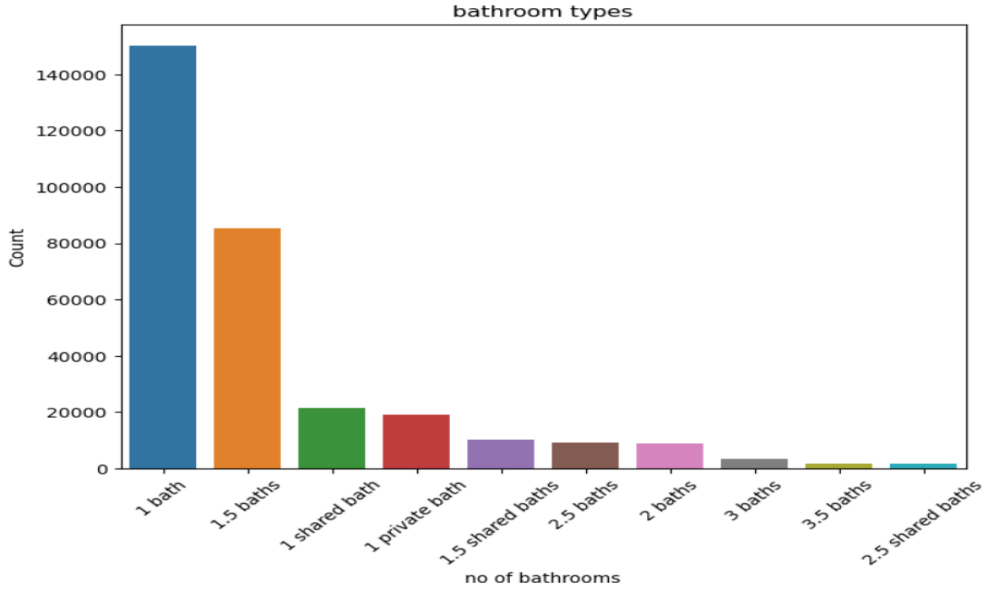
STUDYING THE INTER - COLLINEARITY

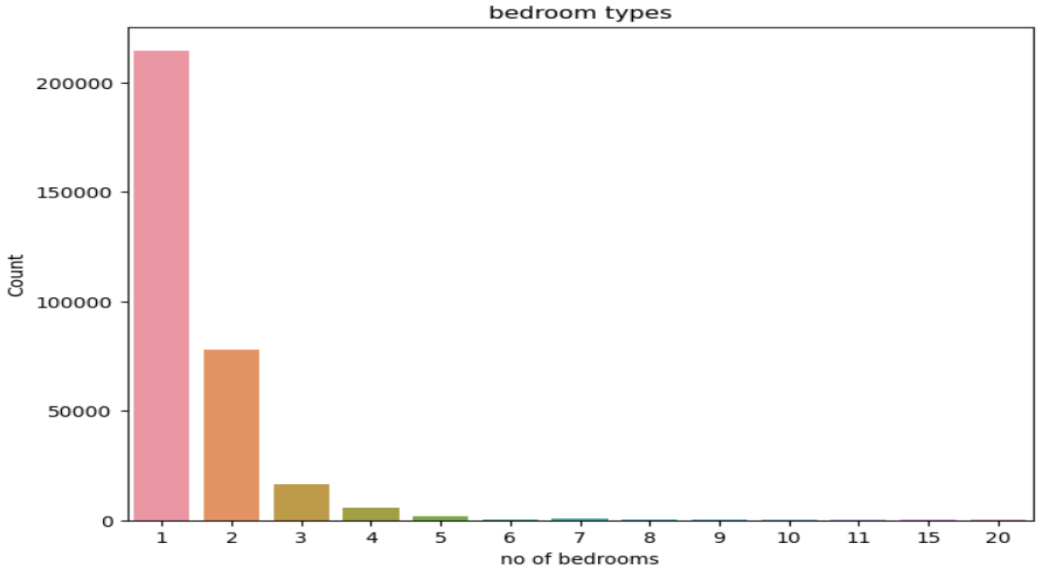


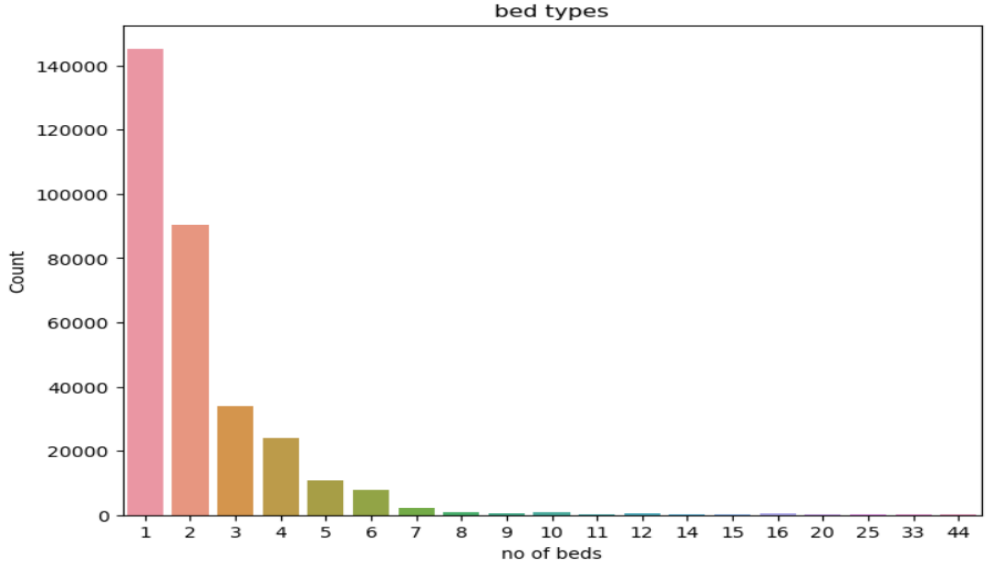


DISTRIBUTION OF THE PREDICTOR VARIABLE

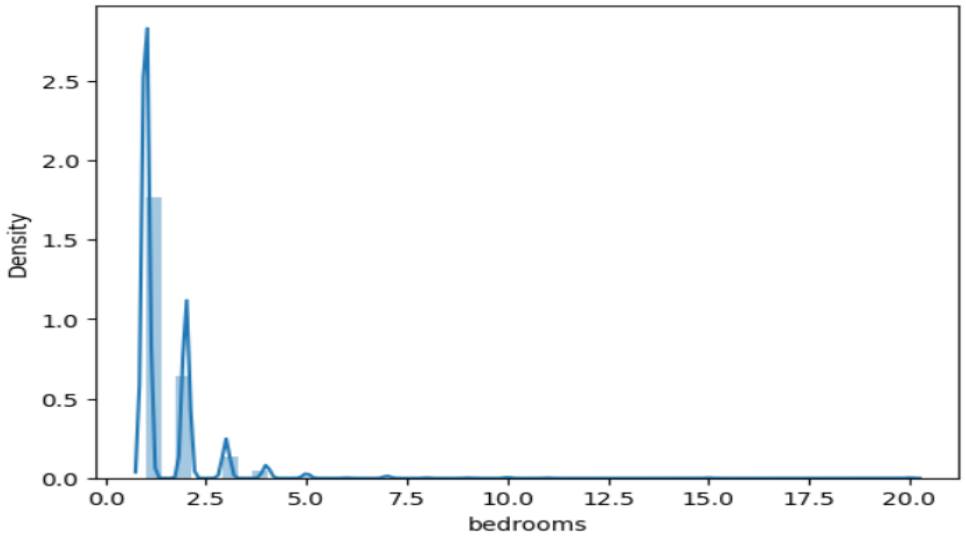


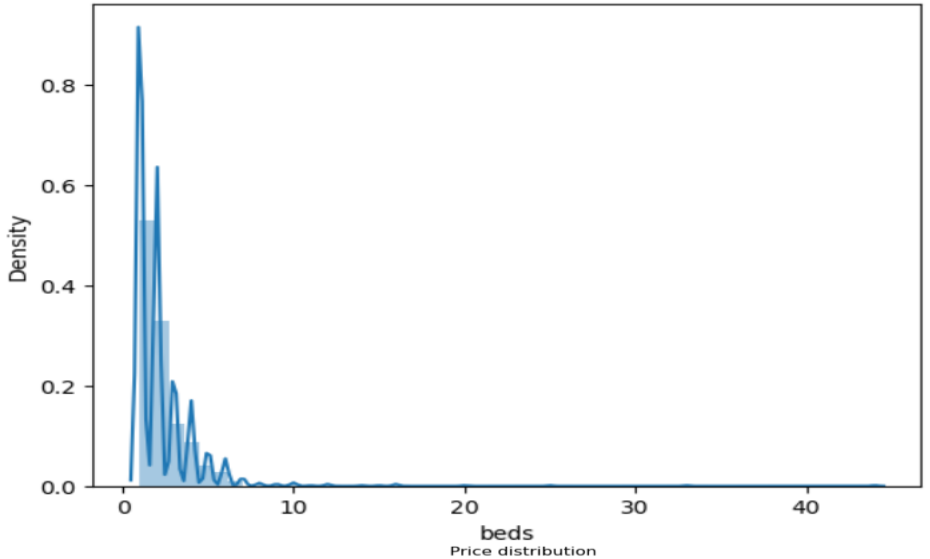


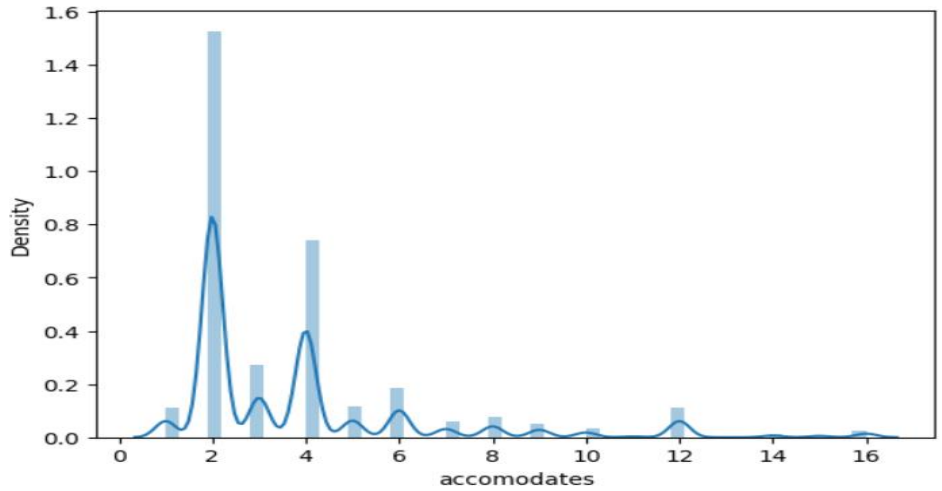


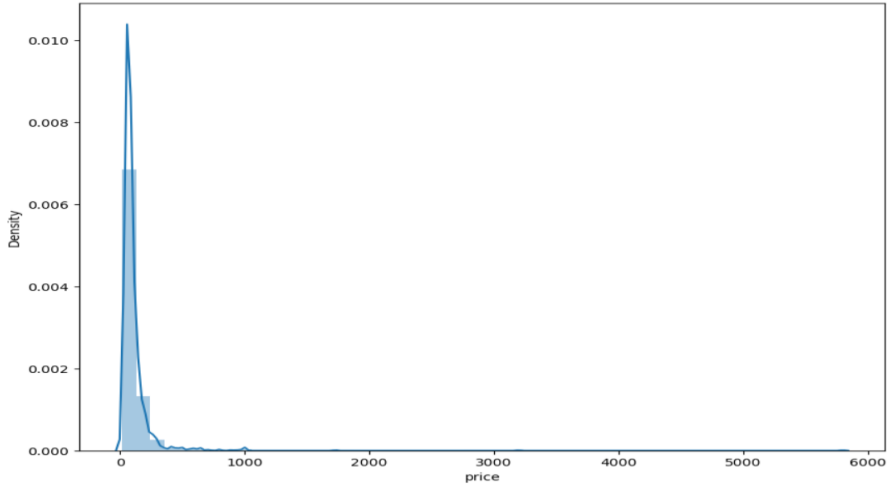


DISTRIBUTION OF THE TARGET VARIABLE





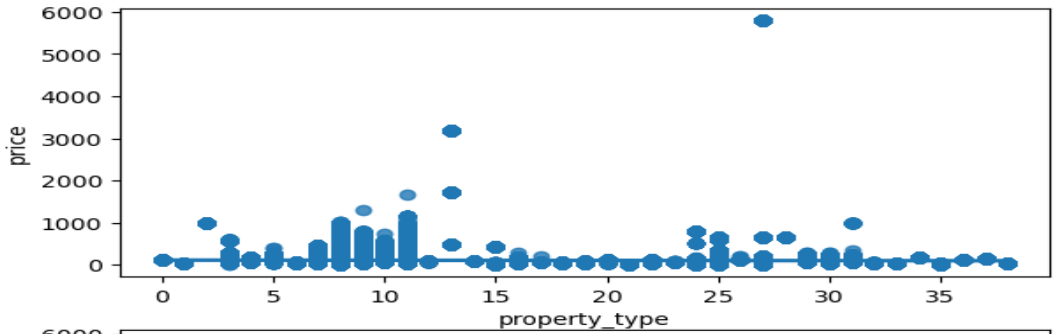


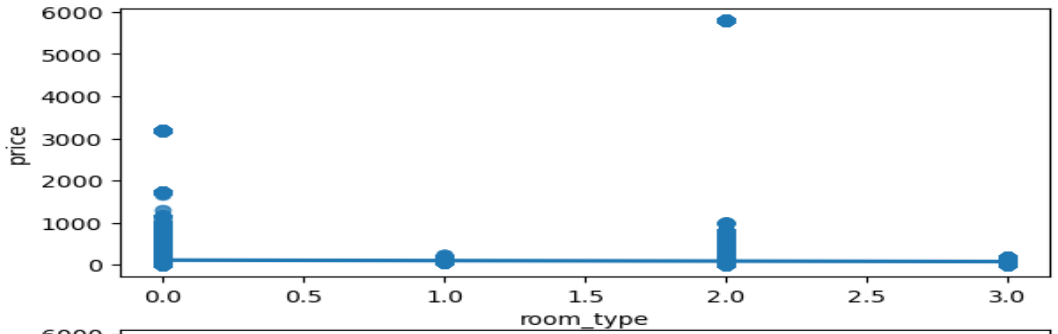


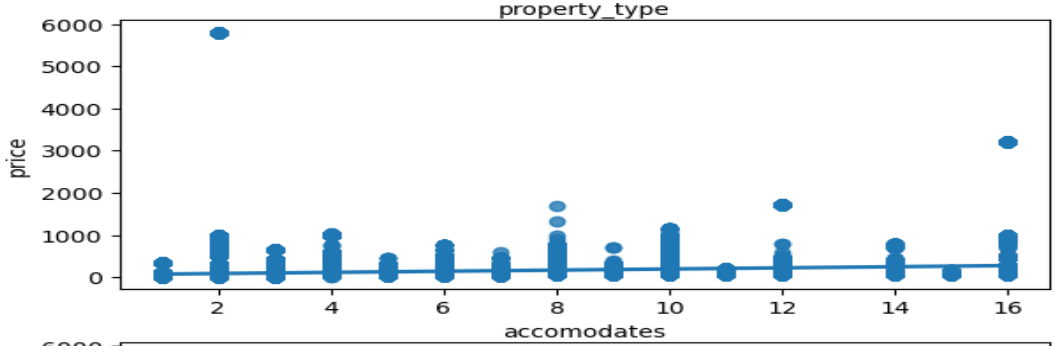
ENCODING THE PREDICTOR VARIABLES

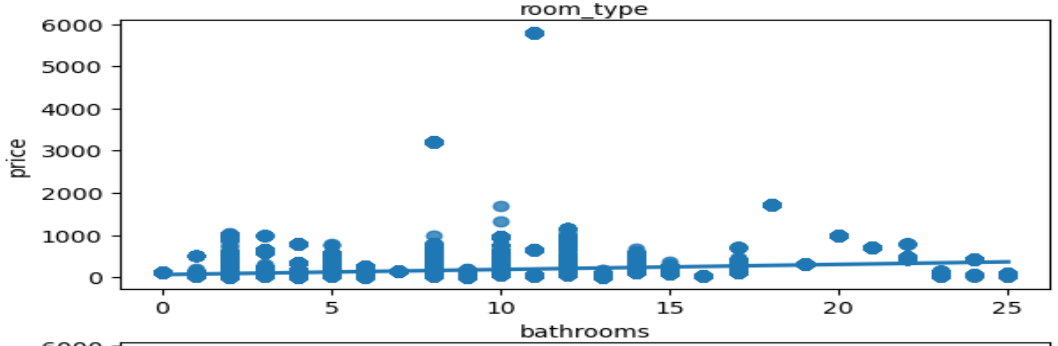
	property_type	room_type	accomodates	bathrooms	bedrooms	beds	price
0	30	1	2	3	1	1	165.0
1	30	1	2	3	1	1	150.0
2	30	1	2	3	1	1	165.0
3	30	1	2	3	1	1	165.0
4	30	1	2	3	1	1	165.0
...	...	...	...	...	...	...	...
319187	8	0	3	5	1	1	150.0
319188	8	0	3	5	1	1	150.0
319189	8	0	3	5	1	1	150.0
319190	8	0	3	5	1	1	150.0
319191	8	0	3	5	1	1	150.0

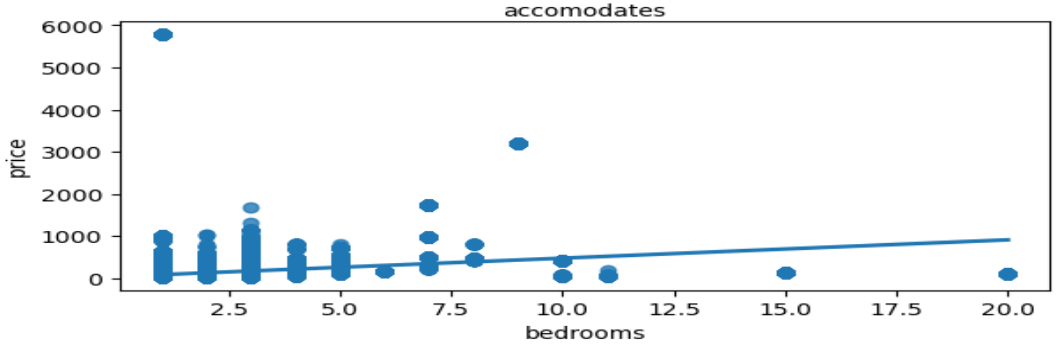
REGRESSION ANALYSIS BETWEEN VARIABLES

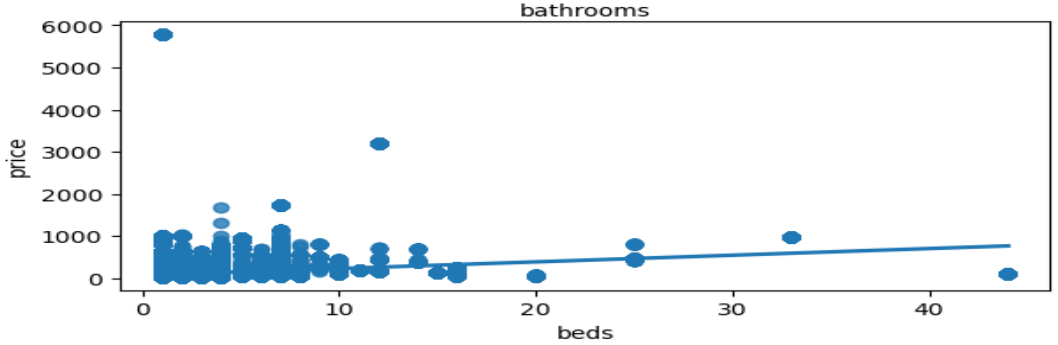












FINDING THE RIGHT MODEL TO PREDICT

- **Not approaching a classification or clustering model as aim is to predict price**

- **Experimenting with linear regression techniques : Multiple Linear Model**
- Identifying the predictor variables by discarding remaining to predict price
- Initially trying with simple linear regression by choosing single predictor variable

- **Experimenting with Random forest Regressor Model and XGBoost Regressor:**
- The idea is to generate prediction using different models and the compare accuracy
- Deciding the final model will be based on the metrics: **MSE, MAE, R2, Adj R2 etc.**

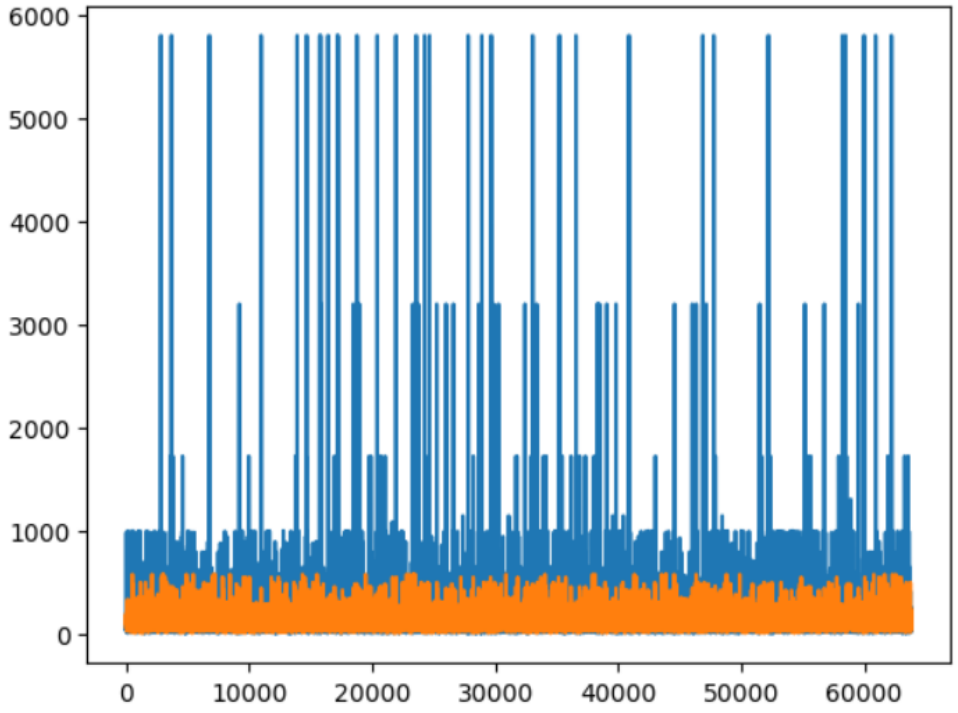
APPROACHES UNDERTAKEN TO IMPLEMENT

- After removing the irrelevant features and subsequently encoding the data, the next step is to experiment with different models.
- The predictor variables are property type, room type, accommodates, bathrooms, bedrooms and beds. The target variable is nothing but the price.
- Using the train_test_split feature of sci-kit learn's model selection, the data is segregated between the training and testing data
- The ratio between training and testing data is kept at a standard 80:20 ratio.
- The same ratio is kept while trying and testing between different models like the linear regression, random forest and XGBoost regressor.
- In the final stage, data is fitted in different models & predicted the target variables.
- Finally, the test results are found and subsequently compared between each other.

EXPERIMENTING WITH LINEAR REGRESSION

```
mae_lr = metrics.mean_absolute_error(y_test, y_pred_lr)
mse_lr = metrics.mean_squared_error(y_test, y_pred_lr)
rmse_lr = np.sqrt(metrics.mean_squared_error(y_test, y_pred_lr))
r2_lr = metrics.r2_score(y_test, y_pred_lr)


print('\nMean Absolute Error of Linear Regression     : ', mae_lr)
print('\nMean Squarred Error of Linear Regression     : ', mse_lr)
print('\nRoot Mean Squarred Error of Linear Regression: ', rmse_lr)
print('\nR2 Score of Linear Regression                : ', r2_lr)
```

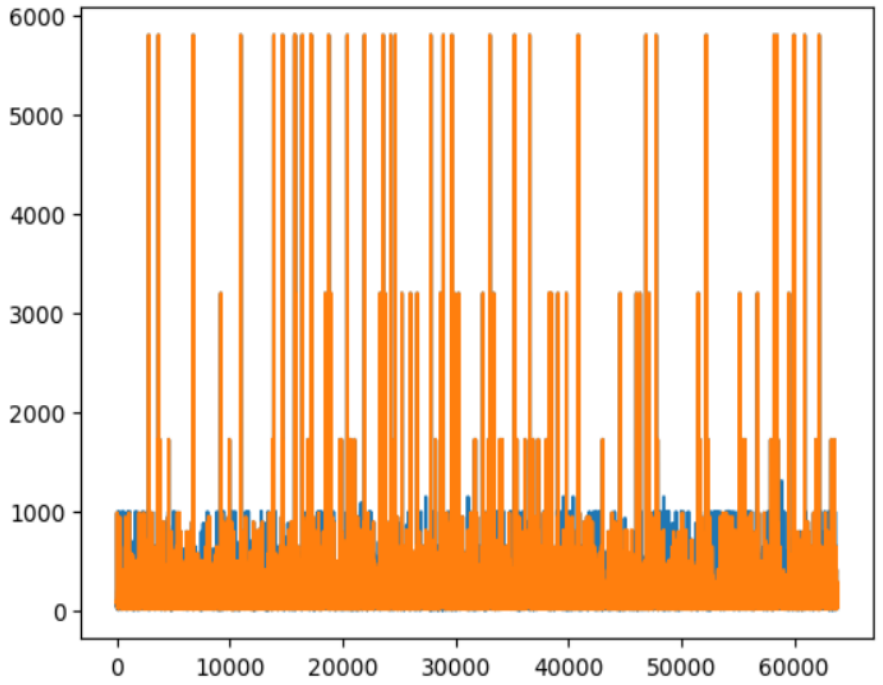
```
Mean Absolute Error of Linear Regression     :  56.35177899960983

Mean Squarred Error of Linear Regression     :  29617.176754340766

Root Mean Squarred Error of Linear Regression:  172.09641702935238

R2 Score of Linear Regression                :  0.08675436600826791
```



EXPERIMENTING WITH RANDOM FOREST

```
mae_rf = metrics.mean_absolute_error(y_test, y_pred_rf)
mse_rf = metrics.mean_squared_error(y_test, y_pred_rf)
rmse_rf = np.sqrt(metrics.mean_squared_error(y_test, y_pred_rf))
r2_rf = metrics.r2_score(y_test, y_pred_rf)


print('\nMean Absolute Error of Random Forest Regressor     : ', mae_rf)
print('\nMean Squarred Error of Random Forest Regressor     : ', mse_rf)
print('\nRoot Mean Squarred Error of Random Forest Regressor: ', rmse_rf)
print('\nR2 Score of Random Forest Regressor                : ', r2_rf)
```

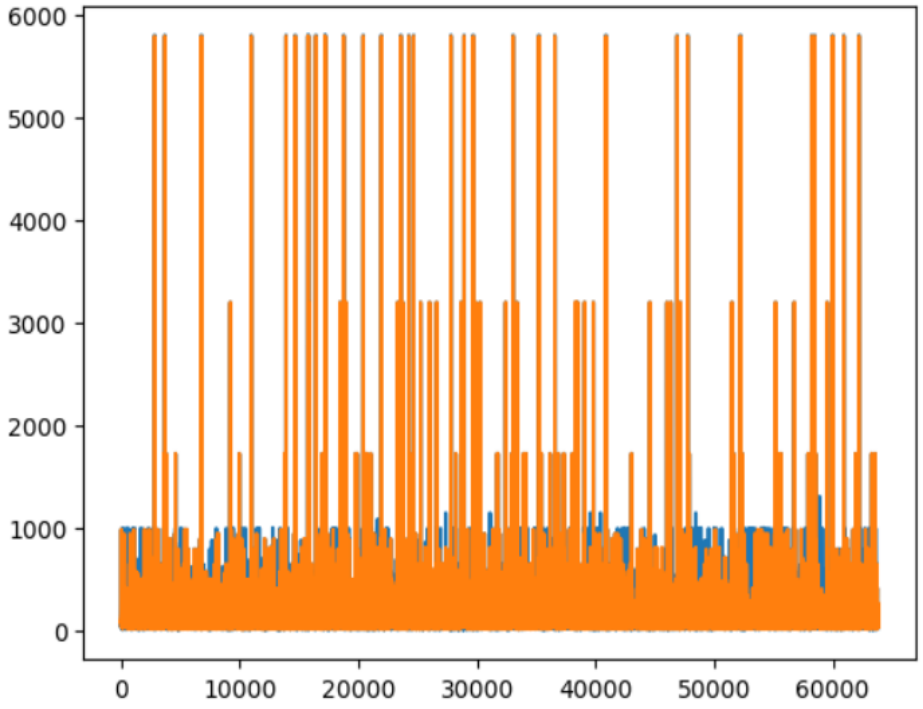
```
Mean Absolute Error of Random Forest Regressor     :  28.61770173221219

Mean Squarred Error of Random Forest Regressor     :  4258.9450301147135

Root Mean Squarred Error of Random Forest Regressor:  65.26059324059744

R2 Score of Random Forest Regressor                :  0.8686754316110499
```



EXPERIMENTING WITH XG BOOST MODEL

```
mae_xgb  = metrics.mean_absolute_error(y_test, y_pred_xgb)
mse_xgb  = metrics.mean_squared_error(y_test, y_pred_xgb)
rmse_xgb = np.sqrt(metrics.mean_squared_error(y_test, y_pred_xgb))
r2_xgb   = metrics.r2_score(y_test, y_pred_xgb)


print('\nMean Absolute Error of XGBoost Regressor     : ', mae_xgb)
print('\nMean Squarred Error of XGBoost Regressor     : ', mse_xgb)
print('\nRoot Mean Squarred Error of XGBoost Regressor: ', rmse_xgb)
print('\nR2 Score of XGBoost Regressor                : ', r2_xgb)
```

```
Mean Absolute Error of XGBoost Regressor     :  29.3864331487217

Mean Squarred Error of XGBoost Regressor     :  4278.851604455569

Root Mean Squarred Error of XGBoost Regressor:  65.41293147731241

R2 Score of XGBoost Regressor                :  0.8680616123987965
```



CONCLUSION: CHOOSING THE RIGHT MODEL

- As seen previously, the linear regression model did not come up with good results. As the accuracy score is less than 10%.
- The model cannot be used for predicting listing prices. Hence some other models were tried and tested.
- With both the Random Forest regressor and XGBoost regressor, the results were much more satisfactory as the R2 score obtained were close to 90% meaning the models can be used to predict the listing prices of Airbnb located in Antwerp.
- It can be concluded that Random forest Regression Model is a good choice to predict the target variable price of listings. Not selecting the RGBoost is mainly due to the complexity.
- Some more models could have been tested but not approached due to lack of time.