Machine Learning for Developers (CIT3C15)

Assignment 2

Joyce Teng Min Li

1907675A

P01

Introduction

- **Topic** : Airbnb Price estimator
- **Problem Statement** : Very often, when someone decides to rent out his apartment and list it on the Airbnb they'd be wondering how much to price it at this is one common question that hosts face especially new host.
- Therefore, this machine learning model is catered for Hosts(especially new host) to predict a price estimation for their AirBnB listing based on some features like city, property type etc.

Data Exploration
Dataset

[2]:

	id	listing_url	scrape_id	last_scraped	name	summary	space	description	experiences_offered	neighborhood_overview	...	instant_bookable	is_business_travel_ready
0	2595	https://www.airbnb.com/rooms/2595	20190806030549	43684	Skylit Midtown Castle	Find your romantic getaway to this beautiful, ...	- Spacious (500+ft²), immaculate and nicely fu...	Find your romantic getaway to this beautiful. ...	none	Centrally located in the heart of Manhattan ju...	...	f	
1	3647	https://www.airbnb.com/rooms/3647	20190806030549	43683	THE VILLAGE OF HARLEM....NEW YORK !	NaN	WELCOME TO OUR INTERNATIONAL URBAN COMMUNITY T...	WELCOME TO OUR INTERNATIONAL URBAN COMMUNITY T...	none	NaN	...	f	
2	3831	https://www.airbnb.com/rooms/3831	20190806030549	43683	Cozy Entire Floor of Brownstone	Urban retreat: enjoy 500 s.f. floor in 1899 br...	Greetings! We own a double-duplex brownst...	Urban retreat: enjoy 500 s.f. floor in 1899 br...	none	Just the right mix of urban center and local n...	...	f	
3	5022	https://www.airbnb.com/rooms/5022	20190806030549	43683	Entire Apt: Spacious Studio/Loft by central park	NaN	Loft apartment with high ceiling and wood floo...	Loft apartment with high ceiling and wood floo...	none	NaN	...	f	
4	5099	https://www.airbnb.com/rooms/5099	20190806030549	43683	Large Cozy 1 BR Apartment In Midtown East	My large 1 bedroom apartment is true New York ...	I have a large 1 bedroom apartment centrally l...	My large 1 bedroom apartment is true New York ...	none	My neighborhood in Midtown East is called Murr...	...	f	

5 rows × 106 columns

The original dataset had 106 columns

[3]:
```
#columns of dataset
dataset_columns = list(df.columns)
dataset_columns
```

[3]:
```
['id',
 'listing_url',
 'scrape_id',
 'last_scraped',
 'name',
 'summary',
 'space',
 'description',
 'experiences_offered',
 'neighborhood_overview',
 'notes',
 'transit',
 'access',
 'interaction',
 'house_rules',
 'thumbnail_url',
 'medium_url',
 'picture_url',
 'xl_picture_url',
 'host_id',
 'host_url',
 'host_name',
 'host_since',
 'host_location',
 'host_about',
 'host_response_time',
 'host_response_rate',
 'host_acceptance_rate',
 'host_is_superhost',
 'host_thumbnail_url',
 'host_picture_url',
 'host_neighbourhood',
 'host_listings_count',
 'host_total_listings_count',
 'host_verifications',
 'host_has_profile_pic',
 'host_identity_verified',
 'street',
 'neighbourhood',
 'neighbourhood_cleansed',
 'neighbourhood_group_cleansed',
```

Using the list() to look at all 106 columns and delete all the unnecessary columns

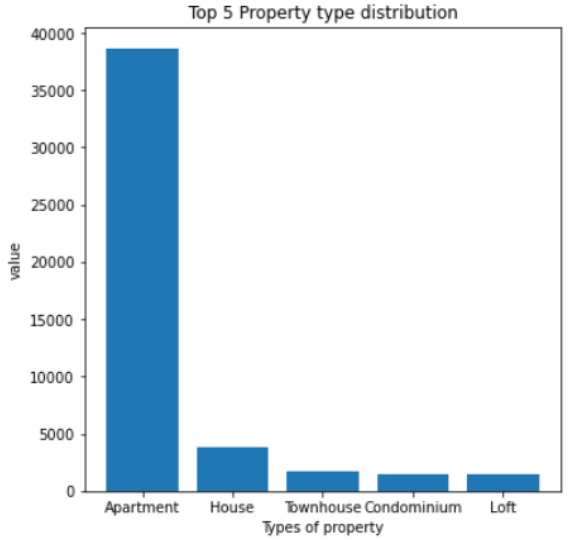
[6]:
```
df.info()
df.shape
```

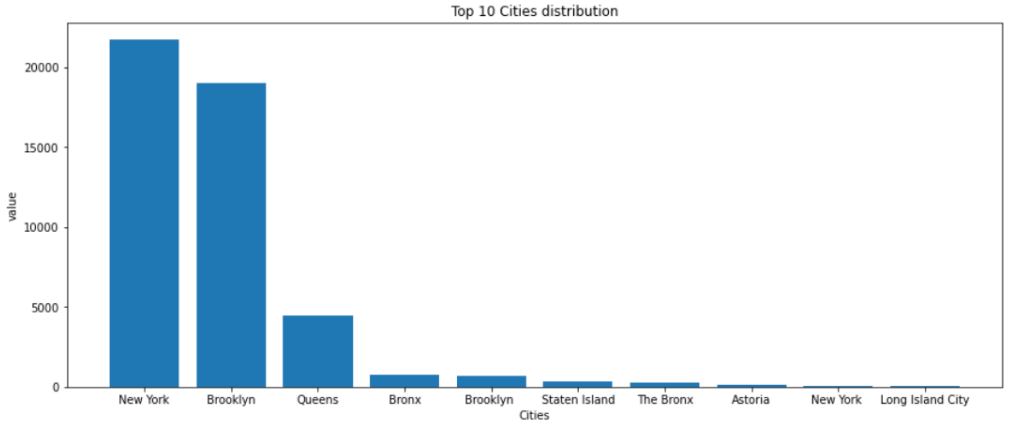
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 48864 entries, 0 to 48863
Data columns (total 23 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   id                           48864 non-null  int64
 1   neighbourhood_group_cleansed 48864 non-null  object
 2   city                         48802 non-null  object
 3   zipcode                      48349 non-null  object
 4   latitude                     48864 non-null  float64
 5   longitude                    48864 non-null  float64
 6   property_type                48864 non-null  object
 7   room_type                    48864 non-null  object
 8   accommodates                 48864 non-null  int64
 9   bathrooms                    48808 non-null  float64
 10  bedrooms                     48837 non-null  float64
 11  beds                         48822 non-null  float64
 12  bed_type                     48864 non-null  object
 13  square_feet                  395 non-null    float64
 14  price                        48864 non-null  int64
 15  guests_included              48864 non-null  int64
 16  availability_365             48864 non-null  int64
 17  number_of_reviews            48864 non-null  int64
 18  first_review                 38733 non-null  float64
 19  last_review                  38733 non-null  float64
 20  review_scores_rating         37760 non-null  float64
 21  review_scores_accuracy       37722 non-null  float64
 22  cancellation_policy          48863 non-null  object
dtypes: float64(10), int64(6), object(7)
memory usage: 8.6+ MB
```

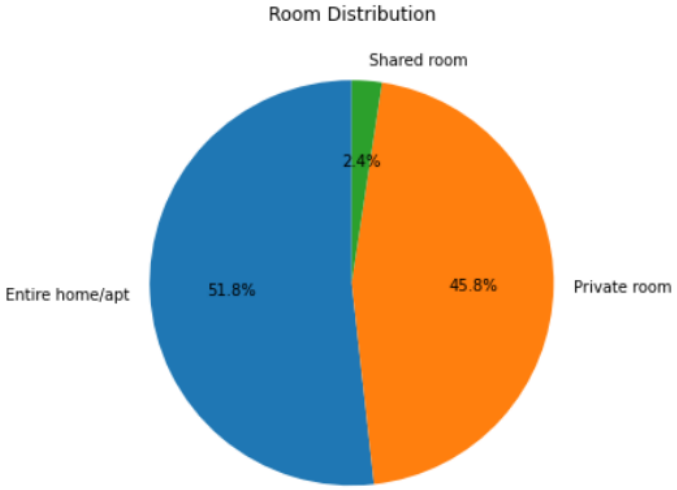
[6]: (48864, 23)

23 columns left after deleting

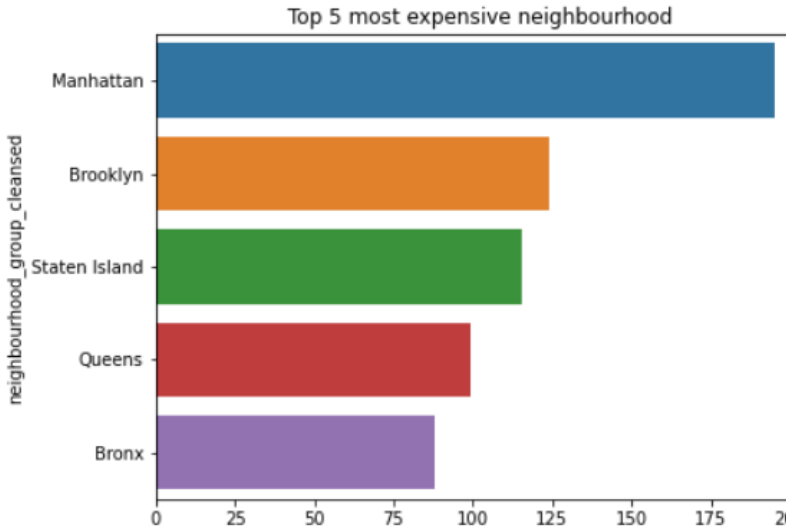
Data Exploration
Features that attributes to price







Data Exploration
Price Relations to other columns

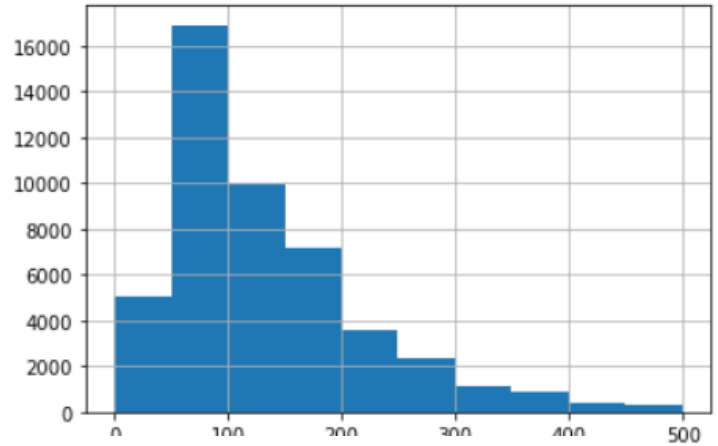


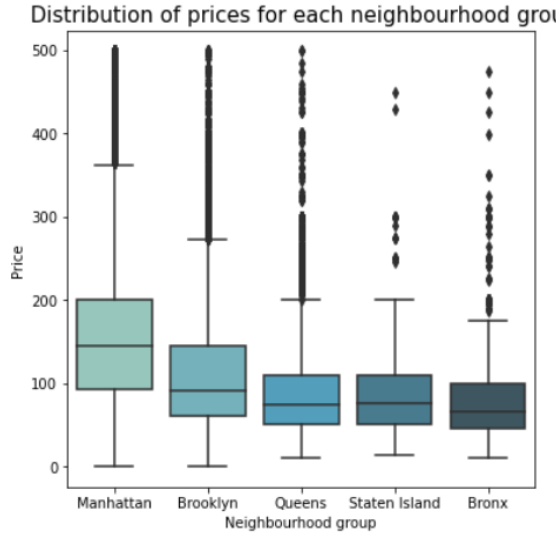

```
# use the desribe() to find the mean/avg price in a listing
df.price.describe()


#based on the price avg price per night is 151.453
```

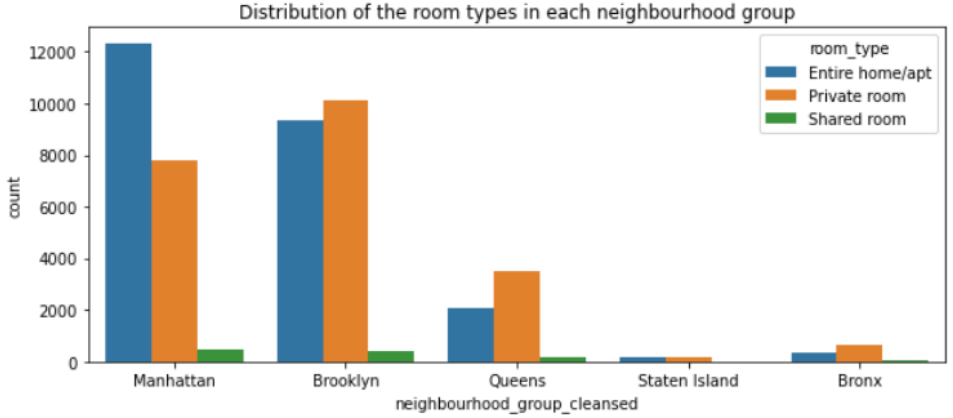
```
count    48864.000000
mean       151.453176
std        236.585525
min          0.000000
25%         69.000000
50%        105.000000
75%        175.000000
max      10000.000000
Name: price, dtype: float64
```

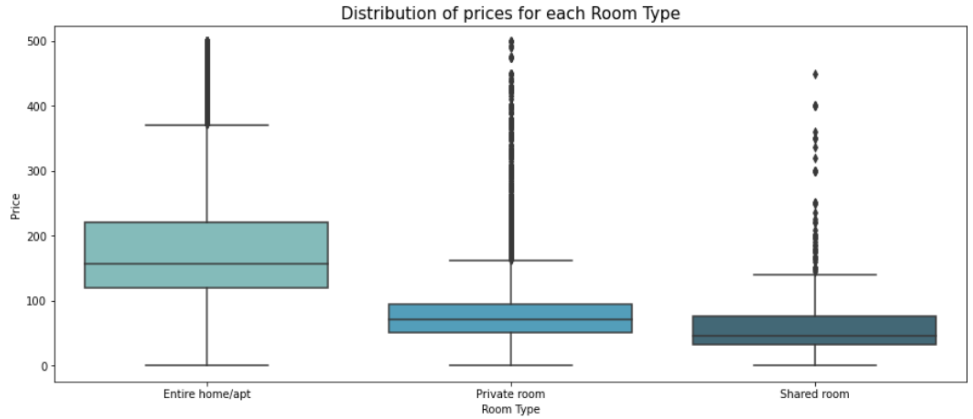
```
[12]: #histogram plot for prices less than $500
hist_price=df['price'][df['price'].values<500].hist()
```

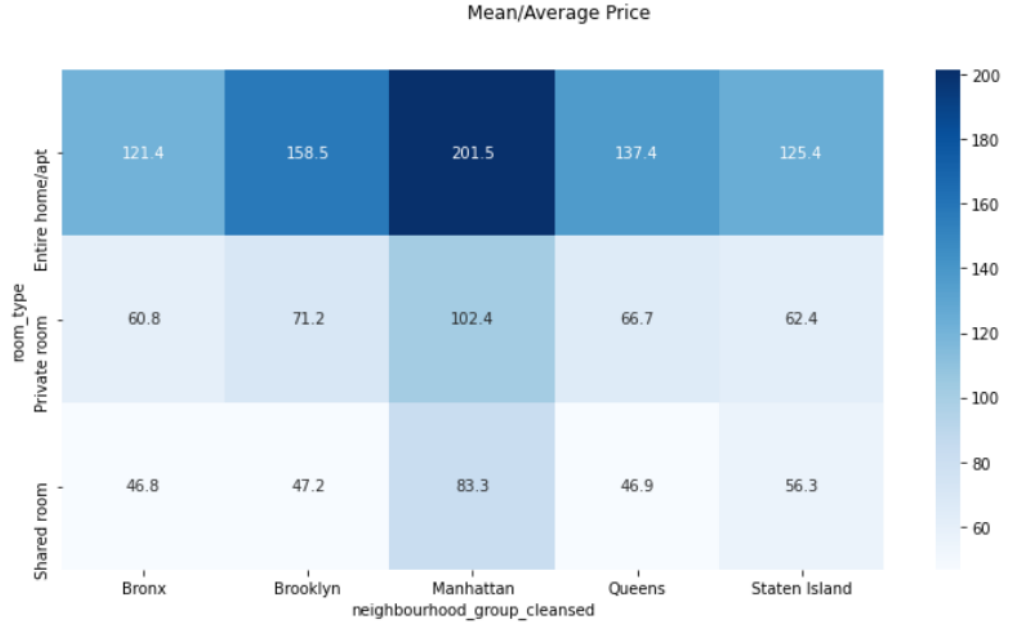




Data Exploration
price relation to other columns



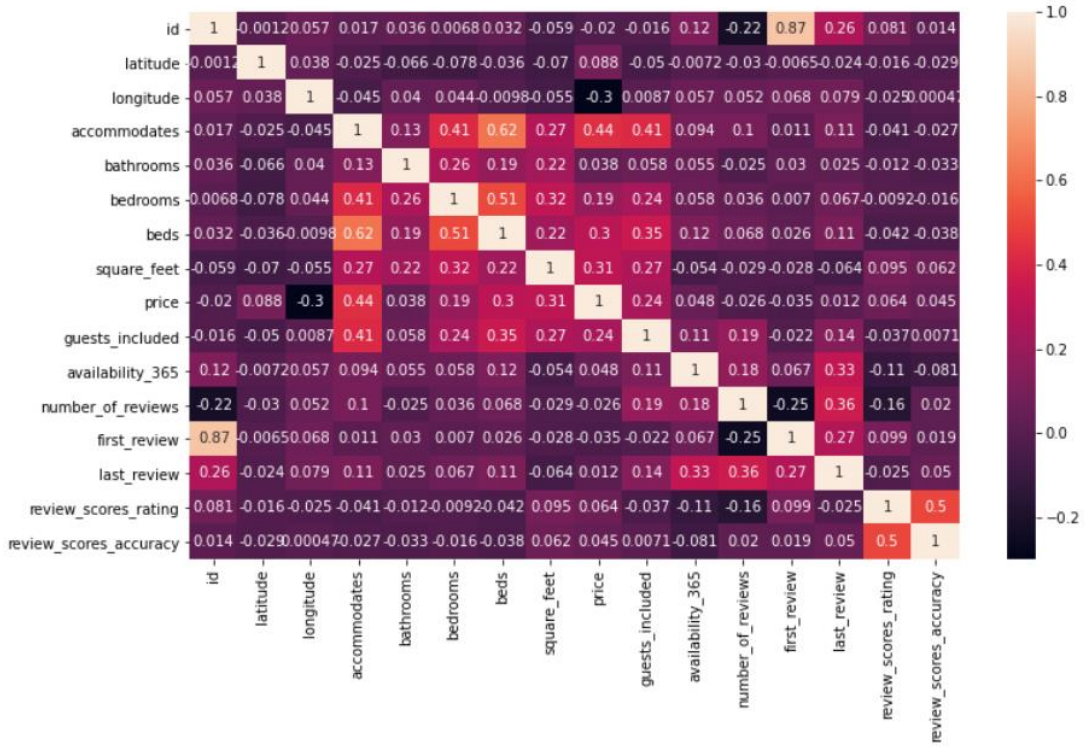




airbnb

Data Preparation

- Check for missing values
  - Handled Missing Values by using the fillna()
  - For city's missing value did df.dropna(subset=['city']) by doing so it removes the rows that are missing in the city column
  - Correlation Map
  - Used it to determine which features has high correlation with price.
  - Based on the result, performed another round of dropping unnecessary columns
- Drop Duplicated Rows



Before

```
#check for  missing values in dataset
df.isnull().sum()
```

```
id                           0
neighbourhood_group_cleansed 0
city                        60
zipcode                    500
latitude                     0
longitude                    0
property_type                0
room_type                    0
accommodates                 0
bathrooms                   53
bedrooms                    27
beds                        39
bed_type                     0
square_feet              47314
price                        0
guests_included              0
availability_365             0
number_of_reviews            0
first_review              9648
last_review               9648
review_scores_rating     10594
review_scores_accuracy   10631
cancellation_policy          1
dtype: int64
```

After

```
[25]: df.dropna(inplace=True)
      df.isnull().sum()
```

```
[25]: neighbourhood_group_cleansed    0
      city                          0
      property_type                 0
      room_type                     0
      accommodates                  0
      bathrooms                     0
      bedrooms                      0
      beds                          0
      price                         0
      guests_included               0
      availability_365              0
      dtype: int64
```

Feature Selection

- Drop Duplicated rows
- Used LabelEncoder() to change categorical features to int

```
[26]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 47636 entries, 0 to 48863
Data columns (total 11 columns):
 #   Column                        Non-Null Count  Dtype
---  ------                        --------------  -----
 0   neighbourhood_group_cleansed  47636 non-null  object
 1   city                          47636 non-null  object
 2   property_type                 47636 non-null  object
 3   room_type                     47636 non-null  object
 4   accommodates                  47636 non-null  int64
 5   bathrooms                     47636 non-null  float64
 6   bedrooms                      47636 non-null  float64
 7   beds                          47636 non-null  float64
 8   price                         47636 non-null  int64
 9   guests_included               47636 non-null  int64
 10  availability_365              47636 non-null  int64
dtypes: float64(3), int64(4), object(4)
memory usage: 4.4+ MB
```

```
[30]: for col in categorical_col:
          df[col] = LabelEncoder().fit_transform(df[col])
      df.dtypes
```

```
[30]: neighbourhood_group_cleansed      int32
      city                              int32
      property_type                     int32
      room_type                         int32
      accommodates                      int64
      bathrooms                       float64
      bedrooms                        float64
      beds                            float64
      price                             int64
      guests_included                   int64
      availability_365                  int64
      dtype: object
```

Methods and Improvements

```
#models
x=df.drop(columns=['price'])
y=df['price']

#split the data set into training set (70%) and test set(30%)
x_train,x_test,y_train,y_test = train_test_split(x,y , test_size=0.3, random_state=7 )
```

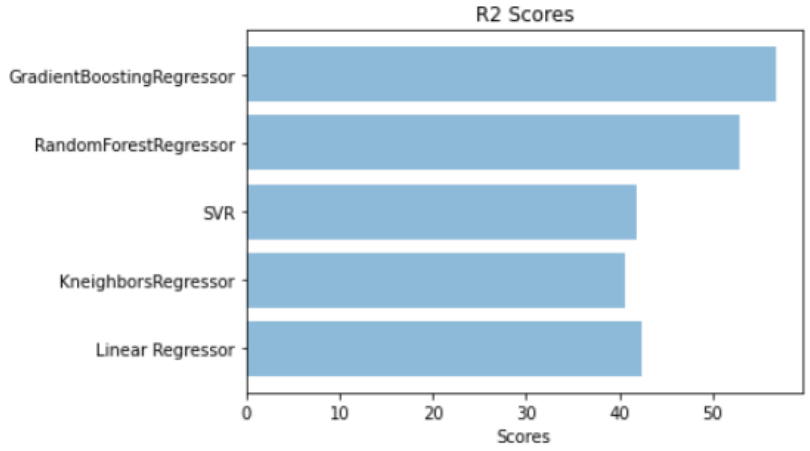
- Algorithms used
  - Linear Regression
  - KneighboursRegression
  - SVR
  - RandomForestRegressor
  - GradientBoostingRegresor

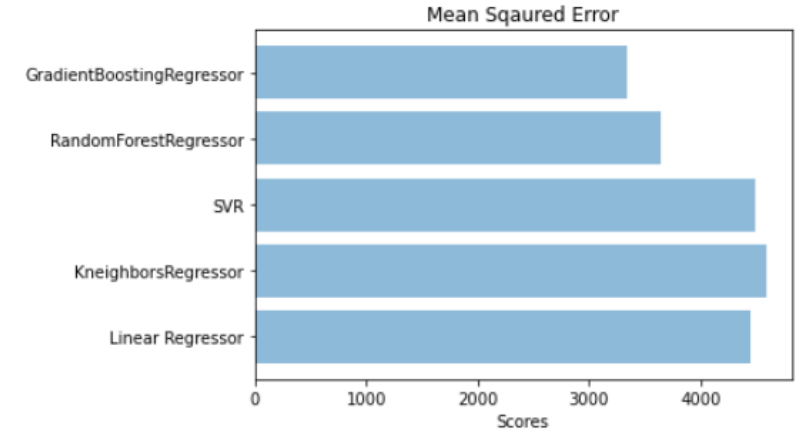
- Improvement
  - Performed a gridsearch for the models and tuned the parameters according to the gridsearch result

airbnb

Result and Analysis

R2 Score : The higher the number the better it is
Mean Squared Error : The lower the number the better it is





GradientBoostingRegressor

After Hyperparameter tuning
R2 score: 56.77098699048435
RMSE: 57.79218745814706
Training Set Mean Absolute Error: 39.3379
Test Set Mean Absolute Error: 41.2883
Mean Squared Error: 3339.93693119761

	Actual Values	Predicted Values
0	249	243.476457
1	66	89.766032
2	85	141.596312
3	139	96.276625
4	190	209.739154
5	300	280.091646
6	400	210.461498
7	80	127.907123
8	49	84.913724
9	125	119.532896
10	95	146.337734
11	53	84.120369
12	48	74.824028
13	169	178.023011
14	61	75.056104
15	200	150.943010
16	73	127.078668
17	120	95.034202
18	80	61.214620
19	165	166.191207

References

- https://scikit-learn.org/stable/supervised_learning.html#supervised-learning
- https://www.kaggle.com/kerneler/starter-ab-ny-august-2019-b8560924-7