UNIVERSITY OF ILLINOIS CHICAGO

Airbnb Pricing Prediction

Giovanni Alvin Prasetya
Naga Bhaskar Muddana
Sagarika Ugendhar
Tushar Yadav

I. Abstract

Nowadays, the number of people involved in staycation as a guest or host are rising. This situation produces availability in an alternative way that allows tourists to customize their rent plan and gain their experience. However, the accommodation price would become more competitive day by day due to high demand, especially in a big city like New York. This research performed with the purpose of acquiring the fittest model and competitive price for airbnb's. The results emphasize the score of each room type (apt room/hotel room/shared room/private room). This research focuses on the linear model and other related models to determine how the price relates to any other factor. The correlation between Airbnb price and the room's type showed how diverse the price for areas combined. The location of Airbnb is also a major factor of the price competitiveness compared to others. To make an analysis out of our datasets, we did exploratory Data Analysis (EDA) which was performed using matplotlib, pandas, and seaborn packages. After undergoing the data analysis on the datasets, we executed a linear regression model, SVR, gradient boost, etc to determine which model suits our objectives. Finally, we evaluate the model using evaluation metric Root Mean Square(RMSE) and Median absolute Error (MAE) since it is less sensitive to outliers compared to other metrics. The graph shows that most of the models are able to predict the price with a median error around 20 to 30 dollars.

Keyword : Airbnb, regression models, location, mean absolute error, price

II.Introduction

Airbnb is a short-term rental platform that allows you to rent out a portion or all of your living space to others. Airbnb has become a shining star in the sharing economy. Airbnb was valued at 113 billion dollars in 2021, up from 75 billion the year before. According to reports, Airbnb accounts for 5% of all tourism lodging revenue. Despite its commercial success, Airbnb has struggled to improve host pricing, potentially missing out on significant financial benefits. Airbnb hosts are predicted to lose up to 46% of additional revenue due to inefficient pricing, which has been cited as one of the company's top difficulties.

Although Airbnb has been developing pricing tools for hosts since 2012, these tools have been relatively basic and have solely focused on simple parameters such as the number of rooms, surrounding properties, and amenities such as parking. Airbnb has selectively launched a new pricing tool that takes both property qualities and demand into account, after years of work on a price algorithm. The pricing of Airbnb properties is complicated because, in addition to standard demand considerations such as seasonal changes, local events, and location, each Airbnb listing has its own unique qualities, and hosts frequently take on additional duties such as concierge, cook, and tour guide.

Airbnb listings face competition from other Airbnbs rather than hotels. It is assumed that potential clients have already decided to use Airbnb and are just looking at Airbnbs in the region, rather than comparing costs to adjacent hotels. Renters will choose other cheaper options if the host charges more than the market price. The hosts will lose out on potential earnings if the nightly rent price is set too low. As a result, we employ machine learning models to anticipate the best prices for the hosts' properties. Dynamic Pricing Optimization for Airbnb listing to optimize yearly profit for hosts. We've decided on three overall business goals to focus on for our project, which we'll expand on in the analytics goals and general model.

To strengthen our work background, we use another source of research related to this topic, which is predicting list prices on airbnb dataset. The purpose of this research is solely to generate competitive prices for a list of airbnb's. Referring to this research, we would like to do some predictive analysis regarding the airbnb pricing with other approaches such as regression algorithms.

III. Related work

This research is based on previous research related to Optimization of Airbnb Dynamic Pricing. This research was conducted with 2 goals which are business goals and analytical goals. Business goals for this research are mainly focused on maximizing yearly profit (via optimization) for a specific listing on Airbnb. All other goals are aimed either at modifying this central goal or constraining it to make the final solution more viable. The second goal is to create a dynamic pricing tool that determines a daily price competitive with other localized Airbnb listings. The research objective was to create a model that was as flexible as possible by determining price at the scale of the smallest possible rental period (daily). This means that the optimization model could be specified down to the specific days that a user would/would not expect to be renting their listing. The third business goal is to create an easily repeatable design process for expansion to other locations. These three objectives would be the main scope to fulfill the business goals.

To complement the business goals, the author made analytical goals to determine how they fulfill the goals. The analytical goal is to calculate the optimal daily price through an optimization model of yearly profit. This goal describes the main business goal in more concrete analytical terms. After the author utilizes an optimization model, they could structure the analysis around the component pieces necessary; a modeled demand variable, intrinsic amenities' additive values for properties, and valuation of costs vs. revenues. By framing the problem as a typical revenue - cost optimization, they could generate the outline of the research general structure and begin to create the pieces necessary for the final model.

IV. Exploratory Data Analysis (EDA)

Exploratory data analysis is a way to better understand your data which helps in further Data preprocessing.

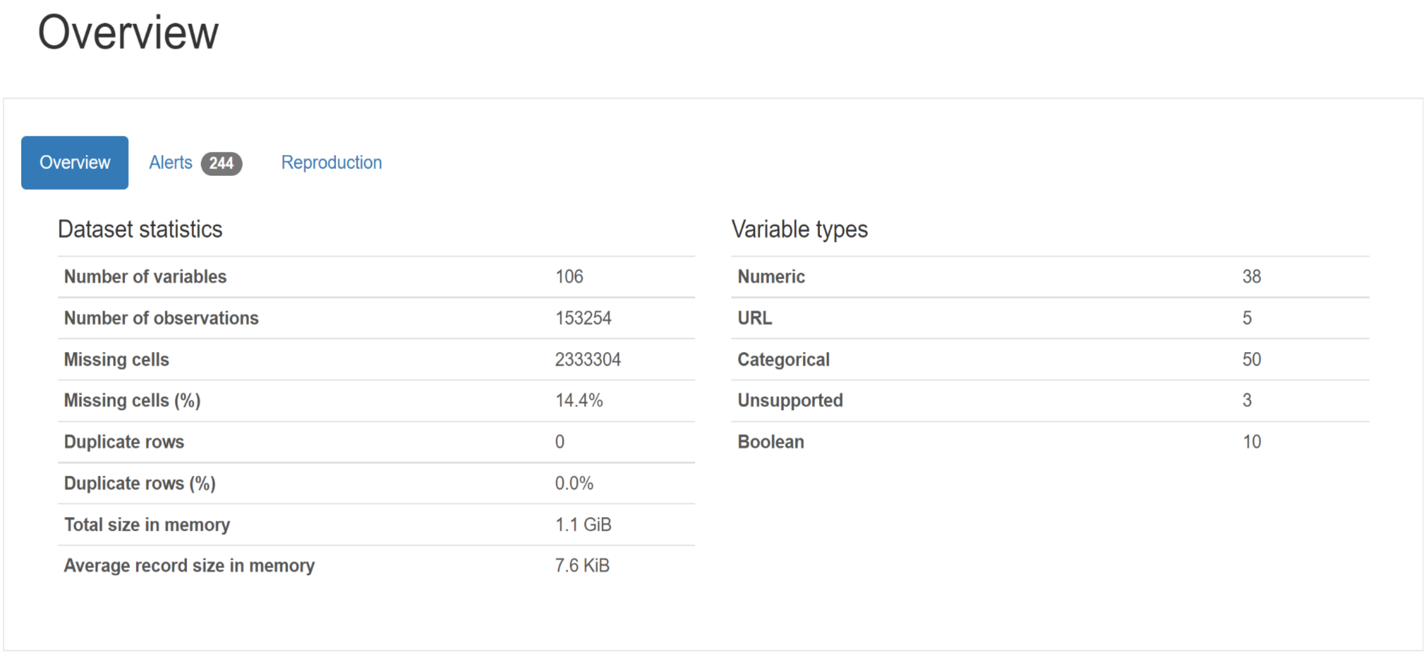
Packages Used :

1. **NUMPY**: To perform mathematical operations on the dataset
2. **PANDAS**: Used for data analysis and cleaning

3. **MATPLOTLIB**: Used to plot the graphs
4. **PANDAS-PROFILING**: Used to better understand the features of the dataset and their statistics

A. Dataset and Features

Dataset is from Kaggle which had New york's Airbnb listings and other geographical metrics to make predictions. The whole dataset had about 153254 rows and 106 columns. We have 'Price' as our target variable. As we are dealing with a target variable of numerical data we will use a regression method to predict the values.



Overview

Overview | Alerts 244 | Reproduction

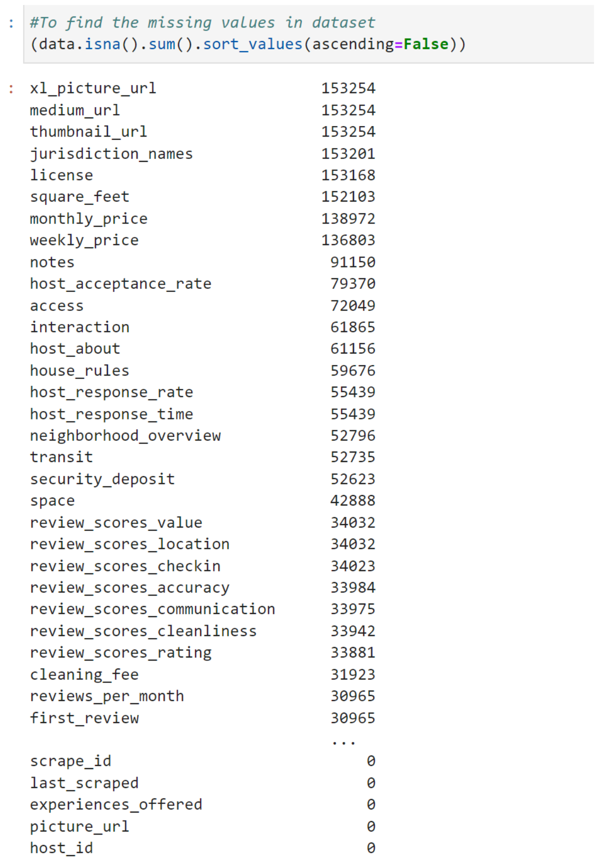
Dataset statistics

Number of variables	106
Number of observations	153254
Missing cells	2333304
Missing cells (%)	14.4%
Duplicate rows	0
Duplicate rows (%)	0.0%
Total size in memory	1.1 GiB
Average record size in memory	7.6 KiB

Variable types

Numeric	38
URL	5
Categorical	50
Unsupported	3
Boolean	10

Fig.1 Missing Value



```
: #To find the missing values in dataset
  (data.isna().sum().sort_values(ascending=False))

: xl_picture_url                  153254
  medium_url                      153254
  thumbnail_url                   153254
  jurisdiction_names              153201
  license                         153168
  square_feet                     152103
  monthly_price                   138972
  weekly_price                    136803
  notes                            91150
  host_acceptance_rate             79370
  access                           72049
  interaction                      61865
  host_about                       61156
  house_rules                      59676
  host_response_rate               55439
  host_response_time               55439
  neighborhood_overview            52796
  transit                          52735
  security_deposit                 52623
  space                            42888
  review_scores_value              34032
  review_scores_location           34032
  review_scores_checkin            34023
  review_scores_accuracy           33984
  review_scores_communication      33975
  review_scores_cleanliness        33942
  review_scores_rating             33881
  cleaning_fee                     31923
  reviews_per_month                30965
  first_review                     30965
                                   ...
  scrape_id                            0
  last_scraped                         0
  experiences_offered                  0
  picture_url                          0
  host_id                              0
```

B. Preprocessing

As the data is scraped from Airbnb website, it has a lot of junk values and missing data. So, the first step is to clean the data like dropping some unnecessary columns, filling the null values and so on. The first thing in the Preprocessing step is to count the number of null values for each column. From Fig. 1, we can see that columns x1_picture_url, medium_url, thumbnail_url, jurisdication_names, license, square_feet, monthly_price, weekly_price, notes have more than 60 percent null values. If an attribute has more than 60 percent of null values, then we can assume that this column has not enough information and is meaningless, and we can drop these columns.

Fig.1 and 2. explains the missing values in the dataset. Fig1. shows the exact number of missing values that has been sorted in descending order. Fig.2 shows the graphical representation of percentage of missing values in each columns

We will now check for missing values in our dataset. In case there are any missing entries, we will impute them with appropriate values (mode in case of categorical feature, and median or mean in case of numerical feature). We will use the isnull() function for this purpose.

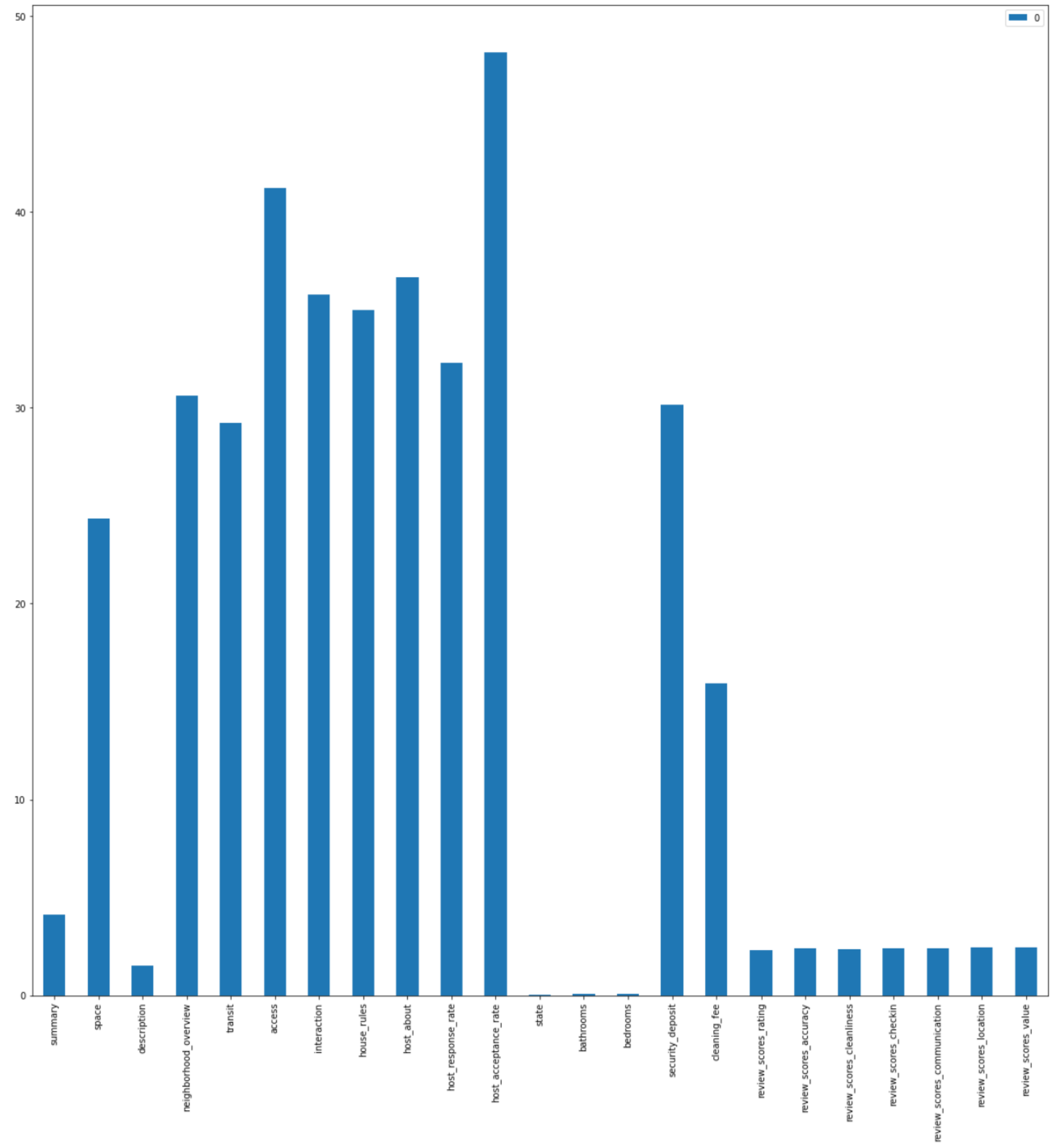
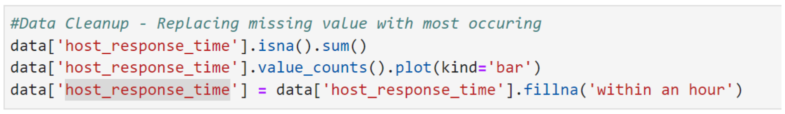


Fig.2 Percentage of Missing values for each column

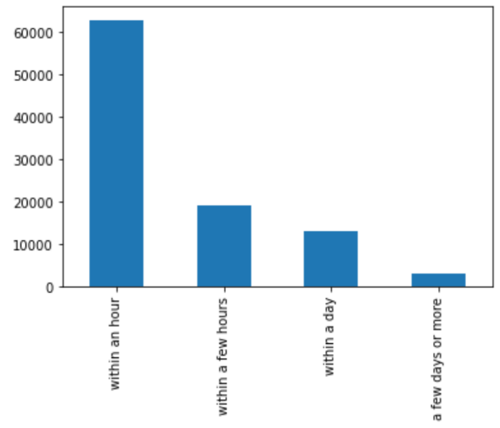
After finding the missing values, the columns that have missing values more than 60% of the dataset are removed as it will not contribute to the model. And then by looking through the details of each attribute, we can further drop more columns. Table 1. shows the columns along with the descriptions for dropping them from the dataset.

Columns	**Description**
listing_url', 'thumbnail_url', 'medium_url', 'picture_url', 'xl_picture_url', 'host_url', 'host_thumbnail_url', 'host_picture_url'	Columns with URL that has no information about the dataset
'neighborhood','calendar_updated','host_neighbourhood', 'city','zipcode','market','first_review','last_review','calendar_last_scraped','experiences_offered','last_scraped','scrape_id','id','host_has_profile_pic','latitude','longitude'	Columns that have redundant information which is already available in the other columns. These are highly correlated
'name','host_name','host_id','host_is_superhost','host_identity_verified',	Columns that has unrelated information about the listings

Table 1. Dropped Columns with description



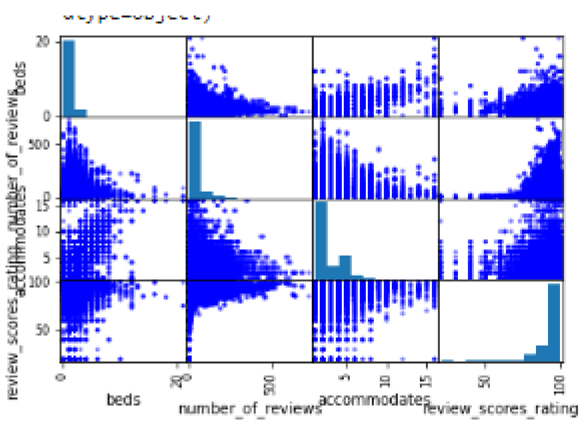
```
#Data Cleanup - Replacing missing value with most occuring
data['host_response_time'].isna().sum()
data['host_response_time'].value_counts().plot(kind='bar')
data['host_response_time'] = data['host_response_time'].fillna('within an hour')
```



We will start with Univariate Analysis to find the distribution of each column. We will be using a bar graph for this purpose. The missing values for categorical columns have been replaced with the most occurring by visualizing the value_counts of the column. In Fig3. The majority of the categories fall under ‘Within an hour’ by looking at the bar plot. Hence the missing values are replaced with ‘Within an hour’ using fillna() method.

Apart from null values, there are a lot of columns with blank ‘ ’ values. All the blank ‘ ’ is replaced with NaN so they can be handled as missing values and will be easy to process.

The next step in EDA is to separate the numerical and categorical columns. Some of the numerical values had string columns due to special characters. Such columns are identified and special characters are removed.



```
time: 10.7 s (started: 2022-04-24 20:08:11 -05:00)
```

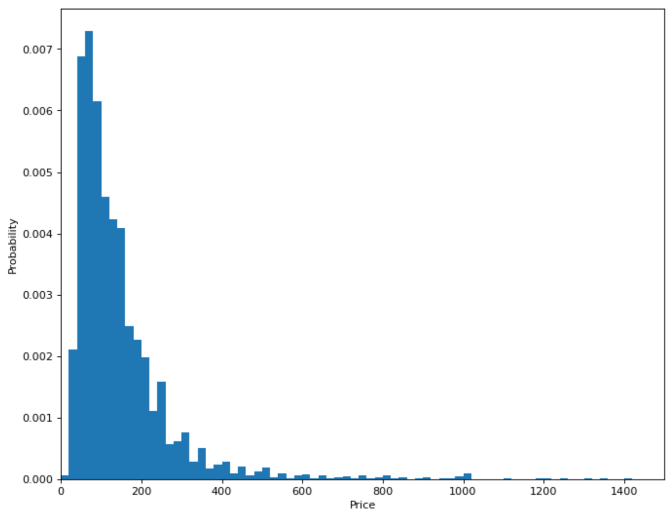
The columns ‘host_response_rate’ & ‘host_acceptance_rate’ had % symbol which was removed and the string columns were converted to numeric.
Similarly, columns ‘price', 'weekly_price', 'monthly_price', 'security_deposit', 'cleaning_fee', 'extra_people' had $ symbol to denote price which was removed and the string columns were converted to numeric.

C. Visualization

Data Visualization represents the text or numerical data in a visual format, which makes it easy to grasp the information of the data. Used data visualization libraries like matplotlib and seaborn to plot the charts. Below graph shows the distribution of the ‘price’ variable and we can see that the distribution is skewed.

Fig4. Probability distribution of Price

```
plt.figure(figsize=(10, 8), dpi=80)
plt.hist(df['price'], density=True, bins=500)
plt.xlim(0,1500)
plt.ylabel('Probability')
plt.xlabel('Price');
```



```
df['price'].describe()
```

```
count    153254.000000
mean        162.669731
std         401.014347
min           0.000000
25%          69.000000
50%         105.000000
```

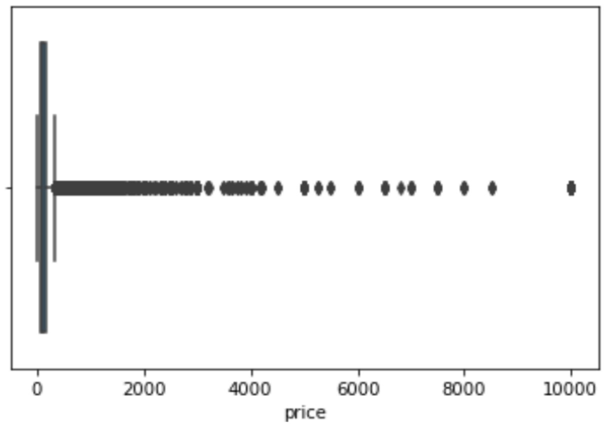
In the price distribution chart, we can see the price values are positively skewed or right skewed where most values are clustered around the left tail of the distribution while the right tail of the distribution is longer.

Outliers detection

Boxplots are a standardized way of displaying the distribution of data based on a five number summary (“minimum”, first quartile (Q1), median, third quartile (Q3), and “maximum”). This type of plot is used to easily detect outliers. It can also tell us if your data is symmetrical, how tightly your data is grouped, and if and how your data is skewed.

```
import warnings
warnings.filterwarnings("ignore")
sns.boxplot(data['price'])
```

```
<AxesSubplot:xlabel='price'>
```

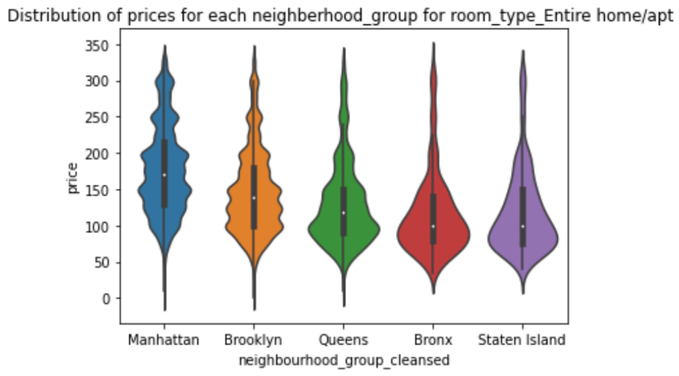


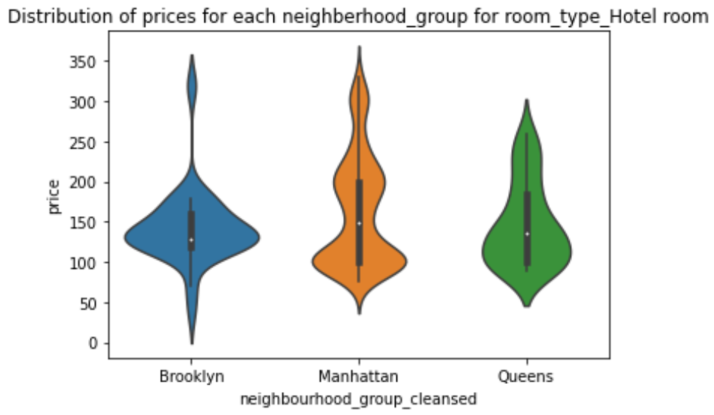
For Skewed distributions: Use Interquartile Range (IQR) proximity rule to remove outliers.
The data points which fall below Q1 – 1.5 IQR or above Q3 + 1.5 IQR are outliers where Q1 and Q3 are the 25th and 75th percentile of the dataset respectively, and IQR represents the interquartile range given by Q3 – Q1.

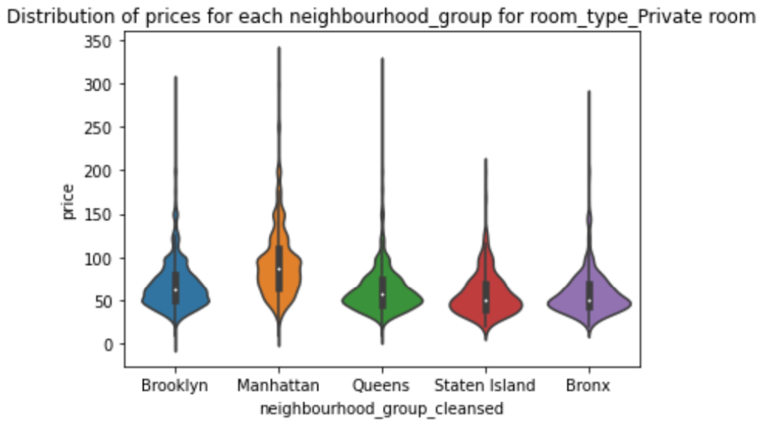
Converting categorical data to numerical

After removing outliers as part of preprocessing, we converted the boolean values to integer using map() method. For columns that had multi- categorical values, dummy variables are created for the categorical variables using get_dummies() function from Pandas library. Thus the categorical variables are converted to numerical using the above approaches.

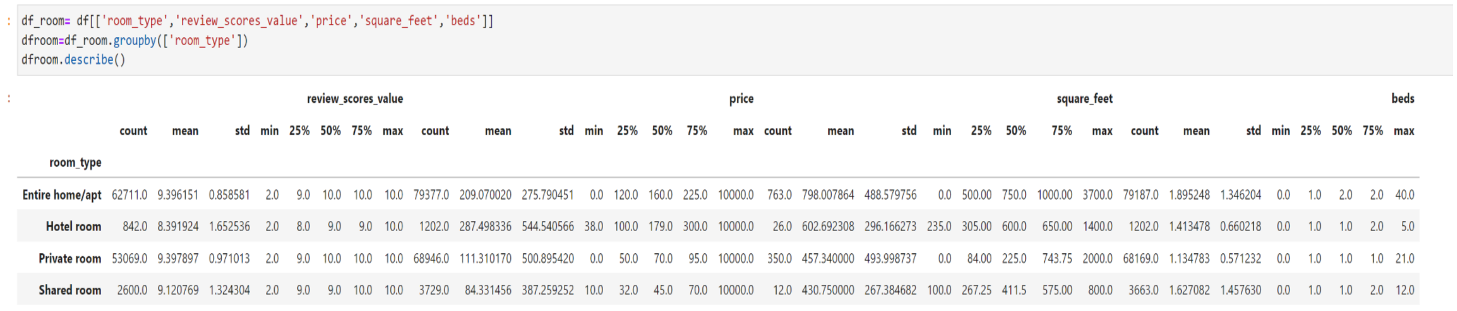
Below Violin graph shows the density and distribution of prices for each neighborhood group filtered by Room type. The median is almost the same across different neighborhoods. The shape of the distribution indicates that the listings are highly concentrated around the median for Bronx & Skaten island for room type 'Entire home/apt'.





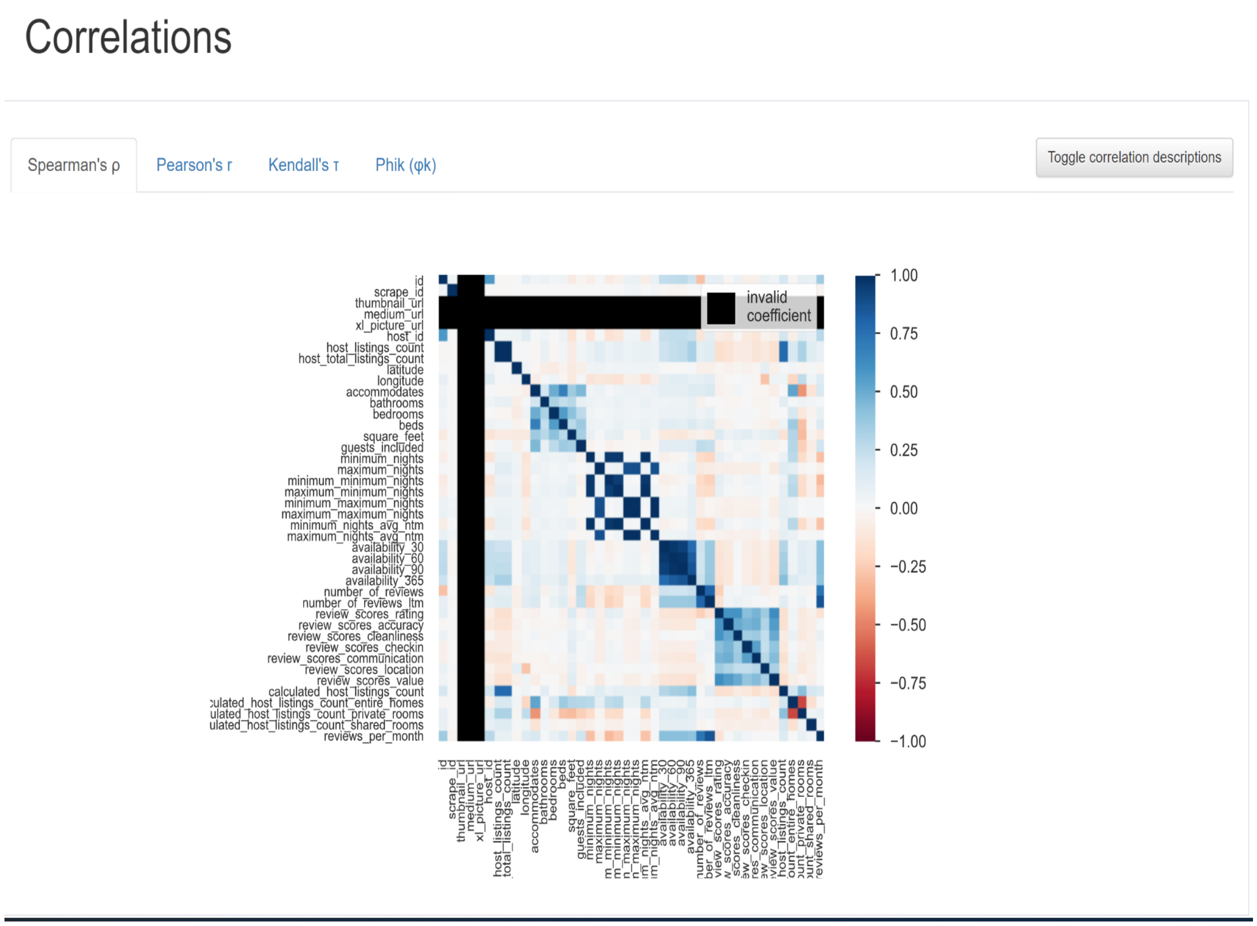


Listing details summary by Room type



```
df_room= df[['room_type','review_scores_value','price','square_feet','beds']]
dfroom=df_room.groupby(['room_type'])
dfroom.describe()
```

	review_scores_value								price								square_feet								beds							
	count	mean	std	min	25%	50%	75%	max	count	mean	std	min	25%	50%	75%	max	count	mean	std	min	25%	50%	75%	max	count	mean	std	min	25%	50%	75%	max
room_type																																
Entire home/apt	62711.0	9.396151	0.858581	2.0	9.0	10.0	10.0	10.0	79377.0	209.070020	275.790451	0.0	120.0	160.0	225.0	10000.0	763.0	798.007864	488.579756	0.0	500.00	750.0	1000.00	3700.0	79187.0	1.895248	1.346204	0.0	1.0	2.0	2.0	40.0
Hotel room	842.0	8.391924	1.652536	2.0	8.0	9.0	9.0	10.0	1202.0	287.498336	544.540566	38.0	100.0	179.0	300.0	10000.0	26.0	602.692308	296.166273	235.0	305.00	600.0	650.00	1400.0	1202.0	1.413478	0.660218	0.0	1.0	1.0	2.0	5.0
Private room	53069.0	9.397897	0.971013	2.0	9.0	10.0	10.0	10.0	68946.0	111.310170	500.895420	0.0	50.0	70.0	95.0	10000.0	350.0	457.340000	493.998737	0.0	84.00	225.0	743.75	2000.0	68169.0	1.134783	0.571232	0.0	1.0	1.0	1.0	21.0
Shared room	2600.0	9.120769	1.324304	2.0	9.0	9.0	10.0	10.0	3729.0	84.331456	387.259252	10.0	32.0	45.0	70.0	10000.0	12.0	430.750000	267.384682	100.0	267.25	411.5	575.00	800.0	3663.0	1.627082	1.457630	0.0	1.0	1.0	2.0	12.0



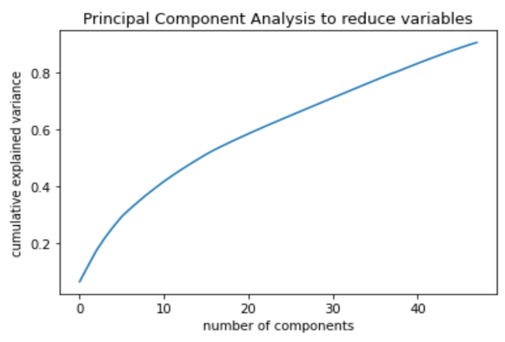
V. Principal Component Analysis (PCA)

PCA is a dimensionality reduction technique that reduces the dimension of the feature space in such a way that new variables are orthogonal to each other.
PCA is affected by scale so it is necessary to standardize the data by scaling the features in a dataset before applying PCA. This is necessary because fitting algorithms highly depend on scaling of the features. By using

StandardScalar() from sklearn.preproccessing , standardize the dataset's features onto unit scale with mean as 0 and variance as 1 which is a requirement for the optimal performance of many machine learning algorithms

```
#this is to perform PCA on our data
plt.plot(np.cumsum(pca.explained_variance_ratio_))
plt.title('Principal Component Analysis to reduce variables')
plt.xlabel('number of components')
plt.ylabel('cumulative explained variance')
```

```
Text(0, 0.5, 'cumulative explained variance')
```



```
#Selection features that exlain atleast 90% of the target variance
from sklearn.decomposition import PCA
pca = PCA(0.90)
pca.fit(X_train_scaled)
```

```
PCA(n_components=0.9)
time: 63 ms (started: 2022-04-24 13:47:24 -05:00)
```

```
pca.n_components_
```

```
48
```

Performed PCA with 0.9 for the number of component parameters. Scikit Learn chooses the minimum number of principal components such that 90% of variance is retained. By doing this, the number of features was reduced from 106 to 48.

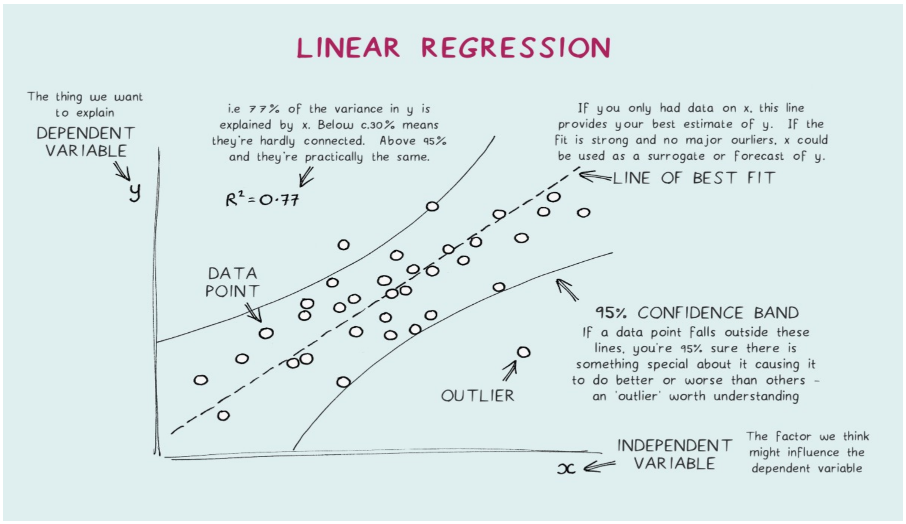
VI. MODELS:

For these specific datasets we used some models and made comparisons out of all models to determine which one suits our objectives. We used **sklearn** for our modeling.

a) Linear Regression

Linear regression is the first model that we used for this project. Linear regression is a traditional model in regression analysis, and it has been discussed several times to solve any problems between a scalar response and one or more explanatory variables. It’s used to predict values within a continuous range. The objective function of Linear Regression is

$$\min_{\theta} \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^i) - y^i)^2$$


LINEAR REGRESSION
The thing we want to explain
DEPENDENT VARIABLE
y
i.e 77% of the variance in y is explained by x. Below c.30% means they're hardly connected. Above 95% and they're practically the same.
R² = 0.77
If you only had data on x, this line provides your best estimate of y. If the fit is strong and no major ourliers, x could be used as a surrogate or forecast of y.
LINE OF BEST FIT
DATA POINT
95% CONFIDENCE BAND
If a data point falls outside these lines, you're 95% sure there is something special about it causing it to do better or worse than others - an 'outlier' worth understanding
OUTLIER
INDEPENDENT VARIABLE
The factor we think might influence the dependent variable
x

b) *Logistic Regression*

Our second model we used isLogistic regression.Logistic regression is a process of modeling the probability of a discrete outcome given an input variable. The purpose of this model is to understand the relationship between the dependent variable and one or more independent variables by estimating probabilities using an equation. Here, we’re trying to determine the likelihood of the explanatory variable on the datasets.

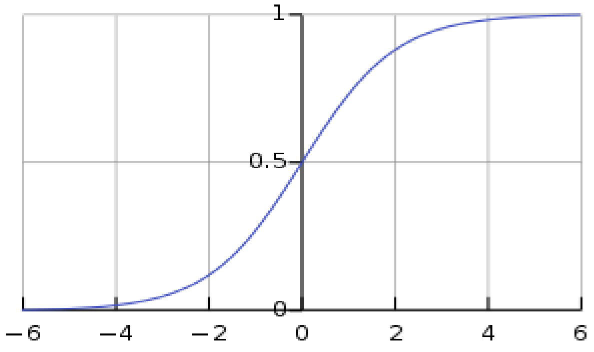
$$f(x) = \frac{L}{1 + e^{-k(x - x_0)}},$$

where

x_0, the x value of the sigmoid's midpoint;

L, the curve's maximum value;

k, the logistic growth rate or steepness of the curve.[1]



x=0,k=1,L=1

c) *Ridge Regression*

The third model we used is ridge regression. Ridge regression is a method of estimating the coefficients of multiple-regression models in scenarios where linearly independent variables are highly correlated. Ridge regression is a complement to least-squares regression, which loses unbiasedness in exchange for high numerical stability, thus obtaining higher computational accuracy.

$$\textbf{Ridge} = \frac{1}{N}\sum_{i=1}^{N}(y_i - (mx_i + z))^2 + \lambda \sum_{i=1}^{p}(mx_i + z)^2$$

d) *Lasso*

Next model we are trying to execute is lasso regression. The goal of lasso regression is to obtain the subset of predictors that minimizes prediction error for a quantitative response variable. Just as what it names, lasso uses shrinkage, where data values are shrunk towards a central point, like the average of the data. Lasso is a modification of linear regression, where the model is penalized for the sum of absolute values of the weights. Thus, the absolute values of weight will be (in general) reduced, and many will tend to be zeros.

$$\textbf{Lasso} = \frac{1}{N}\sum_{i=1}^{N}(y_i - (mx_i + z))^2 + \lambda \sum_{i=1}^{p}(mx_i + z)$$

e) *Elastic Net*

Next model we executed is ElasticNet. The coefficients to the variables are considered to be essential information; nevertheless, ridge regression does not guarantee that all irrelevant coefficients will be removed, which is one of its shortcomings over Elastic Net Regression (ENR). Regularization assists in the resolution of model overfitting issues. Elastic Net is a regression approach that simultaneously does variable selection and regularization. It employs both Lasso and Ridge Regression regularization to eliminate any non-informative coefficients while leaving the informative ones.

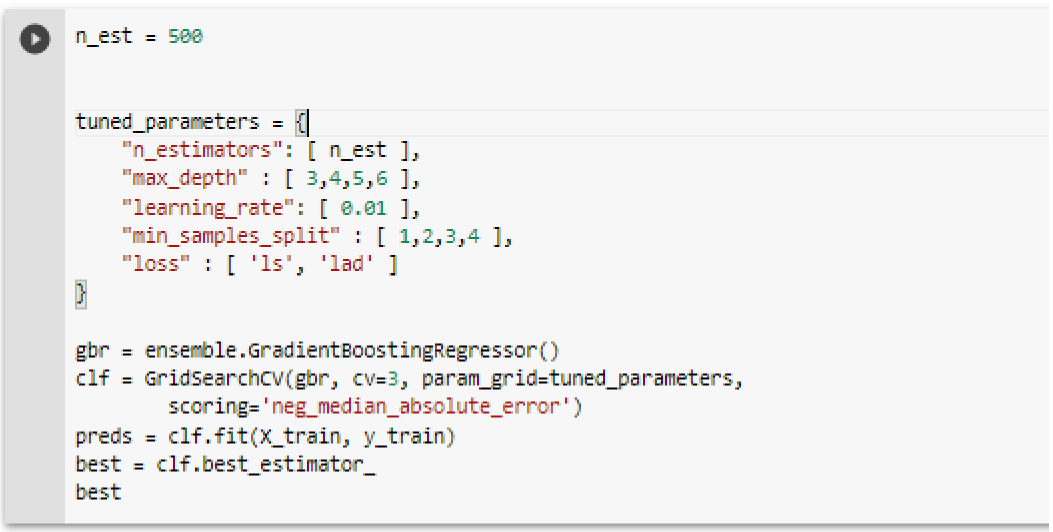
Lasso Regression + Ridge Regression = ENR.

$$\frac{1}{N}\sum_{i=1}^{N}(y_i - (mx_i + z))^2 + \lambda \sum_{i=1}^{p}(mx_i + z)^2 + \lambda \sum_{i=1}^{p}(mx_i + z)$$

f) GradientBoosting Regressor

Gradient Boosting for regression, GB constructs an additive model in a stage-by-stage manner, allowing for the optimization of any differentiable loss function. A regression tree is fitted on the negative gradient of the supplied loss function at each level. It returns a prediction model in the form of an ensemble of weak prediction models, most commonly decision trees. It is a ensemble technique

Hyperparameter tuning :



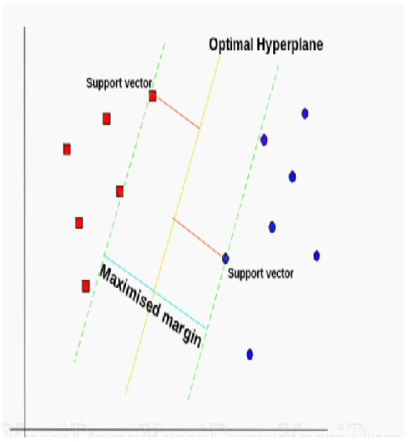
```
n_est = 500


tuned_parameters = {
    "n_estimators": [ n_est ],
    "max_depth" : [ 3,4,5,6 ],
    "learning_rate": [ 0.01 ],
    "min_samples_split" : [ 1,2,3,4 ],
    "loss" : [ 'ls', 'lad' ]
}

gbr = ensemble.GradientBoostingRegressor()
clf = GridSearchCV(gbr, cv=3, param_grid=tuned_parameters,
        scoring='neg_median_absolute_error')
preds = clf.fit(X_train, y_train)
best = clf.best_estimator_
best
```

The result suggested to use

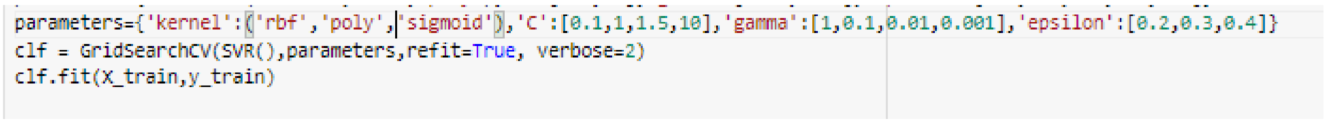
```
learning_rate=0.01, loss='lad', max_depth=4,min_samples_split=2,n_estimators=500)
```

g) SVR (Support Vector Regression)

SVMs and Support Vector Regression are both based on the same principle. SVR's core principle is to locate the best-fitting line. The best fit line in SVR is the hyperplane with the greatest number of points. The supervised learning algorithm Support Vector Regression is used to predict discrete values.Unlike other regression models which tries to minimize the error actual and predicted value, SVR tries to fit the best line within the distance between the hyperplane and boundary by trying to satisfy `-a < y-wx+b < a`



In our modeling we used two different libraries to model SVR , one of them is the LinearSVR() from liblinear which uses a linear kernel whereas the other is implemented using libsvm. We have done hypertuning for the SVR to see if a different combination of parameters would be a good fit for the data.



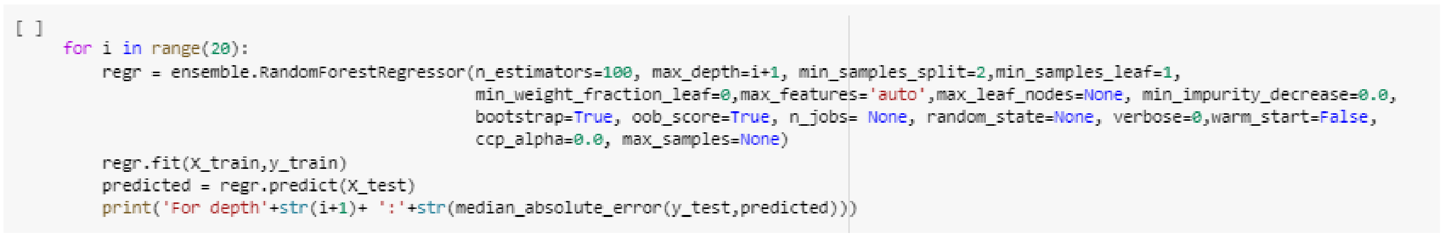
```
parameters={'kernel':('rbf','poly','sigmoid'),'C':[0.1,1,1.5,10],'gamma':[1,0.1,0.01,0.001],'epsilon':[0.2,0.3,0.4]}
clf = GridSearchCV(SVR(),parameters,refit=True, verbose=2)
clf.fit(X_train,y_train)
```

Using GridSearchCV made it easy to find the best hyperparameters used to get the best fit model.The result suggested us to use the following configuration:

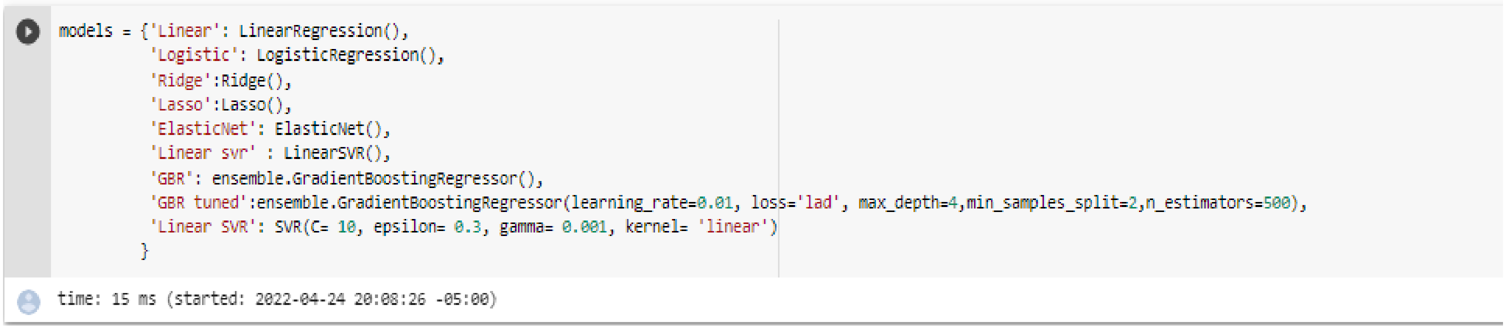
b) RandomForest Regressor

Random Forest Regression is a supervised learning approach for regression that use the ensemble learning method. The ensemble learning method combines predictions from several machine learning algorithms to get a more accurate forecast than a single model.

The models above executed accordingly resulted in the figure below. For the evaluation metrics, we used Root Mean Square Error (RMSE) and Median Absolute Error (MAE). Median absolute error has less sensitivity to outliers than other metrics. This is because the median of all of the absolute values of the residuals, and the median is unaffected by values at the tails. So, this loss function can be used to perform robust regression.



```
[ ]
    for i in range(20):
        regr = ensemble.RandomForestRegressor(n_estimators=100, max_depth=i+1, min_samples_split=2,min_samples_leaf=1,
                                              min_weight_fraction_leaf=0,max_features='auto',max_leaf_nodes=None, min_impurity_decrease=0.0,
                                              bootstrap=True, oob_score=True, n_jobs= None, random_state=None, verbose=0,warm_start=False,
                                              ccp_alpha=0.0, max_samples=None)
        regr.fit(X_train,y_train)
        predicted = regr.predict(X_test)
        print('For depth'+str(i+1)+ ':'+str(median_absolute_error(y_test,predicted)))
```

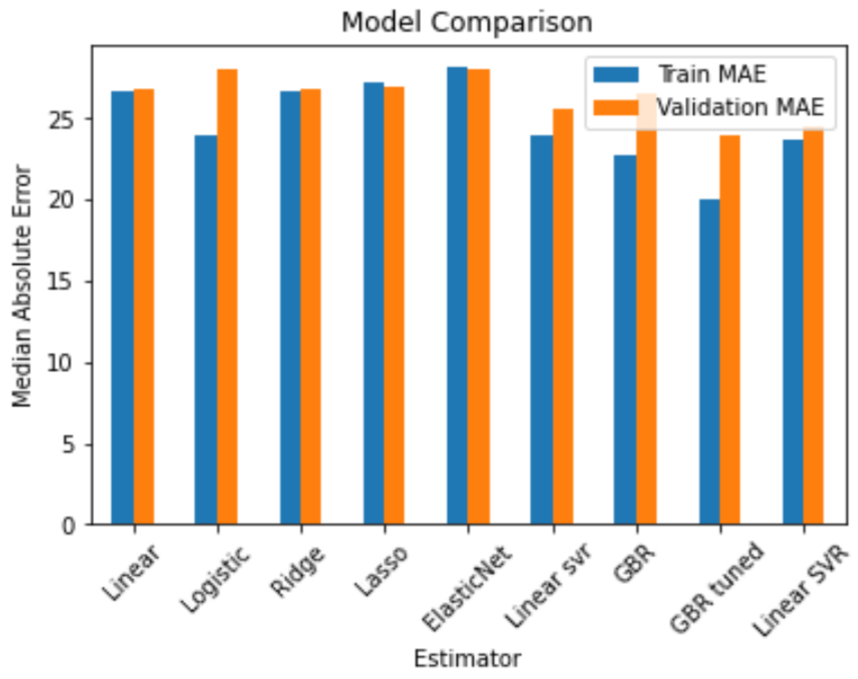


```
models = {'Linear': LinearRegression(),
          'Logistic': LogisticRegression(),
          'Ridge':Ridge(),
          'Lasso':Lasso(),
          'ElasticNet': ElasticNet(),
          'Linear svr' : LinearSVR(),
          'GBR': ensemble.GradientBoostingRegressor(),
          'GBR tuned':ensemble.GradientBoostingRegressor(learning_rate=0.01, loss='lad', max_depth=4,min_samples_split=2,n_estimators=500),
          'Linear SVR': SVR(C= 10, epsilon= 0.3, gamma= 0.001, kernel= 'linear')
         }

time: 15 ms (started: 2022-04-24 20:08:26 -05:00)
```

```
model = SVR(C= 10, epsilon= 0.3, gamma= 0.001, kernel= 'rbf')
model.fit(X_train, y_train)
y_train_pred = model.predict(X_train)
y_test_pred = model.predict(X_test)
tr_mae = median_absolute_error(y_train, y_train_pred)
valid_mae = median_absolute_error(y_test, y_test_pred)
train_rmse = mean_squared_error(y_train,y_train_pred)**0.5
valid_rmse = mean_squared_error(y_test,y_test_pred)**0.5
results_dict['SVR hypertuned'] = [round(tr_mae,3), round(valid_mae,3), round(train_rmse,3), round(valid_rmse,3)]

results_df = pd.DataFrame(results_dict).T
results_df.columns = ["Train MAE", "Validation MAE","Train RMSE","Validation RMSE"]
results_df
```

	Train MAE	Validation MAE	Train RMSE	Validation RMSE
Linear	26.715	26.251	46.863	46.558
Logistic	25.000	25.000	51.136	50.916
Ridge	26.715	26.252	46.863	46.558
Lasso	27.182	26.452	47.219	46.784
ElasticNet	28.456	27.722	47.661	47.099
Linear svr	24.501	24.205	47.841	48.636
GBR	24.815	25.103	43.626	45.051
GBR tuned	22.161	23.173	45.507	46.424
Linear SVR	24.109	23.708	46.909	46.401
SVR hypertuned	24.109	23.708	46.909	46.401

```
time: 2min 11s (started: 2022-05-01 09:34:36 -05:00)
```



IV.Conclusion or Discussion

In conclusion, We have compared different models based on the Root Mean Square Error(RMSE) and Median Absolute Error(MAE) evaluation metric. Linear models are simple, however, it is not quite adequate for prediction on this dataset. Ensemble learning has uncertain performance based on the choice of models. Therefore, in the real-life problem, we cannot depend on only one of a few specific models, we need to compare different models using some high-level techniques. As the main model that we used is SVR, first run was done with the plain model and then use the GridSearchCV to find the best hyperparameters to tune the SVR. MAE(Median Absolute Error) is used as a loss function to compare the models being used because it predicts the error as the difference between the predicted and actual which yields a numeric value that can be easily correlated. Our final model that we suggest for this data Gradient boosting regressor with minimum median absolute error.

In the future, We will try some methods like early stopping and data augmentation to avoid overfitting problems. In addition, We will try some more high-level ensemble learning methods to get a better performance. There's also an opportunity to implement Sentiment analysis on the customer reviews to extract the sentiment score and include it in the model feature to see if it has an impact on the listing's price analysis.

V. References

1. *Learnbnb.com*
2. *https://www.conceptatech.com/blog/price-optimization-how-dynamic-pricing-helps-airbnb-hosts-earn-big*