Airbnb

Airbnb Price Prediction

Submitted By

Ajay S. Nawalkar

(EBEON 0622612053)

(Course: Advanced Certification Program in Data Analytics)

About

Airbnb takes a unique approach toward lodging. Part of the “sharing economy,” Airbnb offers you someone’s home as a place to stay instead of a hotel. On Airbnb, you can find places to crash on your backpacking trip through Europe, or a spot to rent for a month during your internship in Los Angeles. It’s also a great way to explore a town you’re thinking about moving to, or finding an alternative to traditional hotel options in 2022 if you’re just around for a weekend. If you want to rent out extra space in your own home, you can host through Airbnb and make money for allowing a guest to stay the night.

- Airbnb was founded in 2008 as a platform enabling locals to list their homes for short term rental and travelers to have a lodging option alternative to hotels.
- Airbnb net worth as of January 13, 2023 is $64.48B.
- This Dataset is collected in year 2019.
- Along with homes for rent, Airbnb.com allows its users to search through “Experiences”, which include pre-arranged multi destination trips, “Restaurants,” and recently even “Animals,” which stands for experiences where travelers can meet and interact with animals local to their travel destination (e.g., Zoo visit).

Content

Section 1

[A] Introduction

Airbnb takes a unique approach toward lodging. Part of the "sharing economy," Airbnb offers you someone's home as a place to stay instead of a hotel. On Airbnb, you can find places to crash on your backpacking trip through Europe, or a spot to rent for a month during your internship in Los Angeles. It's also a great way to explore a town you're thinking about moving to, or finding an alternative to traditional hotel options in 2022 if you're just around for a weekend. If you want to rent out extra space in your own home, you can host through Airbnb and make money for allowing a guest to stay the night.

i. Airbnb was founded in 2008 as a platform enabling locals to list their homes for short term rental and travelers to have a lodging option alternative to hotels.
ii. Airbnb net worth as of January 13, 2023 is $64.48B.
iii. This Dataset is collected in year 2019.
iv. Along with homes for rent, Airbnb.com allows its users to search through "Experiences", which include pre-arranged multi destination trips, "Restaurants," and recently even "Animals," which stands for experiences where travelers can meet and interact with animals local to their travel destination (e.g., Zoo visit).

[B] Scope & Trend

Many Airbnb hosts in major cities switched to long-term rentals as the short-term market collapsed in March, as a way to maintain a steady (if lower) income. To keep hosts on the platform, Airbnb recently launched monthly stays, similar to a normal rental service, but without the usual yearly rental agreement.

Key statistics

- Airbnb generated $5.9 billion in revenue in 2021, a 73% year-on-year increase.
- Airbnb has 150 million users, though that number has not been updated since 2018.
- In 2021, 300 million bookings were made on Airbnb, a 55% increase on 2020.
- There are over seven million listings on Airbnb, run by four million hosts.

Revenue

- Airbnb saw its revenues increase by 73% in 2021, after a 31% decrease in revenue in 2020 due the coronavirus pandemic shutting down travel.

Profit

- Airbnb reported a net loss of $352 million in 2021, a huge contraction on the $4.5 billion it lost in 2020.

[C] Literature of Existing System

- Airbnb is an online marketplace that connects people who want to rent out their homes with people who are looking for accommodations in specific locales. Airbnb offers people an easy, relatively stress-free way to earn some income from their property.
- Technology is at the core of Airbnb's platform. Each reservation undertaken at Airbnb interacts with machine learning or the artificial intelligence technology generated by the platform.
- Advantage: Airbnb offers people an easy, relatively stress-free way to earn some income from their property. Guests often find Airbnb is cheaper, has more character, and is homier than hotels.
- Disadvantages: A Lengthier Booking Process which result the customer get attracted to the user friendly websites, and prices which are regularly fluctuating. So, it create loss margin for company.

Section 2

[A] Objective

- Predict the Pricing of Airbnb accommodation with the help of customer reviews, rating and with lot of other features in the dataset.
- Experiment with various Regression Models & examine which gives the greatest accuracy.
- Examine Scope & correlations within our dataset.
- Determine which features are most important to predict the pricing and effect of other aspects of dataset.
- In the dataset we seen the type of properties and their popularity for booking.
- The people take booking decision according to the neighborhood and security of that property.

[B] Abstract

- In this Airbnb Dataset we have seen the lots of features which directly affect the price of the bookings. The conditions we have to see for prediction is that how Ratings, Reviews, Neighborhood and etc. are the component which we have to consider as the independent features and price is our dependent because the price is always have variation according to their place, ambience and amenities which are offer with the property for deciding the valuation of our places.
- The Aim of our project is that we have to consider the problem statements and works on that with the help of Exploratory Data Analysis and with Python Language.
- To deal with the problem there is essential need of prediction system for Price with considering Destination and important aspects. Machine learning is the branch of Artificial Intelligence (AI), it provides prestigious support in predicting any kind of event which take training from company datasheet and their sales. In this paper, we calculate accuracy of machine learning algorithms for predicting the best in class price for customers.
- To Deal with this several problems we use Machine Learning Algorithm to train our data.

[C] Methodology

Objective and Methodology of the Project:

- Python Programming
- Python Libraries
- Importing Libraries
- Machine Learning Implementation
- Machine Learning Algorithms

Python Programming

- Machine learning and AI, as a unit, are still developing but are rapidly growing in usage due to the need for automation.
- Artificial Intelligence makes it possible to create innovative solutions to common problems, such as Business Improving Strategy, Pricing, etc.
- The demand for smart solutions to real-world problems necessitates the need to develop AI further in order to automate tasks that are tedious to program without AI.
- Python programming language is considered the best algorithm to help automate such tasks, and it offers greater simplicity and consistency than other programming languages.
- Further, the presence of an engaging python community makes it easy for developers to discuss projects and contribute ideas on how to enhance their code.
- Python includes a modular machine learning library known as SciKit Learn, which provides easy-to-use algorithms for use in machine learning tasks. The best and most reliable coding solutions require a proper structure and tested environment, which is available in the Python frameworks and libraries.

Python Libraries

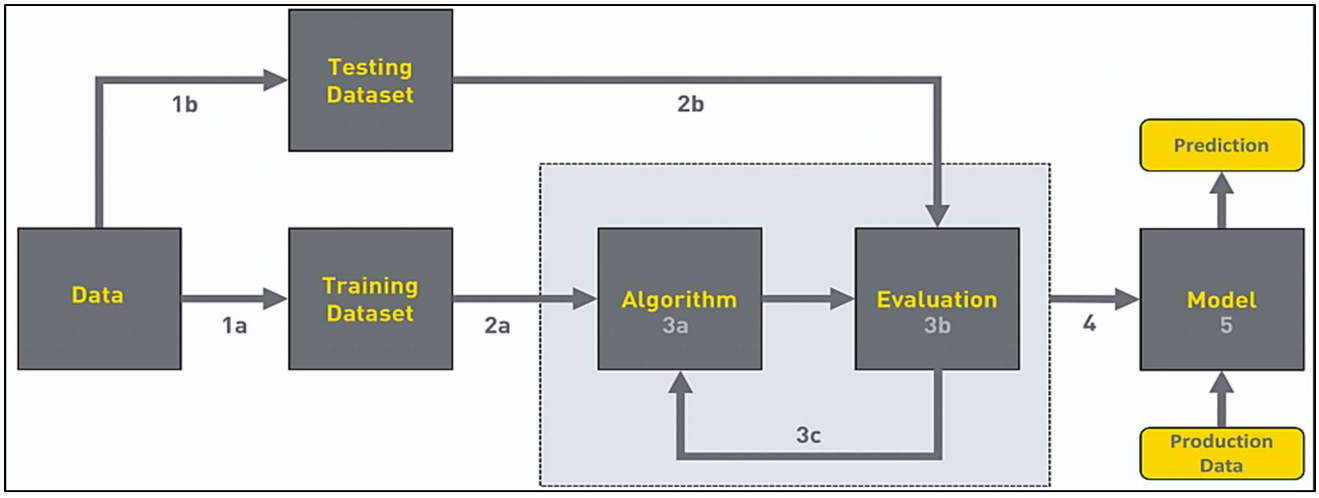
- Numpy = Use for high computation numerical operation. It consists of in-built mathematical functions for easy computations.
- Pandas = It eases data analysis, data manipulation, and cleaning of data. Pandas support operations like Sorting, Re-indexing, Iteration, Concatenation, Conversion of data, Visualizations, Aggregations, etc.
- matplotlib = This library is responsible for plotting numerical data. And that's why it is used in data analysis. It is also an open-source library and plots high-defined figures like pie charts, histograms, scatterplots, graphs, etc.
- Seaborn = Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.
- Scikit Learn = It supports variously supervised and unsupervised algorithms like linear regression, classification, clustering, etc. This library works in association with Numpy and SciPy.
- Warnings.filterwarnings = The warnings filter is initialized by (-w) options passed to the Python interpreter command line and the python warnings environment variable. The interpreter saves the arguments for all supplied entries without interpretation system warning option; the warnings module parses these when it is first imported (invalid options are ignored, after printing a message to system error).

Machine Learning Implementation

- The Algorithms we used are Linear Regression, Decision Tree Regressor, Random Forest Regressor and XGBoost Regressor by using Kaggle repository dataset. For implementation of Python programming Anaconda (Jupyter notebook) is best tool, which have many type of library that make the work more accurate and precise.

Use of Regressor in the Machine Learning Models:

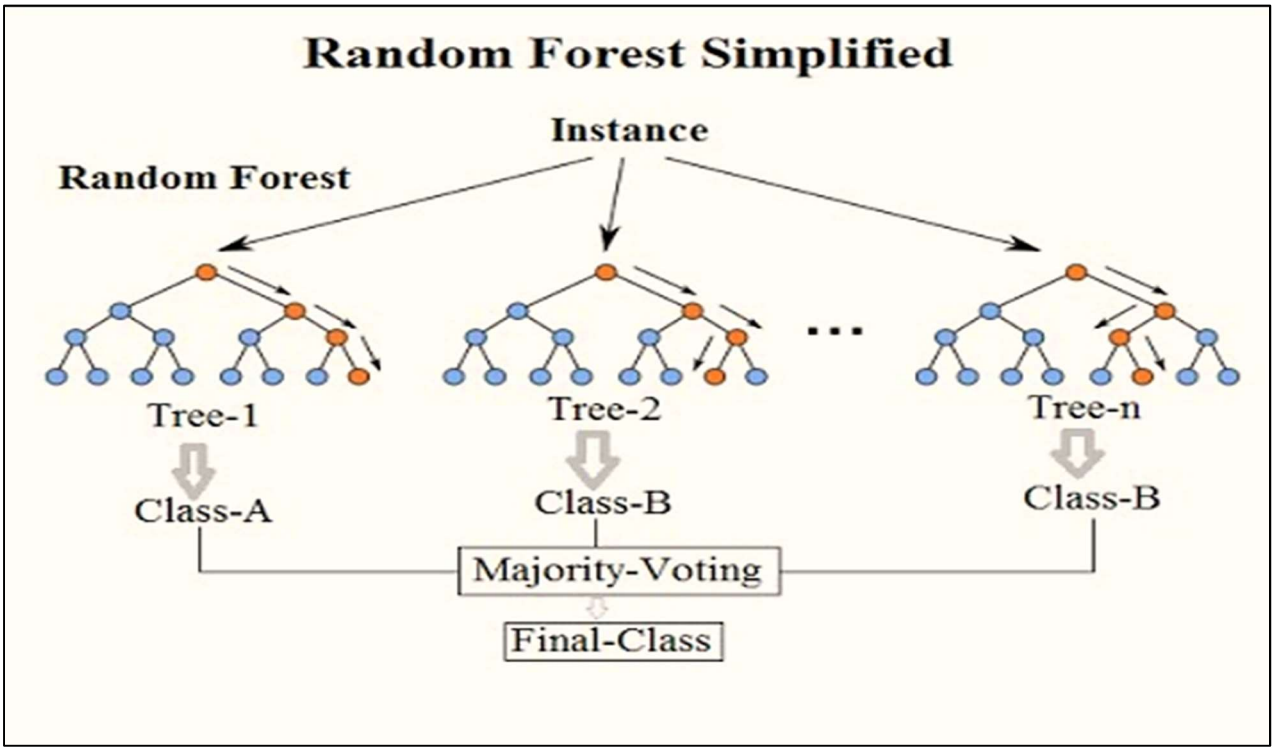
- Regression is a supervised machine learning technique which is used to predict Continuous values. The ultimate goal of the regression algorithm is to plot a best-fit line or a curve between the data.
- Machine Learning Regression is a technique for investigating the relationship between independent variables or features and a dependent variable or outcome. It's used as a method for predictive modelling in machine learning, in which an algorithm is used to predict continuous outcomes.
- There is no categorical Values in the dataset so, will not use the Classifiers in the Machine Learning Models.
- That's why we use Regressor Model in this Dataset.



Machine Learning Algorithms

[1] Random Forest:

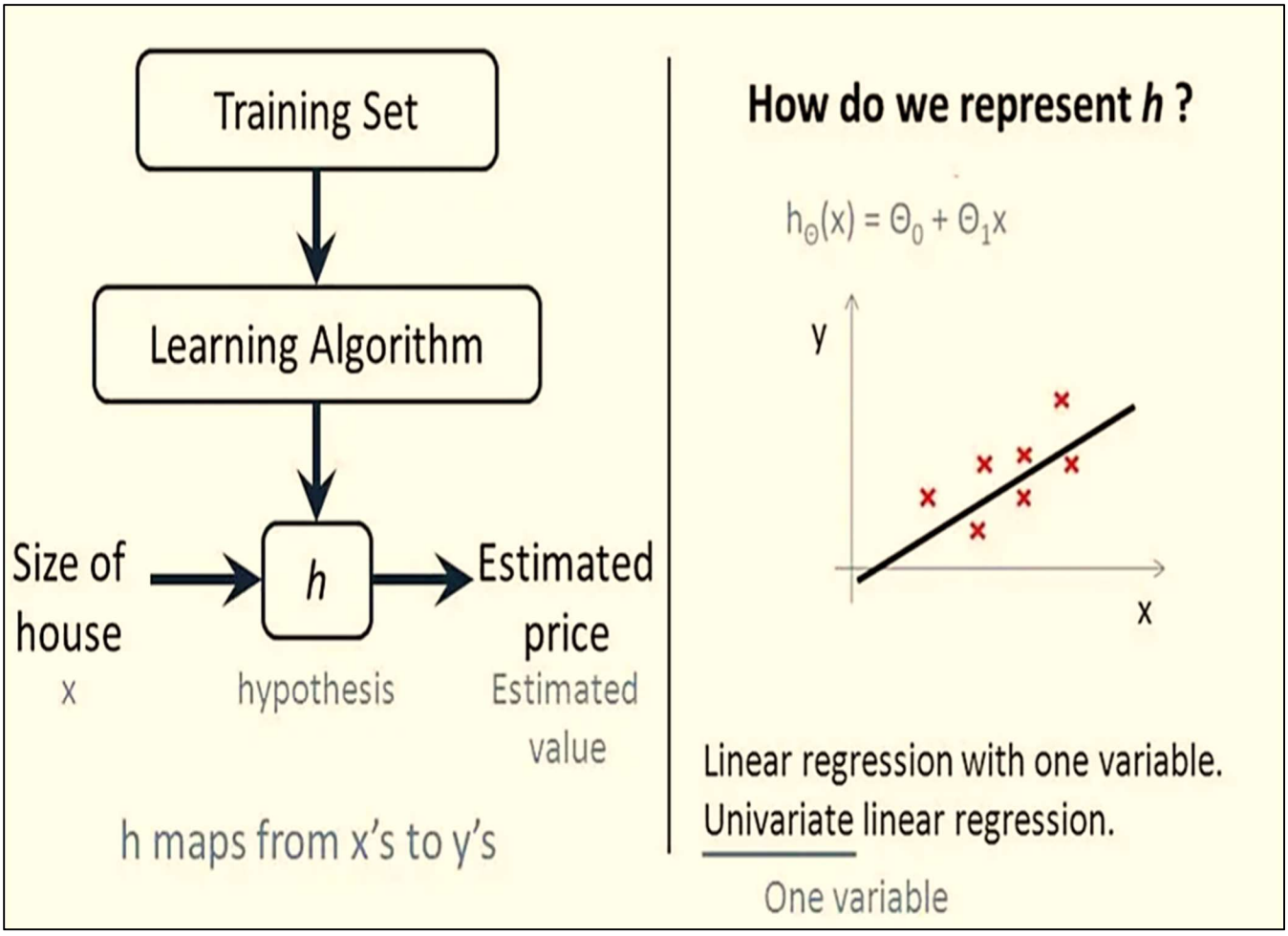
Random Forest is the most famous and it is considered as the best algorithm for machine learning. It is a supervised learning algorithm. **To achieve more accurate and consistent prediction, random forest creates several decision trees and combines them together.** The major benefit of using it is its ability to solve both regression and classification issues. When building each individual tree, it employs bagging and feature randomness in order to produce an uncorrelated tree forest whose collective forecast has much better accuracy than any individual tree's prediction. Bagging enhances accuracy of machine learning methods by grouping them together. In this algorithm, during the splitting of nodes it takes only random subset of nodes into an account. When splitting a node, it looks for the best feature from a random group of features rather than the most significant feature. This results into getting better accuracy. It efficiently deals with the huge datasets. It also solves the issue of overfitting in datasets. It works as follows: First, it'll select random samples from the provided dataset. Next, for every selected sample it'll create a decision tree and it'll receive a forecasted result from every created decision tree. Then for each result which was predicted, it'll perform voting and through voting it will select the best predicted result.



[2] Linear Regression:

Linear regression is one of the easiest and most popular Machine Learning algorithms. It is a statistical method that is used for predictive analysis. Linear regression makes predictions for continuous/real or numeric variables such as **sales, salary, age, product price,** etc.

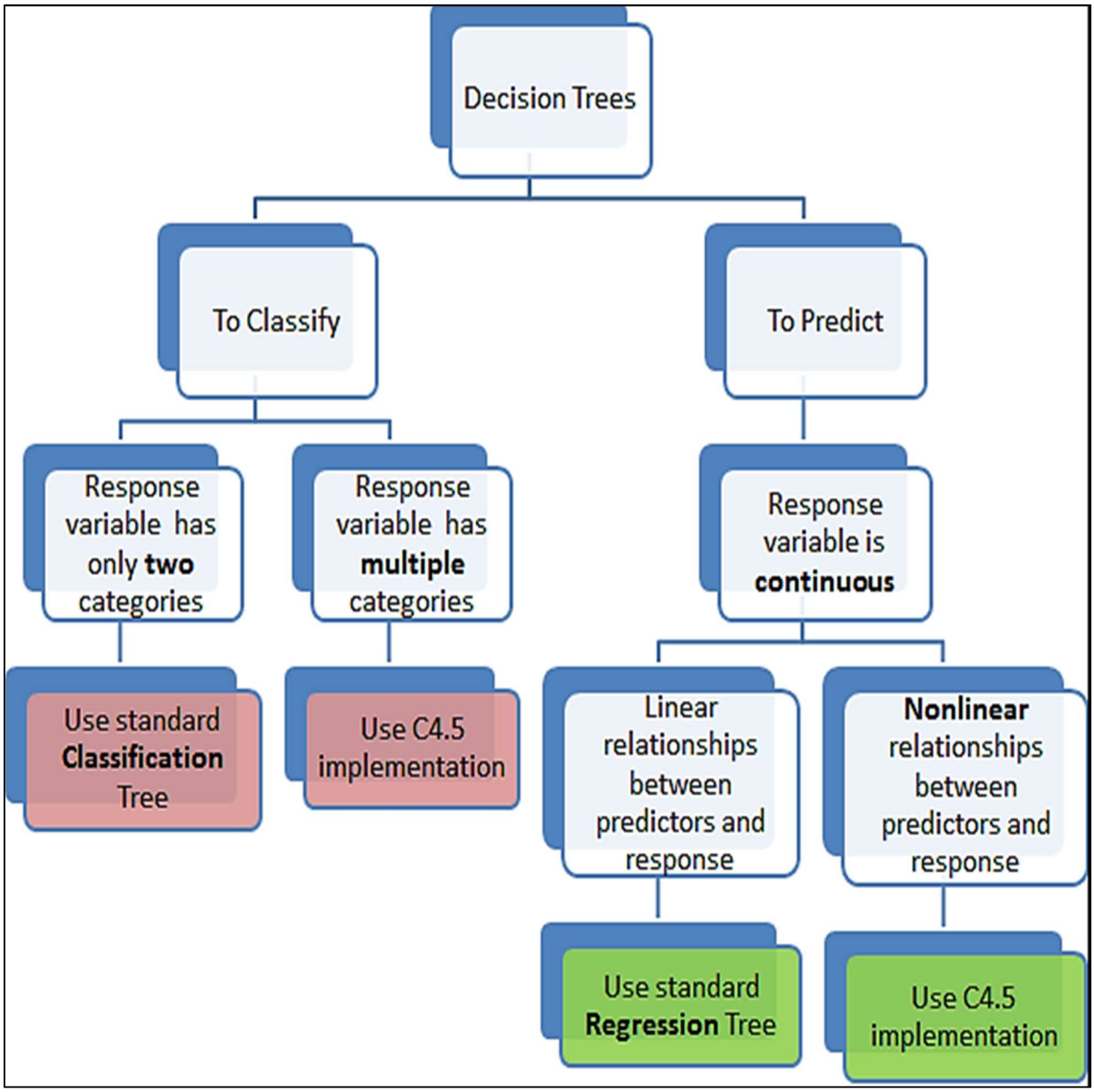
Linear regression algorithm shows a linear relationship between a dependent (y) and one or more independent (x) variables, hence called as linear regression. Since linear regression shows the linear relationship, which means it finds how the value of the dependent variable is changing according to the value of the independent variable.



Formula: Y = MX + C (Slope Formula).

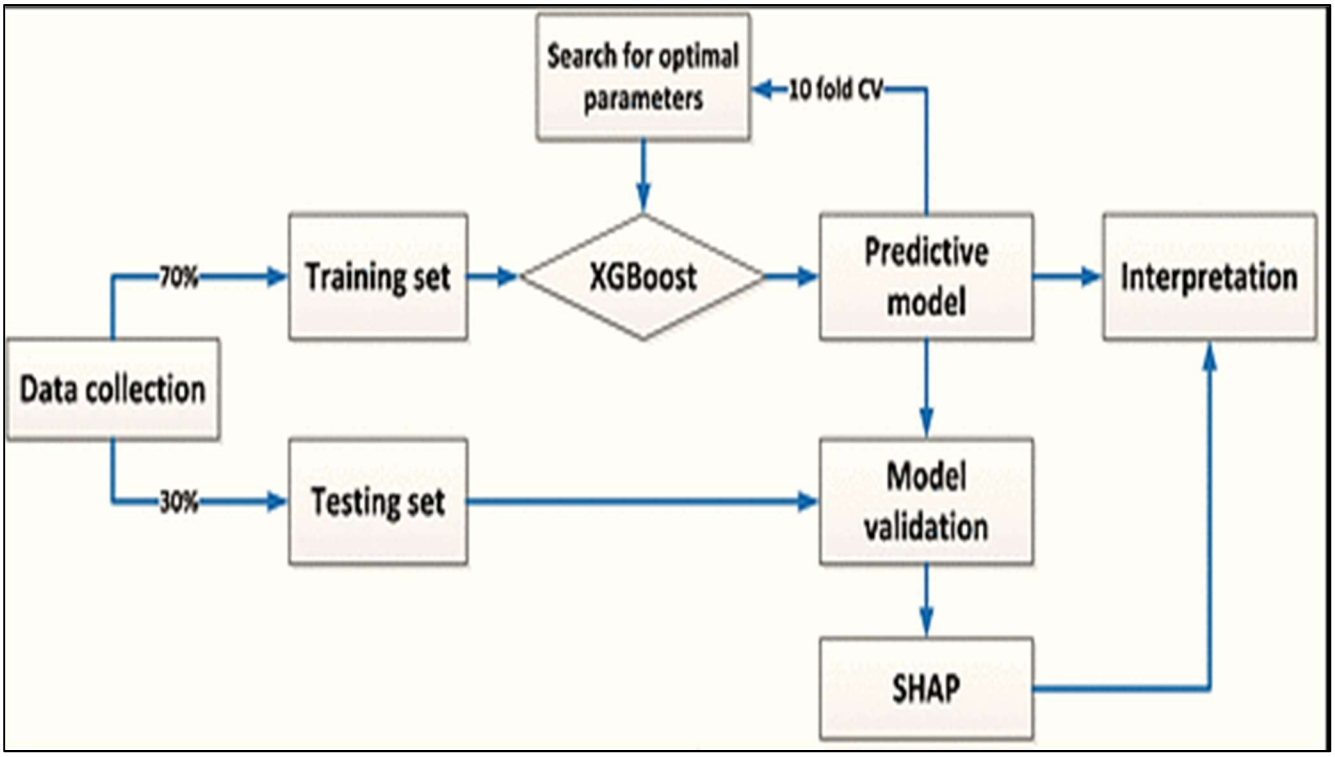
[3] Decision Tree Regressor:

The goal of using a Decision Tree is to create a training model that can use to predict the class or value of the target variable by learning simple decision rules inferred from prior data (training data). In Decision Trees, for predicting a class label for a record we start from the root of the tree. We compare the values of the root attribute with the record's attribute. On the basis of comparison, we follow the branch corresponding to that value and jump to the next node.



[4] XGBoost Regressor:

XGBoost is a powerful approach for building supervised regression models. The validity of this statement can be inferred by knowing about its (XGBoost) objective function and base learners. The objective function contains loss function and a regularization term. It tells about the difference between actual values and predicted values, i.e. how far the model results are from the real values. The most common loss functions in XGBoost for regression problems is reg: linear, and that for binary classification is reg: logistics. Ensemble learning involves training and combining individual models (known as base learners) to get a single prediction, and XGBoost is one of the ensemble learning methods. XGBoost expects to have the base learners which are uniformly bad at the remainder so that when all the predictions are combined, bad predictions cancels out and better one sums up to form final good predictions.



Implementation on Dataset

As we already discussed in the methodology section about some of the implementation details. So, the language used in this project is Python programming. We're running python code in anaconda navigator's Jupyter notebook. Jupyter notebook is much faster than Python IDE tools like PyCharm or Visual studio for implementing ML algorithms. The advantage of Jupyter notebook is that while writing code, it's really helpful for Data visualization and plotting some graphs like histogram and heatmap of correlated matrices. Let's revise implementation steps.

a) Dataset collection.

b) Importing Libraries: Numpy, Pandas, Scikit-learn, Matplotlib and Seaborn libraries were used.

c) Exploratory data analysis: For getting more insights about data.

d) Data cleaning and preprocessing: Checked for null and junk values using isnull() and isna().sum() functions of python .In Preprocessing phase, we did feature engineering on our dataset. As we converted categorical variables into numerical variables using function of Pandas library.

e) Feature Scaling: In this step, we normalize our data by applying Standardization by using fit_transform() functions of scikit-learn library.

f) Model selection : We first separated X's from y's. X's are features or input variables of our datasets and y's are dependent or target variables which are crucial for predicting disease. Then using by the importing model_selection function of the sklearn library, we splitted our X's and y's into train and test split using train_test_split() function of sklearn. We splitted 70% of our data for training and 30% for testing.

g) Applied ML models

h) Deployment of the model which gave the best accuracy.

Importing Libraries

Python:

- **import** numpy **as** np
- **import** pandas **as** pd
- **import** matplotlib.pyplot **as** plt
- **import** seaborn **as** sns
- **import** warnings warnings.filterwarnings('ignore')

Machine Learning:

- **from** sklearn.model_selection **import** train_test_split
- **from** sklearn.linear_model **import** LinearRegression
- **from** sklearn.tree **import** DecisionTreeRegressor
- **from** sklearn.ensemble **import** RandomForestRegressor
- **import** xgboost **as** xgb

Section 3

[A] Analysis of Dataset

1) Data Wrangling:

In this we have to read the raw csv file with the help of pandas library , which gives us access to get the information of feature and record. So after analyzing the data we can modifies and clean the dataset.

- **df = pd.read_csv('AirBnb.csv')**

3.1) Shape of our dataset.

```
1 df.shape
```

```
(102599, 26)
```

Observation = This dataset consist of Records:102599 and Features:26.

Total Values in our dataset is 26,67,574 .

```
1 df.isnull().sum().sum()
```

```
190769
```

Obervation = Total Null Value in this dataset is 190769.

3.4) Information of Features.

```
1 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 102599 entries, 0 to 102598
Data columns (total 26 columns):
 #   Column                          Non-Null Count   Dtype
---  ------                          --------------   -----
 0   id                              102599 non-null  int64
 1   NAME                            102349 non-null  object
 2   host id                         102599 non-null  int64
 3   host_identity_verified          102310 non-null  object
 4   host name                       102193 non-null  object
 5   neighbourhood group             102570 non-null  object
```

3.4) Description of the dataset.

```
1 df.describe()
```

	id	host id	lat	long
count	1.025990e+05	1.025990e+05	102591.000000	102591.000000
mean	2.914623e+07	4.925411e+10	40.728094	-73.949644
std	1.625751e+07	2.853900e+10	0.055857	0.049521
min	1.001254e+06	1.236005e+08	40.499790	-74.249840
25%	1.508581e+07	2.458333e+10	40.688740	-73.982580
50%	2.913660e+07	4.911774e+10	40.722290	-73.954440
75%	4.320120e+07	7.399650e+10	40.762760	-73.932350
max	5.736742e+07	9.876313e+10	40.916970	-73.705220

Observations: The following images are representation of certain csv file reading and data information which gives us brief introduction about our dataset and their numerical observations.

- This dataset consist of Records: 102599 and Features: 26.
- Total Values in our dataset is 26, 67,574.
- Total Null Value in this dataset is 190769.
- The info () method prints information about the Data Frame. The information contains the number of columns, column labels, and column data types.
- The describe () = This method computes and displays summary statistics for a Python data Frame.

2) Data Cleaning/Preparation:

- As the dataset from kaggle was not very suitable for data analysis, we had to change the format of some data in the dataset. We also had to do separate data preparation for exploratory analysis and machine learning.

- Some of our data preparation were:

  I. Convert price of listings from strings to floats and also remove the '$' sign.
  II. Change NaN values to the integer 0.
  III. Cleaning the textual data into a form that would be suitable for Python's Jupyter Notebook.
  IV. Encoding the categorical variables so that it can be fit into the regression models later.
  V. Separating the data into Predicted and Target variables.
  VI. Separating the data into training and testing sets (Training Sets: Testing Sets = 70% : 30%).

4.1) Remove Duplicates

```
1  df.drop_duplicates(inplace=True)
```

```
1  df.shape
```

```
(102058, 26)
```

4.2) Change in Values Type

```
1  df['last review'] = pd.to_datetime(df['last review'])
```

- Convert 'last review' column to Datetime type.

```
1  df['price'] = df['price'].str.replace('$', '')
2  df['price'] = df['price'].str.replace(',', '')
3  df['price'] = df['price'].astype(float)
```

```
df['service fee'] = df['service fee'].str.replace('$', '')
df['service fee'] = df['service fee'].str.replace(',', '')
df['service fee'] = df['service fee'].astype(float)
```

4.3) Treating Null Values

```
df = df.dropna(subset=['price','NAME','host_identity_verified'])
df = df.dropna(subset=['neighbourhood group','neighbourhood'])
df = df.dropna(subset=['cancellation_policy'])
df = df.dropna(subset=['instant_bookable', 'host name','Construction year'])
df = df.dropna(subset=['lat','long'])
```

4.4) Filllng Null Values

```
df['service fee'] = df['service fee'].fillna(0)
```

- Observation = Fill null values in service fee by zeros.

Deal with nulls value by fill with mean and medium.

```
df["reviews per month"].fillna(df["reviews per month"].mean(),inplace=True)
df["minimum nights"].fillna(df["minimum nights"].mean(),inplace=True)
df["availability 365"].fillna(df["availability 365"].mean(),inplace=True)
df["calculated host listings count"].fillna(df["calculated host listings count"].mean(),
                                            inplace=True)
df["number of reviews"].fillna(df["number of reviews"].median(), inplace=True)
```

Fill null date

```
num_days = 10
g = df['last review'].notna().cumsum()
days = pd.to_timedelta(df.groupby(g).cumcount().mul(num_days), unit='d')
df['last review'] = df['last review'].ffill().add(days)
```

- Filling the null date values in last review feature.

```
for col in df.columns:
        if df[col].value_counts().shape[0] == 1:
            df = df.drop(col, axis=1)
```

- Drop columns where value counts is equal to 1.

Observation of Null Value Treatment:

i. 541 duplicated rows has been removed.
ii. After dropping we concludes that we Record=102058 and Feature=24 are left.
iii. Convert Special Character from the integer for readable to machine.
iv. Dropping Column Information:

- 247 row deleted from price
- 250 row deleted from name
- 273 row deleted from host identity verified
- 29 row deleted from neighborhood group
- 76 row deleted from cancellation policy
- 105 row deleted from instant bookable
- 182 row deleted from Construction year
- 390 row deleted from host name
- 14 row deleted from neighborhood
- 7 row deleted from lat & long

Total Rows deleted = 1,442

v. Fill null values in service fee by zeros.
vi. Filling the null date values in last review feature.

3) Feature Engineering:

In the Feature Engineering we change the data type of feature for cleaning and better understanding of data so we can analyze and implement the algorithm on it.

```
df['availability 365'] = np.where(df['availability 365']<0,
                                  df['availability 365']*-1, df['availability 365'])
```

- remove 'availability 365' values that less than 0
- np.where = The numpy.where() function returns the indices of elements in an input array where the given condition is satisfied.

```
df['availability 365'] = np.where(df['availability 365']>365, 365, df['availability 365'])
```

- remove 'availability 365' values that more than 365.

```
df['minimum nights'] = np.where(df['minimum nights']<0,
                                df['minimum nights']*-1, df['minimum nights'])
```

- remove 'minimum nights' values that less than 0 .

Rename Feature

```
df.rename(columns = {'neighbourhood group':'neighbourhood_group'}, inplace = True)
```

```
df = df[df.neighbourhood_group != 'brookln']
```

- remove the row that neighbourhood group is brookln (outliers).

Data Type Conversion

For host_identity_verified

```
df['host_identity_verified'].value_counts()
```

```
unconfirmed    50352
verified       50242
Name: host_identity_verified, dtype: int64
```

```
df.host_identity_verified.replace({'unconfirmed':0 , 'verified':1}, inplace=True)
```

For instant_bookable

```
df['instant_bookable'].value_counts()
```

```
False    50508
True     50086
Name: instant_bookable, dtype: int64
```

```
df.instant_bookable.replace({False:0, True:1}, inplace=True)
```

For neighbourhood_group

```
df['neighbourhood_group'].value_counts()
```

```
Manhattan        42943
Brooklyn         41017
Queens           13043
Bronx             2658
Staten Island      933
Name: neighbourhood_group, dtype: int64
```

```
df.neighbourhood_group.replace({'Brooklyn' : 1 , 'Manhattan' : 2 ,
                                'Bronx' : 0 , 'Queens' : 3 , 'Staten Island' : 4}, inplace=True)
```

For Room Type

```
df['room type'].value_counts()
```

```
Entire home/apt    52691
Private room       45606
Shared room         2182
Hotel room           115
Name: room type, dtype: int64
```

```
df['room type'].replace({'Hotel room' : 1 , 'Private room' : 2 ,
                         'Entire home/apt' : 0 , 'Shared room' : 3}, inplace = True)
```

For cancellation_policy

```
df['cancellation_policy'].value_counts()
```

```
moderate    33708
strict      33478
flexible    33408
Name: cancellation_policy, dtype: int64
```

```
df.cancellation_policy.replace({'moderate' : 1 , 'strict' : 2 , 'flexible' : 0}, inplace = True)
```

Feature Observations:

i. **availability 365**: Remove the negative values and equal to zero values.
ii. **neighbourhood_group**: 5 Type of cites which converted into integers.
iii. **host_identity_verified**: 2 Types of category to confirm host.
iv. **instant_bookable**: It is Binary type in Yes No condition.
v. **cancellation_policy**: 3 Type of options in the booking.
vi. **room type**: 4 Types of room option in the properties.

4) Exploratory Data Analysis & Problem Formulation on Airbnb Dataset:

[A] What are the factors and features of a listing that make an Airbnb listing more Expensive? (**Problem Statements**)

- Problem 1: Host identity verification affect the Airbnb pricing?
- Problem 2: Reviews for Airbnb which effect the Airbnb?
- Problem 3: What is the range of prices in Airbnb?
- Problem 4: Availability according to months.

[B] **Questionnaires**

- How many days do the owners of the house receive the guest in their house throughout the year.
- Which neighborhood do you give the highest ratings?
- What is the range of prices?
- Host identity verified effect on price?
- Relationship with neighborhood group and the price?
- What are the proportions of room types?
- Most Construction year?

Problem Statements

[1] Problem Statement 1 = Host Identity verification.

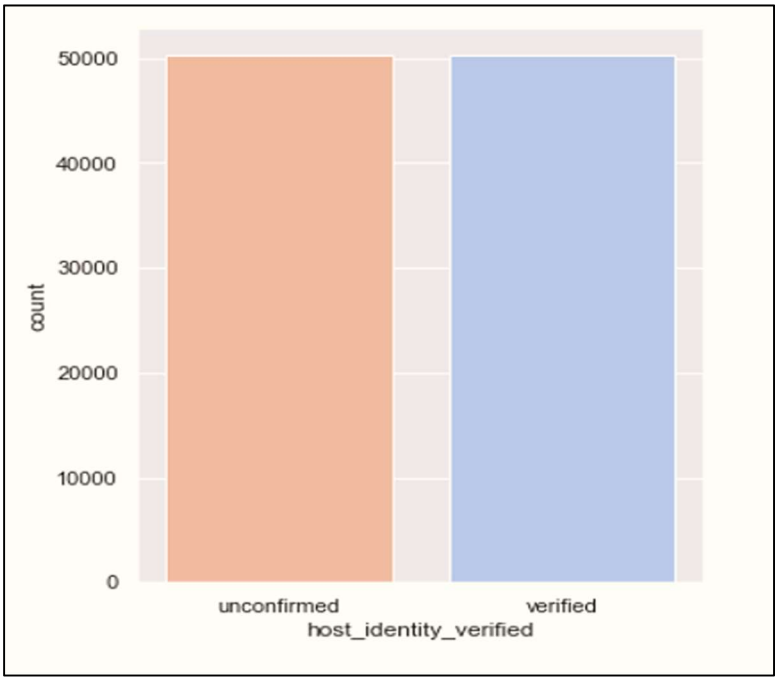
A) Number of Verification and Non-Verification of the Identity of the host.

```
1 df['host_identity_verified'].value_counts()
```

```
unconfirmed    50352
verified       50242
Name: host_identity_verified, dtype: int64
```

B) Count Plot for Host Identity Verification.

```
1 plt.figure(figsize=(5,6))
2 plt.xlabel("feature: {}".format(feature))
3 sns.countplot(data=df, x='host_identity_verified', palette='coolwarm_r')
4 plt.show()
```

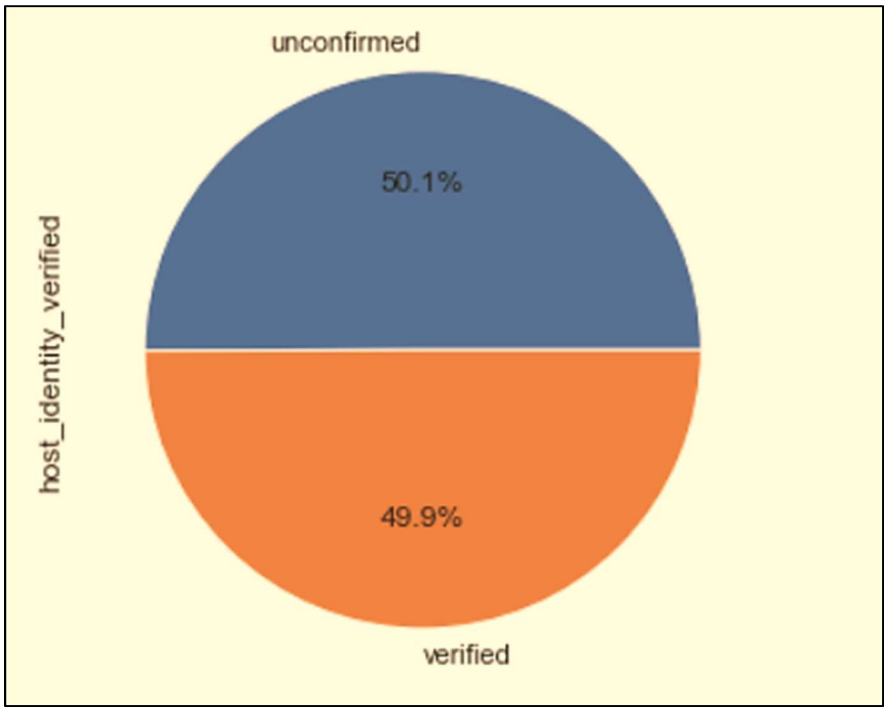


Observation = There Host which Verified thier ID and Unconirm also on the basis of dataset we concludes that there is following numbers. ¶

- Unconfirmed = 50352
- Verified = 50242

C) PIE Plot for Host Identity Verification.

```
plt.figure(figsize=(5,6))
df['host_identity_verified'].value_counts().plot.pie(autopct='%1.1f%%')
plt.show()
```



Observation = From the plot , the number of verification and non-verification is very close, which causes insecurity for the guest, and we can fix this and ensure a more level of safety for the guest by working to increase the number of verification.

➢ Observation:

- **Count Plot** = In the count plot show the count of Unconfirmed = 50352 & Verified = 50242 with counting visuals and color scheme makes it easy to understand the difference.
- **Pie Plot =** In the pie plot visualization we can observe total distribution of the both categories.

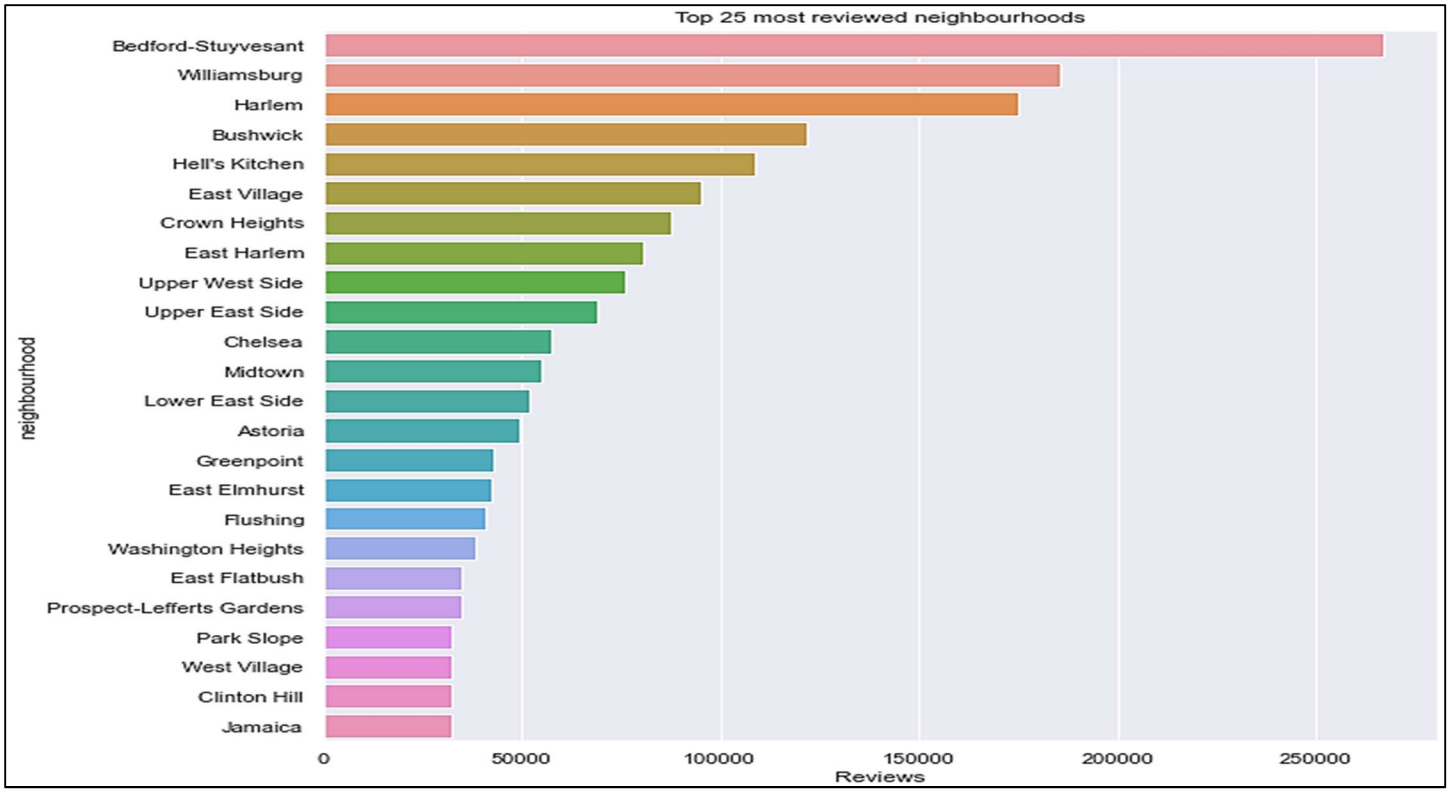
[2] Problem Statement 2 = Number of Reviews for Airbnb.

A) Top 25 reviewed neighbourhoods

```
reviewed_neighbourhoods = df.groupby(['neighbourhood'])
['number of reviews'].sum().sort_values(ascending=False)[0:24]
print(reviewed_neighbourhoods)
```

B) BAR Plot for number of reviews.

```
plt.figure(figsize=(10,10))
plt.title("Top 25 most reviewed neighbourhoods")
plt.ylabel('Neighborhood')
plt.xlabel('Reviews')
sns.barplot(x=reviewed_neighbourhoods.values,y=reviewed_neighbourhoods.index)
plt.show()
```



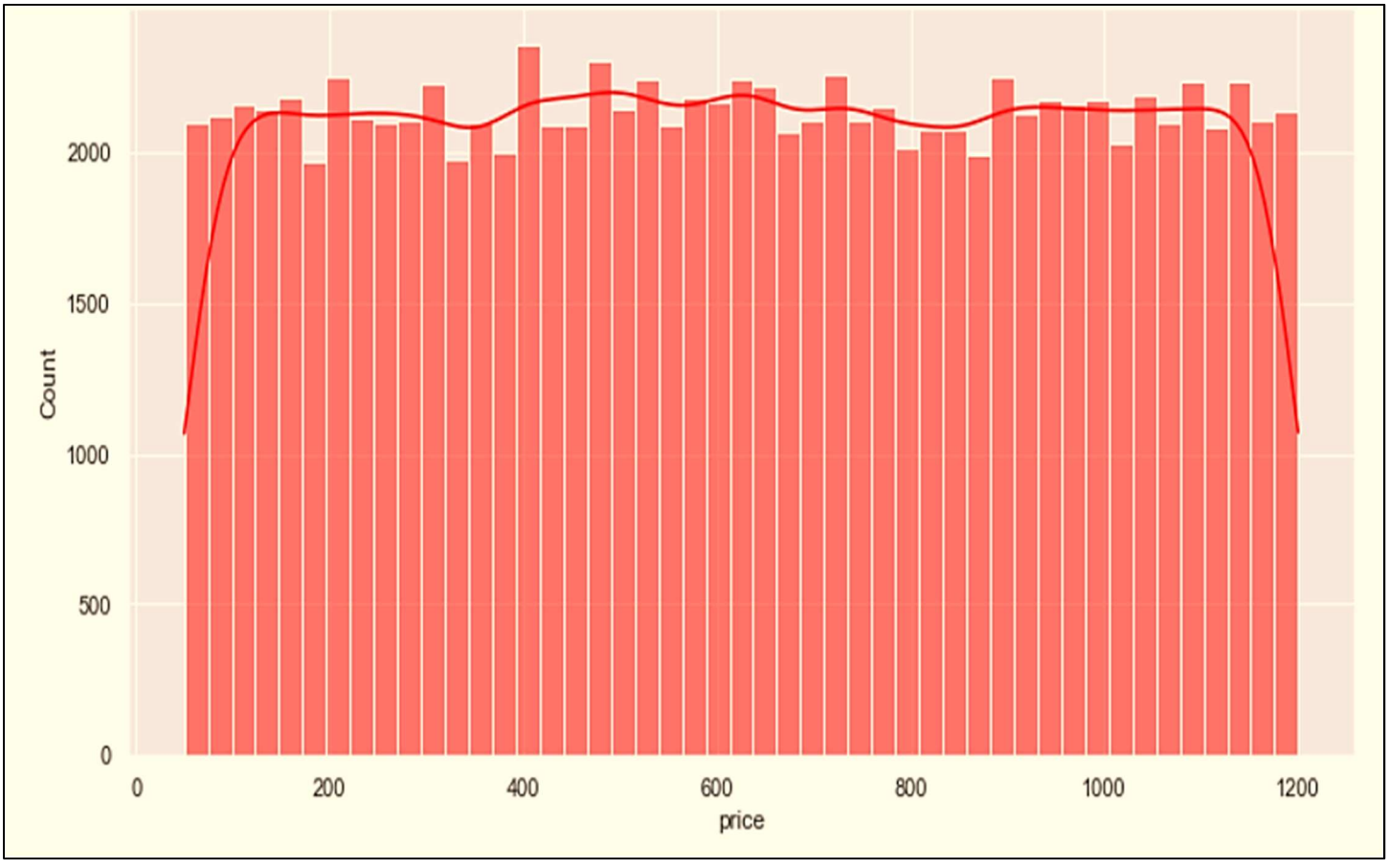
- **Observation =** 'Bedford – Stuyvesant' is on 1 Rank.
- **Bar plot =** It is a Horizontal Bar Plot which shows the reviews according to highest on the basis of neighborhood.

[3] Problem 3: What is the range of prices in Airbnb?

A) histplot

- A histogram is a classic visualization tool that represents the distribution of one or more variables by counting the number of observations that fall within discrete bins.

```
plt.subplots(figsize=(12, 6))
sns.set(font_scale=2)
sns.histplot(data=df, x="price", kde=True, color = 'red')
plt.show()
```



- **Observation**: The price range of properties is comes in between 50 to 1200 Dollar.

[4] Problem 4: Availability according to months.

Grouping the span of 3 months.

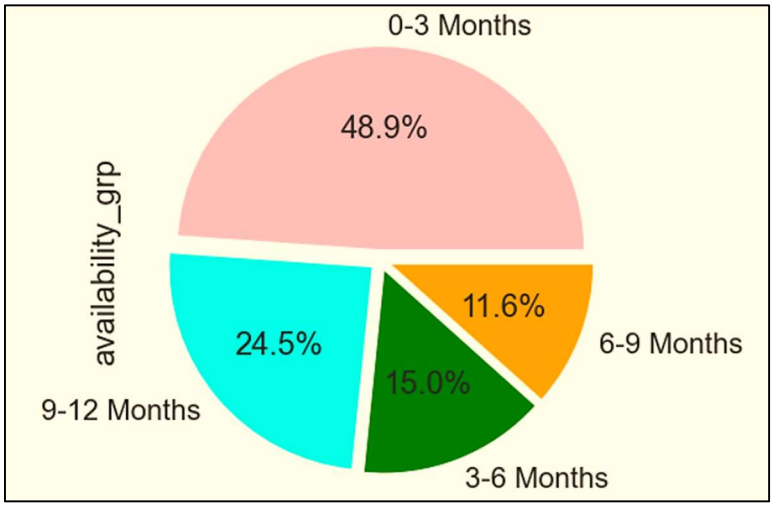
```
df['availability_grp'] = np.where(df['availability 365']<=90, '0-3 Months',
                    np.where((df['availability 365'] > 9) & (df['availability 365'] <= 180),
                             '3-6 Months',
                    np.where((df['availability 365'] > 180) & (df['availability 365'] <= 270),
                             '6-9 Months', '9-12 Months')))
```

```
df['availability_grp'].value_counts()
```

```
0-3 Months     49186
9-12 Months    24652
3-6 Months     15055
6-9 Months     11701
Name: availability_grp, dtype: int64
```

A) PIE Plot

```
round(df['availability_grp'].value_counts()/ df.shape[0]*100,2).plot.pie(autopct = '%1.1f%%',
figsize =(7, 10),explode=(0.05,0.05,0.05,0.05),colors=['pink','cyan','green','orange'])
plt.show()
```



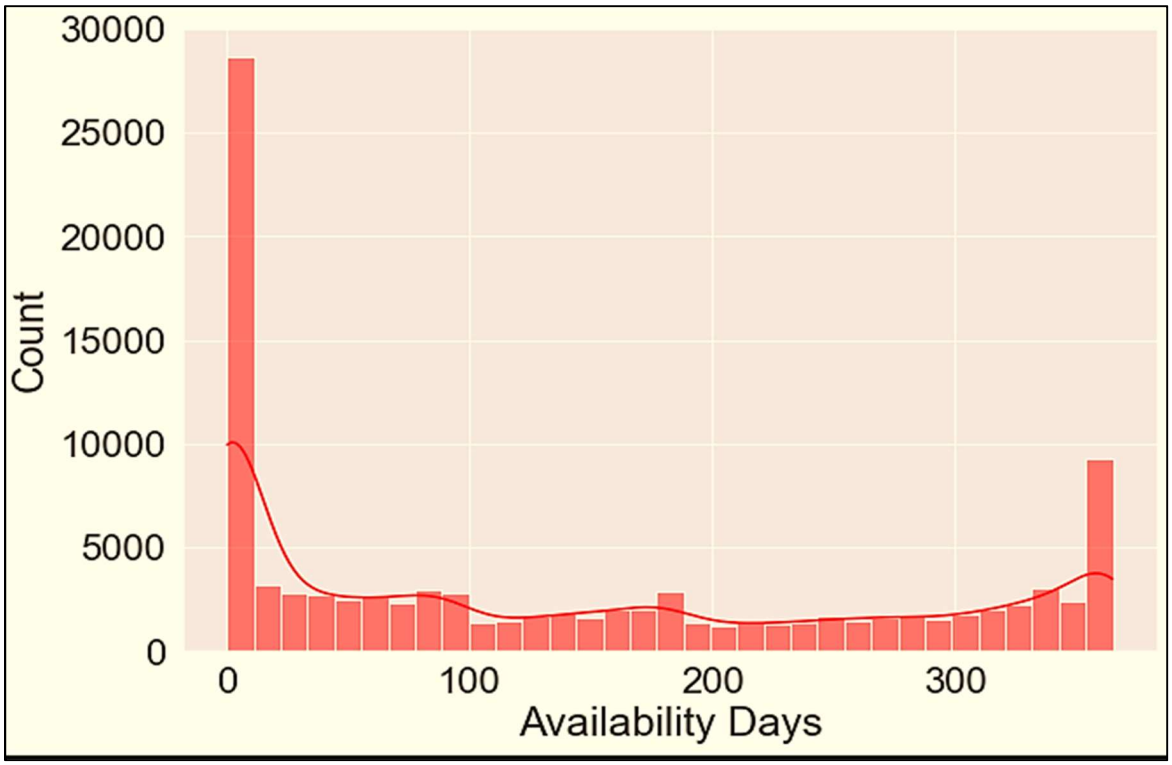
Observation = from 0-3 months is the most.

- Round in pie chart = The round() function returns a floating point number that is a rounded version of the specified number, with the specified number of decimals.
- Autopct = If you want to label the wedges with their numeric value in a pie chart, then use autopct parameter. This parameter allows us to display the percent value using string formatting.
- Explode = To "explode" a pie chart means to make one of the wedges of the pie chart to stand out.

Questionnaires

Q1) How many days do the owners of the house receive the guest in their house throughout the year.

```
plt.figure(figsize=(10,7))
plt.ylabel('Count')
plt.xlabel('Availability Days')
sns.histplot(data=df, x='availability 365', kde=True, color='red')
plt.show()
```

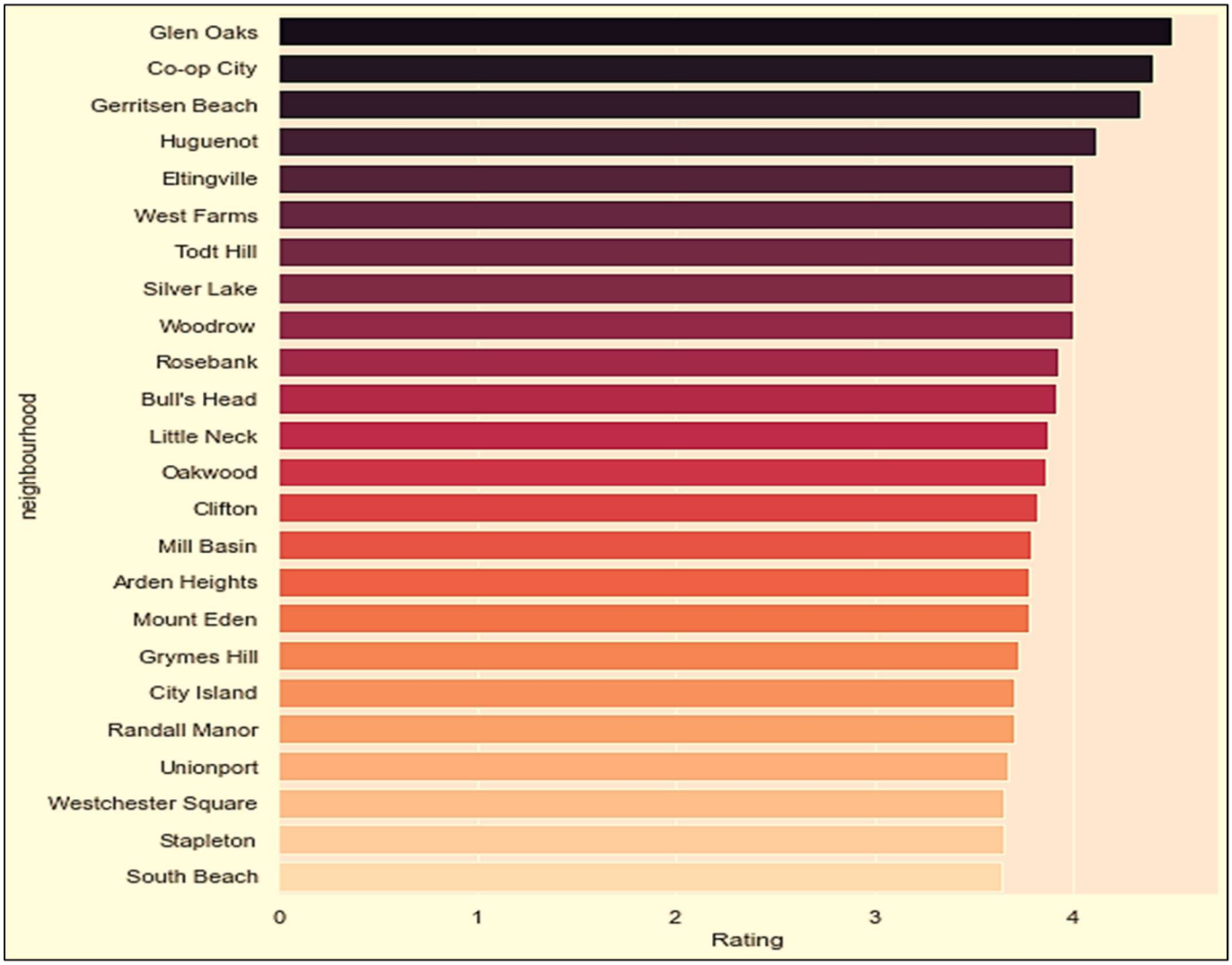


- **Observation** = 0 and 365 day is the most Available.
    - Starting and End of the Year the frequency is improve of guests.

Q2) Which neighborhood do you give the highest ratings

```
avg_rating_per_neighbourhood = df.groupby(['neighbourhood'])
['review rate number'].mean().sort_values(ascending=False)[0:24]
print(avg_rating_per_neighbourhood)
```

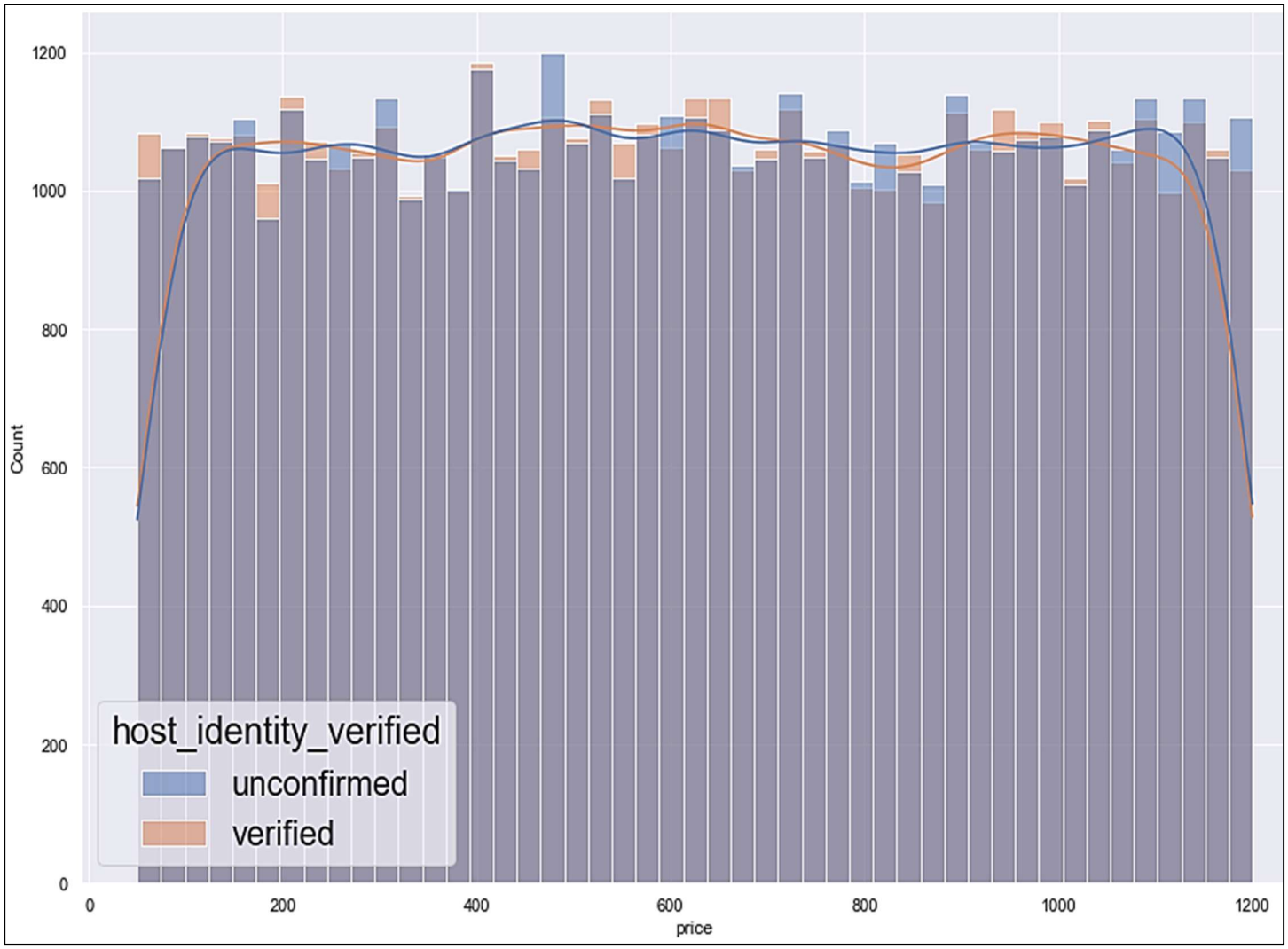
```
plt.figure(figsize=(8,10))
sns.set(font_scale = 1)
plt.ylabel('neighbourhood')
plt.xlabel('Rating')
sns.barplot(x=avg_rating_per_neighbourhood.values,y=avg_rating_per_neighbourhood.index,
            palette='rocket')
plt.show()
```



- **Observation** = Glen Oaks gives the highest ratings with 4.5 Rating.

Q3) What is the range of prices?

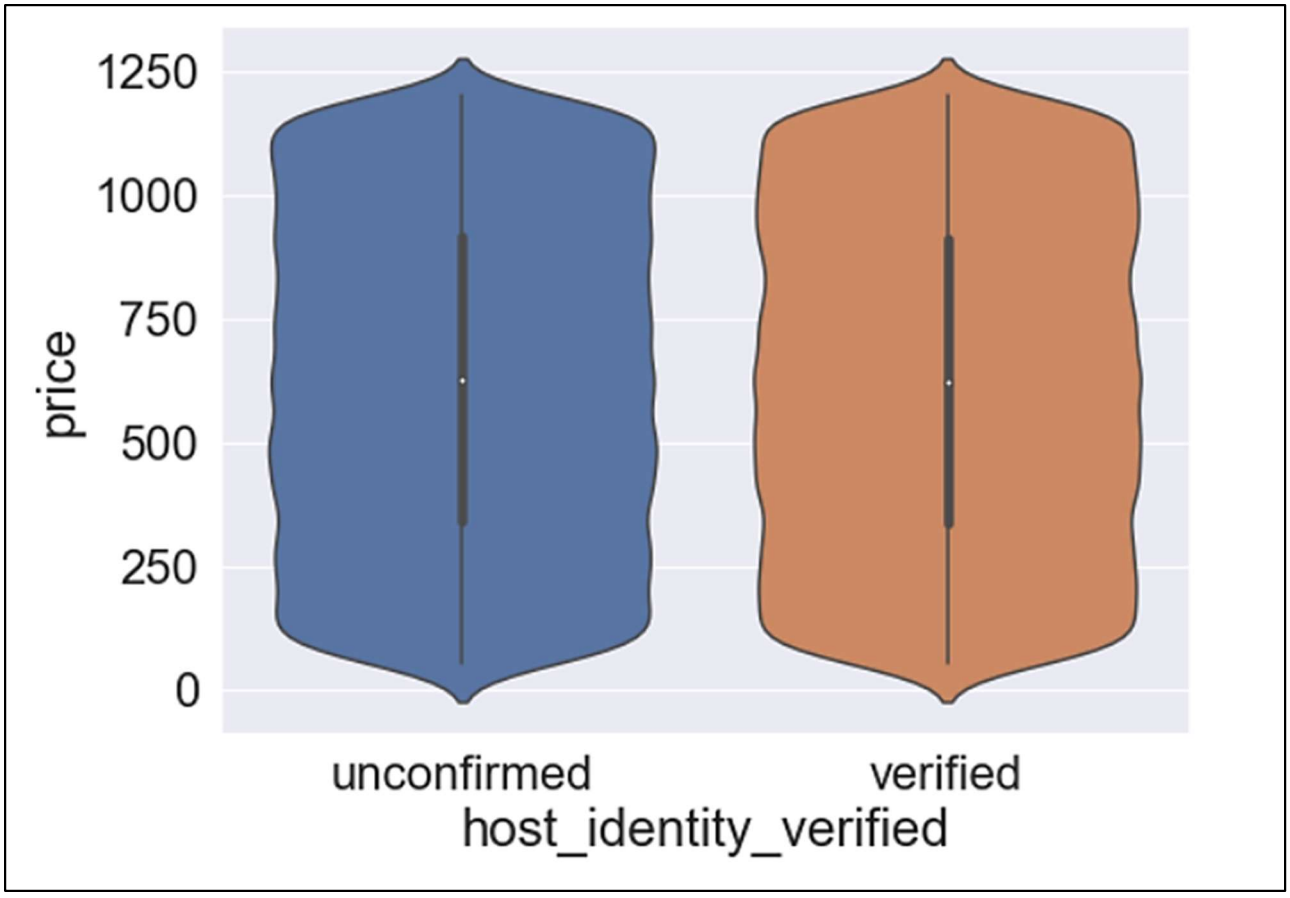
```
plt.subplots(figsize=(15, 10))
sns.set(font_scale=2)
sns.histplot(data=df, x="price", kde=True, color = 'red', hue='host_identity_verified')
plt.show()
```



- **Observation** = Prices range from 50 to 1200 dollars.

Q4) Host Identity verified effect on price?

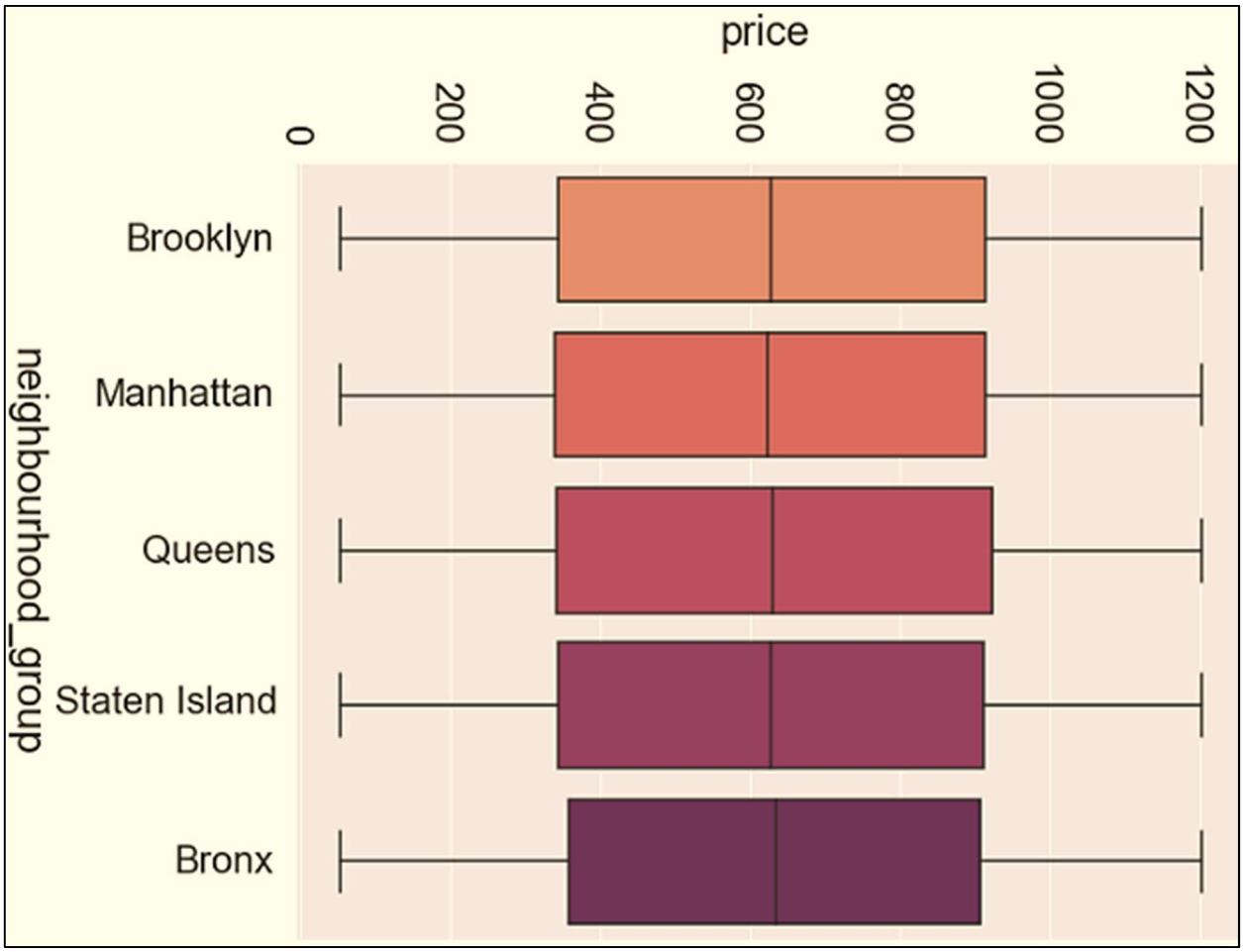
```
plt.subplots(figsize=(8,6))
sns.violinplot(x="host_identity_verified", y="price", data=df)
plt.show()
```



- **Observation** = No, There is no effect on price it's remain same.
- **Violin Plot =** violin plots are used to visualize data distributions, displaying the range, median, and distribution of the data.
  - The dataset should have continuous, numerical features. This is because Violin Plots are used to visualize distributions of continuous data. They display the range, median, and distribution of the data.

Q5) Relationship with neighborhood group and the price?

```
fig, ax = plt.subplots(figsize=(8,10))
sns.set(font_scale=1)
plt.xticks(rotation= 90)
sns.boxplot(x="neighbourhood_group", y="price", data=df,palette="flare")
plt.show()
```



Observation = There is no big difference between the neighborhood group and after them compared to the price, so it is almost the same price range.

- **An Average of $600 per night.**

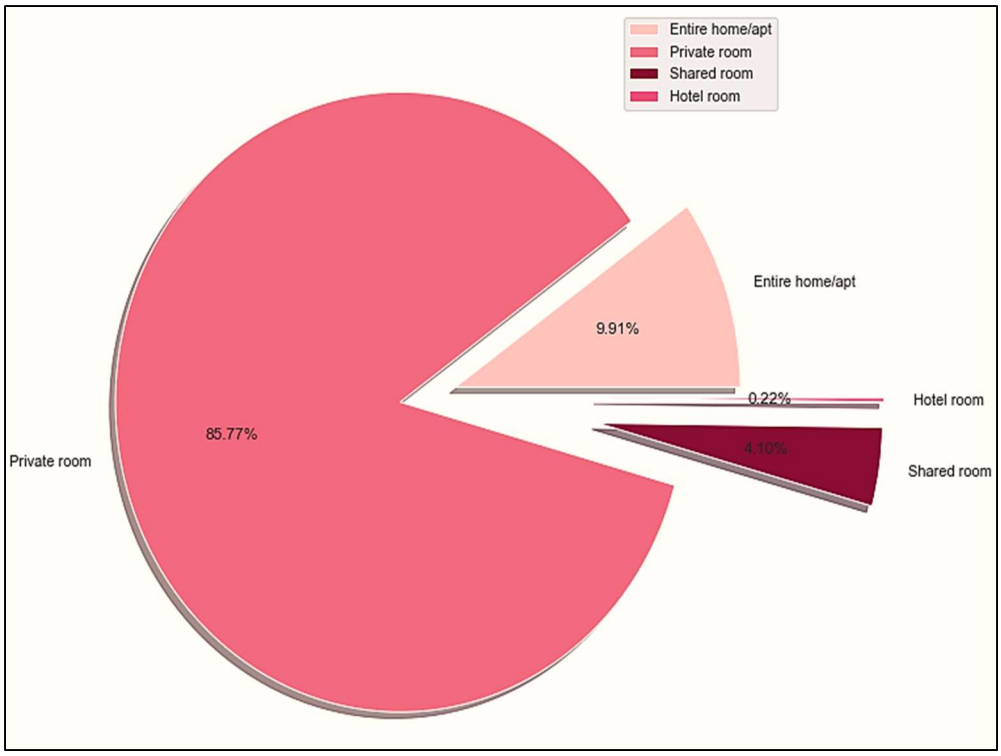
Q6) Type of Rooms.

```
1  df['room type'].value_counts()
```

```
Entire home/apt    52691
Private room       45606
Shared room         2182
Hotel room           115
Name: room type, dtype: int64
```

QA) What are the proportions of room types?

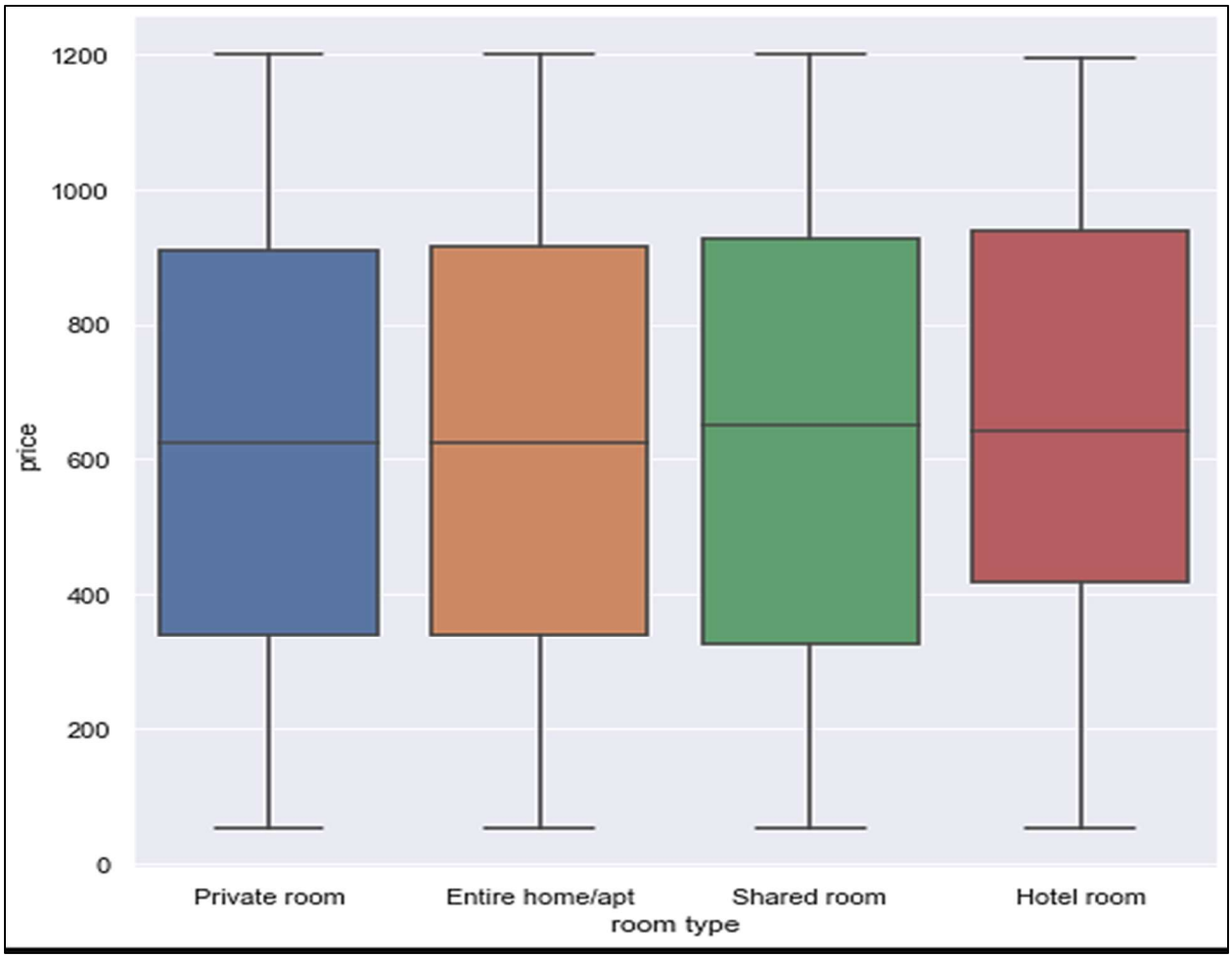
```
plt.figure(figsize= (10, 10))
data  = [5269,45606,2182,115]
labels = ["Entire home/apt", "Private room","Shared room","Hotel room"]
explode = [0.1,0.1,0.6,0.6]
plt.pie(data ,labels= labels , explode = explode , autopct="%1.2f%%", shadow= True,
        colors= ['#FFC4C4','#EE6983','#850E35','#e84a7e'])
plt.legend()
plt.show()
```



- **Observation** = Private Rooms have large proportion.
  - **Color Palettes** in Pink from ['#FFC4C4','#EE6983','#850E35','#e84a7e']

QB) Which of them is the one with the highest average mean.

```
sns.boxplot(x='room type',y='price',data=df)
plt.show()
```

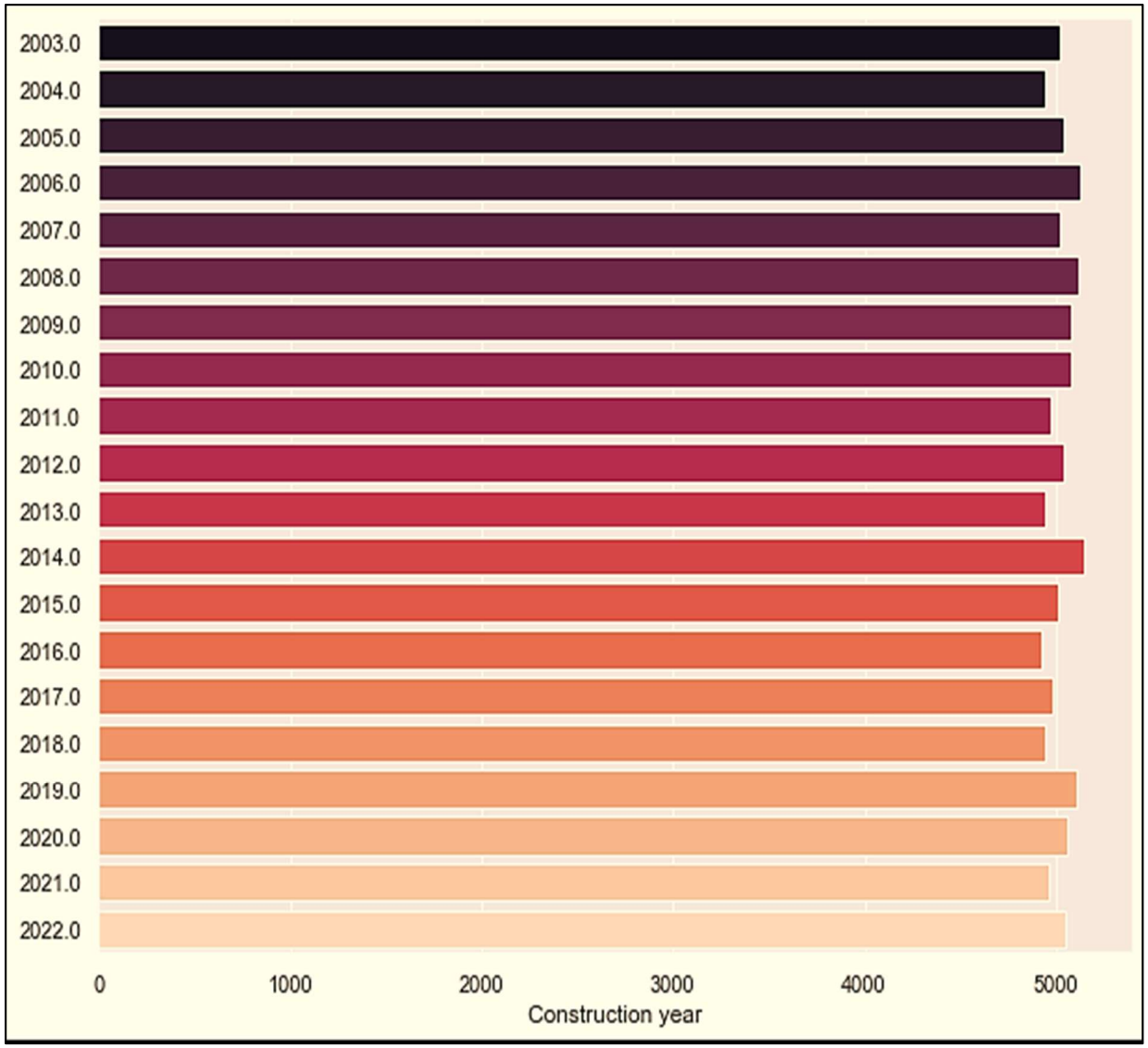


Boxplot Observation = There are no outliers in the distribution of prices in each borough, so the mean is not affected. Therefore, we can use it in order to know which of them the one with the highest average mean is.

Q7) Most Construction year.

```
plt.figure(figsize=(10,8))

sns.barplot(y=df['Construction year'].value_counts().index,
            x=df['Construction year'].value_counts(), palette="rocket", orient='h')
plt.show()
```



- **Observation** = It looks like there are about ~4,000 constructions per year on the dataset.

Machine Learning Model Implementation

(1) Convert Values

For host_identity_verified

```
1 df['host_identity_verified'].value_counts()
```

```
unconfirmed    50352
verified       50242
Name: host_identity_verified, dtype: int64
```

```
1 df.host_identity_verified.replace({'unconfirmed':0 , 'verified':1}, inplace=True)
```

```
1 df['instant_bookable'].value_counts()
```

```
False    50508
True     50086
Name: instant_bookable, dtype: int64
```

```
1 df.instant_bookable.replace({False:0, True:1}, inplace=True)
```

For cancellation_policy

```
1 df['cancellation_policy'].value_counts()
```

```
moderate    33708
strict      33478
flexible    33408
Name: cancellation_policy, dtype: int64
```

```
1 df.cancellation_policy.replace({'moderate' : 1 , 'strict' : 2 , 'flexible' : 0}, inplace = True)
```

For Room Type

```
df['room type'].value_counts()
```

```
Entire home/apt    52691
Private room       45606
Shared room         2182
Hotel room           115
Name: room type, dtype: int64
```

```
df['room type'].replace({'Hotel room' : 1 , 'Private room' : 2 , 'Entire home/apt' : 0
                         , 'Shared room' : 3}, inplace = True)
```

For neighbourhood_group

```
df['neighbourhood_group'].value_counts()
```

```
Manhattan        42943
Brooklyn         41017
Queens           13043
Bronx             2658
Staten Island      933
Name: neighbourhood_group, dtype: int64
```

```
df.neighbourhood_group.replace({'Brooklyn' : 1 , 'Manhattan' : 2 , 'Bronx' : 0 ,
                                'Queens' : 3 , 'Staten Island' : 4}, inplace=True)
```

(2) Train Test Split Data

```
df1 = df[['host_identity_verified','neighbourhood_group','instant_bookable',
          'cancellation_policy','room type','Construction year','minimum nights',
          'number of reviews', 'reviews per month',
          'review rate number', 'calculated host listings count',
          'availability 365','price', 'service fee']]
```

```
x =df1.drop(["price"],axis=1).values
y= df1['price'].values
```

```
from sklearn.model_selection import train_test_split
```

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
```

- We mention data Test Size 0.3 which is 70%-30% split.

```
print(f'The shape of X train dataset is',x_train.shape)
print(f'The shape of X test dataset is',x_test.shape)
print(f'The shape of Y train dataset is',y_train.shape)
print(f'The shape of Y test dataset is',y_test.shape)
print(f'Total shape of dataset',df.shape)
```

```
The shape of X train dataset is (70415, 13)
The shape of X test dataset is (30179, 13)
The shape of Y train dataset is (70415,)
The shape of Y test dataset is (30179,)
Total shape of dataset (100594, 23)
```

Linear Regression Model

```
1  from sklearn.linear_model import LinearRegression
```

```
1  model_LR=LinearRegression()
```

```
1  model_LR.fit(x_train,y_train)
```

```
LinearRegression()
```

```
1  y_predict=model_LR.predict(x_test)
```

```
1  y_predict
```

```
array([1047.47128574,  104.79612203,  780.25147649, ...,  521.28473453,
        438.17998404,  517.61945351])
```

- We have 9 Predict Value in test model thats why there is 9 value.

- If training and implement concept match the model is passing successful.
- Fit function to fit model and train data.
- Score check accuracy of the model that we trained.

```
1  LR = model_LR.score(x_train,y_train)*100
2  LR
```

```
98.94180678545142
```

- Linear Regression Model Gives us a 98% Accuracy.

➢ Linear Regression Model gives us a 98% Accuracy.

Decision Tree Regressor Model

```
1 from sklearn.tree import DecisionTreeRegressor
```

```
1 model_DT = DecisionTreeRegressor()
```

```
1 model_DT
```

```
DecisionTreeRegressor()
```

```
1 model_DT.fit(x_train, y_train)
```

```
DecisionTreeRegressor()
```

```
1 BDT=model_DT.score(x_test, y_test)*100
2 BDT
```

```
99.47478706645533
```

- Decision Tree Regressor gives us 99% of Accuracy.

➢ Decision Tree Regressor gives us 99% of Accuracy.

Random Forest Regressor Model

```
1 from sklearn.ensemble import RandomForestRegressor
```

```
1 model_RFR = RandomForestRegressor()
```

```
1 model_RFR
```

```
RandomForestRegressor()
```

```
1 model_RFR.fit(x_train,y_train)
```

```
RandomForestRegressor()
```

```
1 RFR =model_RFR.score(x_test, y_test)*100
2 RFR
```

```
99.70426454323305
```

- Random Forest Regressor gives us 99% of Accuracy.

➢ Random Forest Regressor gives us 99% of Accuracy.

XGBoost Model

```
1 import xgboost as xgb
```

```
1 model_xgb = xgb.XGBRegressor()
```

```
1 model_xgb
```

```
XGBRegressor(base_score=None, booster=None, callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=None, early_stopping_rounds=None,
             enable_categorical=False, eval_metric=None, feature_types=None,
             gamma=None, gpu_id=None, grow_policy=None, importance_type=None,
             interaction_constraints=None, learning_rate=None, max_bin=None,
             max_cat_threshold=None, max_cat_to_onehot=None,
             max_delta_step=None, max_depth=None, max_leaves=None,
             min_child_weight=None, missing=nan, monotone_constraints=None,
             n_estimators=100, n_jobs=None, num_parallel_tree=None,
             predictor=None, random_state=None, ...)
```

```
1 model_xgb.fit(x_train, y_train)
```

```
XGBRegressor(base_score=0.5, booster='gbtree', callbacks=None,
             colsample_bylevel=1, colsample_bynode=1, colsample_bytree=1,
             early_stopping_rounds=None, enable_categorical=False,
             eval_metric=None, feature_types=None, gamma=0, gpu_id=-1,
             grow_policy='depthwise', importance_type=None,
             interaction_constraints='', learning_rate=0.300000012, max_bin=256,
             max_cat_threshold=64, max_cat_to_onehot=4, max_delta_step=0,
             max_depth=6, max_leaves=0, min_child_weight=1, missing=nan,
             monotone_constraints='()', n_estimators=100, n_jobs=0,
             num_parallel_tree=1, predictor='auto', random_state=0, ...)
```

```
1 xgb = model_xgb.score(x_test, y_test)*100
```

```
1 xgb
```

```
99.61083921906663
```

- XGBoost Regressor gives us 99% of Accuracy.

➢ XGBoost Regressor gives us 99% of Accuracy.

Section 4

Analysis of Result

➢ Performance Evaluation of the Algorithm Implemented

```
Models = pd.DataFrame({'Model': ['Random Forest', 'Decision Tree',
                                 'Linear Regression','XGBoost'],
                       'Accuracy': [RFR, BDT, LR , xgb]})
```

```
1 Models
```

	Model	Accuracy
0	Random Forest	99.704265
1	Decision Tree	99.474787
2	Linear Regression	98.941807
3	XGBoost	99.610839

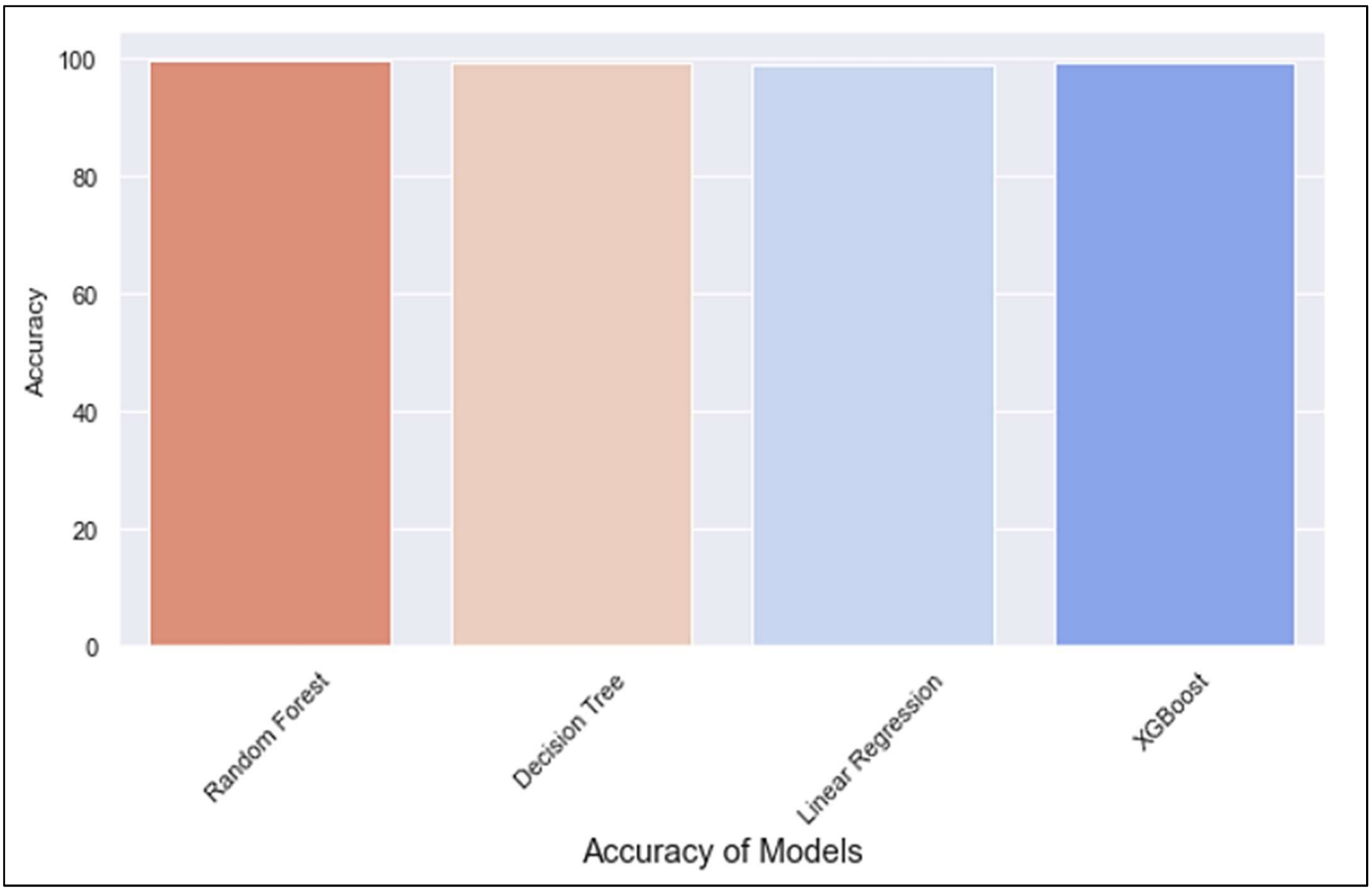
➢ Observations

i. In the following table we have seen Accuracy scored by our Model is very good and it will predict the accurate result for the future predictions.

ii. The dependent variable which has to be predicted is Price so, we can see the all perspective for the prediction of model In various cities where the Airbnb is operational.

Comparison Visuals

```
plt.figure(figsize=(10,5))
sns.barplot(x='Model', y = 'Accuracy', data = Models, palette= 'coolwarm_r')
plt.xticks(rotation = 45)
plt.xlabel('Accuracy of Models', fontsize =15)
plt.show()
```



➢ Observation

i. In the following Bar plot we seen that the all model gives best accuracy .

ii. In the comparison of all four models Random Forest Regressor is best model for prediction.

Accuracy Table

Sr. No.	Machine Learning Models	Accuracy
1	Random Forest Regressor	99.70
2	Decision Tree Regressor	99.47
3	Linear Regression	98.94
4	XGBoost	99.61

K-Fold Cross Validation

Cross-validation is a technique for validating the model efficiency by training it on the subset of input data and testing on previously unseen subset of the input data. We can also say that it is a technique to check how a statistical model generalizes to an independent dataset.

- Generalization refers to your model's ability to adapt properly to new, previously unseen data, drawn from the same distribution as the one used to create the model.
- Due to high accuracy of model that's why we check with k fold cross validation in it.
- **Observation** = The Above calculations of all the model shows that , Model is clean and free from outliers so that it gives us the best accuracy from all the models.

Import Cross_Val_Score Using Scikit Learn Library.

```
1 from sklearn.model_selection import cross_val_score
```

[1] Cross Val in Random Forest.

```
1 randon_forest_kfold = cross_val_score(RandomForestRegressor(), x,y, cv=3)
```

```
1 RF_CV = np.mean(randon_forest_kfold)*100
```

```
1 RF_CV
```

```
99.74510602804595
```

- Observation = Using cross val score our model prediction is 99%.

[2] Cross Val in Linear Regression.

```
1 Linear_Regression_kfold = cross_val_score(LinearRegression(), x,y, cv=3)
```

```
1 LR_CV = np.mean(Linear_Regression_kfold)*100
```

```
1 LR_CV
```

```
98.84084797433793
```

- Observation = Using cross val score our model prediction is 98%.

[3] Cross Val in Decision Tree Regressor.

```
1 Decision_tree_regressor_kfold = cross_val_score(DecisionTreeRegressor(), x,y, cv=3)
```

```
1 BDT_CV = np.mean(Decision_tree_regressor_kfold)*100
```

```
1 BDT_CV
```

```
99.55737417421847
```

- Observation = Using cross val score our model prediction is 99%.

[4] Cross Val in XGBoost Regressor.

```
1 XGBoost_Regression_kfold = cross_val_score(model_xgb, x,y, cv=3)
```

```
1 XGB_CV = np.mean(XGBoost_Regression_kfold)*100
```

```
1 XGB_CV
```

```
99.67964120024314
```

- Observation = Using cross val score our model prediction is 99%.

Comparison

```
Models = pd.DataFrame({'Model': ['Random Forest', 'Decision Tree',
                                 'Linear Regression','XGBoost'],
                       'Accuracy': [RFR, BDT, LR , xgb],
                      'KFold_Accuracy':[RF_CV, BDT_CV, LR_CV, XGB_CV]})
```

	Model	Accuracy	KFold_Accuracy
0	Random Forest	99.740573	99.745106
1	Decision Tree	99.345541	99.557374
2	Linear Regression	98.807011	98.840848
3	XGBoost	99.672490	99.679641

Predictor

```
1  model_LR.predict([[1,3,1,1,2,2011,30,45,8,4,5,3,200]])
```

```
array([997.47117913])
```

1) Inputs

- host_identity_verified = 1 (Verified)
- neighbourhood_group = 3 (Queens City)
- instant_bookable = 1 (Yes)
- cancellation_policy = 1 (Moderate)
- room type = 2 (Private room)
- Construction year = 2011
- minimum nights = 30 nights
- number of reviews = 45 Points
- reviews per month = 8 Star
- review rate number = 4 Star
- calculated host listings count = 5
- availability 365 = 3
- service fee = $200

2) Output

- Price Of Property.

Price of Our Property is 997.47 Dollars.

➢ Observation Of Prediction:

The following features used in the prediction are,

1. host_identity_verified = 1 (Verified)
2. neighbourhood_group = 3 (Queens City)
3. instant_bookable = 1 (Yes)
4. cancellation_policy = 1 (Moderate)
5. room type = 2 (Private room)
6. Construction year = 2011
7. minimum nights = 30 nights
8. number of reviews = 45 Points
9. reviews per month = 8 Star
10. review rate number = 4 Star
11. calculated host listings count = 5
12. availability 365 = 3
13. service fee = $200

- So, this feature help us to predict us the price of property.
- By the following values we predict that the price of property is **997.47 Dollars**.

Section 5

Conclusion

From all the analysis done above, we can confidently answer our initial question of the factors that make a listing more expensive. An aspiring Airbnb host, if investing on a new properties, should focus on the following factors to maximize the price of his listing. Additionally a traveller who wants to pay the lowest - possible price for a listing might want to avoid having these features in his prospective housing.

- Entire properties listed instead of just a single room fetch the highest prices.
- Apartment and landed house tend to be the most expensive and the most abundant properties in Airbnb.
- The more bedrooms a property has, the higher its price. The highest prices are fetched by 6 room properties.
- There are plenty of listings in Belltown or West Queen Anne and they tend to be expensive.
- Words like: view, modern & walk all frequently appear in the summary of the more expensive listing.
- Ammenities such as: Washer, Dryer, Heating, Wireless Internet, Smoke Detector, Free Parking, Kid Friendly, TV, HotTub/Sauna/Pool, Gyms and Elevators are all common among the more expensive listings.
- The reviews a listing gets (quality or quantity) does not have much of an impact in its price.

Section 6

References

1. J.R. Quinlan (1986). "Induction of Decision Trees". Machine Learning. **1**: 81–106. doi:10.1007/BF00116251.
2. Nick Littlestone (1988). "Learning Quickly When Irrelevant Attributes Abound: A New Linear-threshold Algorithm"
3. "Editorial Board of the Kluwer Journal, Machine Learning: Resignation Letter". SIGIR Forum. **35** (2). 2001.
4. "General Python FAQ — Python 3.9.2 documentation". docs.python.org. Archived from the original on 24 October 2012. Retrieved 28 March 2021.
5. The Python Library Reference by Guido van Rossum Fred L. Drake, Jr., editor.
6. Dataset from Kaggle : https://www.kaggle.com/datasets/arianazmoudeh/airbnbopendata .