DATA MINING PROJECT REPORT

CSCI-B565

Airbnb Recommendation System

Team:

DILEEP SAI ELLANKI
UID: 2001058995
Email: dellanki@iu.edu

HARSHITHA REDDY GARLAPATI
UID: 2001082373
Email: hgarlapa@iu.edu

SATWIK CHOWDARY INAMPUDI
UID: 2001079254
Email: sinampu@iu.edu

INDIANA UNIVERSITY

ABSTRACT

Airbnb is a platform that helps hosts to lease homestays on a short-term basis. Guests can view dates available, pricing options, location, neighborhood, amenities, etc. This is the most widely used platform across the globe. The guests must surface through all the options according to their preference with the help of various filters available. With several options available, it confuses the user in selecting the right option. We had to develop a system where the user enters their annual income to predict the room type and location most suitable for the user since New York City (NYC) is one of the most happening cities in the world. We have selected four different portions of New York City as inputs for our model. We have collected data with different room types which are most preferred by customers. We figured out the parameters that effect the cost of living in a certain allocation and fed that to the recommender system.

KEYWORDS

- **Data Mining:** Data Mining is a process of extracting useful information from data. Several data mining techniques can be applied to huge repositories to infer patterns.
- **Dataset:** Collection of records (data objects). Also known as record, entity, instance.
- **Attribute:** Characteristics of an object. Also known as feature, dimension, field.
- **Data Preprocessing:** Process of making the raw data suitable for analysis. Includes data cleaning, transforming, reduction.
- **Dimensionality Reduction:** Reducing the number of dimensions of the data by cutting down unwanted features which do not aid in the analysis process.
- **Feature Subset Selection:** Using a subset of features from the dataset so that the resultant dataset is free of redundant or irrelevant attributes.
- **Sampling:** Representative of the original dataset consisting of the same properties.
- **Mean Squared Error:** measures how close a regression line is to a set of data points

INTRODUCTION

About the dataset:

The dataset is about the listings and metrics in New York City. This dataset includes information about hosts, location, reviews, availability, neighborhood, etc.

Dataset consists of close to 50000 listings consisting of 15 attributes.

Attributes:

- ***Name****:* Provides the name of the listing
- ***Host_name****:* Name of the host
- ***Neighbourhood_group****:* The neighbourhood group to which the property/listing belongs to
- ***Neighbourhood****:* The neighbourhood to which the property/listing belongs to
- ***Latitude***: Lattitude of the property
- ***Longitude***: Longitude of the property
- ***Room_type****:* Gives information about the type of room. For instance, private room/
- ***Price***: Provides the price of the listing according to the room type
- ***Minimum_nights:*** The minimum number of nights the guest should book to avail the property
- ***Number_of_reviews***: Reviews the guests provides for the listing
- ***Last_review:*** Date on the which the last review was provided
- ***Reviews_per_month***: Number of reviews per month a listing gets
- ***Availability_365:*** Shows the availability of the listing throughout the year
- ***Days_since_last_review:*** Information about the number of days since the last review

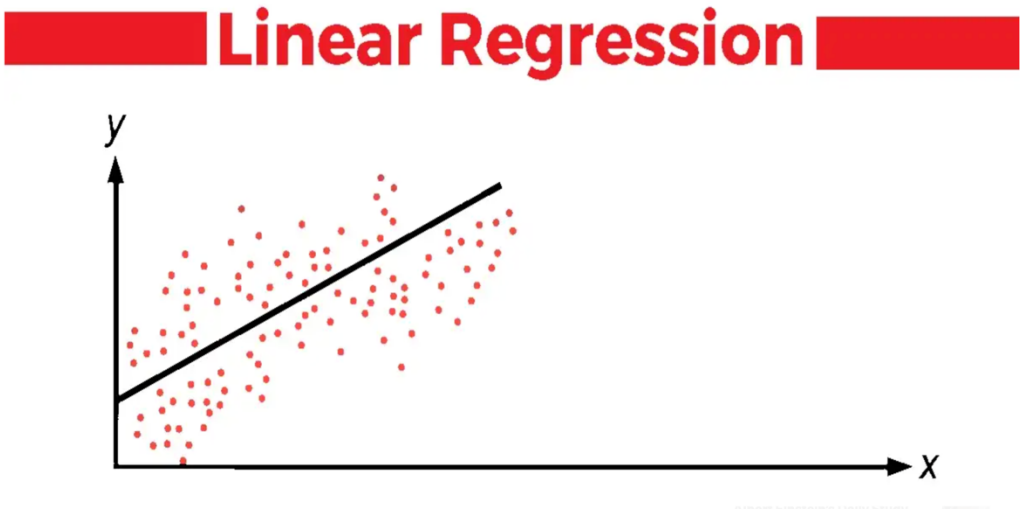
MODELS USED

Linear regression [1]

The Linear Regression model is used to interpret the relationship between variables. It helps us to interpret the association between the dependent and independent variables.

Before trying to apply a linear model, check whether a relationship is present between the variables. For instance,

Creating a scatter plot helps in determining if linear regression is helpful. If there are no trends observed in the scatter plot, the model may not be helpful.



K-Neighbors Classifier[2][3]

It is one of the most basic and extensively used classification methods, in which a new data point is classified based on similarities in a certain set of nearby data points. This produces a competitive outcome.

Step 1: Find the value of K:

In the first step, we figure out what is the value of k. The calculation of the K value varies substantially based on the situation. The default value of K in the Scikit-Learn Library is 5.

Step 2: Calculate the distance between new and training data.

Euclidean Distance, Manhattan Distance, and Minkowski Distance are three distance metrics that are frequently used to determine distances.

Step 3: Calculate the distance between new and training data.

Euclidean Distance, Manhattan Distance, and Minkowski Distance are three distance metrics that are frequently used to determine distances.

The default distance utilized by Scikit-Learn is Euclidean. The Minkowski distance formula contains a Hyperparameter p. If p = 1 is specified, the Manhattan distance is used, while p = 2 is Euclidean.

Step 4: Using the new data, find the nearest K-neighbors.

After we calculated the distance, look for K-Neighbors which are the closest to the new data. If K = 4, look for 4 training data that are closest to the new data. that is the most like the new data.

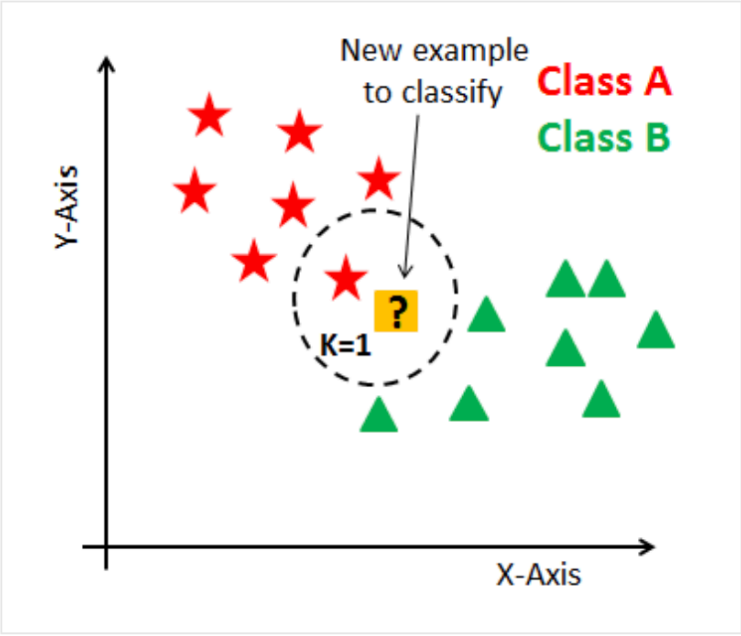
Step 5: Prediction of New Data Classes

To decide the class of new data, choose the training data class that is closest to the new data and has the most of it.

Step 6: Assessment.

Calculate the model's accuracy; if the accuracy remains poor, the process can be repeated from step 1. With the hyperparameter-tuned values that best fit the model to give us more accuracy.

It is true that KNN reduces overfitting. On the other hand, the best value for K must be chosen.



Naive Bayes Classifier[4]

A machine learning model that is used for classification tasks is called a Naive Bayes classifier. It is a method of classification that is based on Bayes' Theorem and assumes that predictors are independent of one another. A Naive Bayes classifier, to put it in more layman's terms, works under the assumption that the existence of one character in a class is unrelated to the presence of any other feature.

The Naive Bayes model is simple to construct and is especially helpful for working with very big data sets. In addition to its ease of use, the Naive Bayes method is renowned for its ability to outperform even the most complex categorization approaches. Types of Naive Bayes Classifier:

Multinomial Naive Bayes:

Most often, this is applied to the question of how to categorize a document, such as whether it deals with sports, politics, technology, etc. The word frequency in the document serves as the features/predictors for the classifier.

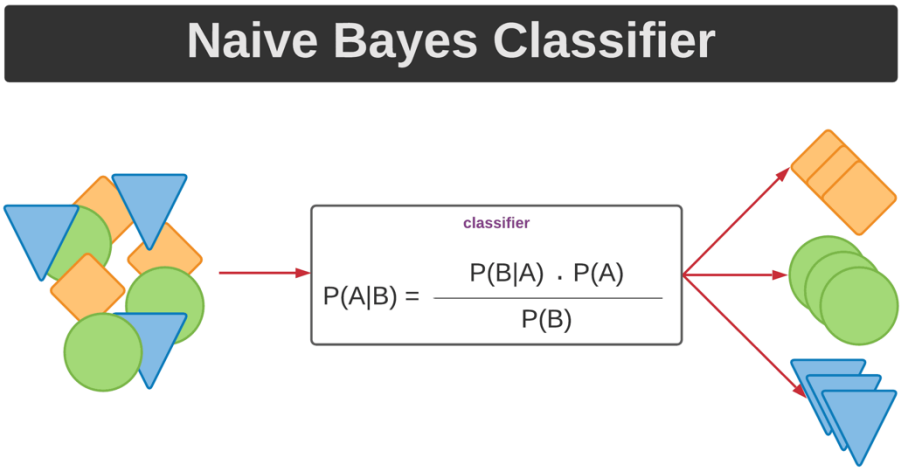
Bernoulli Naive Bayes:

Similar to multinomial naive Bayes, but with Boolean variables as predictors instead. All the parameters we need to make predictions about the class variable are either yes or no, such as whether or not a given word appears in the text.

Gaussian Naive Bayes:

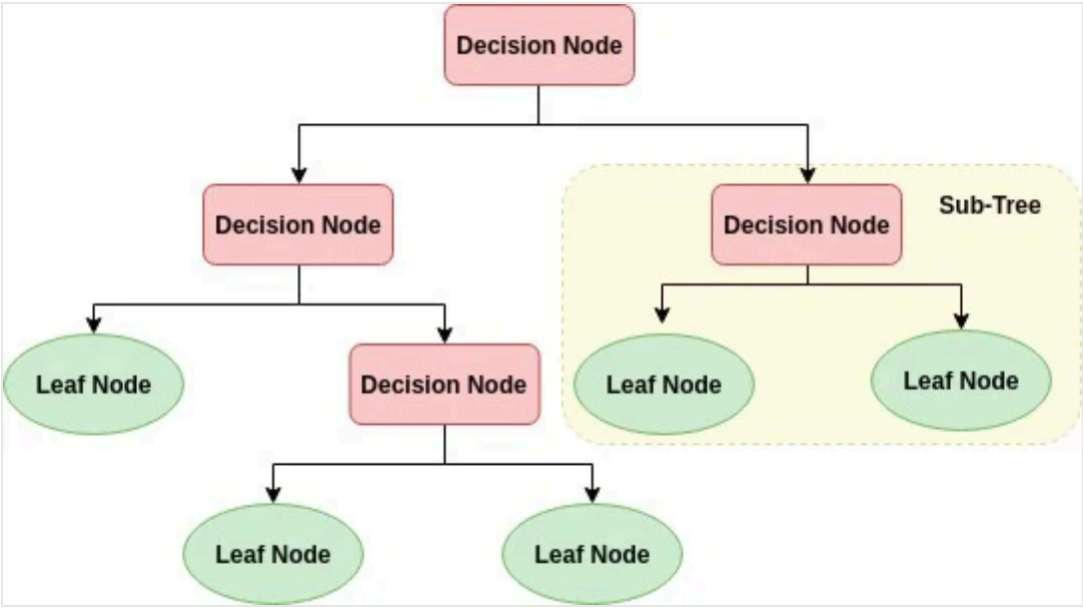
If the predictors' values are not binary, we assume they are drawn at random from a Gaussian distribution if they are continuous.

Sentiment analysis, spam filtering, recommendation systems, etc. are typical applications of Naive Bayes algorithms. They can be implemented quickly and with little effort, but their main drawback is that it is necessary for predictors to be impartial. In most practical settings, the predictors are interdependent, which reduces the classifier's efficacy.



Decision Tree Classifier[9]

A decision tree is a tree structure similar to a flowchart in which each internal node represents a feature or attribute, each branch represents a decision rule, and each leaf node reflects the outcome. The node at the very top of a decision tree is called the root node. It discovers how to split based on the attribute value. Recursive partitioning is the partitioning of a tree in a recursive fashion. This layout like a flowchart facilitates decision making. It is a visual representation in the form of a flowchart that easily imitates human-level thought. Therefore, decision trees are simple to comprehend and interpret.



Decision Tree is a sort of ML method known as a white box. It provides access to internal decision-making mechanism that is unavailable in black-box algorithms such as Neural Network. It has a shorter training period than the neural network algorithm. The time complexity of decision trees is proportional to the quantity of data records and characteristics. The decision tree is a non-parametric or distribution-free strategy that does not rely on probability distribution assumptions. Accurately manage high-dimensional data with decision trees.

PROCESS

Data Preprocessing

Step1:

Here we see the data after reading the CSV file. In the CSV file, we have unwanted columns so initially, we eliminate them using the drop command.

```
[1] import numpy as np
    import pandas as pd
    import matplotlib.pyplot as plt
    import seaborn as sns
```

```
[3] path = '/content/Project.csv'
    data = pd.read_csv(path)
```

```
[4] data.head()
```

	id	name	host_id	host_name	neighbourhood_group	neighbourhood	latitude	longitude	room_type	price	minimum_nights	number_of_reviews
0	2539	Clean & quiet apt home by the park	2787	John	Brooklyn	Kensington	40.64749	-73.97237	Private room	149	1	9
1	2595	Skylit Midtown Castle	2845	Jennifer	Manhattan	Midtown	40.75362	-73.98377	Entire home/apt	225	1	45
2	3647	THE VILLAGE OF HARLEM....NEW YORK !	4632	Elisabeth	Manhattan	Harlem	40.80902	-73.94190	Private room	150	3	0
3	3831	Cozy Entire Floor of Brownstone	4869	LisaRoxanne	Brooklyn	Clinton Hill	40.68514	-73.95976	Entire home/apt	89	1	270
4	5022	Entire Apt: Spacious Studio/Loft by central park	7192	Laura	Manhattan	East Harlem	40.79851	-73.94399	Entire home/apt	80	10	9

Here we can see the unwanted columns are dropped

```
[5] data = data.drop(['name','id','host_name','last_review','neighbourhood'],axis = 1)
```

```
[6] data.head()
```

	host_id	neighbourhood_group	latitude	longitude	room_type	price	minimum_nights	number_of_reviews	reviews_per_month	calculated_host_listings_count
0	2787	Brooklyn	40.64749	-73.97237	Private room	149	1	9	0.21	6
1	2845	Manhattan	40.75362	-73.98377	Entire home/apt	225	1	45	0.38	2
2	4632	Manhattan	40.80902	-73.94190	Private room	150	3	0	NaN	1
3	4869	Brooklyn	40.68514	-73.95976	Entire home/apt	89	1	270	4.64	1
4	7192	Manhattan	40.79851	-73.94399	Entire home/apt	80	10	9	0.10	1

Step 2:

We have eliminated the nan values and replaced them with 0 in the riviews_per_month column. We have also replaced nan values in the price column with the mode value of the column

```
data['reviews_per_month'].fillna(0,inplace=True)

a=data['price'].mode()
data['price'].fillna(a,inplace=True)
```

Step 3:

We have converted categorical data to numerical data which we used for simplification of the model training and drawing insights from them.

```
[9] NB_group = {'Brooklyn': 1,'Manhattan': 1,'Queens':3,'Staten Island':2,'Bronx':4}
    data.neighbourhood_group= [NB_group[item] for item in data.neighbourhood_group]
    data.head()
```

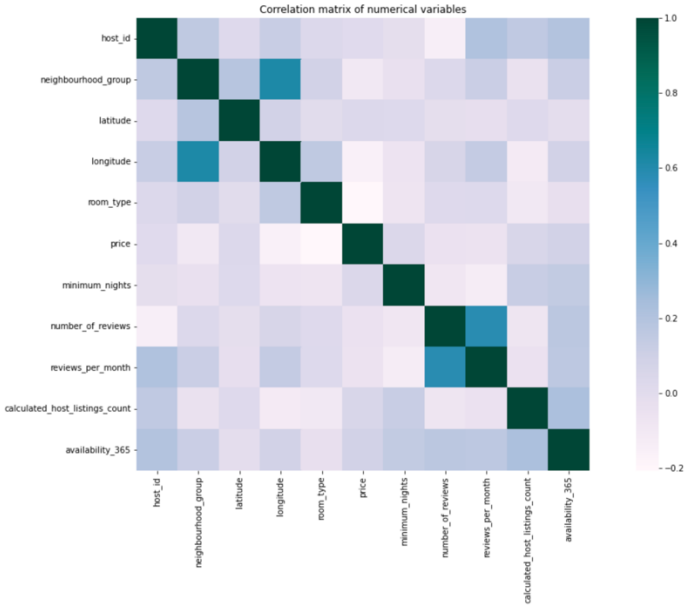
	host_id	neighbourhood_group	latitude	longitude	room_type	price	minimum_nights	number_of_reviews	reviews_per_month	calculated_host_listings_count
0	2787	1	40.65	-73.97	Private room	149	1	9	0.21	6
1	2845	1	40.75	-73.98	Entire home/apt	225	1	45	0.38	2
2	4632	1	40.81	-73.94	Private room	150	3	0	0.00	1
3	4869	1	40.69	-73.96	Entire home/apt	89	1	270	4.64	1
4	7192	1	40.80	-73.94	Entire home/apt	80	10	9	0.10	1

```
[10]
    RM_Type = {'Shared room': 1,'Entire home/apt': 2,'Private room':3}
    data.room_type= [RM_Type[item] for item in data.room_type]
    data.head()
```

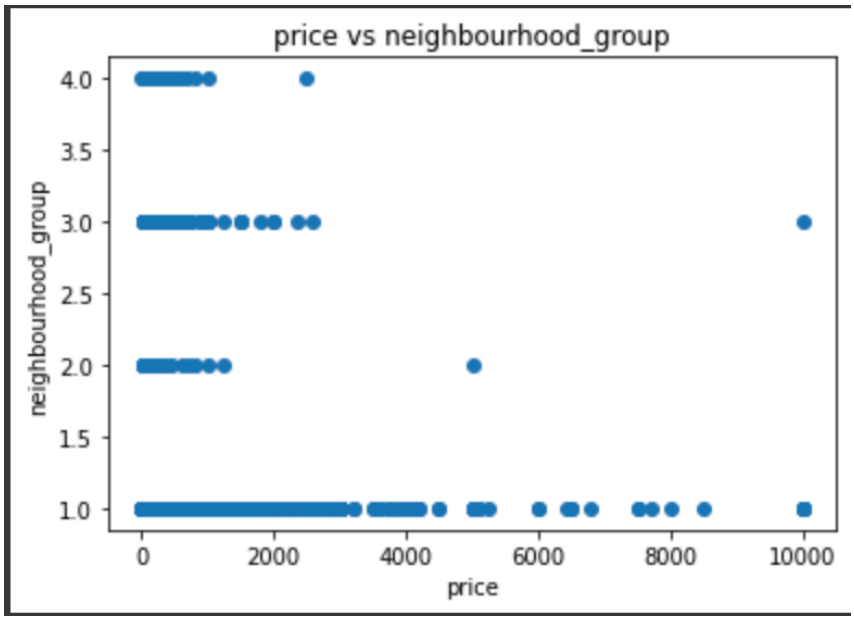
	host_id	neighbourhood_group	latitude	longitude	room_type	price	minimum_nights	number_of_reviews	reviews_per_month	calculated_host_listings_count
0	2787	1	40.65	-73.97	3	149	1	9	0.21	6
1	2845	1	40.75	-73.98	2	225	1	45	0.38	2
2	4632	1	40.81	-73.94	3	150	3	0	0.00	1
3	4869	1	40.69	-73.96	2	89	1	270	4.64	1
4	7192	1	40.80	-73.94	2	80	10	9	0.10	1

Data Visualization

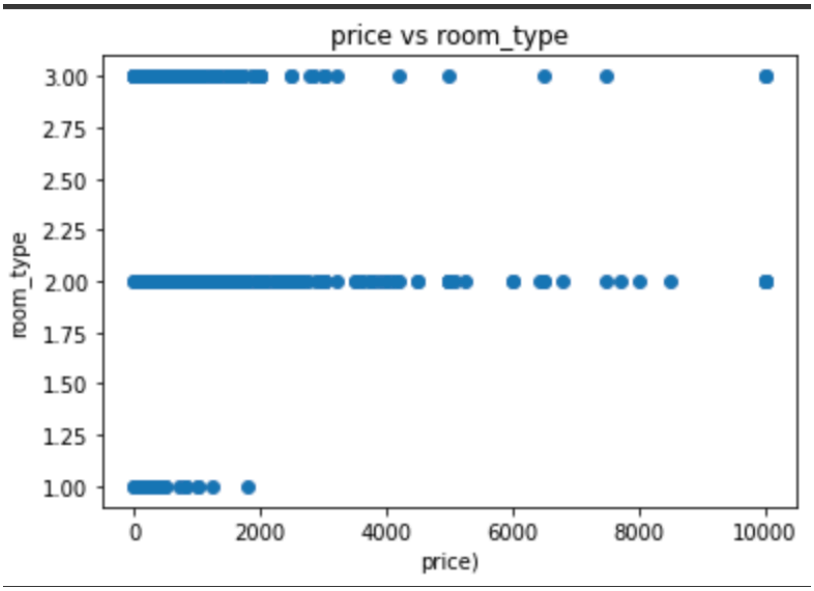
1) We have generated the correlation matrix which shows how the attributes are correlated



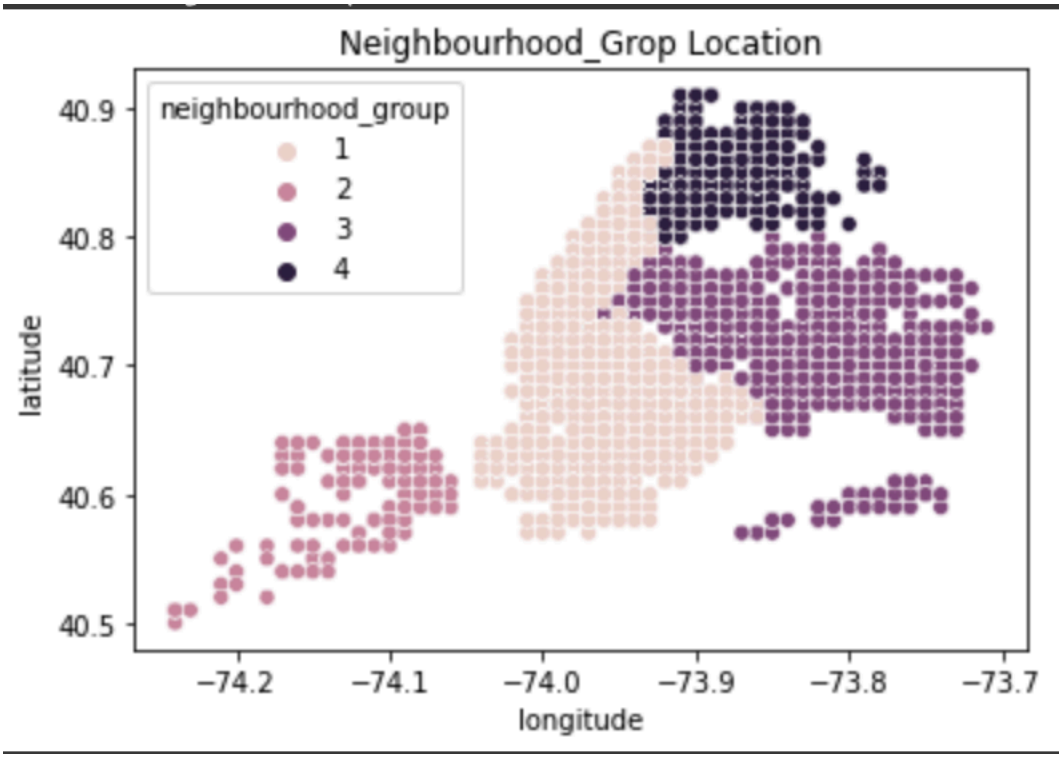
2) We have plotted a scatter plot between neighborhood groups and prices to find how price changes from one neighborhood to other.



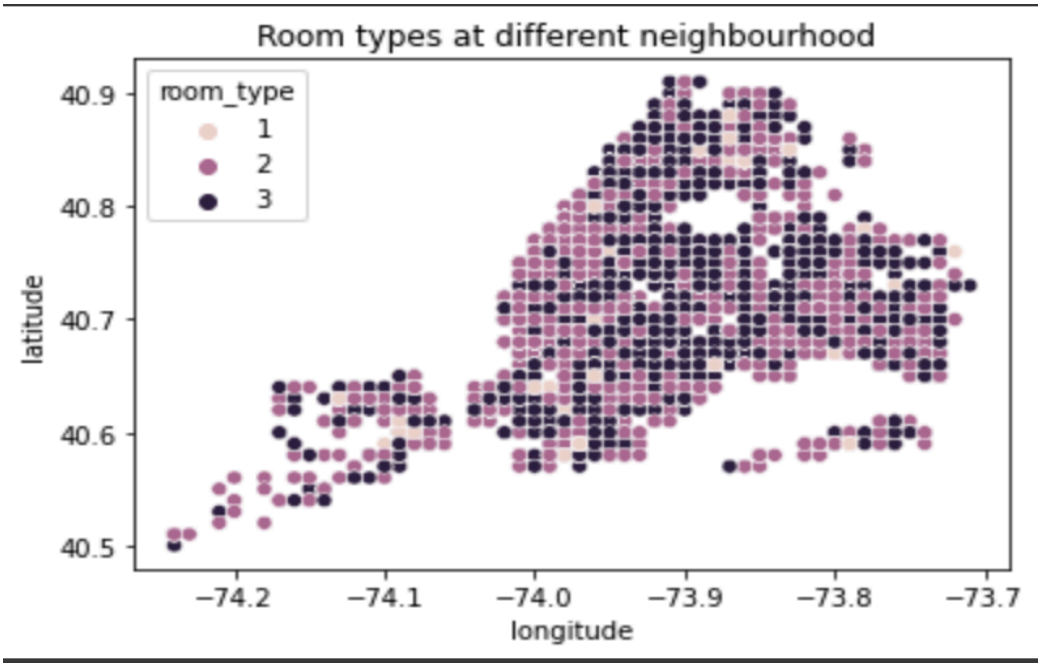
3) We have plotted a scatter plot between Room type and prices to find how price changes with the type of room.



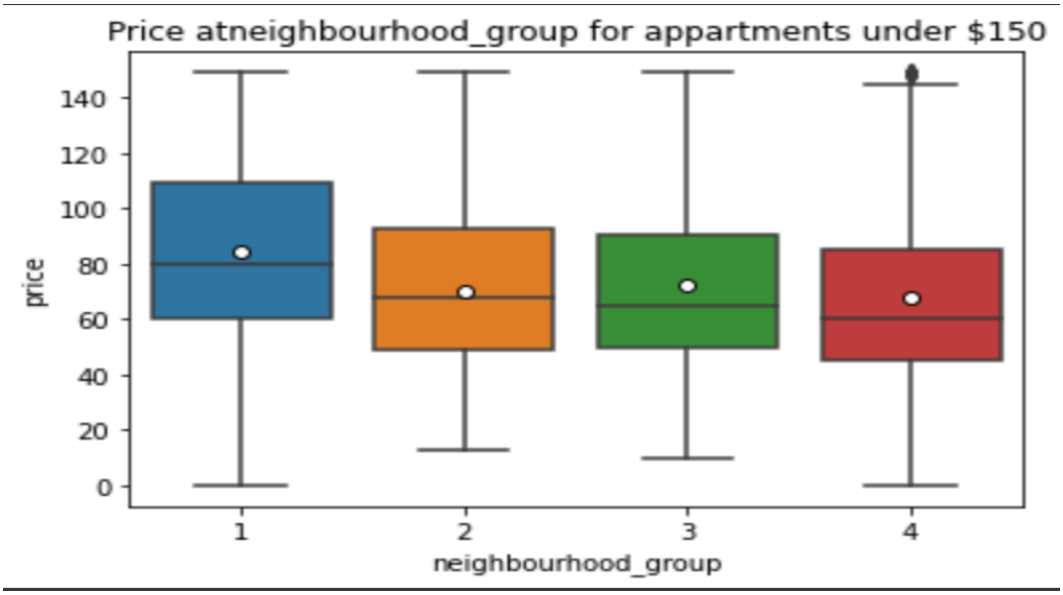
4) We plotted map neighborhood group location using latitudes and longitudes from the data[2]



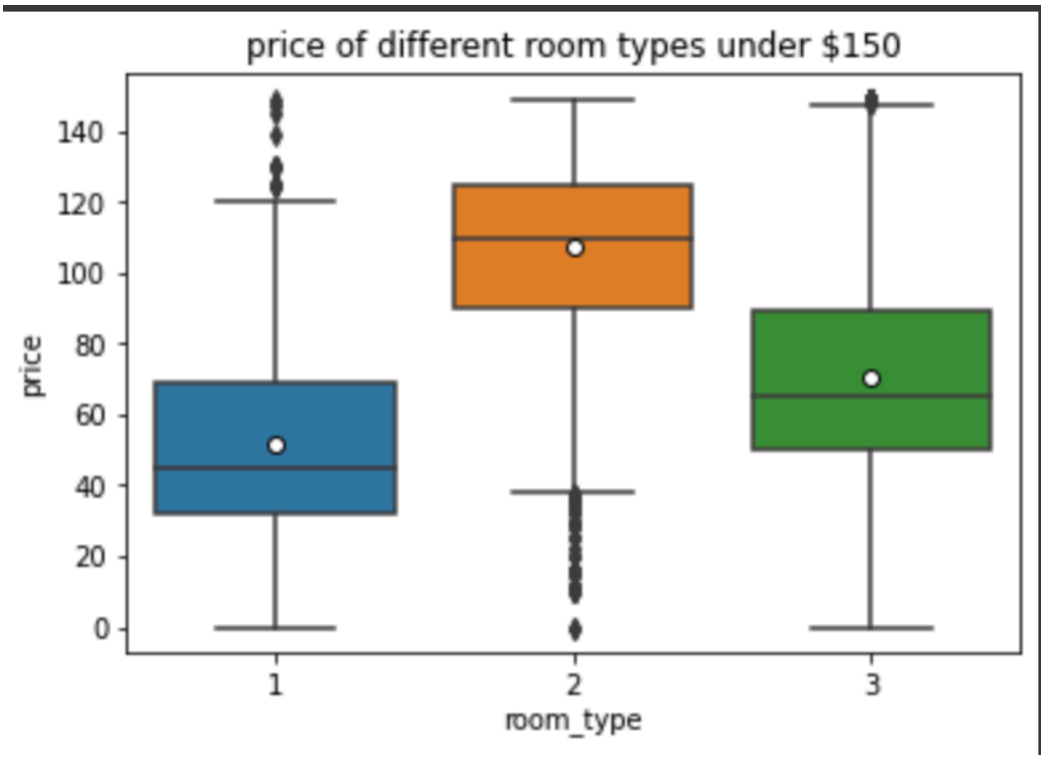
5) We plotted map room type neighborhood group location using latitudes and longitudes from the data as we can see that all the room types are distributed equally



6) We plotted the price per property under 150$ in a certain neighborhood and the [1,2,3,4,5] represents the numerical data of the converted categorical data in pre-processing.



7) We plotted the price per property under 150$ of a certain room type and the [1,2,3] represents the numerical data of the converted categorical data in pre-processing.



LINEAR REGRESSION

Linear regression is done between highly correlated attributes which are derived from the correlation matrix.

The above linear regression is between the total number of reviews and reviews per month from which we can see that the relation between the two attributes are linear and they are mutually dependent.

Classification

K-Neighbor classifier[2]

```
[13] #Creating KNN classifier
     K_N_Class = KNeighborsClassifier(n_neighbors=25)

     #Trainig the model with the training data sets
     K_N_Class .fit(X_train, y_train)

     #Predicting the data
     y_pred1 = K_N_Class .predict(X_test)
```

```
[14] # calucating the model accuracy
     print("Accuracy Accuracy after hyper parameter tuning :",metrics.accuracy_score(y_test, y_pred1)*100)

     Accuracy Accuracy after hyper parameter tuning : 85.0296543731679
```

```
[15] temp = {'classifier': 'K_N_Class', 'Accuracy': metrics.accuracy_score(y_test, y_pred1)*100}
     expLog = expLog.append(temp, ignore_index = True)
```

```
[16] expLog
```

	classifier	Accuracy
0	K_N_Class	85.029654

- In the above classification we have used the k-neighbor classifier where:
- We have divided the data set into train and test data, and we have predicted the accuracy of the model with the given dataset.
- After finding the accuracy we appended the value to a table.
- We got an accuracy of 85.02%

K-Neighbor classifier after Hyperparameter tuning[5]

- We have applied hyperparameter tuning to find the parameters that can increase the accuracy.
- We found leaf size, number of neighbors, and p-value that can increase the accuracy.
- We got the leaf size as 3 and neighbors as 13 and applied the metrics to the new K-NN classifier.
- So our accuracy has increased and we got an accuracy of 85.08%.
- It is not a big change in accuracy but it can be increased if k values are increased.

```
[17] #Hyper parameter tuning and the things to be tuned
     leaf_size = list(range(1,10))
     n_neighbors = list(range(1,30))
     p=[1,2]
     #Converting to dictionary
     hyperparameters = dict(leaf_size=leaf_size, n_neighbors=n_neighbors, p=p)
     #creating new KNN objectss
     knn_2 = KNeighborsClassifier()
     #GridSearchCV is applied
     clf = GridSearchCV(knn_2, hyperparameters, cv=10)
     #fitting the model
     best_model = clf.fit(X,y)
     #printting the hyperparameters for increasing the accuracy
     print('Best leaf_size:', best_model.best_estimator_.get_params()['leaf_size'])
     print('Best p:', best_model.best_estimator_.get_params()['p'])
     print('Best n_neighbors:', best_model.best_estimator_.get_params()['n_neighbors'])
```

```
Best leaf_size: 3
Best p: 1
Best n_neighbors: 16
```

```
#Import knearest neighbors Classifier model
from sklearn.neighbors import KNeighborsClassifier

#Create KNN Classifier
K_N_Class  = KNeighborsClassifier(leaf_size=4,n_neighbors=21,p=1)

#Train the model using the training sets
K_N_Class .fit(X_train, y_train)

#Predict the response for test dataset
y_pred2 = K_N_Class .predict(X_test)
```

```
[19] #Import scikit-learn metrics module for accuracy calculation
     from sklearn import metrics
     # Model Accuracy, how often is the classifier correct?
     print("Accuracy after hyper parameter tuning :",metrics.accuracy_score(y_test, y_pred2)*100)
```

```
Accuracy after hyper parameter tuning : 85.08419115140772
```

NAIVE_BAYES CLASSIFIER

NAIVE_BAYES CLASSIFIER

```
[22]
     from sklearn.naive_bayes import MultinomialNB

     #multinomial function is used in Naive Bayes model
     gnb = MultinomialNB()
     # training the model on training set
     gnb.fit(X_train, y_train)
     # Applying predictions on test data set
     y_pred3 = gnb.predict(X_test)

     # finding the ourput result
     print("Accuracy of MultinomialNB Naive Bayes model :", metrics.accuracy_score(y_test, y_pred3)*100)
```

```
Accuracy of MultinomialNB Naive Bayes model : 85.11827663780763
```

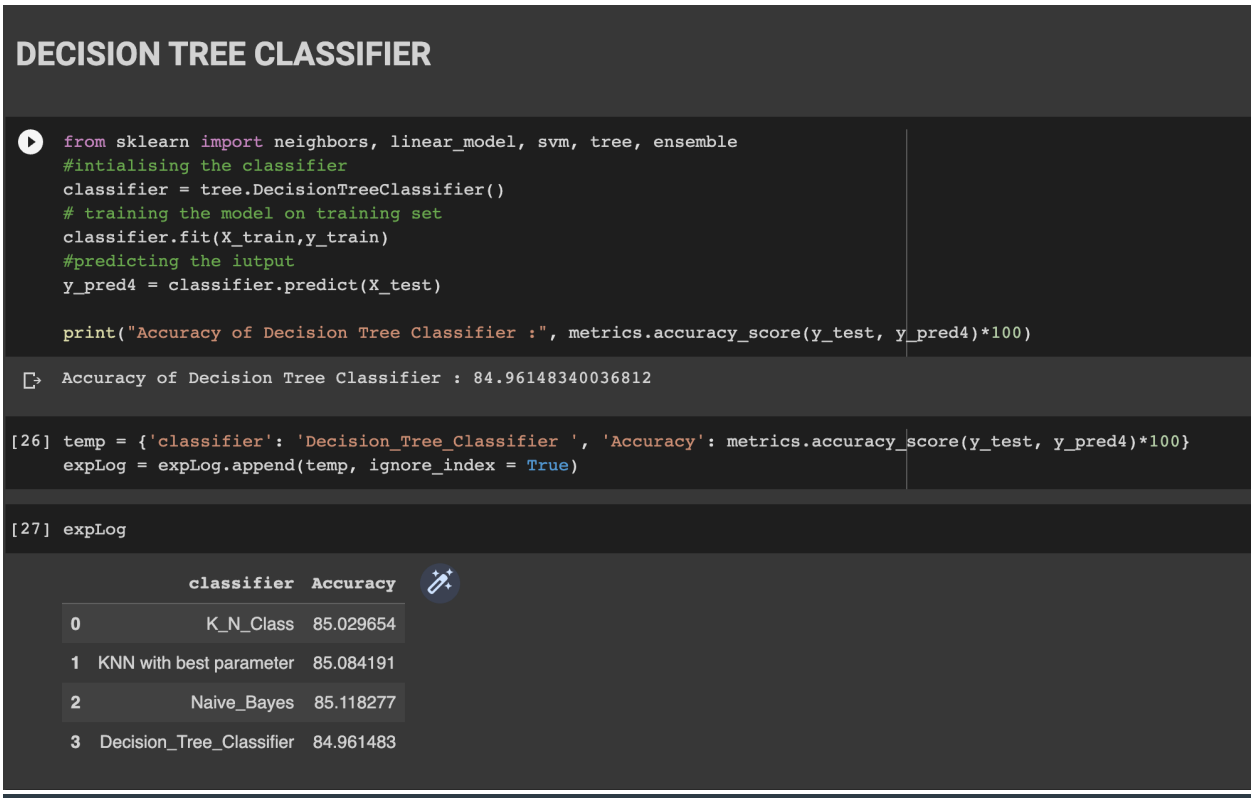
```
[23] temp = {'classifier': 'Naive_Bayes ', 'Accuracy': metrics.accuracy_score(y_test, y_pred3)*100}
     expLog = expLog.append(temp, ignore_index = True)
```

```
expLog
```

	classifier	Accuracy
0	K_N_Class	85.029654
1	KNN with best parameter	85.084191
2	Naive_Bayes	85.118277

- In the above Naïve Bayes classifier, we used the Multinomial Naïve Bayes classification.
- We have used the fit function to fit the model for the split train and test data set.
- After performing the classification, we got an accuracy of 85.11% and we have recorded this accuracy

DECISION TREE CLASSIFIER



DECISION TREE CLASSIFIER

```
from sklearn import neighbors, linear_model, svm, tree, ensemble
#intialising the classifier
classifier = tree.DecisionTreeClassifier()
# training the model on training set
classifier.fit(X_train,y_train)
#predicting the iutput
y_pred4 = classifier.predict(X_test)

print("Accuracy of Decision Tree Classifier :", metrics.accuracy_score(y_test, y_pred4)*100)
```

```
Accuracy of Decision Tree Classifier : 84.96148340036812
```

```
[26] temp = {'classifier': 'Decision_Tree_Classifier ', 'Accuracy': metrics.accuracy_score(y_test, y_pred4)*100}
     expLog = expLog.append(temp, ignore_index = True)
```

```
[27] expLog
```

	classifier	Accuracy
0	K_N_Class	85.029654
1	KNN with best parameter	85.084191
2	Naive_Bayes	85.118277
3	Decision_Tree_Classifier	84.961483

- In the above Decision tree classifier, we have built a model with the help of a tree.
- We have used the fit function to fit the model for the split train and test data set.
- After performing the classification, we got an accuracy of 84.96% and we have recorded this accuracy

RECOMMENDATION SYSTEM

```
booked_rooms = data.groupby(['neighbourhood_group','room_type'], as_index = False)['booking'].mean()\
               .sort_values('booking', ascending = False).reset_index(drop = True)
booked_rooms
```

	neighbourhood_group	room_type	booking
0	2	3	0.909574
1	2	1	0.888889
2	2	2	0.863636
3	4	2	0.839050
4	4	3	0.837423
5	4	1	0.833333
6	3	1	0.818182
7	3	3	0.766904
8	3	2	0.739504
9	1	1	0.721165
10	1	2	0.637710
11	1	3	0.588605

```
[36] #Finding the rate of occupancy of the room over 365 days in certain neighborhood_group and type of room
     data['Rate_of_Occupancy'] = (365 - data['availability_365'])/365

     occup_rate = data.groupby(['neighbourhood_group', 'room_type'], as_index = False)['Rate_of_Occupancy'].mean()\
                  .sort_values('Rate_of_Occupancy', ascending = False).reset_index(drop = True)
     occup_rate
```

	neighbourhood_group	room_type	Rate_of_Occupancy
0	2	1	0.822527
1	1	3	0.723926

- We started building our recommendation system by creating new data frames as booked_rooms where we found which type of rooms are booked in a certain location.
- In the next step, we found the rate of occupancy of a certain room type and in which location trough out the year.
- After that, we padded the values to occup_rate

```
[37] # we created an income with respect to the booking rate in a location with respwect to type of room

     income_BR = data.groupby(['neighbourhood_group','room_type']).agg({'booking':'mean', 'price':'mean'})
     income_BR
```

neighbourhood_group	room_type	booking	price
1	1	0.721165	71.194849
	2	0.637710	219.454258
	3	0.588605	94.248095
2	1	0.888889	57.444444
	2	0.863636	173.846591
	3	0.909574	62.292553
3	1	0.818182	69.020202
	2	0.739504	147.050573
	3	0.766904	71.762456
4	1	0.833333	59.800000
	2	0.839050	127.506596
	3	0.837423	66.788344

```
#create model dataframe by appending the income_BR and occup_rate
model = pd.merge(income_BR, occup_rate , how = 'left', on=['neighbourhood_group', 'room_type'])

#createed a new colum of income
model['WAIncome'] = model['price'] * 365 * model['Rate_of_Occupancy'] * model['booking']
model = model.sort_values(['Rate_of_Occupancy'], ascending = True).reset_index(drop = True)

model
```

	neighbourhood_group	room_type	booking	price	Rate_of_Occupancy	WAIncome
0	2	3	0.909574	62.292553	0.379831	7855.206575
1	3	1	0.818182	69.020202	0.473461	9758.943192
2	2	2	0.863636	173.846591	0.512126	28065.134528

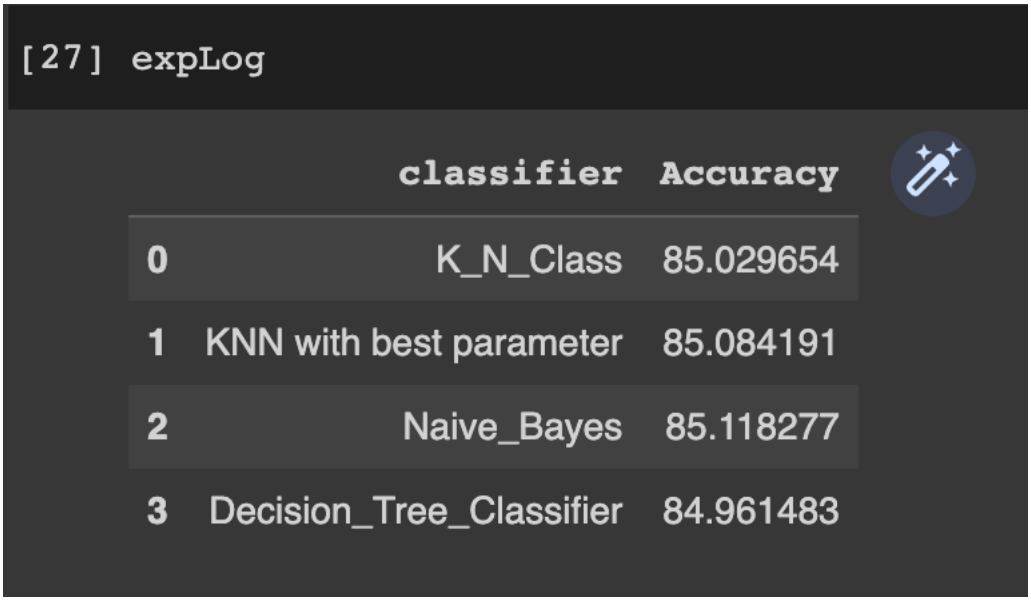
- We created an income with respect to the booking rate in a location with respect to the type of room.
- Created a model data frame by appending the income_BR and occup_rate
- And created a new column for income and generated the values using how booking is done and the rate of occupancy and price as factors

```
[46] #setting a threshold value of income
     thresh =  model['WAIncome'].mean()

     #select the property with the highest income that exceeds the threshold and the lowest

     for i, row, in model[['WAIncome', 'neighbourhood_group', 'room_type']].iterrows():
         if model.loc[i, 'WAIncome'] > thresh:
             if model.loc[i, 'WAIncome'] > model.loc[i+1, 'WAIncome']:
                 print(f"Location Recomended for You: {model.loc[i, 'neighbourhood_group']}"
                 print(f"Best room type for you: {model.loc[i, 'room_type']}")
                 print(f"Estimated  Income is: {model.loc[i, 'WAIncome']}")
                 break

     Location Recomended for You: 2
     Best room type for you: 2
     Estimated  Income is: 28065.134527904298
```

Finally, we got a predicted system with a threshold value and which location is recommended with which type of room based on the income calculated.

RESULTS AND CONCLUSIONS

- From the models above we have classified the neighborhood groups with respect to the type of room and price
- We got an accuracy of 85.02 percent of accuracy from the KNN classification. But after hyperparametric tuning, we got an accuracy of 85.08 percent
- We got an accuracy of 85.11 percent from the Naïve Bayes classification
- We got an accuracy of 85.11 percent from Decision tree classification
- We have build our recommendation system successfully.



```
[27] expLog
```

	classifier	Accuracy
0	K_N_Class	85.029654
1	KNN with best parameter	85.084191
2	Naive_Bayes	85.118277
3	Decision_Tree_Classifier	84.961483

The above is our experiment log and the accuracies of our model

REFERENCES

[1] https://en.wikipedia.org/wiki/Linear_regression

[2]https://scikitlearn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

[3]https://www.datacamp.com/tutorial/k-nearest-neighbor-classification-scikit-learn

[4]https://towardsdatascience.com/naive-bayes-classifier-81d512f50a7c

[5]https://medium.datadriveninvestor.com/k-nearest-neighbors-in-python-hyperparameters-tuning-716734bc557f

[6]https://www.kaggle.com/code/chirag9073/airbnb-analysis-visualization-and-prediction

[7]https://www.kaggle.com/code/siddharthm1698/airbnb-regression-classification-hypothesis-test#Linear-Regression

[8] https://scikit-learn.org/stable/modules/tree.html

[9]https://www.datacamp.com/tutorial/decision-tree-classification-python