AIRBNB RECOMMENDATION SYSTEM

TEAM MEMBERS

HARSHITHA REDDY GARLAPATI

SATWIK CHOWDARY INAMPUDI

DILEEP SAI ELLANKI

PROBLEM STATEMENT

Assisting a user to find the best rooms according to their
chosen room types based on the mean price for
a particular neighbourhood location

CONTENTS

ABOUT DATASET

DATASET SIZE - CLOSE TO 50K

DATATYPES OF ATTRIBUTES

ATTRIBUTES:

HOST ID
NEIGHBOURHOOD GROUP
NEIGHBOURHOOD
LATTITUDE
LONGITUDE
ROOM TYPE
PRICE
MINIMUM NIGHTS
NUMBER OF REVIEWS
AVAILABILTY
HOST LISTINGS

0.2s

```python
data.head()
```

Table Raw Visualize Statistics

	host_id	neighbour...	latitude	longitude	room_type	price	minimum
0	2787	Brooklyn	40.65	-73.97	Private room	149	
1	2845	Manhattan	40.75	-73.98	Entire home/apt	225	
2	4632	Manhattan	40.81	-73.94	Private room	150	
3	4869	Brooklyn	40.69	-73.96	Entire home/apt	89	
4	7192	Manhattan	40.8	-73.94	Entire home/apt	80	

5 rows x 12 columns Jump to top Jump to bottom

0.2s

```python
data.shape
```

```
(48895, 11)
```

[9] 0.2s

```
data.describe()
```

Table Raw Visualize Statistics

	host_id	latitude	longitude	price	minimum_n...	number_of...	rev
cou…	48895.0	48895.0	48895.0	48895.0	48895.0	48895.0	
mean	67620010.64661008	40.72895715308314	-73.95212721137132	152.7206871868289	7.029962163820431	23.27446591788528	1.37
std	78610967.03266661	0.054564756581244…	0.046270100209320…	240.15416974718758	20.51054953317987	44.55058226668393	1.68
min	2438.0	40.5	-74.24	0.0	1.0	0.0	
25%	7822033.0	40.69	-73.98	69.0	1.0	1.0	
50%	30793816.0	40.72	-73.96	106.0	3.0	5.0	
75%	107434423.0	40.76	-73.94	175.0	5.0	24.0	
max	274321313.0	40.91	-73.71	10000.0	1250.0	629.0	

8 rows x 10 columns

Jump to top Jump to bottom

DATA PREPROCESSING

- DELETING UNNECESSARY COLUMNS WHICH DOES NOT AID IN ANALYSIS PROCESS
- CHECK FOR NULL VALUES
- REMOVING NAN COLUMNS FROM THE COLUMNS
- REPLACING NAN WITH OTHER VALUES SUCH AS MEAN
- CONVERTING ATTRIBUTE VALUES FROM CATEGORICAL TO NUMERICAL

Removing unnesary columns

[5]

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

[6]

```
data['latitude'] = data['latitude'].apply(lambda x: round(x, 2))
data['longitude'] = data['longitude'].apply(lambda x: round(x, 2))
```

[10] ▷ 0.2s

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

calculated_host_listings_count

[11]

```
data.head()
```

Table Raw Visualize Statistics

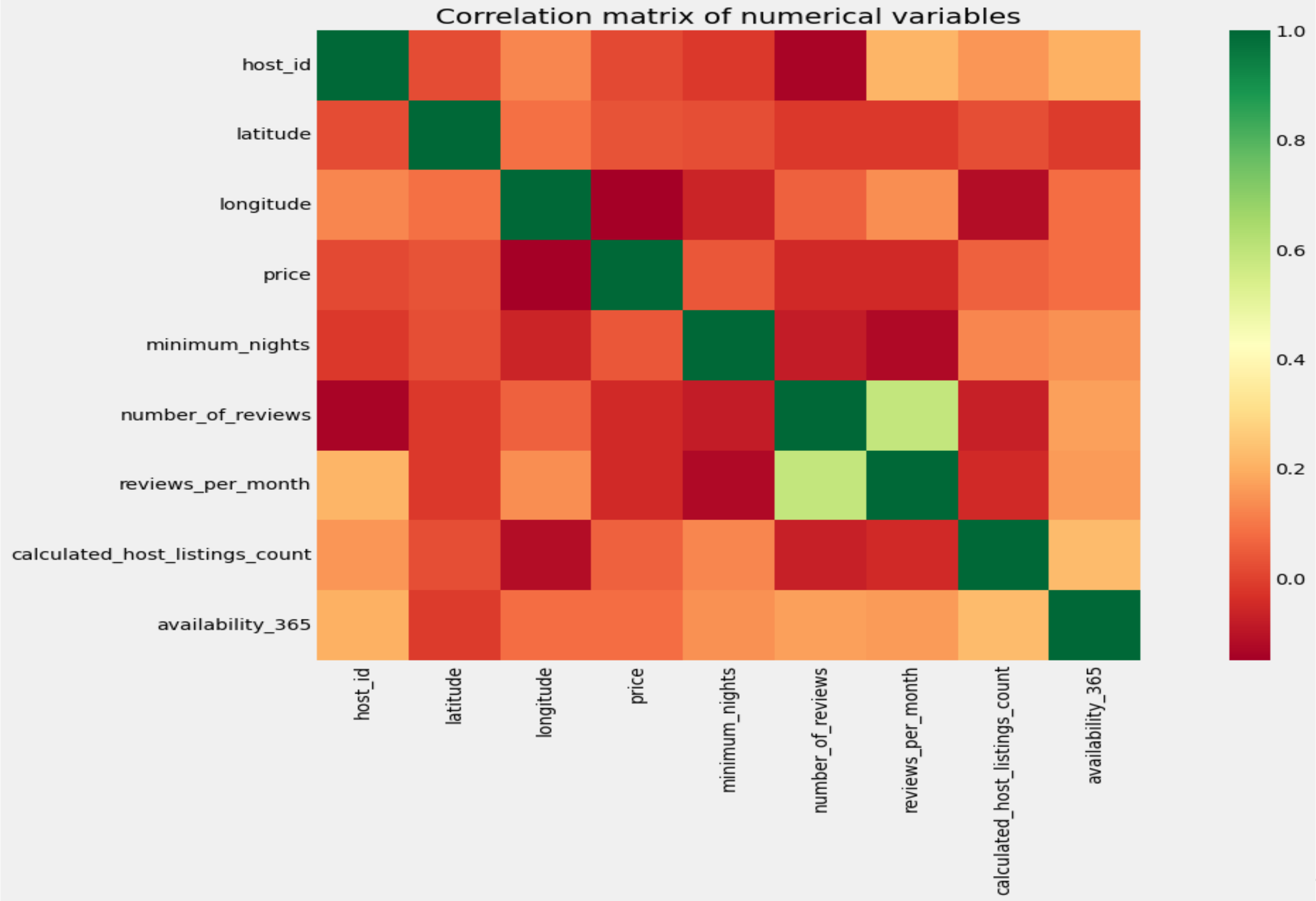
	host_id	neighbour...	latitude	longitude	room_type	price	minimum
0	2787	Brooklyn	40.65	-73.97	Private room	149	
1	2845	Manhattan	40.75	-73.98	Entire home/apt	225	
2	4632	Manhattan	40.81	-73.94	Private room	150	
3	4869	Brooklyn	40.69	-73.96	Entire home/apt	89	
4	7192	Manhattan	40.8	-73.94	Entire home/apt	80	

5 rows x 12 columns Jump to top Jump to bottom

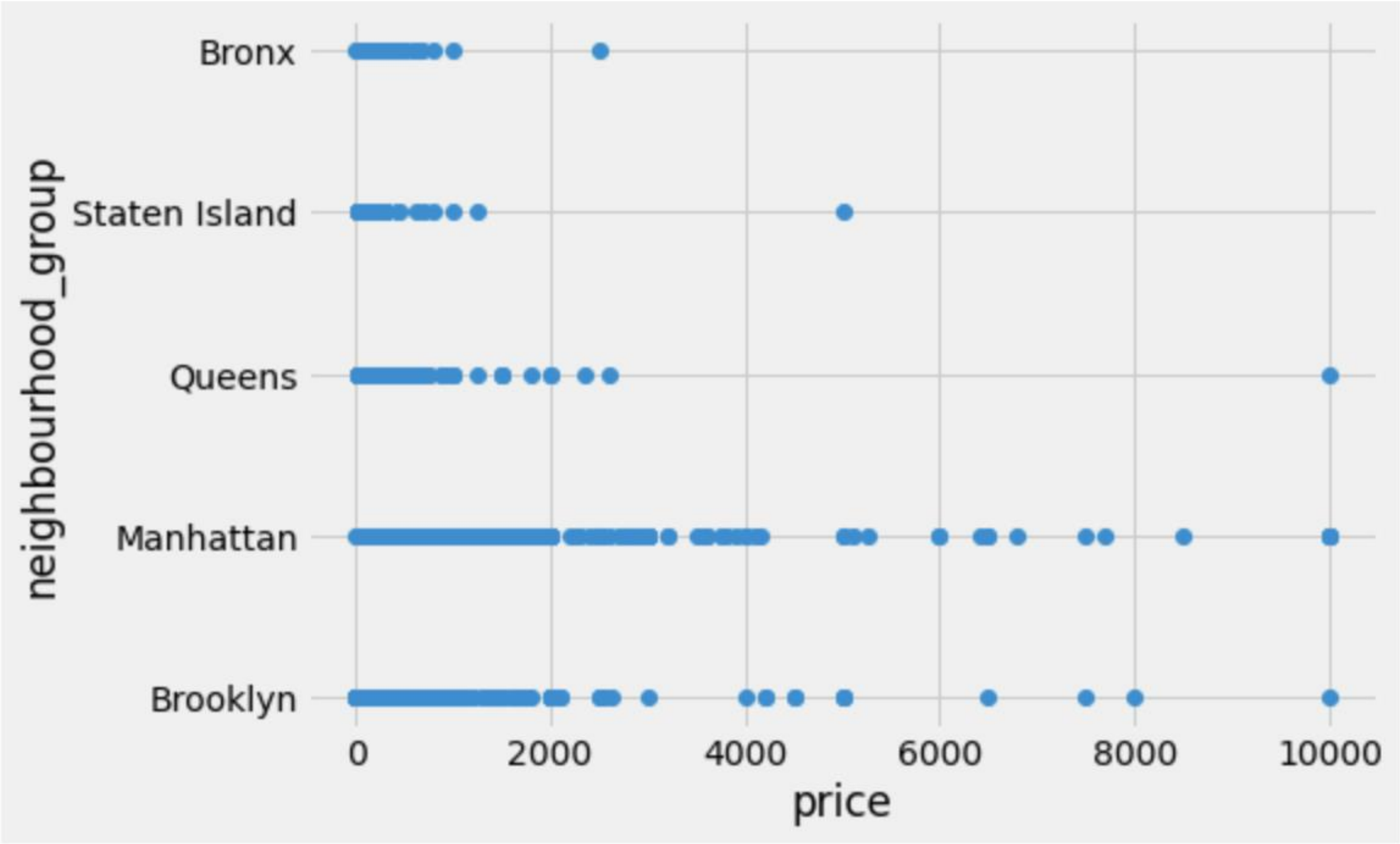
DATA VIZUALIZATION

- FINDING CORRELATION BETWEEN THE ATTRIBUTE USING HEATMAP
- PLOTTING A SCATTER PLOT BETWEEN NEIGHBOURHOOD GROUP AND PRICE
- SCATTER PLOT BETWEEN ROOM TYPE AND PRICE
- NEIGHBOURHOOD GROUP LOCATION USING LATTITUE AND LONGITUDE
- ROOM TYPE AND NEIGHBOURHOOD GROUP LOCATION
- MEDIAN PRICE PER NEIGHBOURHOOD GROUP
- PRICE PER NEIGHBOURHOOD GROUP FOR PROPERTIES UNDER 150$
- PRICE PER ROOMTYPE FOR PROPERTIES UNDER 150$

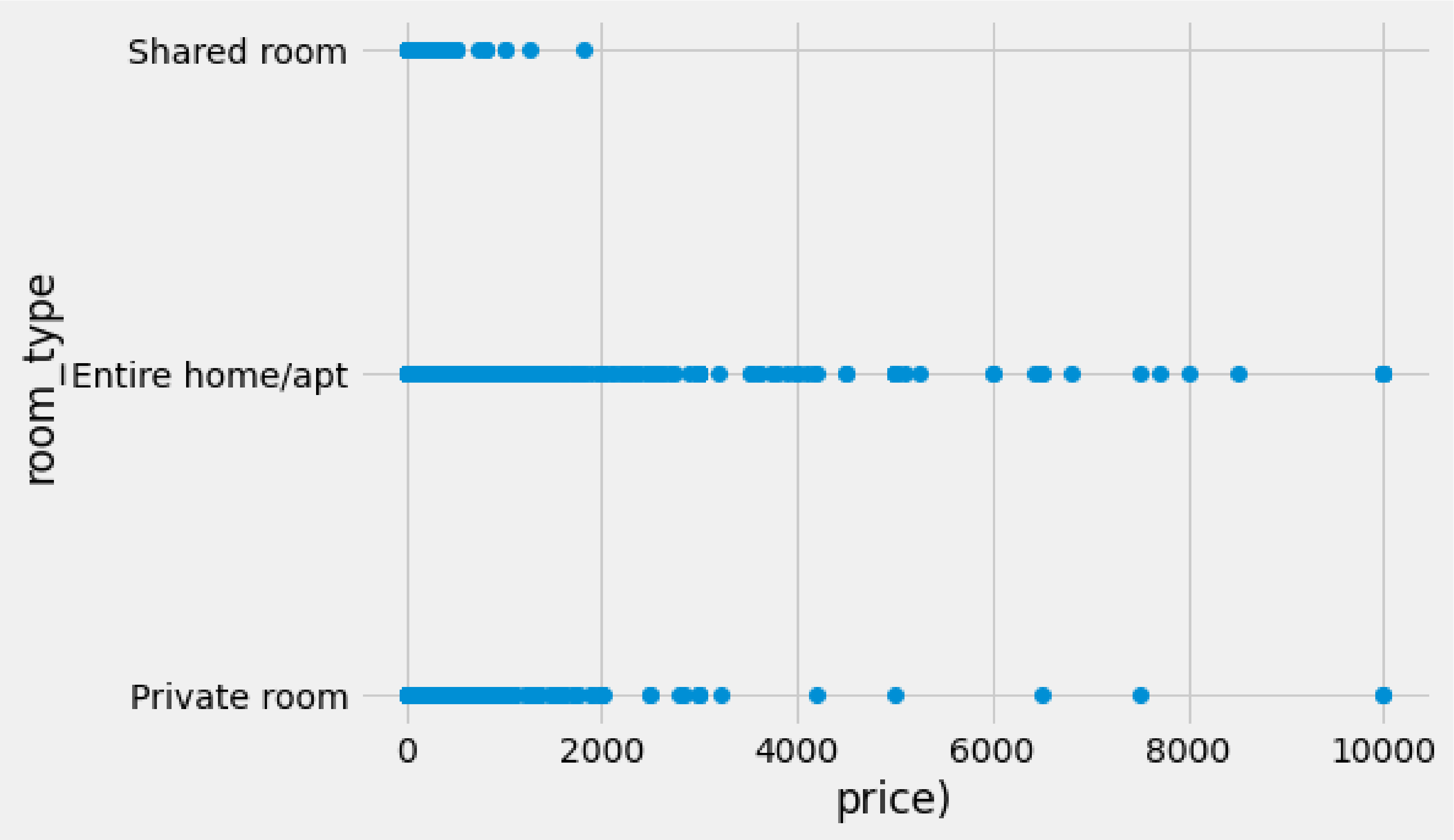
FINDING CORRELATION BETWEEN THE ATTRIBUTE USING HEATMAP



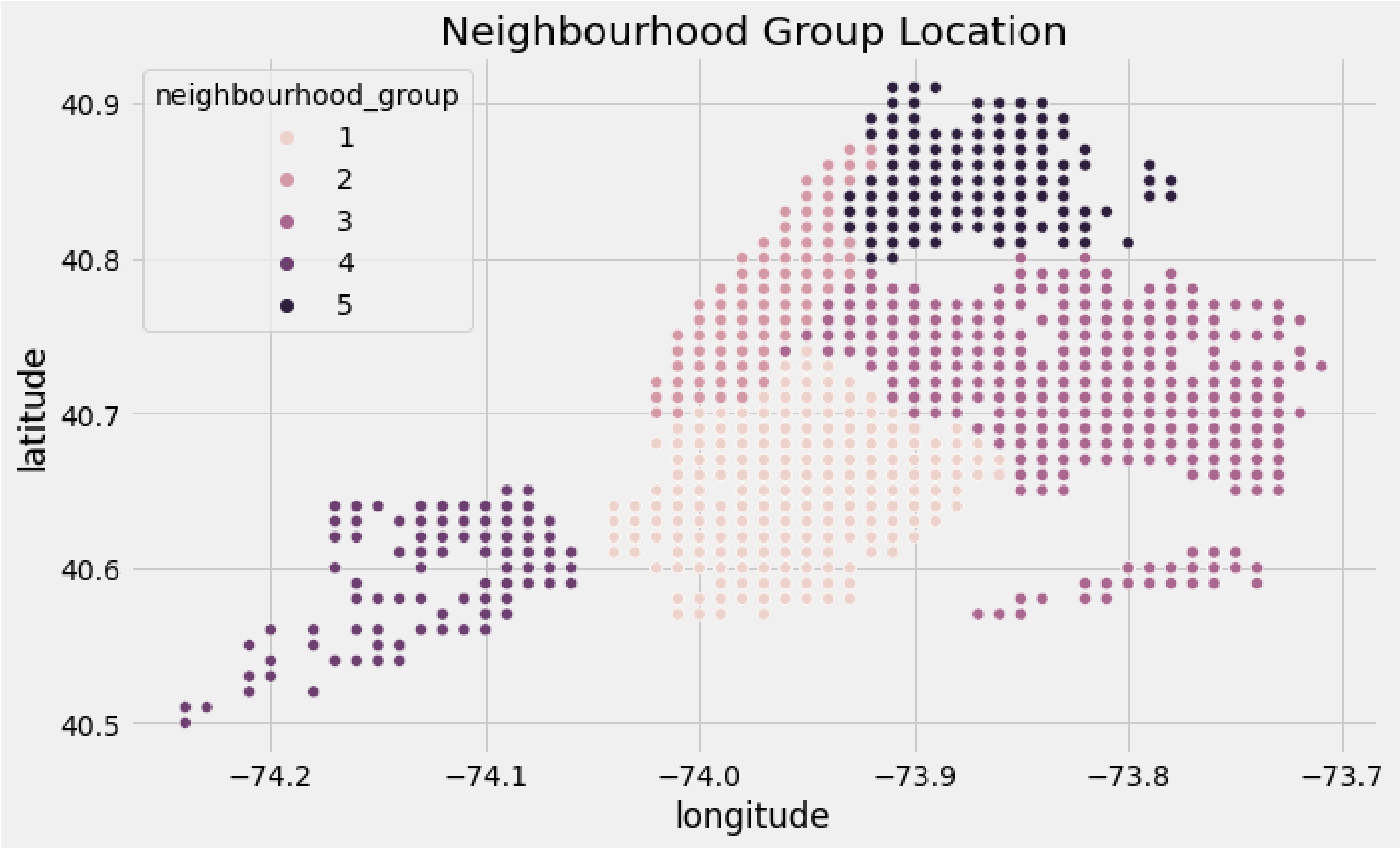
PLOTTING A SCATTER PLOT BETWEEN NEIGHBOURHOOD GROUP AND PRICE



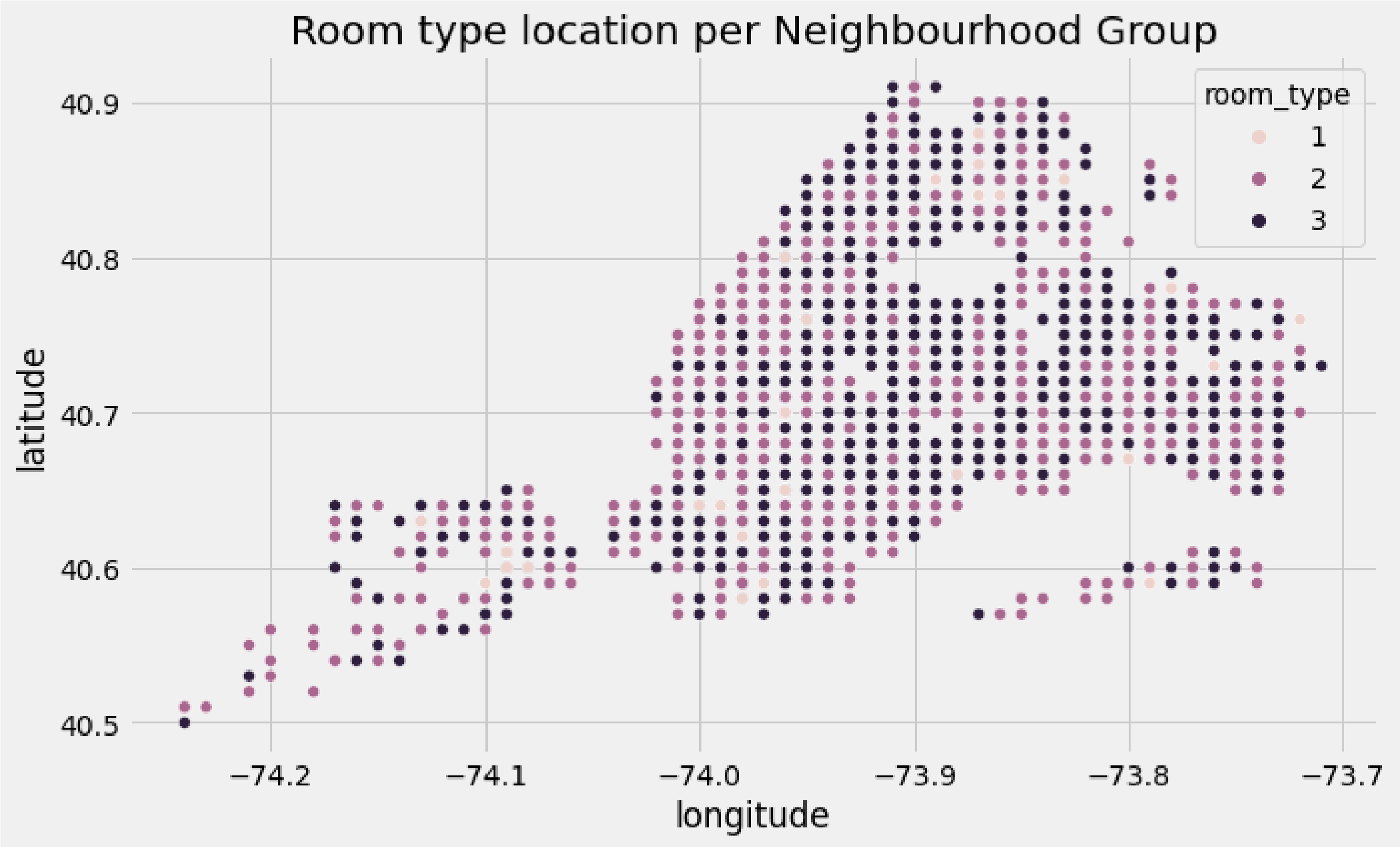
SCATTER PLOT BETWEEN ROOM TYPE AND PRICE



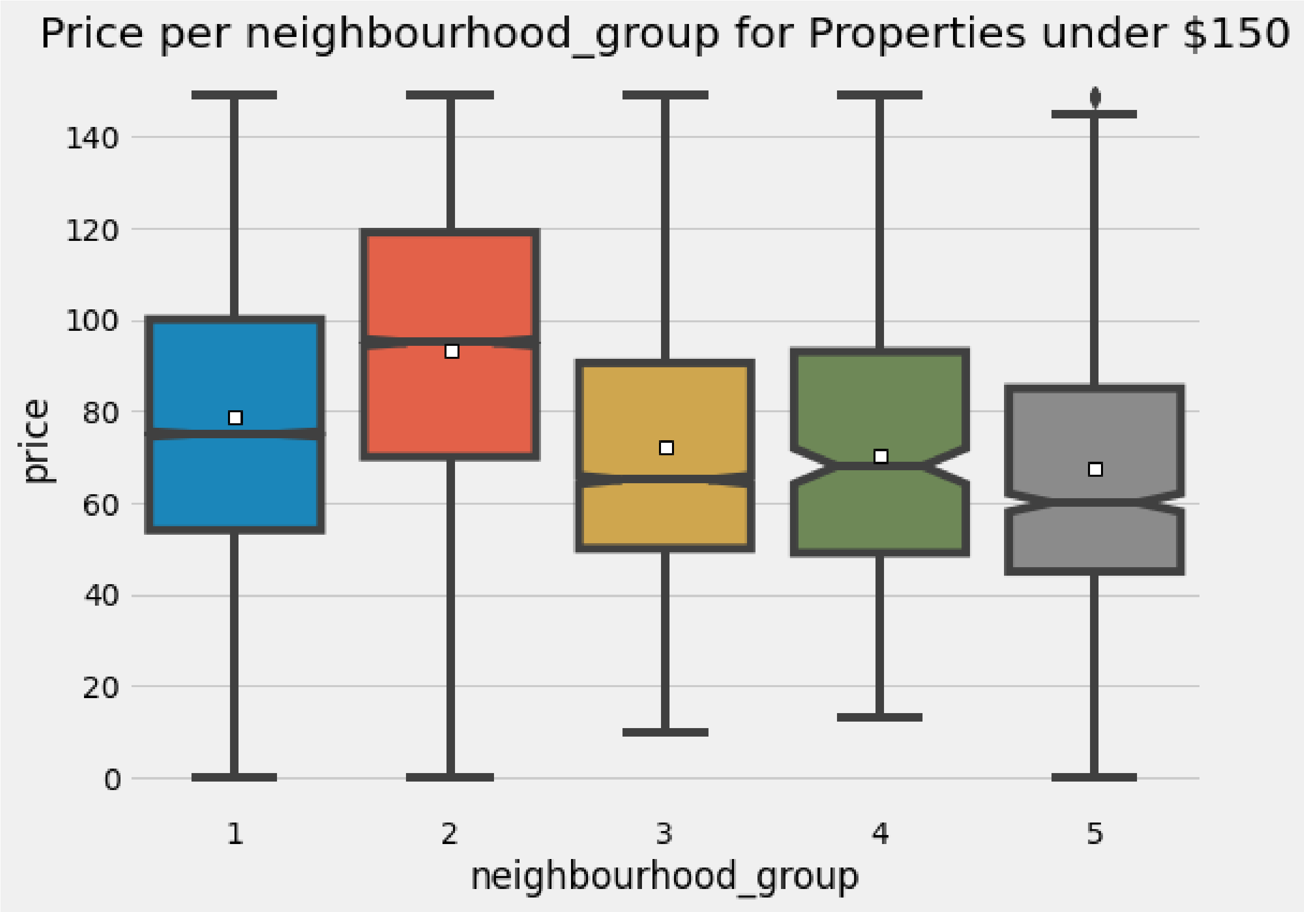
NEIGHBOURHOOD GROUP LOCATION USING LATTITUE AND LONGITUDE



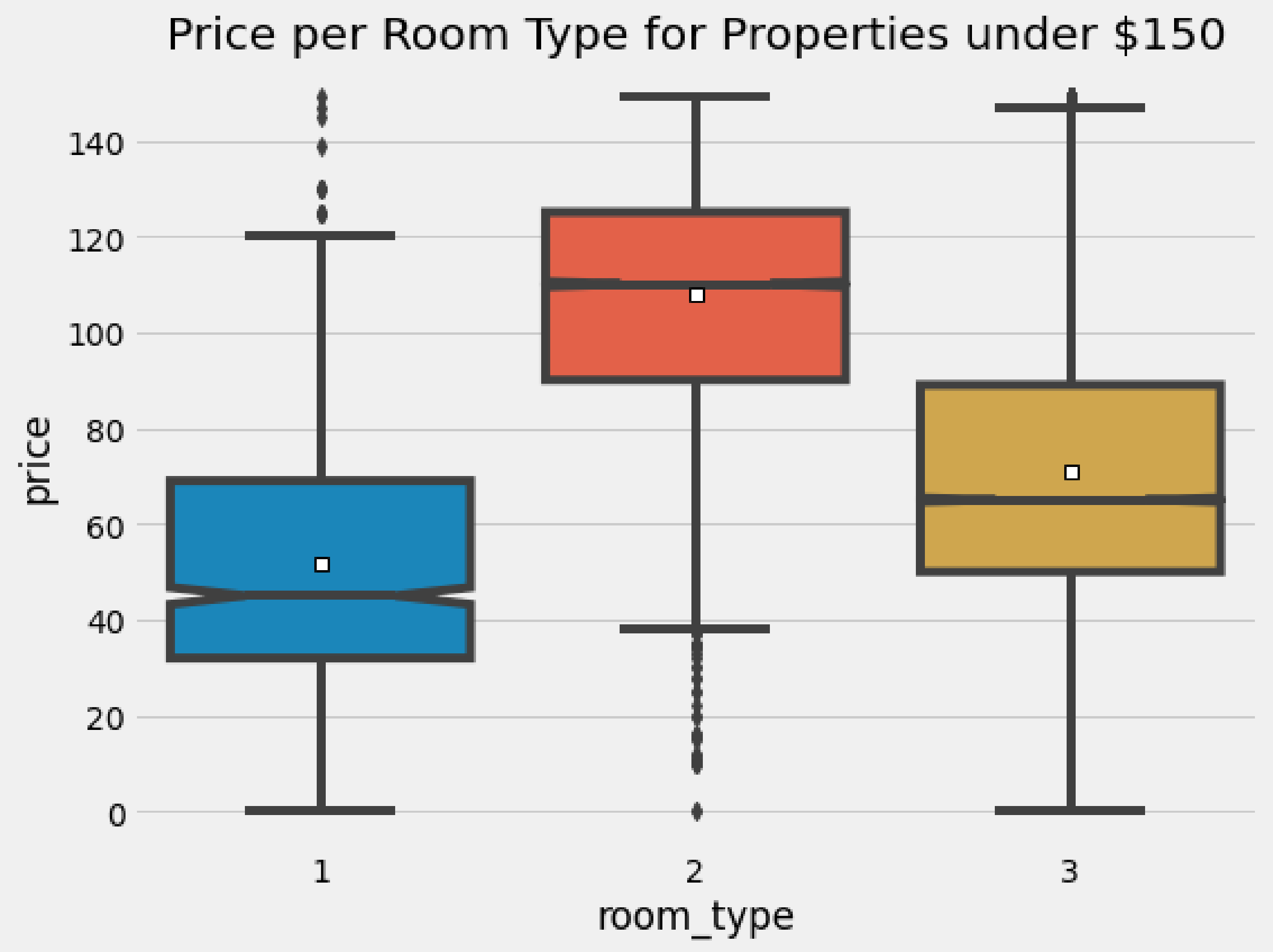
ROOM TYPE AND NEIGHBOURHOOD GROUP LOCATION



PRICE PER NEIGHBOURHOOD GROUP FOR PROPERTIES UNDER 150$



PRICE PER ROOMTYPE FOR PROPERTIES UNDER 150$



MODELS

- LINEAR REGRESSION ON NEIGHBOURHOOD GROUP AND PRICE ATTRIBUTES
- LOGISTIC REGRESSION ON AVAILABITY OF ROOMS OVER 365 DAYS
- K NEIGHBOUR CLASSIFICATION ON AVAILABILITY OF ROOMS OVER 365 DAYS
- DECISION TREE ON AVAILABITY OF ROOMS OVER 365 DAYS

MODELS NEEDED TO BE COMPUTED:

- REGRESSION AND CLASSIFICATION MODELS ON PRICE AND AVAILABILITY OF ROOMS
- REGRESSION AND CLASSIFICATION MODELS NEIGHBOURHOOD GROUP AND AVAILABILITY OF ROOMS

LINEAR REGRESSION ON NEIGHBOURHOOD GROUP AND PRICE ATTRIBUTES

[50]

```
X =  data[['room_type']].values
y = data[['price','neighbourhood_group']].values


sc_X = StandardScaler()
sc_y = StandardScaler()

X = sc_X.fit_transform(X)
y = sc_X.fit_transform(y)
```

[51] 0.3s

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state=42)
```

[52]

```
print(X_train.shape)
print(y_train.shape)
print(X_test.shape)
print(y_test.shape)
```

```
(34226, 1)
(34226, 2)
(14669, 1)
(14669, 2)
```

[53]

```
# instantiate
linreg = LinearRegression()

# fit the model to the training data (learn the coefficients)
linreg.fit(X_train, y_train)

# print the intercept and coefficients
print("intercept is: ",linreg.intercept_)

print("coefficients are: ",linreg.coef_)
```

1.7s

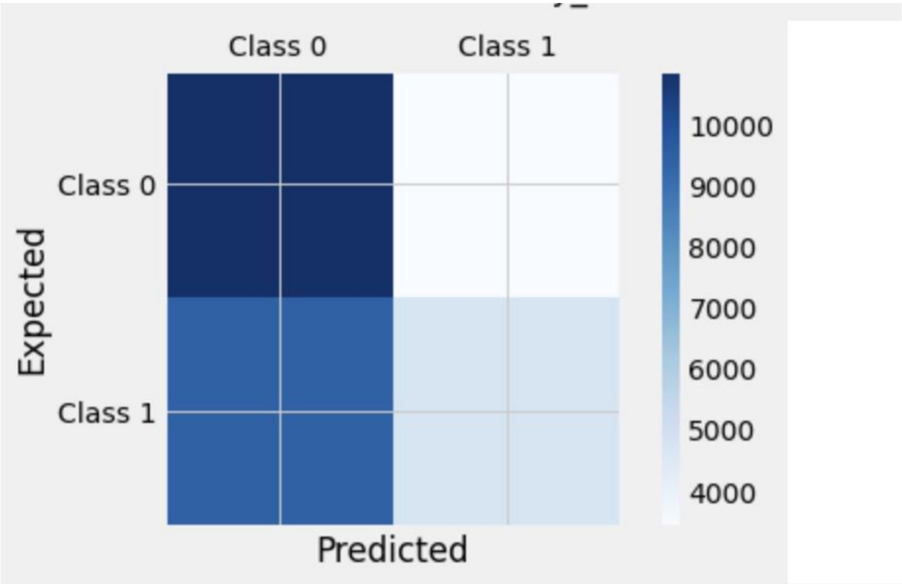
```
pipeline = Pipeline(steps=[('model', LinearRegression())])

from sklearn.model_selection import cross_val_score

# Multiply by -1 since sklearn calculates *negative* scores
scores1 =  1 * cross_val_score(pipeline, X, y,
                               cv=10,
                               scoring='r2')
scores2 = -1 * cross_val_score(pipeline, X, y,
                               cv=10,
                               scoring='neg_mean_absolute_error')
scores3 = -1 * cross_val_score(pipeline, X, y,
                               cv=10,
                               scoring='neg_root_mean_squared_error')

print("R squared scores:\n", scores1)
print("Average R :",scores1.mean())


print("RMSE scores:\n", scores3)
print("Average RMSE score:",scores3.mean())
```

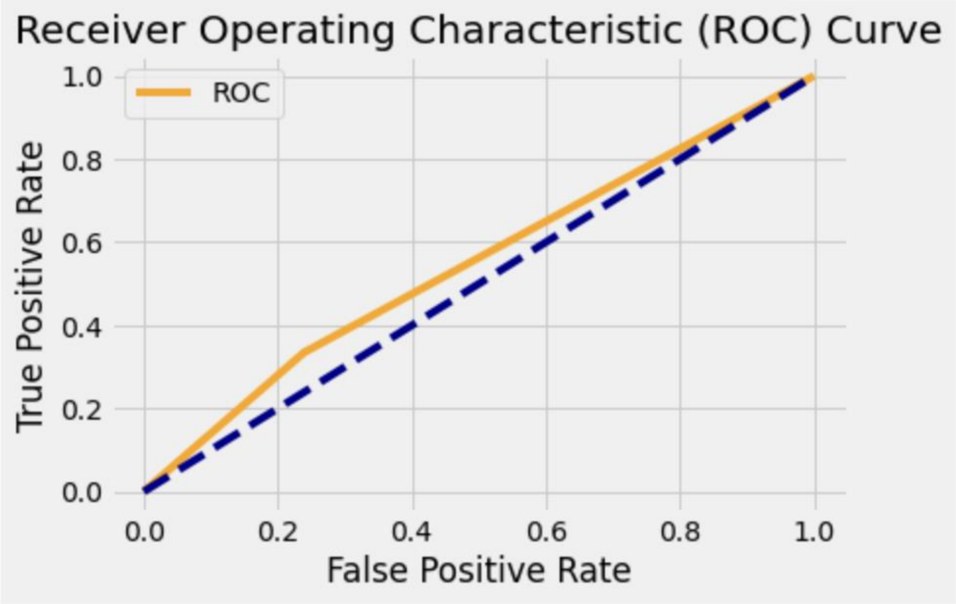
```
R squared scores:
 [0.29260765 0.38739206 0.37423347 0.36551587 0.34510415 0.40695468
 0.32323162 0.36902534 0.36763048 0.33044815]
Average R squared score (across experiments): 0.35621434788376827
RMSE scores:
 [0.73127889 0.69449903 0.67717075 0.75067888 0.83983606 0.79793824
 0.87941837 0.84410726 0.85331137 0.88023837]
Average RMSE score (across experiments): 0.7948477218588952
```

LOGISTIC REGRESSION ON AVAILABITY OF ROOMS OVER 365 DAYS

```
from sklearn.linear_model import LogisticRegression
classifier = GaussianNB()

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state=42)

classifier.fit(X_train,y_train)

y_pred = classifier.predict(X_test)

from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test,y_pred)


classif_results()
```

```
Confusion matrix:
 [[10914  3425]
 [ 9466  4755]]
Accuracy 0.5486344537815127
              precision    recall  f1-score   support

           0       0.54      0.76      0.63     14339
           1       0.58      0.33      0.42     14221

    accuracy                           0.55     28560
   macro avg       0.56      0.55      0.53     28560
weighted avg       0.56      0.55      0.53     28560

AUC Score:
0.5477528081209202
```


Class 0
Class 1
Class 0
Class 1
Expected
Predicted
10000
9000
8000
7000
6000
5000
4000

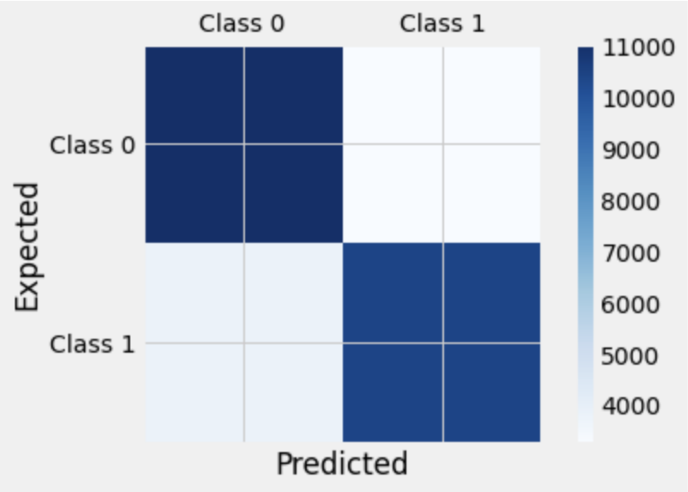

Receiver Operating Characteristic (ROC) Curve
ROC
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0

K NEIGHBOUR CLASSIFICATION ON AVAILABILITY OF ROOMS OVER 365 DAYS

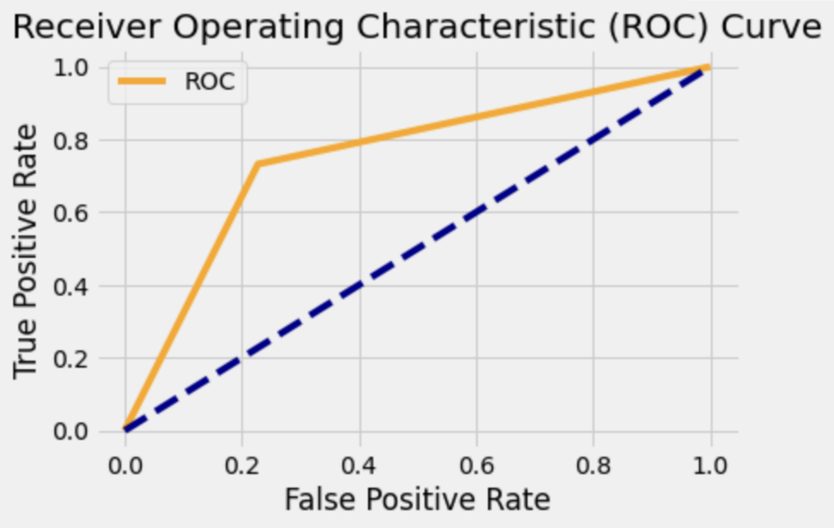
[66]

```
classifier = KNeighborsClassifier()

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state=42)

classifier.fit(X_train,y_train)

y_pred = classifier.predict(X_test)

from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test,y_pred)


classif_results()
```

```
Confusion matrix:
 [[11075  3264]
 [ 3809 10412]]
Accuracy 0.7523459383753501
              precision    recall  f1-score   support

           0       0.74      0.77      0.76     14339
           1       0.76      0.73      0.75     14221

    accuracy                           0.75     28560
   macro avg       0.75      0.75      0.75     28560
weighted avg       0.75      0.75      0.75     28560

AUC Score:
0.7522628665536728
```



Class 0
Class 1
11000
10000
9000
8000
7000
6000
5000
4000
Class 0
Class 1
Expected
Predicted



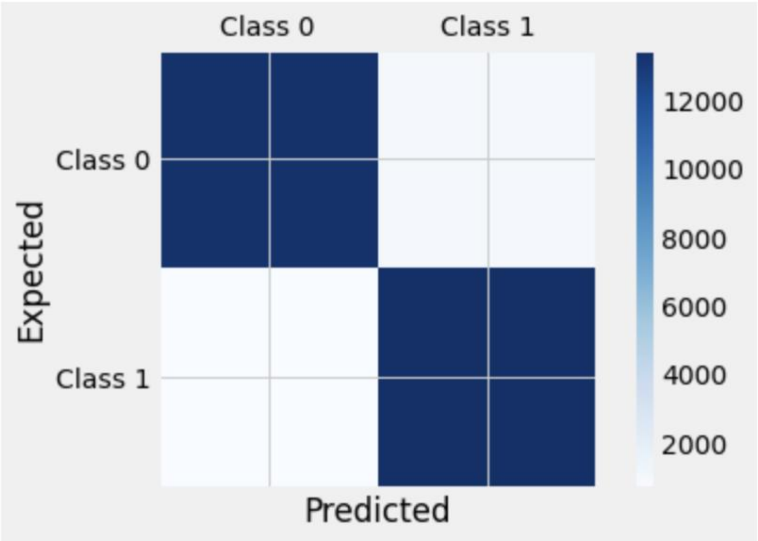
Receiver Operating Characteristic (ROC) Curve
ROC
1.0
0.8
0.6
0.4
0.2
0.0
True Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
False Positive Rate

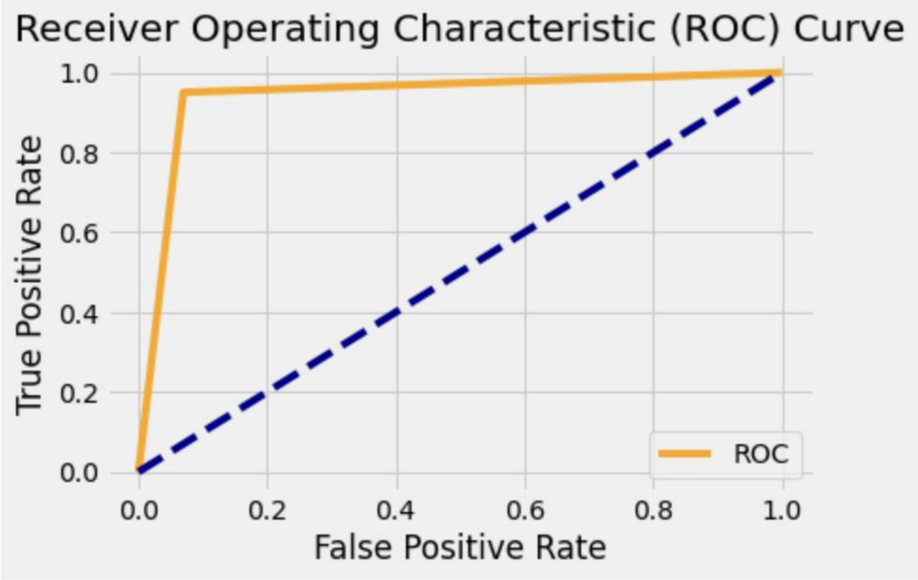
DECISION TREE ON AVAILABITY OF ROOMS OVER 365 DAYS

[68] 1.4s

```
classifier = tree.DecisionTreeClassifier()
classifier.fit(X_train,y_train)

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state=42)

y_pred = classifier.predict(X_test)

from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test,y_pred)


classif_results()
```

```
Confusion matrix:
 [[13338  1001]
 [  706 13515]]
Accuracy 0.9402310924369748
              precision    recall  f1-score   support

           0       0.95      0.93      0.94     14339
           1       0.93      0.95      0.94     14221

    accuracy                           0.94     28560
   macro avg       0.94      0.94      0.94     28560
weighted avg       0.94      0.94      0.94     28560

AUC Score:
0.9402727492440119
```


Class 0
Class 1
Class 0
Class 1
Expected
Predicted
12000
10000
8000
6000
4000
2000


Receiver Operating Characteristic (ROC) Curve
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
ROC

CONCLUSION SO FAR..

- We had an accuracy of 60% in regression models
- Accuracy of 94% was observed in classification model so far