Sri Lanka Institute of Information Technology

Rental Price Prediction in China

Final Report

FDM Mini Project 2022

Submitted by:

1. IT20204716 – (Hemachandra K.M.T.A.)
2. IT20149680– (Athurugiriya A.A.A.G.)
3. IT20203412 – (Dinoja N.)
4. IT20123772 – (Pallawala P.K.B.D.S.)
5. IT20123840 – (Sewwandi R.A.U.T)

Table of Contents

Contents

Introduction and Problem Definition

We were advised to find out a real-world problem and propose a solution for it as our mini project. Data Mining and Machine Learning are the techniques that we can use. After discussing with the group members, we decided to take a rental price prediction problem which is very useful for property owners as well as the buyers.

Problem:

If a property owner who do not have a better knowledge about rental prices, wants to predict the price for his/her property, that person has to waste a lot of time to find the details of rental properties. Not only that, but also buyers also have to put in a lot of effort to find the budget of a property that suits their needs. These people do the predictions according to their knowledge. Sometimes they are successful, sometimes they are not.

We identified 3 main problems that property owners and buyers face.

1. Property owners do not know how to predict the price of their property correctly and buyers also do not have an idea about the price of a property that they need.
2. Often a person is not aware of whether it is appropriate to cancel a booking. It can be a problem for buyers because they do not know the policies.
3. When a person wants to develop his/her rental business, it is difficult to decide which type of properties are best for rent.

By analyzing historical data of properties, we can mine these data and identify patterns that will finally conclude the rental price of a property as well as the cancellation policy. And also, we can predict which type of properties are best to expand the business.

This will help property owners to predict the price of their properties, and also the buyers to find the price of a property according to the features that they want. Buyers can predict the cancellation policy type of the property, so that they can get an idea about the risk of canceling a booked property. Property owners can get a better idea about which types of properties can gain a good profit.

The dataset will be used to create models and to make the following predictions:

1. Predict the price of a property
2. Predict the cancellation policy of a booking
3. Predict the most demanding property type

The application uses python for backend and Streamlit for frontend and deployed in Heroku.
https://ororentalpredictions.herokuapp.com/

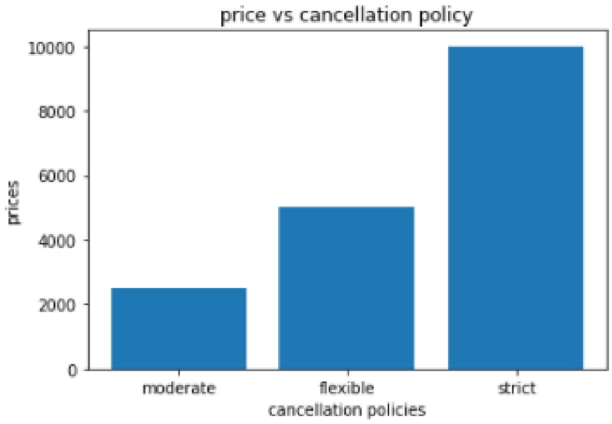
Description of Variables in Dataset

In our dataset we have data of 5834 current rental properties. There are 26 features that have been collected as below,

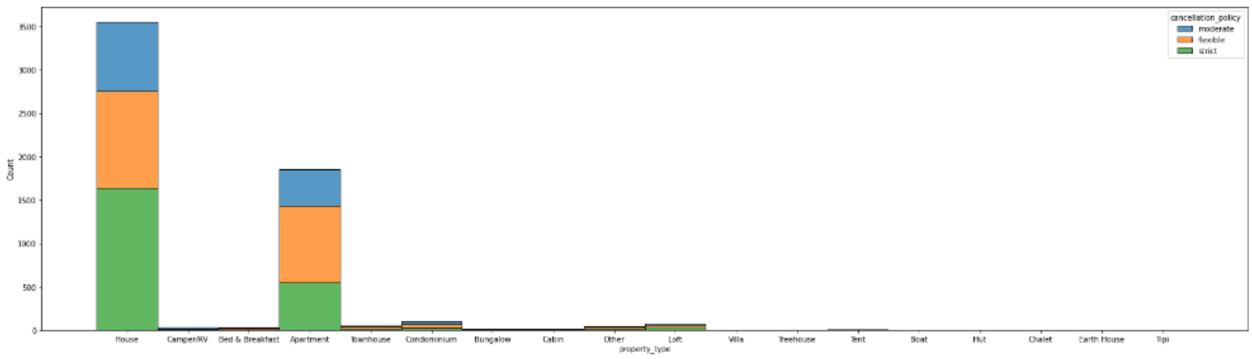
Var#	Variable Name	Description	Variable Type
1.	accommodates	Accommodates	Numeric
2.	amenities	Amenities/facilities available	Categorical
3.	availability_30	Available for upcoming 30 days	Numeric
4.	bathrooms	No of bathrooms	Numeric
5.	bed_type	bed_type (‘Real bed, Airbed’, 'Futon’,’ Couch’, ‘Pull-out sofa’)	Categorical
6.	bedrooms	No of bedrooms	Numeric
7.	Beds	No of Beds	Numeric
8.	calculated_host_listings_count	host_listings_count in trending list	Numeric
9.	guests_included	No of guests included	Numeric
10.	has_availability	Has availability? (Yes or No)	Binary
11.	host_is_superhost	Is the host superhost? (Yes or No)	Binary
12.	host_listings_count	host_listings_count	Numeric
13.	instant_bookable	Instant bookable available? (Yes or No)	Binary
14.	latitude (North)	Latitude-North (Location)	Numeric
15.	longitude (East)	Longitude-East (Location)	Numeric
16.	maximum_nights	No of maximum nights	Numeric
17.	number_of_reviews	Number of reviews	Numeric
18.	property_type	Property type (‘Apartment’, ‘Bed & Breakfast’, ‘Boat’, ‘Bungalow’, ‘Cabin’, ‘Camper/RV’, ‘Chalet’, ‘Condominium’, ‘Earth House’, ‘House’, ‘Hut’, ‘Loft’, ‘Other’, ‘Tent’, ‘Tipi’, ‘Townhouse’, ‘Treehouse’, ’Villa’)	Categorical
19.	review_scores_checkin	Checkin review score	Numeric
20.	review_scores_communication	Communication review score	Numeric
21.	review_scores_location	Location review scores	Numeric
22.	review_scores_rating	Rating review score	Numeric
23.	review_scores_value	Review scores value	Numeric
24.	room_type	Room type (‘Private room’, ‘Shared room’, ‘Entire home/apt’)	Categorical
25.	price	rental price	Numeric
26.	cancellation_policy	Cancellation policy (‘strict’, ‘flexible’, ‘moderate’)	Categorical

Data Visualization

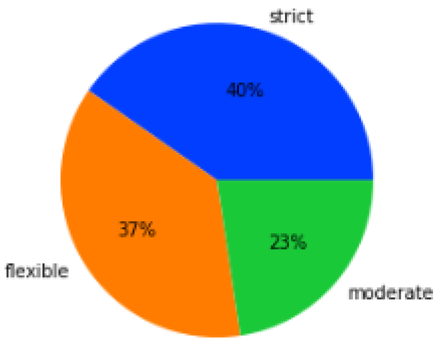
To visualize the data in the dataset, graphs and charts were prepared using the python, and these graphs can be found in our Application.



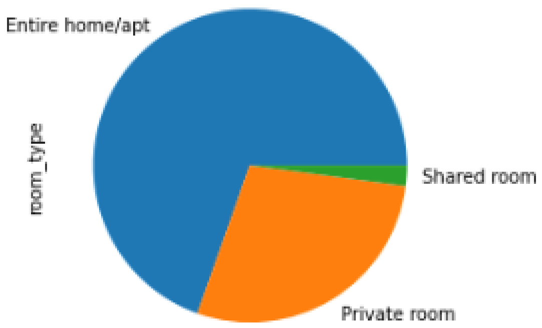
The above bar chart displays how the cancellation policy variates with respect to the price.



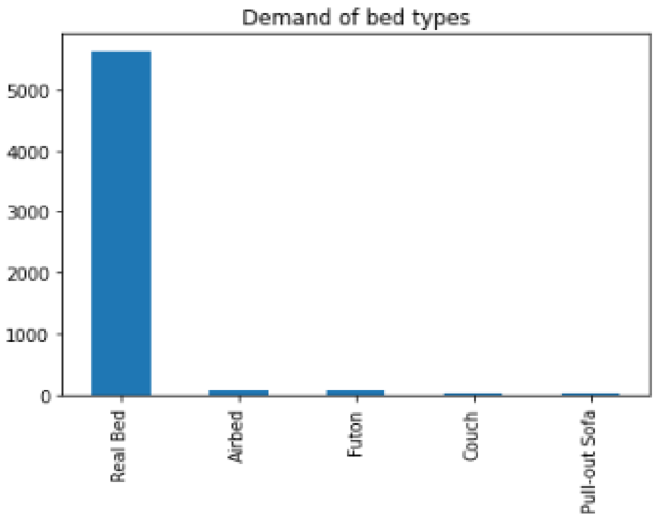
The above bar chart depicts the cancellation policy according to each property type.



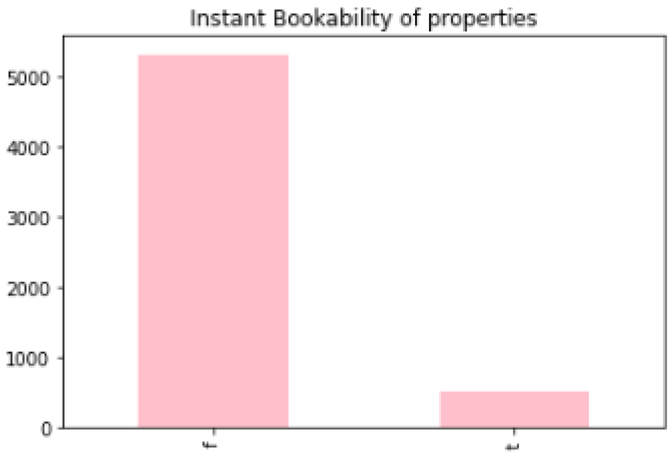
The above pie chart displays the percentages of cancellation policies out of the whole.



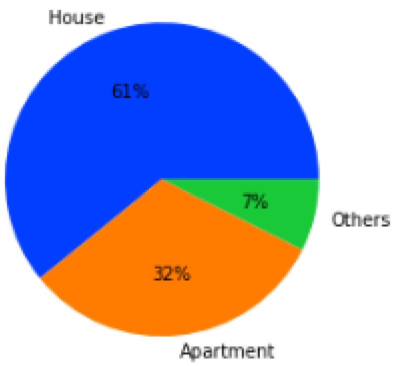
The above pie chart displays the demand for each room type.



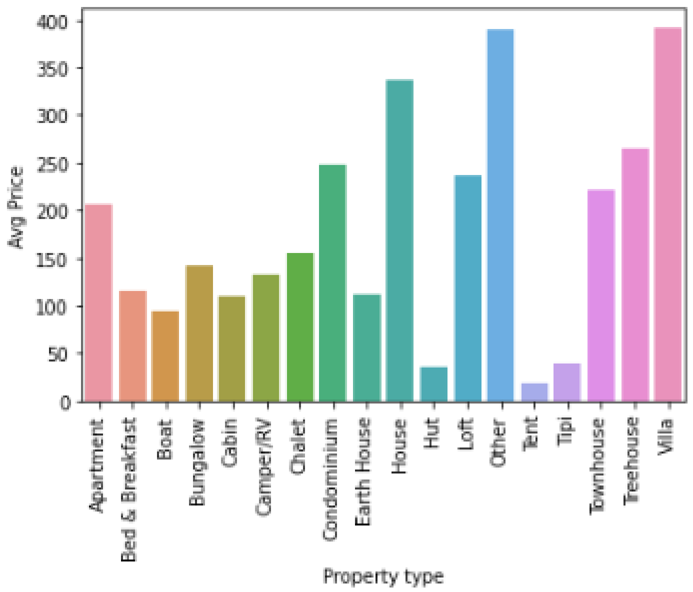
Looking at the above bar chart we can deduce that the mostly demanded bed type is 'Real bed'



Looking at the above bar chart we can conclude that most of the property types are not instantly bookable.



Looking at the above pie chart we can conclude that ‘house’ property type is highly demanded.



Above bar graph shows the average price of each property price and villas are the most expensive.

Data Preparation

The data set is partitioned into:

Training set – 70% - 4083 records

Testing set – 30% - 1751 records

Data Preprocessing

Importing dataset & split dataset

In [128]:
```
%cd F:\FDM Group Project
```
```
F:\FDM Group Project
```

In [129]:
```
!pip install missingpy
!pip install miceforest
```
```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: missingpy in c:\users\thari\appdata\roaming\python\python39\site-packages (0.2.0)
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: miceforest in c:\users\thari\appdata\roaming\python\python39\site-packages (5.6.2)
Requirement already satisfied: lightgbm>=3.3.1 in c:\users\thari\appdata\roaming\python\python39\site-packages (from micefores
t) (3.3.3)
Requirement already satisfied: blosc in c:\users\thari\appdata\roaming\python\python39\site-packages (from miceforest) (1.10.6)
Requirement already satisfied: numpy in c:\programdata\anaconda3\lib\site-packages (from miceforest) (1.21.5)
Requirement already satisfied: dill in c:\programdata\anaconda3\lib\site-packages (from miceforest) (0.3.5.1)
Requirement already satisfied: wheel in c:\programdata\anaconda3\lib\site-packages (from lightgbm>=3.3.1->miceforest) (0.37.1)
Requirement already satisfied: scikit-learn!=0.22.0 in c:\users\thari\appdata\roaming\python\python39\site-packages (from light
gbm>=3.3.1->miceforest) (1.1.2)
Requirement already satisfied: scipy in c:\programdata\anaconda3\lib\site-packages (from lightgbm>=3.3.1->miceforest) (1.7.3)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\programdata\anaconda3\lib\site-packages (from scikit-learn!=0.22.0->l
ightgbm>=3.3.1->miceforest) (2.2.0)
Requirement already satisfied: joblib>=1.0.0 in c:\programdata\anaconda3\lib\site-packages (from scikit-learn!=0.22.0->lightgbm
>=3.3.1->miceforest) (1.1.0)
```

In [130]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import miceforest as mf
import random
import sklearn.neighbors._base
import sys
sys.modules['sklearn.neighbors.base'] = sklearn.neighbors._base
from missingpy import MissForest
from sklearn.impute import KNNImputer
```

In [131]:
```
data = pd.read_csv(r'Data\rental_price_data.csv')
```

In [132]:
```
data.head()
```

Out[132]:

	accommodates	amenities	availability_30	bathrooms	bed_type	bedrooms	beds	calculated_host_listings_count	cancellation_policy	gue
0	6	{"Cable TV",Internet,"Wireless Internet","Air ...	0	2.5	Real Bed	1.0	3.0	1	moderate	
1	2	{"Air Conditioning",Heating,"Family/Kid Friend...	29	0.0	Futon	1.0	1.0	1	moderate	
2	2	{TV,"Cable TV",Internet,"Wireless Internet","A...	30	1.5	Real Bed	1.0	1.0	2	flexible	
3	2	{TV,"Cable TV",Internet,"Wireless Internet","A...	30	1.5	Real Bed	1.0	1.0	2	flexible	
4	6	{TV,"Cable TV",Internet,"Wireless Internet","A...	27	2.0	Real Bed	3.0	3.0	2	strict	

In [133]:
```
#preprocess the amentities column.
#extracting the values as features.
amentities_key = set()
def extract_feature(x):
    s1 = x.replace('{', '').replace('}', '').replace('"', '')
    s1 = s1.split(',')
    for s in s1:
        amentities_key.add(s)
        #print(s)

data['amenities'].apply(extract_feature)
```

Out[133]:
```
0       None
1       None
2       None
3       None
4       None
        ...
5829    None
5830    None
5831    None
5832    None
5833    None
Name: amenities, Length: 5834, dtype: object
```

In [137]:
```python
#we store each value as a variable in the dataset.
#and then storing 1 for the variables if the amintities column is having the value. otherwise 0.
for amentities in amentities_key:
    data['amenities__'+amentities.replace(' ','_')] = data['amenities'].str.contains(amentities).astype(int)
```

```
C:\Users\thari\AppData\Local\Temp\ipykernel_18276\3560883285.py:4: UserWarning: This pattern is interpreted as a regular expres
sion, and has match groups. To actually get the groups, use str.extract.
  data['amenities__'+amentities.replace(' ','_')] = data['amenities'].str.contains(amentities).astype(int)
C:\Users\thari\AppData\Local\Temp\ipykernel_18276\3560883285.py:4: UserWarning: This pattern is interpreted as a regular expres
sion, and has match groups. To actually get the groups, use str.extract.
  data['amenities__'+amentities.replace(' ','_')] = data['amenities'].str.contains(amentities).astype(int)
C:\Users\thari\AppData\Local\Temp\ipykernel_18276\3560883285.py:4: UserWarning: This pattern is interpreted as a regular expres
sion, and has match groups. To actually get the groups, use str.extract.
  data['amenities__'+amentities.replace(' ','_')] = data['amenities'].str.contains(amentities).astype(int)
```

In [138]:
```python
#now all together we have 67 variables.
data.head()
```

Out[138]:

	accommodates	amenities	availability_30	bathrooms	bed_type	bedrooms	beds	calculated_host_listings_count	cancellation_policy	gue
0	6	{"Cable TV",Internet,"Wireless Internet","Air ...	0	2.5	Real Bed	1.0	3.0	1	moderate	
1	2	{"Air Conditioning",Heating,"Family/Kid Friend...	29	0.0	Futon	1.0	1.0	1	moderate	
2	2	{TV,"Cable TV",Internet,"Wireless Internet","A...	30	1.5	Real Bed	1.0	1.0	2	flexible	
3	2	{TV,"Cable TV",Internet,"Wireless Internet","A...	30	1.5	Real Bed	1.0	1.0	2	flexible	
4	6	{TV,"Cable TV",Internet,"Wireless Internet","A...	27	2.0	Real Bed	3.0	3.0	2	strict	

In [139]:
```python
#no need of amentities column now.
data.drop(['amenities'], axis=1, inplace=True)
```

In [142]:
```python
#replace $ sign
data['price'] = data['price'].str.replace('$','')
#replace , sign
data['price'] = data['price'].str.replace(',','')
```

```
C:\Users\thari\AppData\Local\Temp\ipykernel_18276\3560235492.py:2: FutureWarning: The default value of regex will change from T
rue to False in a future version. In addition, single character regular expressions will *not* be treated as literal strings wh
en regex=True.
  data['price'] = data['price'].str.replace('$','')
```

In [143]:
```python
data.head()
```

Out[143]:

	accommodates	availability_30	bathrooms	bed_type	bedrooms	beds	calculated_host_listings_count	cancellation_policy	guests_included	has_availability
0	6	0	2.5	Real Bed	1.0	3.0	1	moderate	1	t
1	2	29	0.0	Futon	1.0	1.0	1	moderate	1	t
2	2	30	1.5	Real Bed	1.0	1.0	2	flexible	1	t
3	2	30	1.5	Real Bed	1.0	1.0	2	flexible	1	t
4	6	27	2.0	Real Bed	3.0	3.0	2	strict	1	t

In [144]:
```
data.dtypes
```
Out[144]:
```
accommodates                      int64
availability_30                   int64
bathrooms                       float64
bed_type                         object
bedrooms                        float64
                                  ...
amenities__Wheelchair_Accessible  int32
amenities__Family/Kid_Friendly    int32
amenities__Fire_Extinguisher      int32
amenities__Hangers                int32
amenities__Smoking_Allowed        int32
Length: 67, dtype: object
```
In [148]:
```
data['bed_type'] = data['bed_type'].astype('category')
data['cancellation_policy'] = data['cancellation_policy'].astype('category')
data['instant_bookable'] = data['instant_bookable'].astype('category')
data['property_type'] = data['property_type'].astype('category')
data['room_type'] = data['room_type'].astype('category')
data['has_availability'] = data['has_availability'].astype('category')
data['host_is_superhost'] = data['host_is_superhost'].astype('category')
data['price'] = data['price'].astype('float')
```
In [149]:
```
df = pd.DataFrame(data.dtypes)
df.to_csv('cols.csv')
```
In [150]:
```
kernel = mf.ImputationKernel(
  data=data,
  save_all_iterations=True,
  random_state=10
)

kernel.mice(5,verbose=True)
#print(kernel)
completed_dataset = kernel.complete_data(dataset=0, inplace=False)
```
```
C:\Users\thari\AppData\Roaming\Python\Python39\site-packages\miceforest\ImputationKernel.py:369: UserWarning: [cancellation_pol
icy,property_type] have very rare categories, it is a good idea to group these, or set the min_data_in_leaf parameter to preven
tlightgbm from outputting 0.0 probabilities.
  warn(
```
```
Initialized logger with name mice 1-5
Dataset 0
1  | bedrooms | host_is_superhost | host_listings_count | beds | bathrooms | review_scores_rating | review_scores_location | re
view_scores_checkin | review_scores_value | review_scores_communication
2  | bedrooms | host_is_superhost | host_listings_count | beds | bathrooms | review_scores_rating | review_scores_location | re
view_scores_checkin | review_scores_value | review_scores_communication
3  | bedrooms | host_is_superhost | host_listings_count | beds | bathrooms | review_scores_rating | review_scores_location | re
view_scores_checkin | review_scores_value | review_scores_communication
4  | bedrooms | host_is_superhost | host_listings_count | beds | bathrooms | review_scores_rating | review_scores_location | re
view_scores_checkin | review_scores_value | review_scores_communication
5  | bedrooms | host_is_superhost | host_listings_count | beds | bathrooms | review_scores_rating | review_scores_location | re
view_scores_checkin | review_scores_value | review_scores_communication
```
In [151]:
```
completed_dataset['bathrooms'].round(0)
completed_dataset['beds'].round(0)
```
Out[151]:
```
0       3.0
1       1.0
2       1.0
3       1.0
4       3.0
       ...
5829    2.0
5830    2.0
5831    2.0
5832    1.0
5833    1.0
Name: beds, Length: 5834, dtype: float64
```
In [152]:
```
completed_dataset['bathrooms'] = completed_dataset['bathrooms'].astype('int')
completed_dataset['beds'] = completed_dataset['beds'].astype('int')
```
In [154]:
```
completed_dataset.to_csv('preprocessed_dataset_1.csv')
```

We drop columns that have too many unknown data

Prediction 01 - Predict the price of a rental property

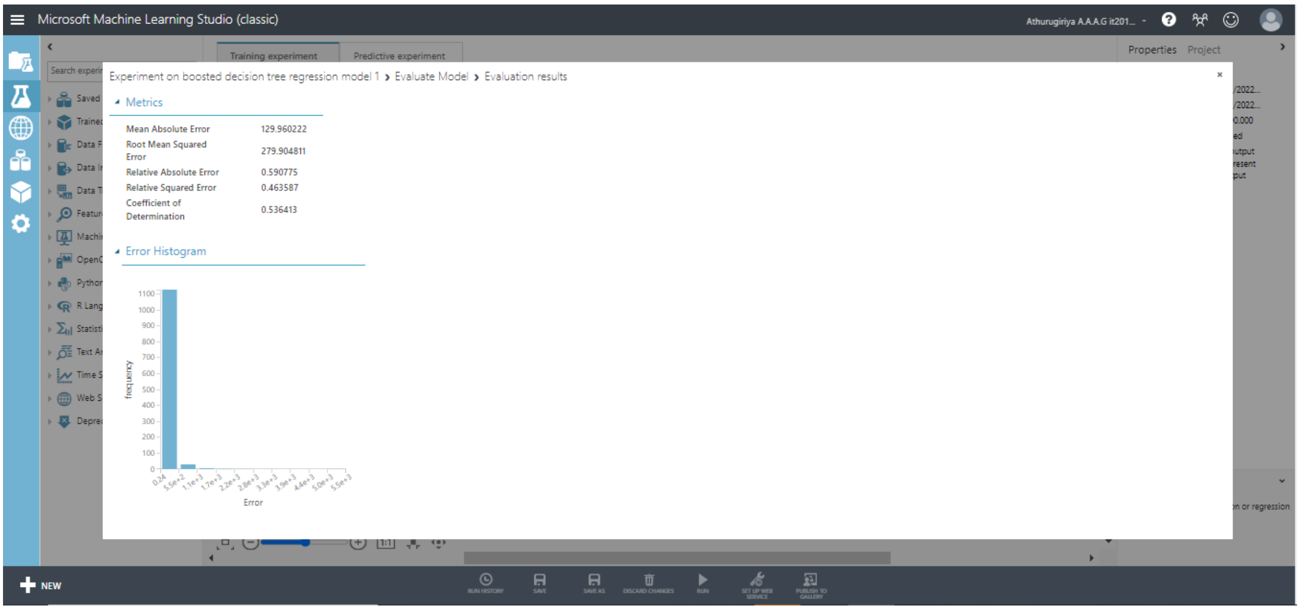
- ## Description

  Predict the price of a rental property using historical data. Prediction is done by using the price column.

- ## Analyzing Regression Models

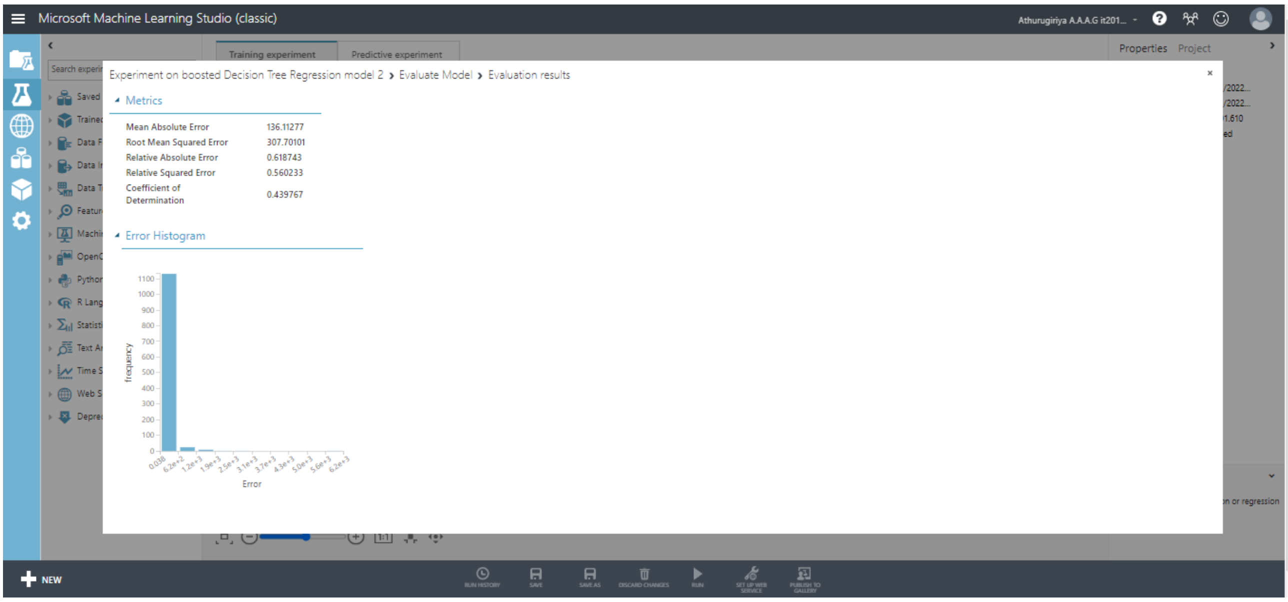
  We used regression models and predicted the Root Mean Squared Errors for them by using Azure machine Learning Studio.

Boosted Decision Tree Regression



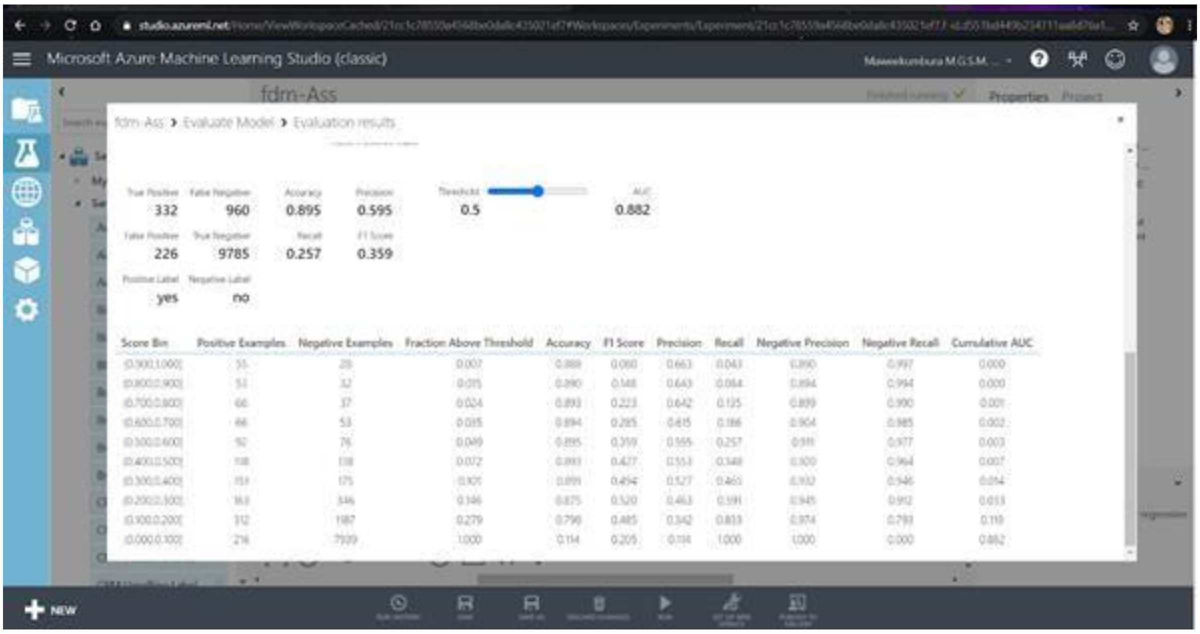
No of Features = 14
RMS Error = 279.904811

Boosted Decision Tree Regression



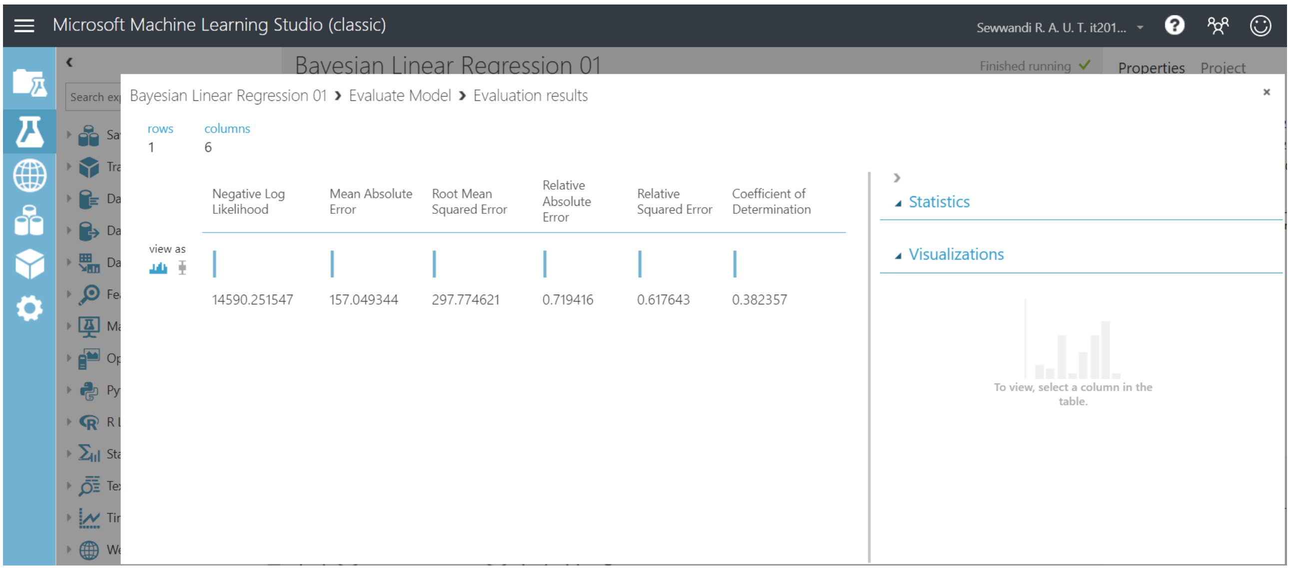
No of Features = 10
RMS Error = 307.70101

Boosted Decision Tree Regression



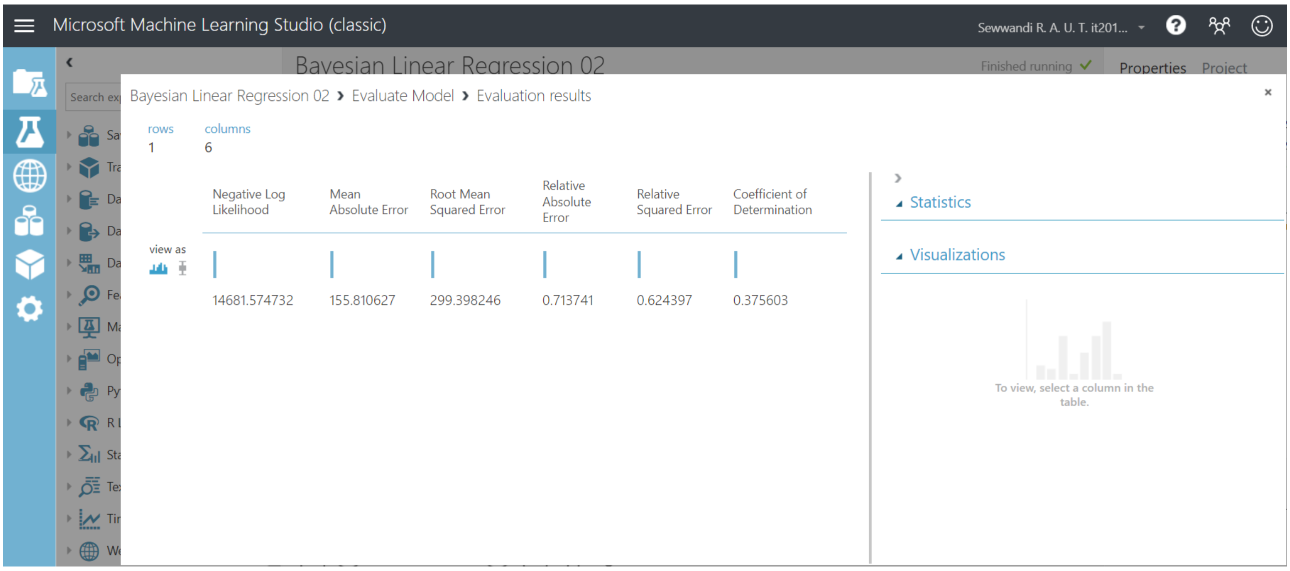
No of Features = 13
RMS Error = 297.013309

Bayesian Linear Regression



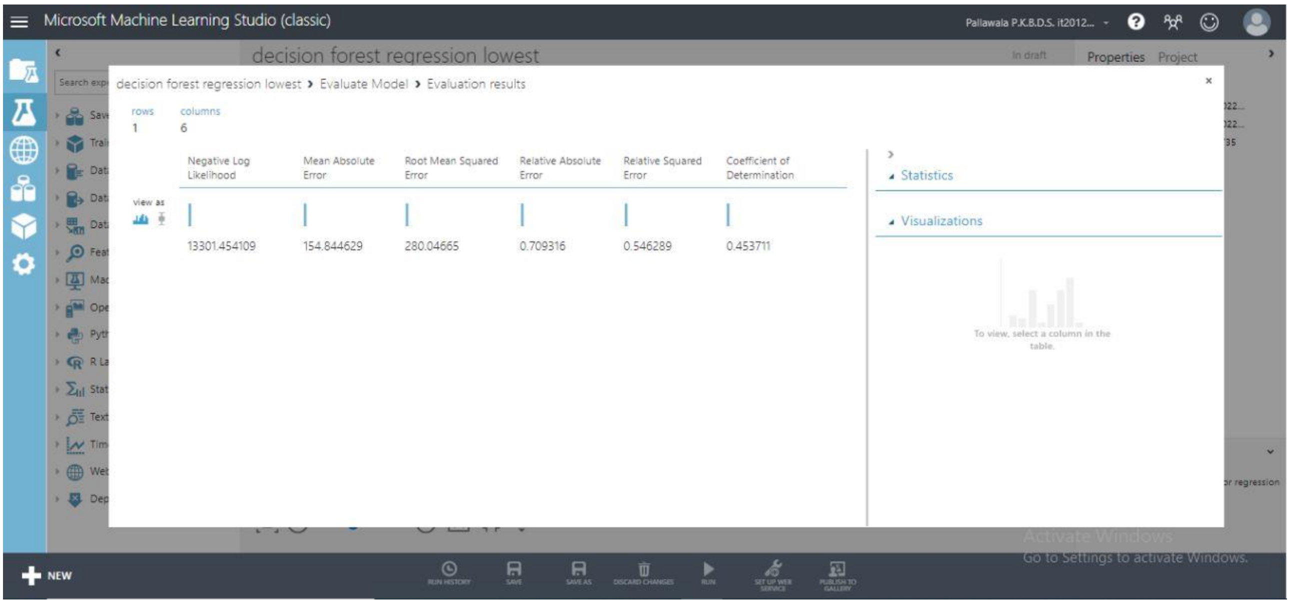
No of Features = 15
RMS Error = 297.774621

Bayesian Linear Regression



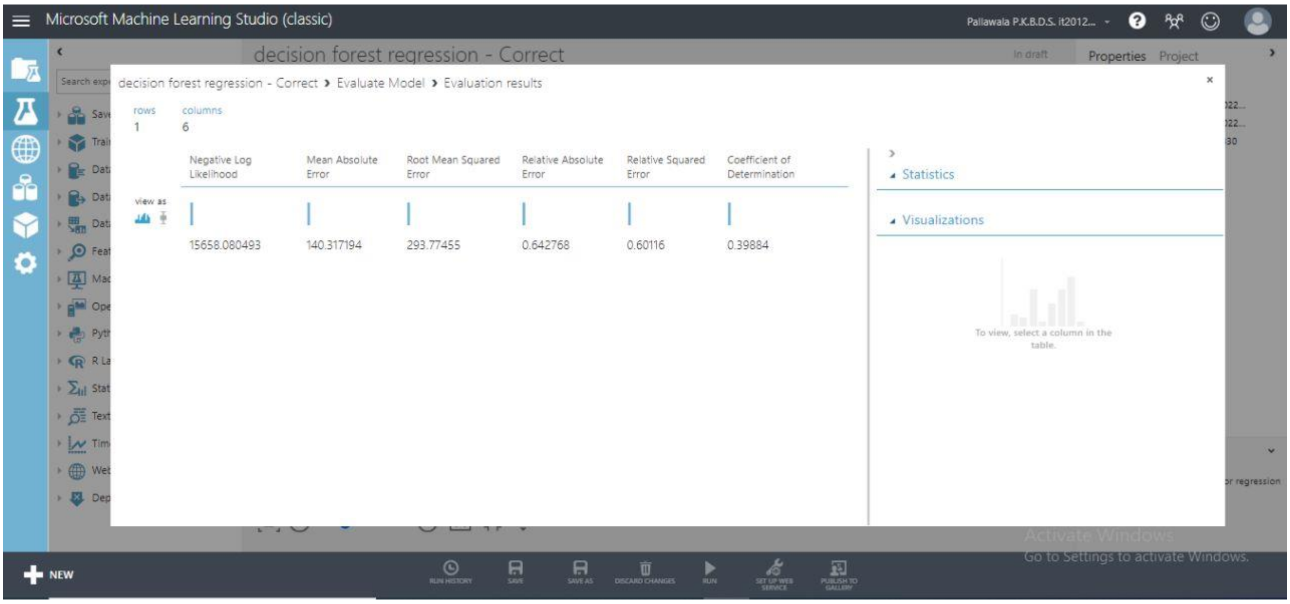
No of Features=13
RMS Error=299.398264

Decision Forest Regression



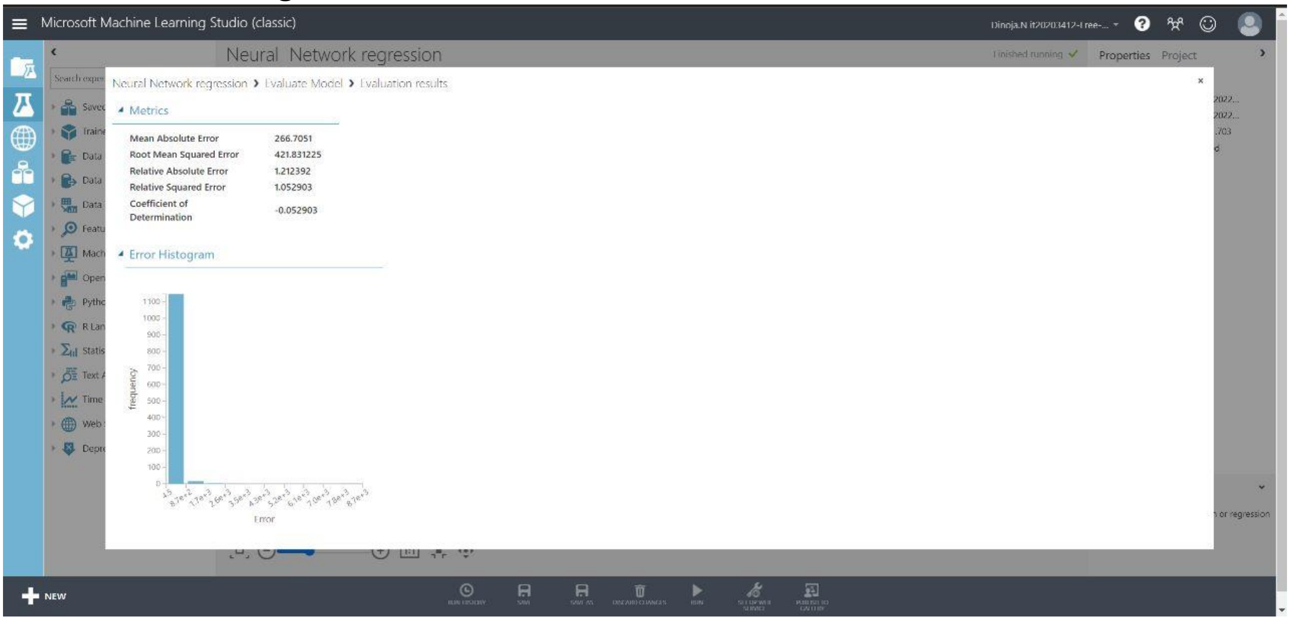
No of Features = 6
RMS Error = 280.04665

Decision Forest Regression



No of Features = 15
RMS Error = 293.77455

Neural Network Regression



No of Features = 13

RMS Error = 266.7051

After doing the analysis, it was concluded that the lowest Root Mean Squared Error was obtained when 17 variables were used. They are:

Accommodates, availability_30, bathrooms, bed_type, bedrooms, beds, cancellation_policy, guests_included, instant_bookable, maximum_nights, property_type, room_type, price, amenities__Elevator_in_Building, amenities__Kitchen, amenities__Internet, amenities__Air_Conditioning.

Splitting the dataset and cleaning

```
#import the dataset
data = pd.read_csv("/content/preprocessed_dataset_2.csv.xls")
data = pd.DataFrame(data)
data.columns

#dropping the unnecessary columns
df3 = data.drop(columns = ['calculated_host_listings_count','has_availability','host_is_superhost','host_listings_count','latitude(North)','longitude(East)','number_of_reviews','rev
        'review_scores_rating', 'review_scores_value','Unnamed: 0', 'amenities__Dryer', 'amenities__Other_pet(s)',
        'amenities__Carbon_Monoxide_Detector', 'amenities__', 'amenities__Iron', 'amenities__Cable_TV',
        'amenities__Heating', 'amenities__Shampoo',
        'amenities__Wireless_Internet', 'amenities__Dog(s)',
        'amenities__Doorman', 'amenities__24-Hour_Check-in', 'amenities__TV',
        'amenities__First_Aid_Kit', 'amenities__Pets_live_on_this_property',
        'amenities__Essentials', 'amenities__Laptop_Friendly_Workspace',
        'amenities__Hair_Dryer', 'amenities__Gym', 'amenities__Safety_Card',
        'amenities__Pool','amenities__Pets_Allowed', 'amenities__Buzzer/Wireless_Intercom',
        'amenities__Breakfast', 'amenities__Washer_/_Dryer',
        'amenities__Suitable_for_Events', 'amenities__Free_Parking_on_Premises',
        'amenities__Lock_on_Bedroom_Door', 'amenities__Washer',
        'amenities__Hot_Tub', 'amenities__Cat(s)', 'amenities__Indoor_Fireplace',
        'amenities__Smoke_Detector', 'amenities__Wheelchair_Accessible',
        'amenities__Family/Kid_Friendly', 'amenities__Fire_Extinguisher',
        'amenities__Hangers', 'amenities__Smoking_Allowed'])

#one-hot encoding for categorical variable
onehot_columns = ['bed_type', 'cancellation_policy', 'instant_bookable', 'property_type','room_type']
onehot_df = df3[onehot_columns]
onehot_df = pd.get_dummies(onehot_df, columns = onehot_columns)
score_onehot_drop = df3.drop(onehot_columns, axis = 1)
score_onehot = pd.concat([score_onehot_drop, onehot_df], axis = 1)
```



```
#Splitting data and target
X = score_onehot.drop(['price'],axis=1)
Y = score_onehot['price']

#splitting data into training and testing data
X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size=0.3,random_state=42)
```

- ## Models Accuracy

Random Forest Regressor

```
#loading the model
rfr= RandomForestRegressor(max_depth=10)

#training the model with X_train
rfr.fit(X_train,Y_train)

#prediction on training data
#accuracy for prediction on training data
training_data_prediction = rfr.predict(X_train)
print(training_data_prediction)

#R squared error
score_1= metrics.r2_score(Y_train,training_data_prediction)

#Mean Absolute Error
score_2 = mean_absolute_error(Y_train,training_data_prediction)

#Calculate the root mean squared error
rmse1 = np.sqrt(mean_squared_error(Y_train,training_data_prediction))

print("Root mean squared error(training data):",rmse1)
print("R squared error:",score_1)
print("Mean absolute error:",score_2)
```

```
print("Root mean squared error(training data):",rmse1)
print("R squared error:",score_1)
print("Mean absolute error:",score_2)

#prediction on testing data
#accuracy for prediction on testing data
testing_data_prediction = rfr.predict(X_test)
print(testing_data_prediction)

#R squared error
score_1= metrics.r2_score(Y_test,testing_data_prediction)

#Mean Absolute Error
score_2 = mean_absolute_error(Y_test,testing_data_prediction)

#Calculate the root mean squared error
rmse2 = np.sqrt(mean_squared_error(Y_test,testing_data_prediction))

print("Root mean squared error(testing data):",rmse2)

print("R squared error:",score_1)
print("Mean absolute error:",score_2)

#create a pickle file using serialization
rfr_pickle=open("rfregressor.pkl","wb")
pickle.dump(rfr,rfr_pickle)
rfr_pickle.close()
filename = 'rfregressor.sav'
pickle.dump(rfr, open(filename,'wb'))
```

XG Boost Regressor

```
[ ] #loading the model
    model= XGBRegressor()

    #training the model with X_train
    model.fit(X_train,Y_train)

    #prediction on training data

    #accuracy for prediction on training data
    training_data_prediction = model.predict(X_train)
    print(training_data_prediction)

    #R squared error
    score_1= metrics.r2_score(Y_train,training_data_prediction)

    #Mean Absolute Error
    score_2 = mean_absolute_error(Y_train,training_data_prediction)

    #Calculate the root mean squared error
    rmse1 = np.sqrt(mean_squared_error(Y_train,training_data_prediction))

    print("Root mean squared error(training data):",rmse1)
    print("R squared error(training data):",score_1)
    print("Mean absolute error(training data):",score_2)
```

```
#prediction on testing data

#accuracy for prediction on testing data
testing_data_prediction = model.predict(X_test)
print(testing_data_prediction)

#R squared error
score_1= metrics.r2_score(Y_test,testing_data_prediction)

#Mean Absolute Error
score_2 = mean_absolute_error(Y_test,testing_data_prediction)

#Calculate the root mean squared error
rmse2 = np.sqrt(mean_squared_error(Y_test,testing_data_prediction))

print("Root mean squared error(testing data):",rmse2)
print("R squared error(testing data):",score_1)
print("Mean absolute error(testing data):",score_2)
```

```
[04:24:30] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:
[618.13995  111.75796   49.288445 ... 765.3672   108.03666  260.49222 ]
Root mean squared error(training data): 216.16914644374174
R squared error(training data): 0.712147592803385
Mean absolute error(training data): 116.6857307464242
[ 176.53801   585.0223   1944.6909   ...  127.77199   189.6472
   99.976295]
Root mean squared error(testing data): 287.2645837431405
```

KNeighbors Regressor

```
#loading the model
knn= KNeighborsRegressor(n_neighbors=10)

#training the model with X_train
knn.fit(X_train,Y_train)

#prediction on training data

#accuracy for prediction on training data
training_data_prediction = knn.predict(X_train)
print(training_data_prediction)

#R squared error
score_1= metrics.r2_score(Y_train,training_data_prediction)

#Mean Absolute Error
score_2 = mean_absolute_error(Y_train,training_data_prediction)

#Calculate the root mean squared error
rmse1 = np.sqrt(mean_squared_error(Y_train,training_data_prediction))

print("Root mean squared error(training data):",rmse1)
print("R squared error:",score_1)
print("Mean absolute error:",score_2)
```

```
#prediction on testing data

#accuracy for prediction on testing data
testing_data_prediction = knn.predict(X_test)
print(testing_data_prediction)

#R squared error
score_1= metrics.r2_score(Y_test,testing_data_prediction)

#Mean Absolute Error
score_2 = mean_absolute_error(Y_test,testing_data_prediction)

#Calculate the root mean squared error
rmse2 = np.sqrt(mean_squared_error(Y_test,testing_data_prediction))

print("Root mean squared error(testing data):",rmse2)

print("R squared error:",score_1)
print("Mean absolute error:",score_2)
```

```
[473.   102.2  89.9 ... 709.    86.7 275. ]
Root mean squared error(training data): 298.2227081094483
R squared error: 0.4521472203456679
Mean absolute error: 129.3677932892481
[ 150.8  867.3 1158.8 ...  151.8  225.8   90.6]
```

Gradient Boost Regressor

```
#loading the model
gbr= GradientBoostingRegressor(n_estimators=250)

#training the model with X_train
gbr.fit(X_train,Y_train)

#prediction on training data

#accuracy for prediction on training data
training_data_prediction = gbr.predict(X_train)
print(training_data_prediction)

#R squared error
score_1= metrics.r2_score(Y_train,training_data_prediction)

#Mean Absolute Error
score_2 = mean_absolute_error(Y_train,training_data_prediction)

#Calculate the root mean squared error
rmse1 = np.sqrt(mean_squared_error(Y_train,training_data_prediction))

print("Root mean squared error(training data):",rmse1)
print("R squared error:",score_1)
print("Mean absolute error:",score_2)
```

```
#prediction on testing data

#accuracy for prediction on testing data
testing_data_prediction = gbr.predict(X_test)
print(testing_data_prediction)

#R squared error
score_1= metrics.r2_score(Y_test,testing_data_prediction)

#Mean Absolute Error
score_2 = mean_absolute_error(Y_test,testing_data_prediction)

#Calculate the root mean squared error
rmse2 = np.sqrt(mean_squared_error(Y_test,testing_data_prediction))

print("Root mean squared error(testing data):",rmse2)

print("R squared error:",score_1)
print("Mean absolute error:",score_2)
```

```
[569.34760464 113.02010889  53.20523909 ... 811.89873976 115.44413064
 259.96835721]
Root mean squared error(training data): 193.91851939946173
R squared error: 0.7683560172289081
Mean absolute error: 107.33644070665486
[ 177.59184959  588.60104495 1835.33560411 ...  129.80573002  211.18920122
  100.71881151]
Root mean squared error(testing data): 291.74509892247704
R squared error: 0.4781878817009735
```

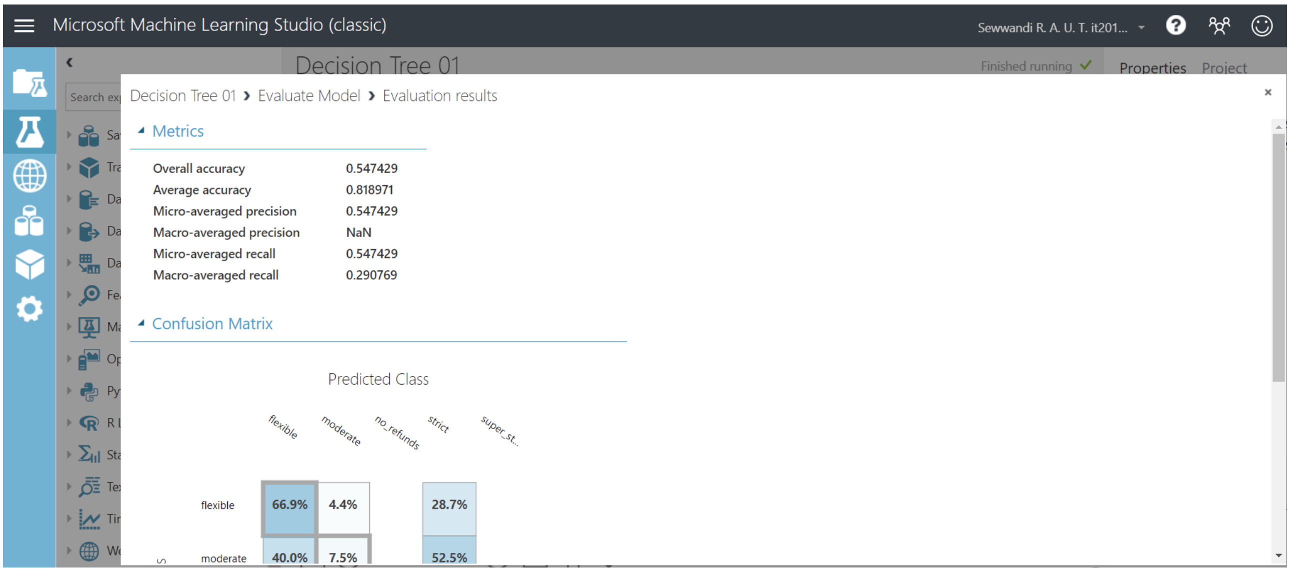
Prediction 2: Predict the cancellation policy of a property

Description:

This is used to predict the cancellation policy type of a property. it helps buyers to decide whether cancelling a booking is appropriate or not. We use the default column, which tells us the cancellation policy types

Analyzing Classification Models

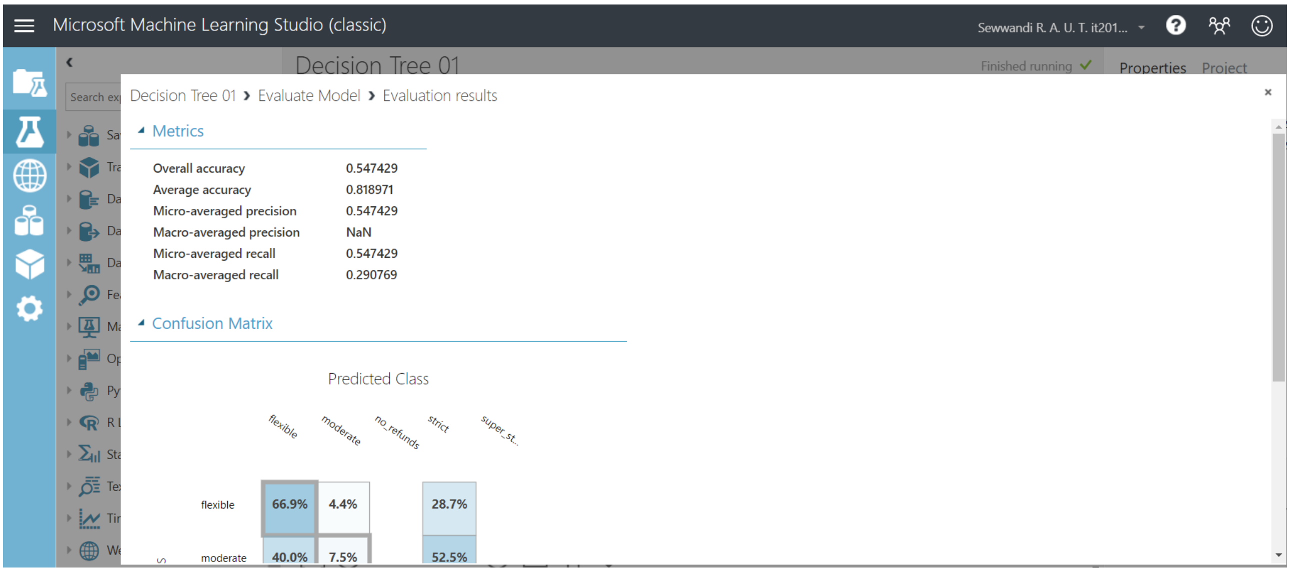
Multiclass Decision Jungle



No of Features = 15
Accuracy = 0.54742

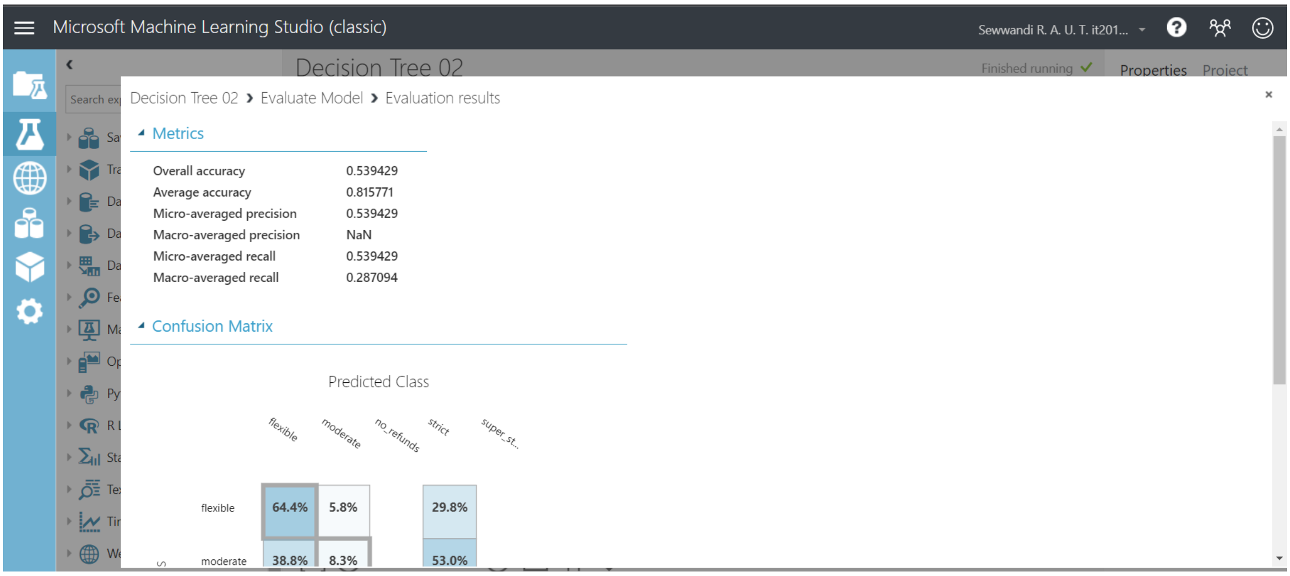
Analyzing Classification Models

Multiclass Decision Jungle



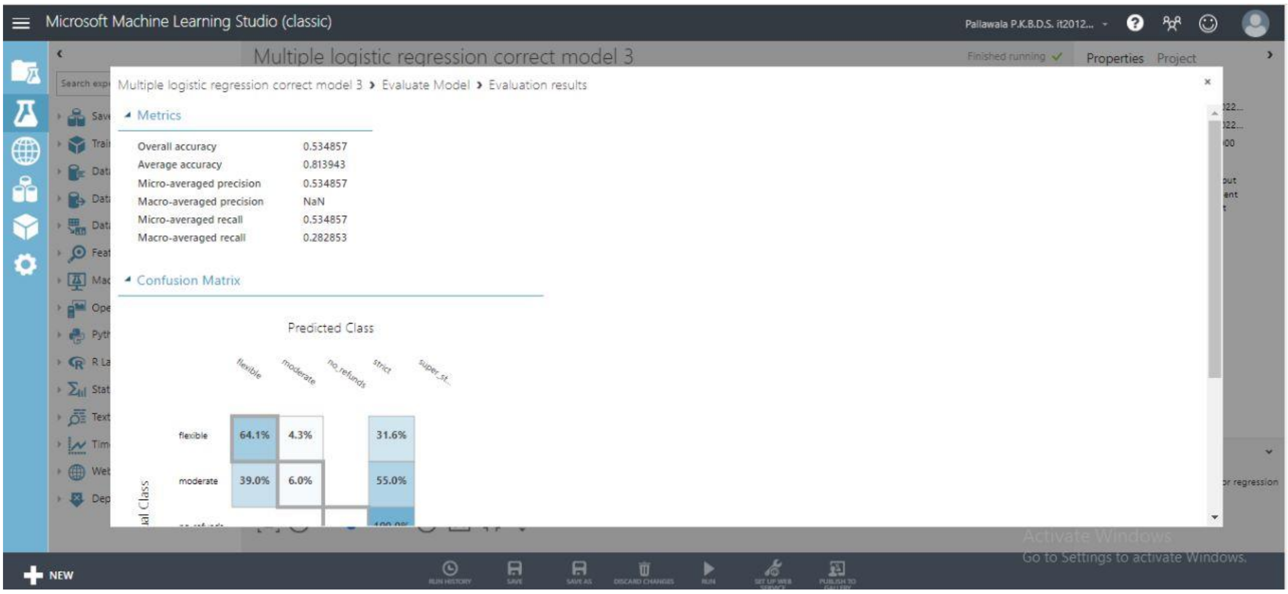
No of Features = 15
Accuracy = 0.54742

Multiclass Decision Jungle



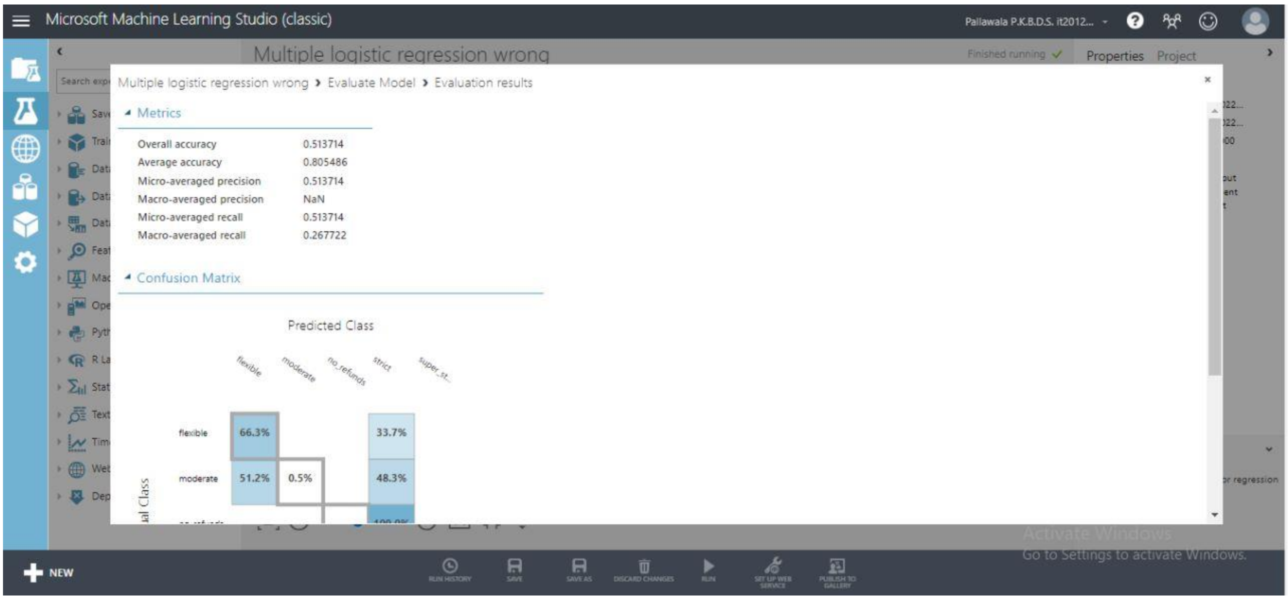
No of Features = 17
Accuracy = 0.53942

Multiclass Logistic Regression



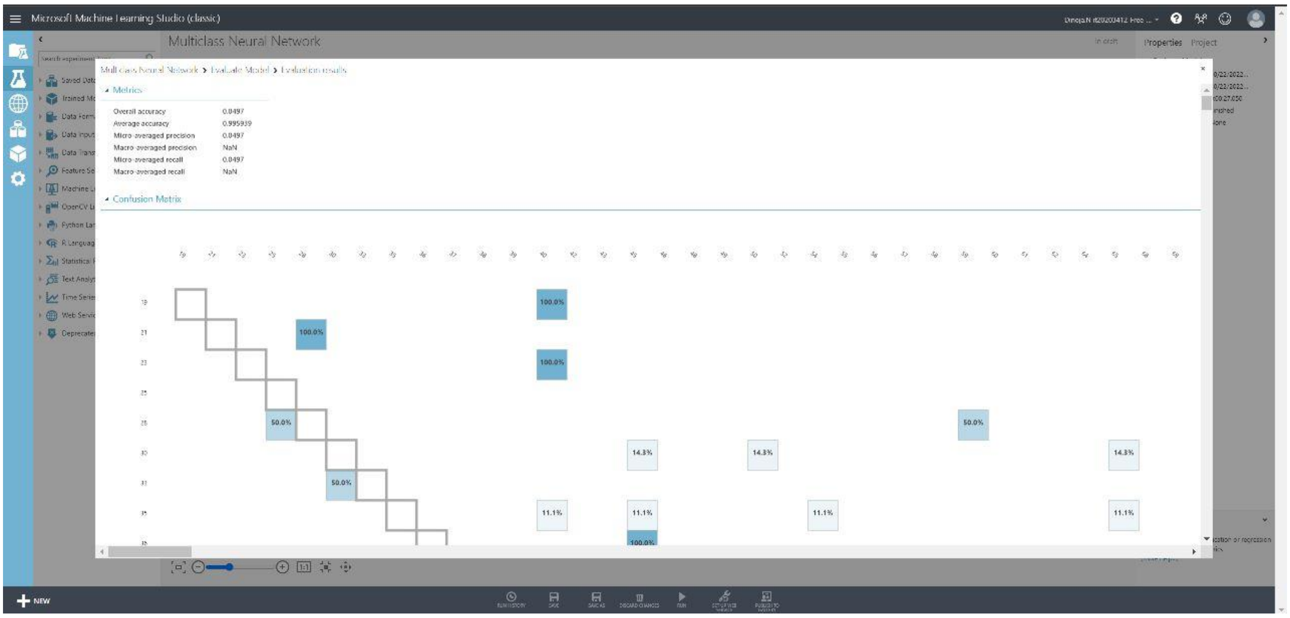
No of Features = 11
Accuracy = 0.53485

Multiclass Logistic Regression



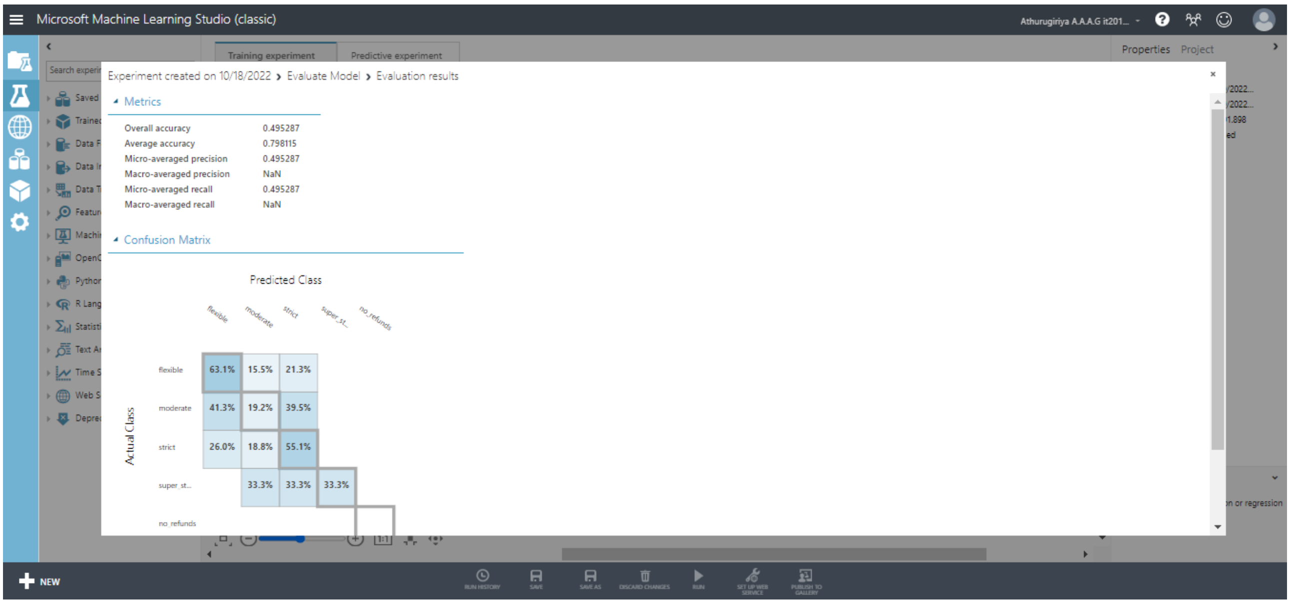
No of Features = 11
Accuracy = 0.51371

Multiclass Neural Network



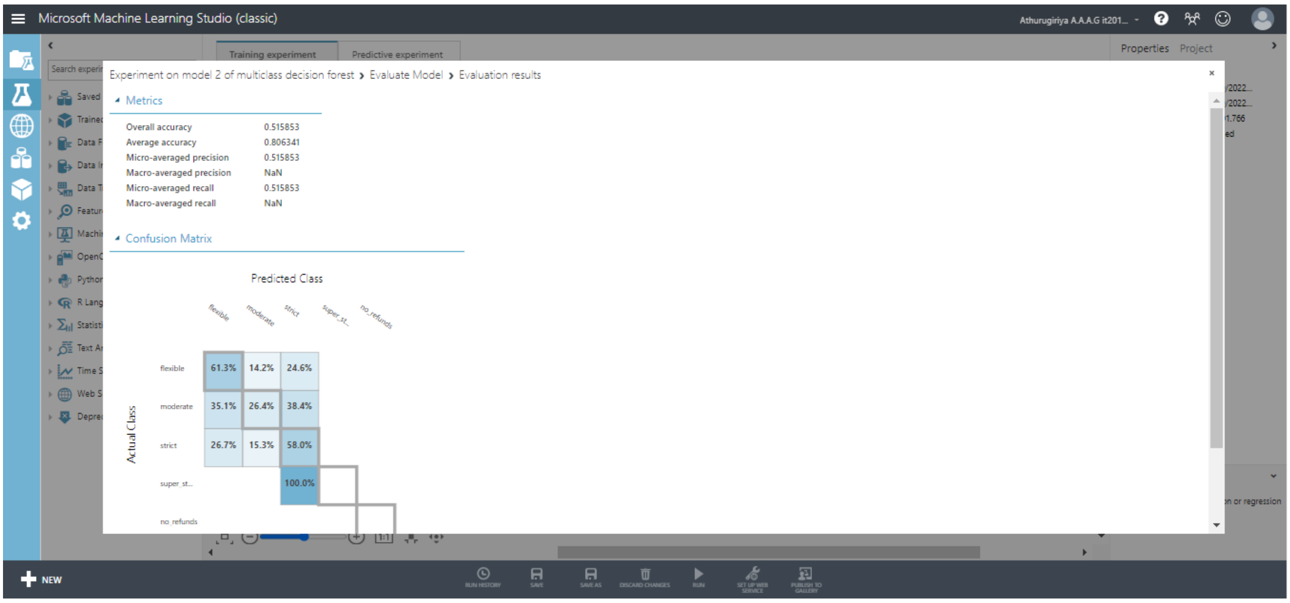
No of Features = 13
Accuracy = 0.0497

Multiclass Decision Forest



No of Features = 14
Accuracy = 0.486718

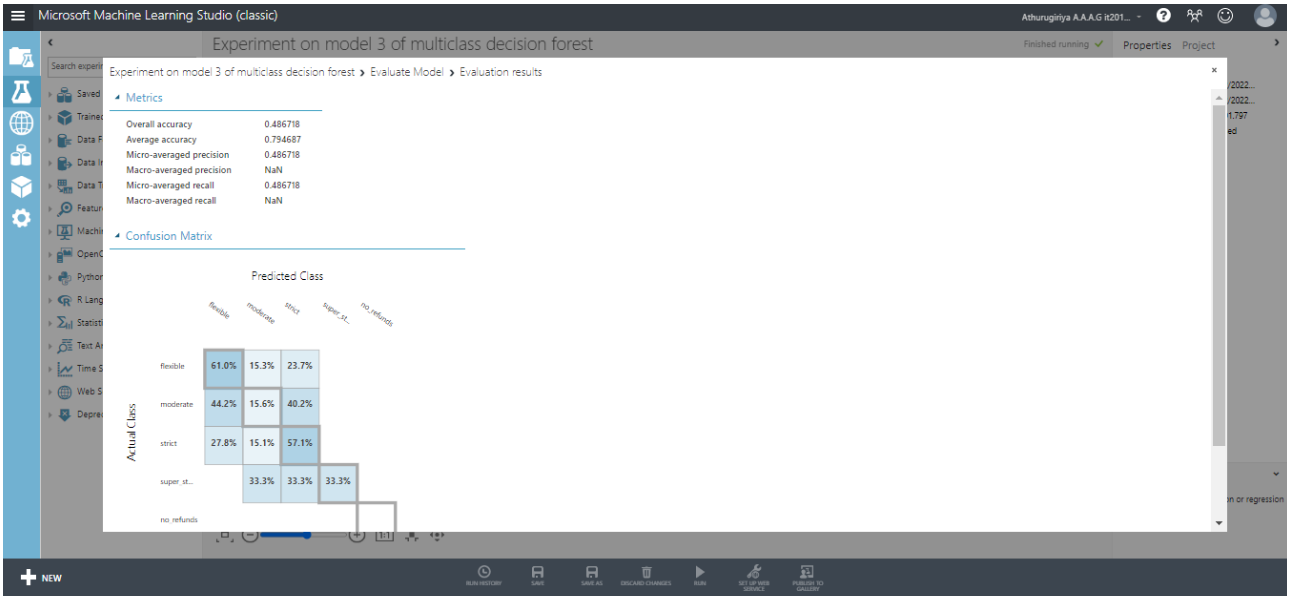
Multiclass Decision Forest



No of Features = 14

Accuracy = 0.515853

Multiclass Decision Forest



No of Features = 9

Accuracy = 0.486718

After doing the analysis, it was concluded that the highest accuracy was obtained when 15 variables were used. They are:

> accommodates, availability_30, bathrooms, bed_type, bedrooms, beds, cancellation_policy, guests_included, instant_bookable, latitude (North), longitude (East), maximum_nights, property_type, room_type, price

Splitting the dataset and cleaning

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, MinMaxScaler
import xgboost as xgb

data = pd.read_csv("/content/preprocessed_dataset_2.csv")
data = pd.DataFrame(data)
data
```

	Unnamed: 0	accommodates	availability_30	bathrooms	bed_type	bedrooms	beds	calculated_host_listings_count	cancellation_policy	guests_ir
0	0	6	0	2	Real Bed	1	3	1	moderate	
1	1	2	29	0	Futon	1	1	1	moderate	
2	2	2	30	1	Real Bed	1	1	2	flexible	
3	3	2	30	1	Real Bed	1	1	2	flexible	
4	4	6	27	2	Real Bed	3	3	2	strict	
...	...	...	...	...	...	...	...	...	...	
5829	5829	4	17	1	Real Bed	1	2	3	moderate	
5830	5830	4	15	1	Real Bed	1	2	11	strict	
5831	5831	2	11	1	Real Bed	1	2	2	moderate	
5832	5832	1	30	1	Real Bed	1	1	1	flexible	
5833	5833	2	2	1	Real Bed	1	1	1	strict	

```
df3 = data.drop(columns = ['calculated_host_listings_count', 'host_listings_count', 'has_availability', 'host_is_superhost', 'number_of_reviews'
                           'review_scores_checkin', 'review_scores_communication', 'review_scores_location', 'review_scores_rating',
                           'review_scores_value', 'Unnamed: 0', 'amenities__Dryer', 'amenities__Other_pet(s)', 'amenities__Carbon_Monoxide_Detec
                           'amenities__', 'amenities__Iron', 'amenities__Elevator_in_Building', 'amenities__Cable_TV', 'amenities__Heating',
                           'amenities__Shampoo', 'amenities__Wireless_Internet', 'amenities__Dog(s)', 'amenities__Doorman',
                           'amenities__24-Hour_Check-in', 'amenities__TV', 'amenities__First_Aid_Kit', 'amenities__Pets_live_on_this_property',
                           'amenities__Essentials', 'amenities__Laptop_Friendly_Workspace','amenities__Hair_Dryer', 'amenities__Gym',
                           'amenities__Safety_Card', 'amenities__Pool', 'amenities__Internet', 'amenities__Kitchen', 'amenities__Pets_Allowed',
                           'amenities__Buzzer/Wireless_Intercom', 'amenities__Breakfast', 'amenities__Washer_/_Dryer', 'amenities__Suitable_for_
                           'amenities__Free_Parking_on_Premises', 'amenities__Lock_on_Bedroom_Door', 'amenities__Washer', 'amenities__Hot_Tub',
                           'amenities__Cat(s)', 'amenities__Air_Conditioning', 'amenities__Indoor_Fireplace', 'amenities__Smoke_Detector',
                           'amenities__Wheelchair_Accessible', 'amenities__Family/Kid_Friendly', 'amenities__Fire_Extinguisher', 'amenities__Har

df3.columns
```

```
Index(['accommodates', 'availability_30', 'bathrooms', 'bed_type', 'bedrooms',
       'beds', 'cancellation_policy', 'guests_included', 'instant_bookable',
       'latitude(North)', 'longitude(East)', 'maximum_nights', 'property_type',
       'room_type', 'price'],
      dtype='object')
```

```
df3['bed_type'] = df3['bed_type'].map({'Real Bed': 1, 'Futon': 2, 'Airbed': 3, 'Pull-out Sofa': 4, 'Couch': 5})
df3['cancellation_policy'] = df3['cancellation_policy'].map({'moderate':1, 'flexible':2, 'strict': 3})
df3['instant_bookable'] = df3['instant_bookable'].map({'t':1, 'f':0})
df3['property_type'] = df3['property_type'].map({'House':1, 'Camper/RV':2, 'Bed & Breakfast':3, 'Apartment':4, 'Townhouse':5, 'Condominium':6, '
df3['room_type'] = df3['room_type'].map({'Private room':1, 'Entire home/apt':2, 'Shared room':3})

df3
```

	accommodates	availability_30	bathrooms	bed_type	bedrooms	beds	cancellation_policy	guests_included	instant_bookable	latitude(North)
0	6	0	2	1	1	3	1	1	0	22.542900
1	2	29	0	2	1	1	1	1	0	22.539490
2	2	30	1	1	1	1	2	1	0	22.508573
3	2	30	1	1	1	1	2	1	0	22.508697
4	6	27	2	1	3	3	3	1	0	22.509502
...	...	...	...	...	...	...	...	...	...	...
5829	4	17	1	1	1	2	1	1	0	22.619618
5830	4	15	1	1	1	2	3	3	0	22.606116
5831	2	11	1	1	1	2	1	1	0	22.606302
5832	1	30	1	1	1	1	2	1	0	22.819196
5833	2	2	1	1	1	1	3	1	0	22.769541

5834 rows × 15 columns

```
df3['bed_type'].round(0)
df3['cancellation_policy'].round(0)
df3['instant_bookable'].round(0)
df3['property_type'].round(0)
df3['room_type'].round(0)
```

```
0       1
1       1
2       1
3       1
4       2
       ..
5829    2
5830    2
5831    1
5832    3
5833    2
Name: room_type, Length: 5834, dtype: int64
```


```
df3['bed_type'].round(0)
df3['cancellation_policy'].round(0)
df3['instant_bookable'].round(0)
df3['property_type'].round(0)
df3['room_type'].round(0)
```

```
0       1
1       1
2       1
3       1
4       2
       ..
5829    2
5830    2
5831    1
5832    3
5833    2
Name: room_type, Length: 5834, dtype: int64
```


```
df3['bed_type'] = df3['bed_type'].astype('Int64')
df3['cancellation_policy'] = df3['cancellation_policy'].astype('Int64')
df3['instant_bookable'] = df3['instant_bookable'].astype('Int64')
df3['property_type'] = df3['property_type'].astype('Int64')
df3['room_type'] = df3['room_type'].astype('Int64')

df3
```

	accommodates	availability_30	bathrooms	bed_type	bedrooms	beds	cancellation_policy	guests_included	instant_bookable	latitude(North)
0	6	0	2	1	1	3	1	1	0	22.542900
1	2	29	0	2	1	1	1	1	0	22.539490
2	2	30	1	1	1	1	2	1	0	22.508573
3	2	30	1	1	1	1	2	1	0	22.508697
4	6	27	2	1	3	3	3	1	0	22.509502
...	...	...	...	...	...	...	...	...	...	...
5829	4	17	1	1	1	2	1	1	0	22.619618
5830	4	15	1	1	1	2	3	3	0	22.606116
5831	2	11	1	1	1	2	1	1	0	22.606302
5832	1	30	1	1	1	1	2	1	0	22.819196
5833	2	2	1	1	1	1	3	1	0	22.769541

5834 rows × 15 columns

```
d = df3.drop(columns = ['cancellation_policy'])
X = d.iloc[:,:]
y = df3.iloc[:,6]
y=y.astype('int')
```

```
[15] # Splitting data into training and testing data
     X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.3,random_state=42)
```

Models Accuracy

Decision Tree Classifier

```
[16] # Implement Decision Tree Classifier

     from sklearn.tree import DecisionTreeClassifier

     dtc = DecisionTreeClassifier()
     dtc.fit(X_train,y_train)

     DecisionTreeClassifier()
```

```
[17] DTy_pred = dtc.predict(X_test)
```

```
[18] from sklearn.metrics import accuracy_score

     DTscore = accuracy_score(y_test, DTy_pred)
     DTscore

     0.4842946887492861
```

```
import pickle

DT_pickle_out = open("FD-DecisionTreeClassifier.pk1", "wb")
pickle.dump(dtc, DT_pickle_out)
DT_pickle_out.close()
```

Logistic Regression Classifier

```
from sklearn.linear_model import LogisticRegression

logmodel = LogisticRegression(solver='lbfgs', max_iter=10000)
logmodel.fit(X_train,y_train)

y_predict = logmodel.predict(X_test)

from sklearn import metrics
lg_acc = metrics.accuracy_score(y_test, y_predict)
print(lg_acc)
```

```
0.5248429468874929
```

Ada Boost Classifier

```
from sklearn.ensemble import AdaBoostClassifier
from sklearn.tree import DecisionTreeClassifier
import pickle

adaBoost = AdaBoostClassifier(
    base_estimator = DecisionTreeClassifier(max_depth = 2),
    n_estimators=100,
    learning_rate=0.1,
    random_state = 7
)

adaBoost.fit(X_train, y_train)

y_predict = adaBoost.predict(X_test)

from sklearn import metrics
ad_acc = metrics.accuracy_score(y_test, y_predict)
print(ad_acc)

#ADA_pickle_out = open("FD-AdaBoostClassifier.pk1", "wb")
pickle.dump(adaBoost, open("FD-AdaBoostClassifier.pk1", "wb"))
filename = 'FD-AdaBoostClassifier.sav'
pickle.dump(adaBoost, open(filename, 'wb'))
#ADA_pickle_out.close()
```

```
0.5859508852084523
```

Random Forest Classifier

```
from sklearn.ensemble import RandomForestClassifier
RFclassifier=RandomForestClassifier()
RFclassifier.fit(X_train,y_train)
y_pred= RFclassifier.predict(X_test)
from sklearn.metrics import accuracy_score
acc= accuracy_score(y_test,y_pred)
acc
```

```
0.5659623072529982
```

Prediction 03 - Predict the most demanding property type

Using clustering model to predict the most demanding property type.

3 variables are suitable to predict most demanding property type. Those variables are:
Longitude, latitude, price

Creating models

In [1]:
```
#import libraries
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
from sklearn.cluster import KMeans
```

In [2]:
```
#load the dataset
df = pd.read_csv(r'F:\preprocessed_dataset_1.csv')
```

In [3]:
```
df.head()
```

Out[3]:

	Unnamed: 0	accommodates	availability_30	bathrooms	bed_type	bedrooms	beds	calculated_host_listings_count	cancellation_policy	guests_included	...	ar
0	0	6	0	2	Real Bed	1.0	3	1	moderate	1	...	
1	1	2	29	0	Futon	1.0	1	1	moderate	1	...	
2	2	2	30	1	Real Bed	1.0	1	2	flexible	1	...	
3	3	2	30	1	Real Bed	1.0	1	2	flexible	1	...	
4	4	6	27	2	Real Bed	3.0	3	2	strict	1	...	

5 rows × 68 columns

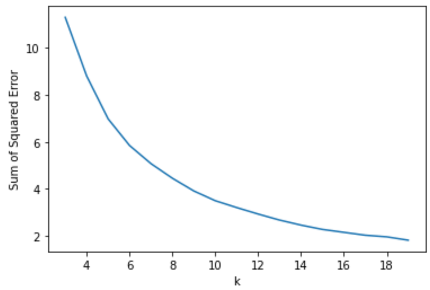
In [6]:
```
sse = []
k_range = range(3, 20)
for k in k_range:
    km = KMeans(n_clusters = k)
    km.fit(df[['latitude(North)', 'longitude(East)']])
    sse.append(km.inertia_)  #inertia will give the sse
```

In [7]:
```
plt.xlabel('k')
plt.ylabel('Sum of Squared Error')
plt.plot(k_range, sse)
```

Out[7]: [<matplotlib.lines.Line2D at 0x2e880afe220>]

In [7]:
```
plt.xlabel('k')
plt.ylabel('Sum of Squared Error')
plt.plot(k_range, sse)
```

Out[7]: [<matplotlib.lines.Line2D at 0x2e880afe220>]



In [9]:
```
#initialize the clusters
km = KMeans(10)
```

In [10]:
```
km
```

Out[10]:

```
KMeans
KMeans(n_clusters=10)
```

In [11]:
```
#fit and predict
y_pred = km.fit_predict(df[['latitude(North)', 'longitude(East)']])
y_pred
```

Out[11]: array([1, 1, 1, ..., 2, 6, 6])

In [16]:
```
df['Cluster_Lon_Lat'] = y_pred
df.head()
```

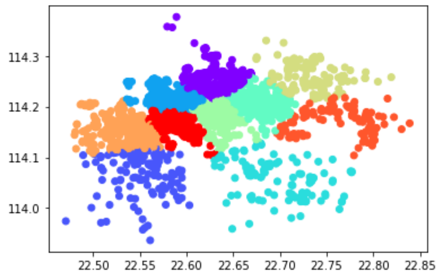
Out[16]:

	Unnamed: 0	accommodates	availability_30	bathrooms	bed_type	bedrooms	beds	calculated_host_listings_count	cancellation_policy	guests_included	...	ar
0	0	6	0	2	Real Bed	1.0	3	1	moderate	1	...	
1	1	2	29	0	Futon	1.0	1	1	moderate	1	...	
2	2	2	30	1	Real Bed	1.0	1	2	flexible	1	...	
3	3	2	30	1	Real Bed	1.0	1	2	flexible	1	...	
4	4	6	27	2	Real Bed	3.0	3	2	strict	1	...	

5 rows × 70 columns

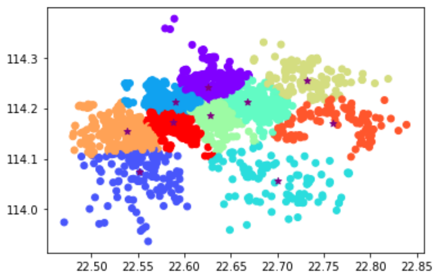
In [18]:
```
#plot the data
plt.scatter(df['latitude(North)'], df['longitude(East)'], c = df['Cluster_Lon_Lat'], cmap = 'rainbow')
```

Out[18]: <matplotlib.collections.PathCollection at 0x2e881bb4e80>

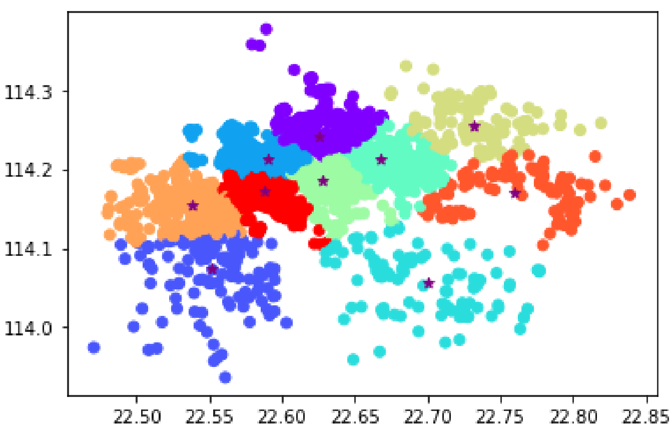


In [19]:
```
#plot the data
plt.scatter(df['latitude(North)'], df['longitude(East)'], c = df['Cluster_Lon_Lat'], cmap = 'rainbow')
plt.scatter(km.cluster_centers_[:,0], km.cluster_centers_[:, 1], color = 'purple', marker = '*')
```

Out[19]: <matplotlib.collections.PathCollection at 0x2e881e1c670>

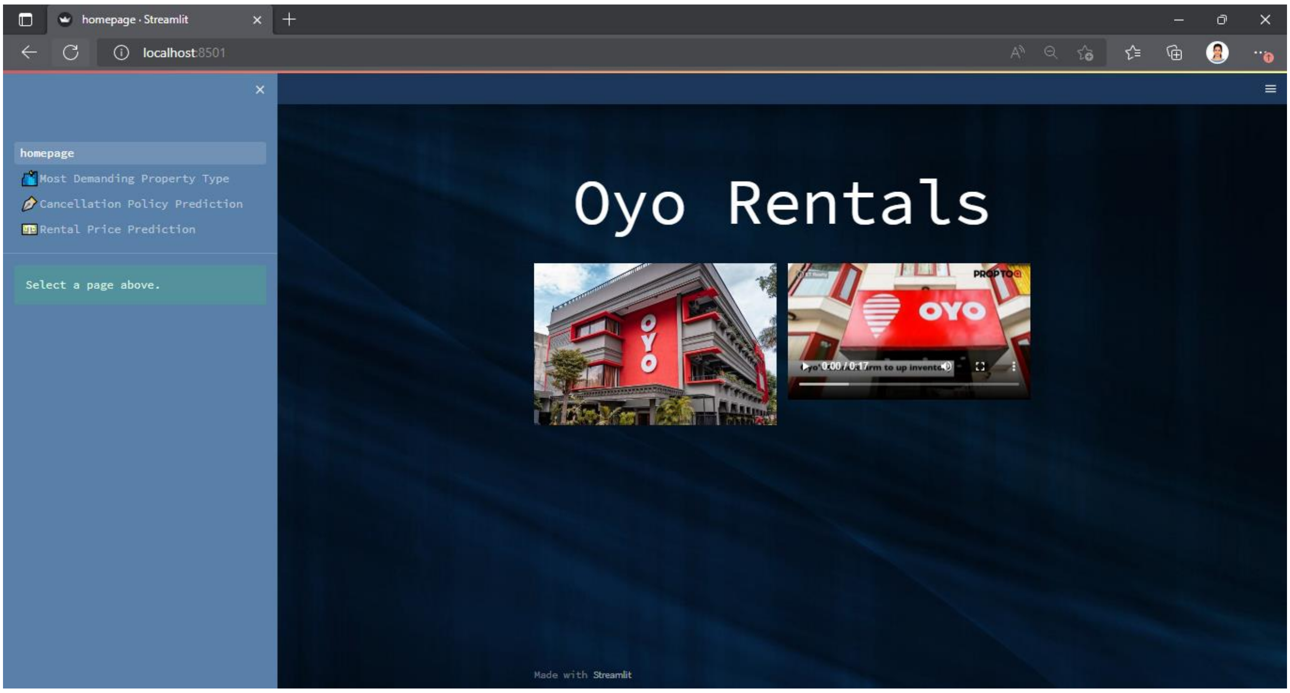


In [20]:
```
df.to_csv(r'F:\preprocessed_dataset_1.csv')
```

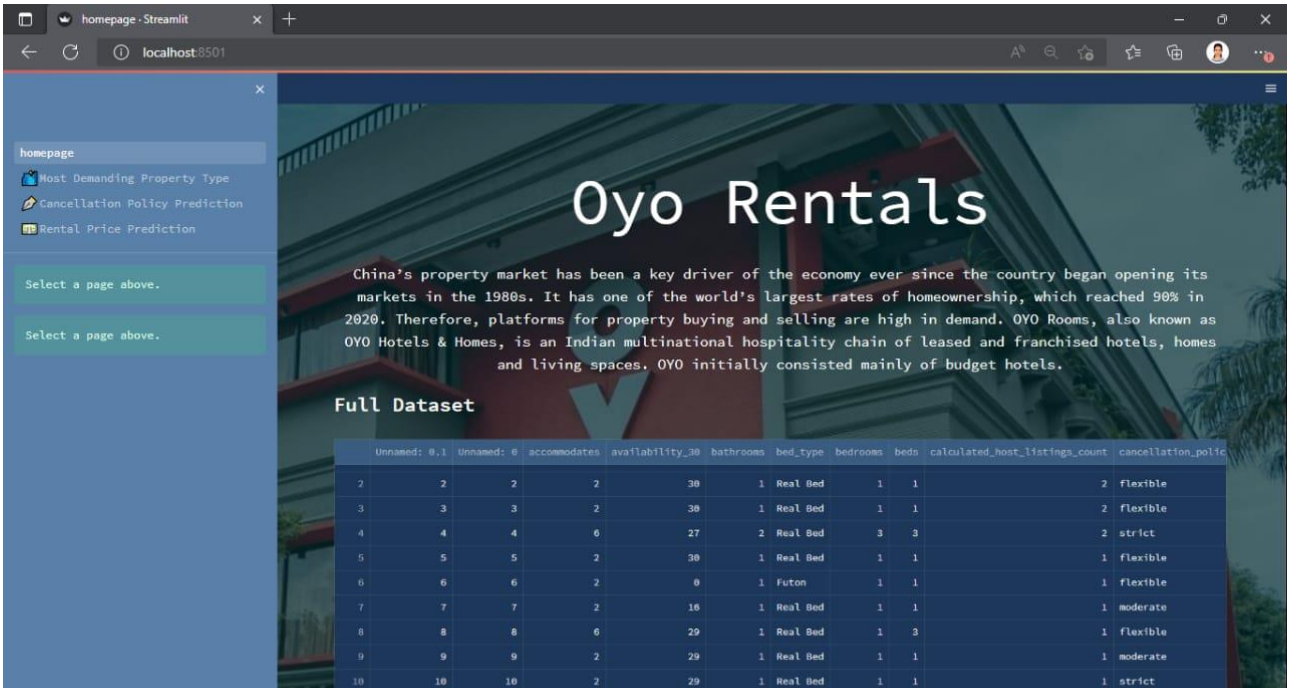


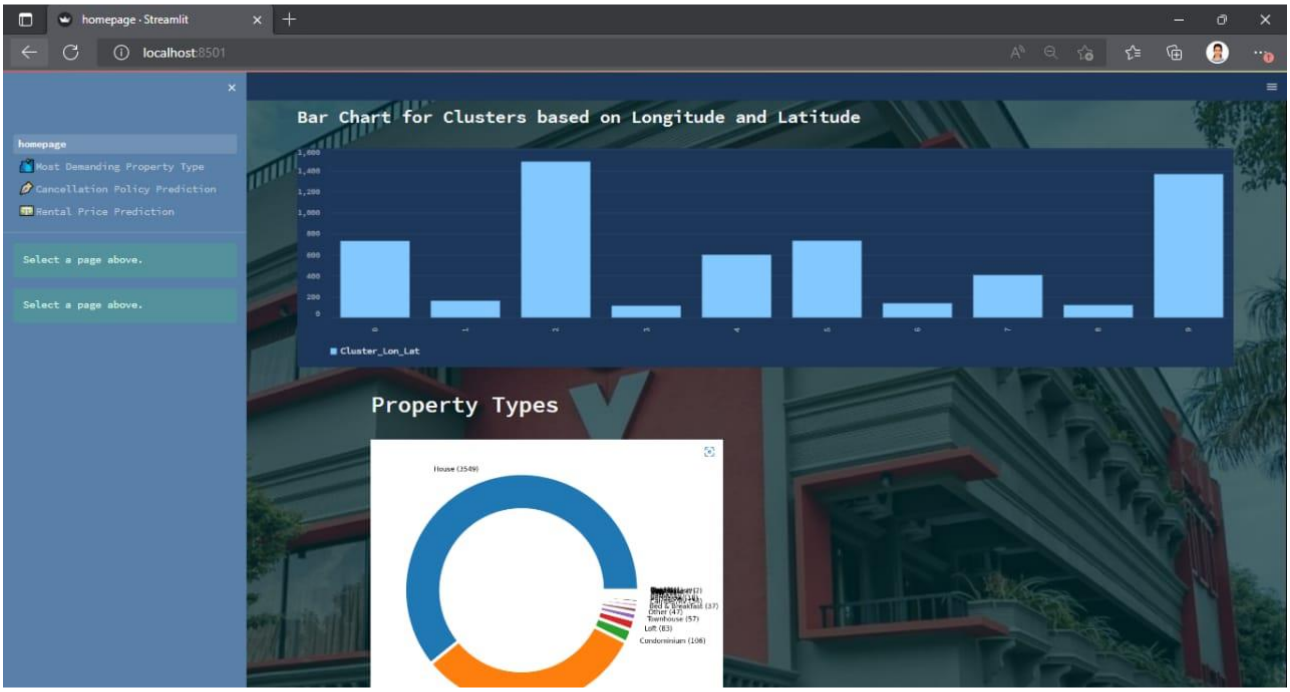
User Interfaces

1. Home page

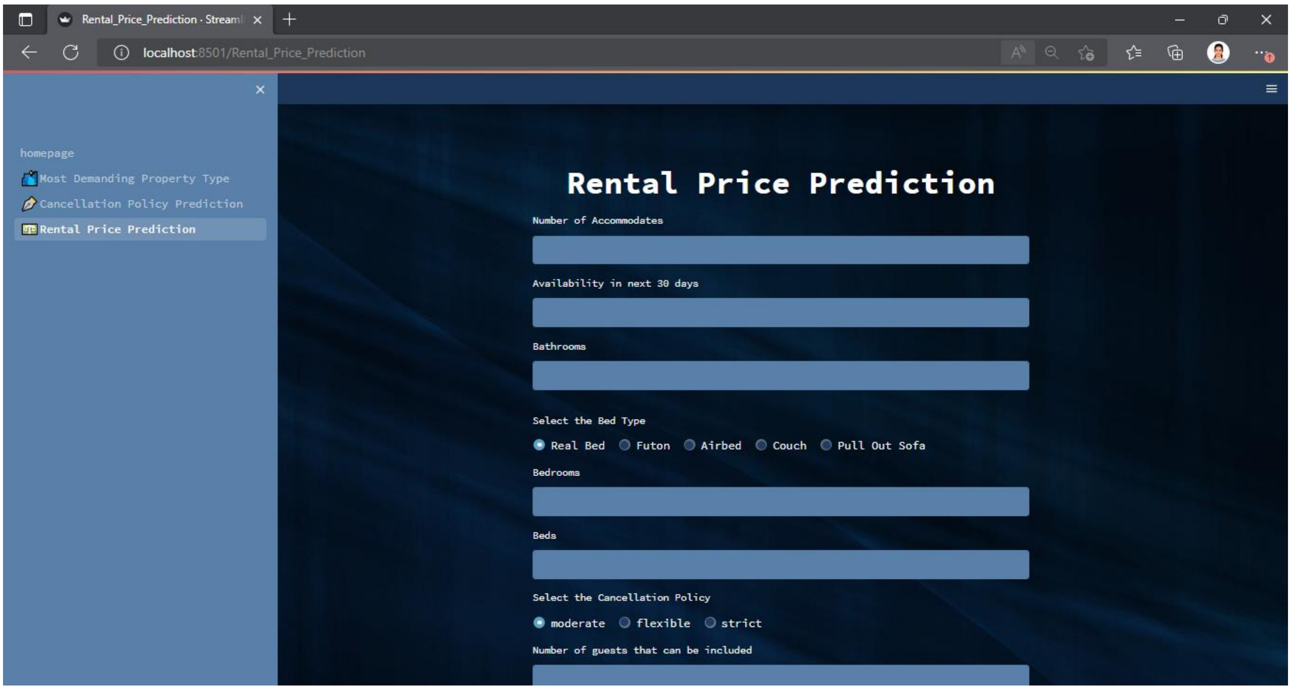


2. Visualize data page

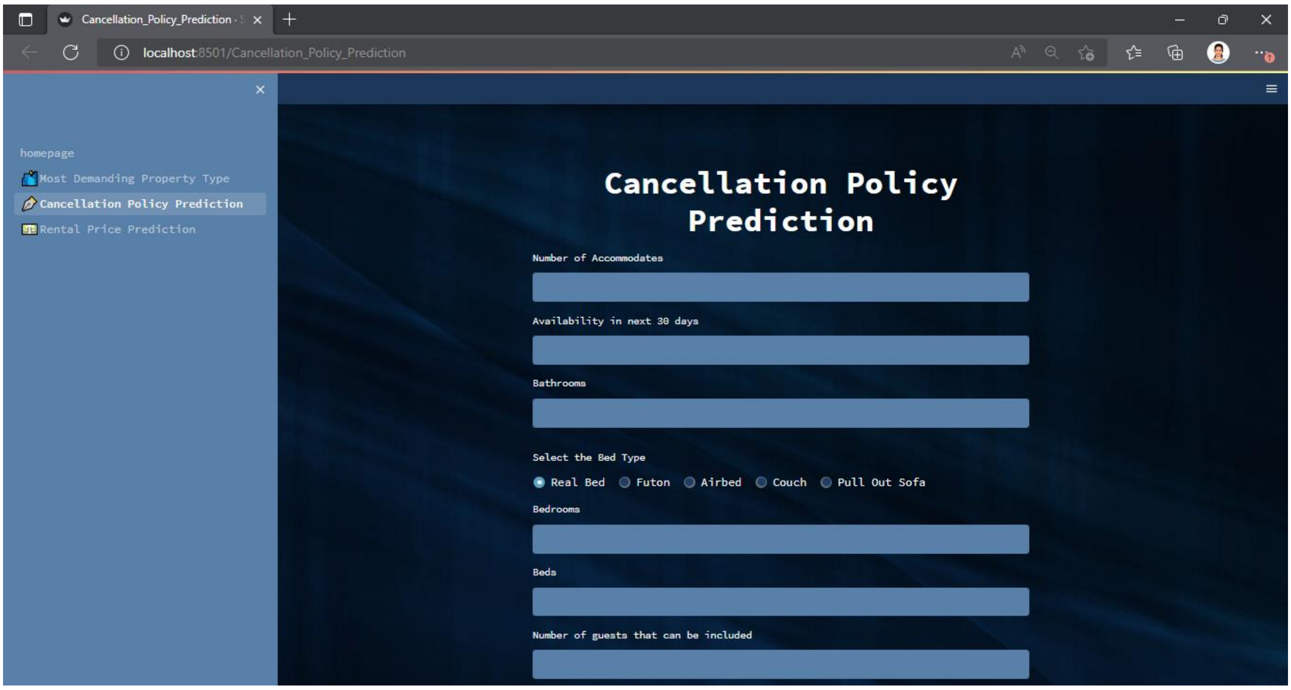




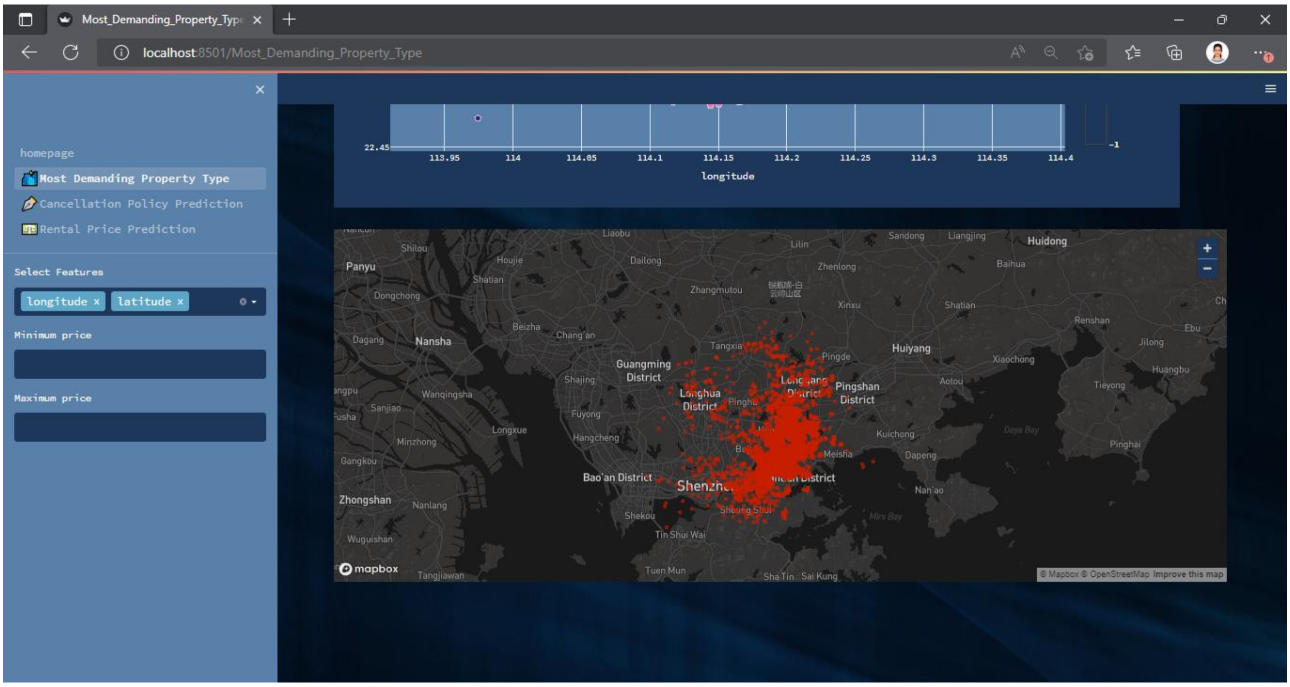
3. Rental Price Prediction Page



4. Cancellation Policy Prediction Page



5. Most Demanding Property Type Prediction Page



Report Conclusion

- Lot of money can be wasted due to incorrect predictions about most demanded property types in different areas. This application will be very important for property owners as they can save a lot of money which is spent for those kinds of constructions, as they can directly identify the demanding property types.
- When property owners want to expand their business, they can decide the most demanding property type to gain a good profit in each location through this application.
- This application is very useful to property owners to predict the price of a property. It is not a huge problem even if the owner does not have a better idea about the prices of properties according to their features.
- Buyers can reduce the effort to find the budget of a property that suits their needs.
- When the buyer is not sure about their choice and might need to cancel the booking, he/she can have an idea about the cancellation policies before confirming the booking.