Fundamentals of Data Mining – IT3051

Final Report

Group Number: 14

Group Members:

Name	Registration Number
B.D.A.D.Hettiarachchi	IT21029868
U.D.K.Navaratne	IT20708276
A.N.Elvitigala	IT21088582
V.L.K.Tennekoon	IT21015212

Table of Content

Table of Figures

1. Background

Dubai is not just a city, but it's a country of its own that has everything you need to experience life to its fullest. Living in Dubai is like living in the heart of everything, where you can have mind-blowing experiences every day. The flourishing economy of Dubai has led to a leap in its real estate market. Choosing the properties which range from apartments and houses, to villas, lands and commercial spaces is not restricted to investors and businessmen only, and there are many individuals who prefer to have a permanent residence in Dubai. Freehold property policies, tax-free policies, a diversified economy and the assurance of overall safety and security are some of the main reasons why one could benefit from buying properties in Dubai.

Are you planning to buy a property in Dubai? If you already live in Dubai, you probably have a fairly good idea about what sort of property you would like to buy. If not, then it's worth thinking about what factors are essential.

Dubai is one of the fastest-growing metropolises in the world, considered as an incredible place to live. The decision to buy a property in Dubai will either boost your current investment portfolio or secure a home in one of the best property markets in the world.

Choosing the best place to buy a property depends on your priorities, the size of your family, budgetary concerns; and if you are an investor, your financial goals. Moreover, there are so many reasons as to why you should own a property in Dubai. The biggest benefit of investing in the property market of Dubai is that you do not need to pay any taxes. This means that once you buy a property in Dubai you would not need to pay any extra taxes for it. Another major advantage is the low rate of crime. The likelihood of minor crimes is extremely less as well. The consistent development and innovation within the country, the UAE property visa where one can become eligible to get a resident visa based on your property purchase, and good rental yields, are some of the other benefits for a property holder in Dubai.

2. Target and Business Goals

As mentioned above, our target is a prospective market of property investors, along with individual investors who wish to purchase property in Dubai. This data mining case study will offer information that can guide those investors to make proper decisions concerning the acquisition of properties.

In this case, the main business purpose was to create a predictive model to predict the price and the quality of properties in Dubai. Such a model would examine diverse issues including property types, facilities of property prices and its quality. This involves the development of an instrument that will allow investors to estimate the value of a property according to their own criteria.

3. About the Dataset

Over the last two decades, Dubai has emerged as one of the most popular destinations for real estate investments, owing to the gorgeous skyscrapers, unlimited retail and dining options, low crime rate, excellent transportation routes and favorable return on investments that the city has to offer. The "Dubai Properties" dataset available in Kaggle was used to fit a predictive model using advanced analysis, after performing exploratory data analysis in the first step. Creating a more effective data product where not only property buyers but sellers can also easily and efficiently find the price of a property with desired requirements, would be the main objective of the project and thus the best fit model for the data would be identified in this step.

In this study the response variable is price and quality. Here, we have focused on two predictive methods, Regression and Decision Tree Classification. The Regression model has been used to predict the "Price", while the Decision Tree Classification model has been used to predict the "Quality" of the property.

Dataset: https://www.kaggle.com/datasets/dataregress/dubai-properties-dataset

4. Choices of Technology

Task	Technology
Model development	Python
Frontend development	HTML, CSS, JS, Bootstrap
Frontend Application Framework	Flask

Figure 1: Choice of Technology

5. Description of the original dataset

Variable name	Definition	Type of the variable
Price	market price of the apartment	Int
ID	property id	Int
Neighborhood	neighborhood of the property (54 places in Dubai)	Object
latitude	location of the dataset	Float
Longitude	location of the dataset	Float
Size_in_sqft	covered area of the apartment	Int
Price_per_sqft	price per square feet for the apartment	Int
No_of_bedrooms	number of bedrooms in apartment	Int
No_of_bathrooms	number of bathrooms in apartment	Int
Quality	quality based on number of amenities	Object
Maid_room	whether the apartment has a maid room	Boolean
Unfurnished	whether the apartment is unfurnished	Boolean
Balcony	whether the apartment has a balcony	Boolean
Barberque_area	whether the apartment has a barbeque area	Boolean
Built_in_wardrobes	whether the apartment has a wardrobe	Boolean
Central_ac	whether the apartment has central ac	Boolean
Childrens_play_area	whether the apartment has a children play area	Boolean
Childrens pool	whether the apartment has a children pool	Boolean
Concierge	whether the apartment has concierge service	Boolean
Covered_parking	whether the apartment has covered parking	Boolean
Kitchen_appliances	whether the apartment has kitchen appliances	Boolean
Loby_in_the_building	whether the apartment has a lobby	Boolean
Maid_service	whether the apartment has a maid service	Boolean
Networked	whether the apartment is networked	Boolean
Pets_allowed	whether the apartment is allowed for keeping pets	Boolean
Private_garden	whether the apartment has a private garden	Boolean
Private_gym	whether the apartment has a private gym	Boolean
Private_jacuzzi	whether the apartment has a private Jacuzzi	Boolean
Private_pool	whether the apartment has a private pool	Boolean
Security	whether the apartment has a security service	Boolean
Shared_gym	whether the apartment has a shared gym	Boolean
Shared_pool	whether the apartment has a shared pool	Boolean
Shared_spa	whether the apartment has a shared spa	Boolean
Study	whether the apartment has a study area	Boolean
Vastu_compliant	whether the apartment is vastu compliant	Boolean
View_of_landmark	whether the apartment has a view of a landmark	Boolean
View_of_water	whether the apartment has a view of water	Boolean
Walk_in_closet	whether the apartment has a walking closet	Boolean

Figure 2: Description and Types of the Data Columns

***In the data processing step, we have converted the object data types into categorical data in order to encode the labels of the columns. Here, the Boolean data type columns are encoded as well.**

6. Data Identification

- Summary of the response variable-1 ‘Price’

```
import pandas as pd

dataset = pd.read_csv('properties_data.csv')
price_summary = dataset['price'].describe().apply('{:,.2f}'.format)
print(price_summary)
```
```
✓ 0.0s

count        1,905.00
mean     2,085,829.87
std      2,913,199.96
min        220,000.00
25%        890,000.00
50%      1,400,000.00
75%      2,200,000.00
max     35,000,000.00
```

Figure 3: Price Variable Summary

Figure 4:Distribution of the Response Variable1 -Price

In the Dubai property dataset, there does not exist a property with price quoted as zero, therefore we can agree that the available data is realistic. Here, we can clearly see that the price distribution is positively skewed with a long tail.

- Summary of the response variable [2] ‘Quality’

```
import pandas as pd

dataset = pd.read_csv('properties_data.csv')
price_summary = dataset['quality'].describe()
print(price_summary)
```

```
✓ 0.0s
```

```
count       1905
unique         4
top       Medium
freq        1146
Name: quality, dtype: object
```

Figure 5: Quality Variable Summary

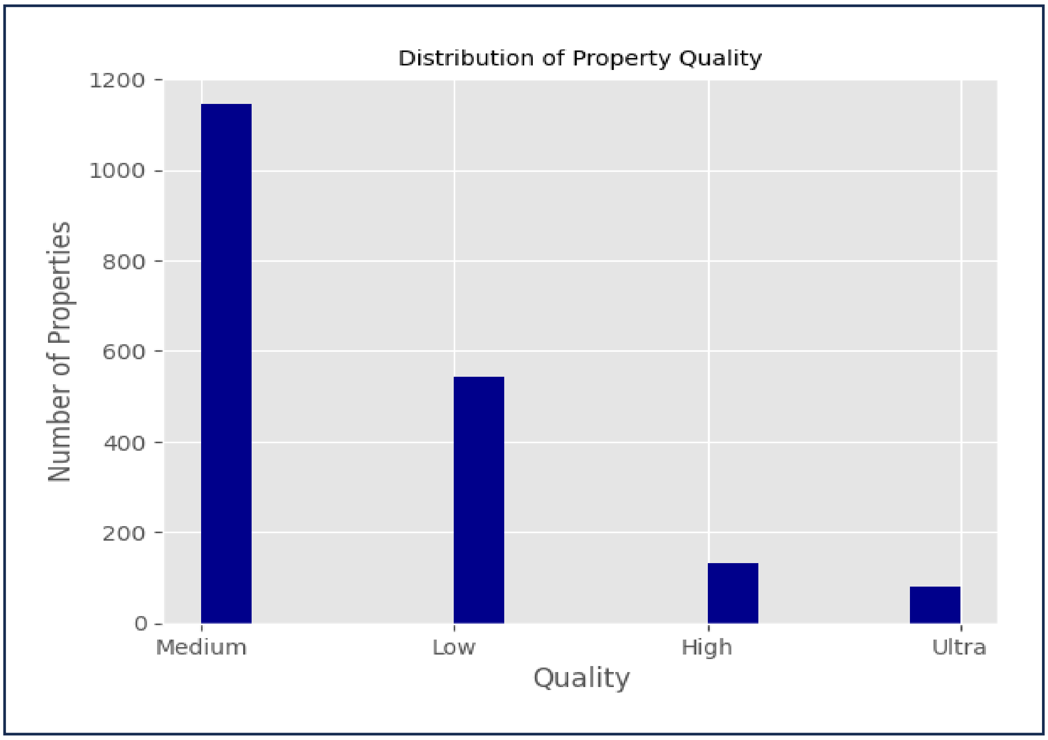

Figure 6: Distribution of the Response Variable-2 Quality

According to the histogram, most of the properties in the dataset are medium quality properties which have a limited number of features associated with them.

- Correlation matrix of the dataset

```
import matplotlib.pyplot as mp
import pandas as pd
import seaborn as sns

# plotting correlation heatmap
dataplot = sns.heatmap(dataset2.corr())

# displaying heatmap
mp.show()
✓ 0.2s
```

Figure 7: Correlation HeatMap

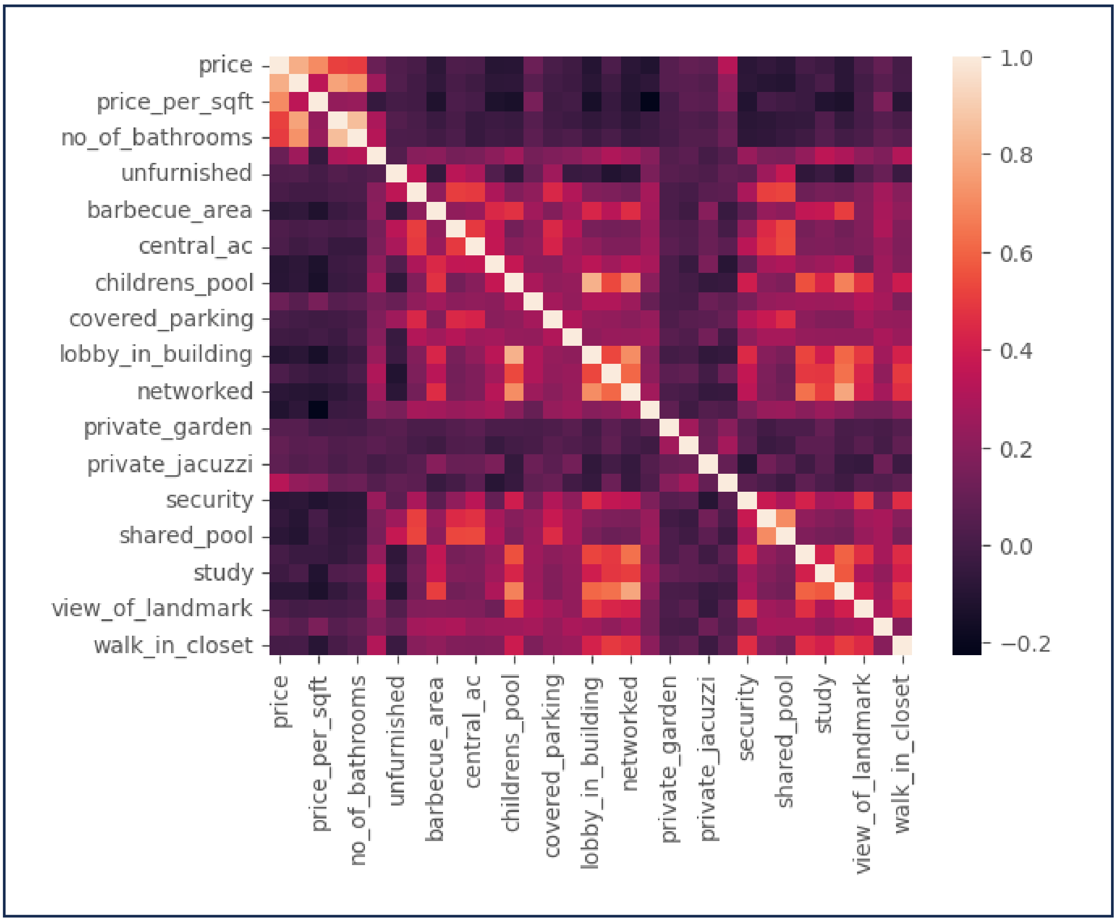

Figure 8: Correlation HeatMap for Property Dataset

The features ”price” and ”price_per_sqft” are highly correlated. This may be due to the fact that the price is calculated by multiplying ”price_per_sqft” by ”size_in_sqft”, which was a column dropped in the previous steps. The features ”no_of_bathrooms”, ”view_of_landmark”,”security”, and ”private_garden” are also highly correlated with the price. This suggests that these features are also important in determining the price of a house. This may be due to the fact that these factors are crucial for much easier and comforting accommodation. The features” unfurnished and ”barbecue_area” are negatively correlated with the price. This suggests that these features may actually decrease the value of a house.

Unfurnished properties can be less appealing to buyers, especially to those who are looking to move in quickly and easily. Buyers may have to purchase furniture and appliances, which can be expensive and time-consuming. Additionally, unfurnished properties can feel empty and impersonal. Also Barbecue areas can be seen as a negative feature by some buyers, especially those who are concerned about noise, smoke, and potential fire hazards. Additionally, barbecue areas may not be suitable for all buyers, such as those with allergies or respiratory problems.

Overall, the heatmap suggests that the features in your dataset are highly correlated with each other. This is a good thing for machine learning models, as it means that the models will be able to learn the relationships between the features more easily.

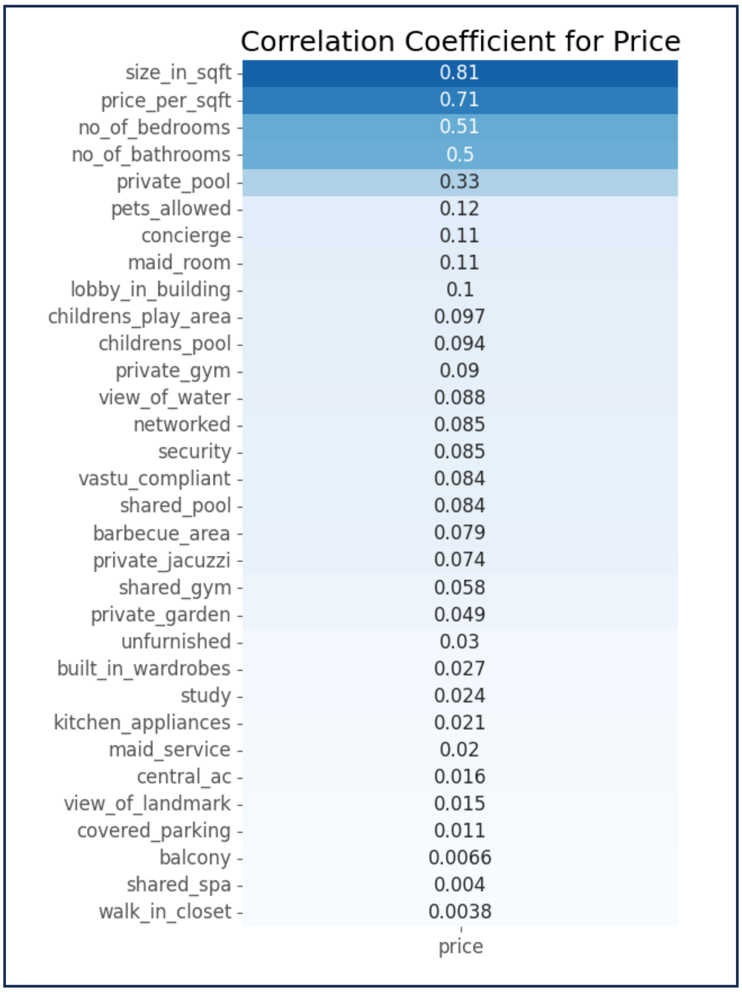

Figure 9: Table of Correlation Coefficient for Price Variable

7. Data Preprocessing and Transformation

7.1 Checking missing values

As the first step of the data preprocessing, we have checked whether any of the data columns contain missing/null values.

```
dataset.isna().sum()
[118]
...
id                    0
neighborhood          0
latitude              0
longitude             0
price                 0
size_in_sqft          0
price_per_sqft        0
no_of_bedrooms        0
no_of_bathrooms       0
quality               0
maid_room             0
unfurnished           0
balcony               0
barbecue_area         0
built_in_wardrobes    0
central_ac            0
childrens_play_area   0
childrens_pool        0
concierge             0
covered_parking       0
kitchen_appliances    0
lobby_in_building     0
maid_service          0
networked             0
pets_allowed          0
...
vastu_compliant       0
view_of_landmark      0
view_of_water         0
walk_in_closet        0
dtype: int64
```

Figure 10: Checking the Null Values

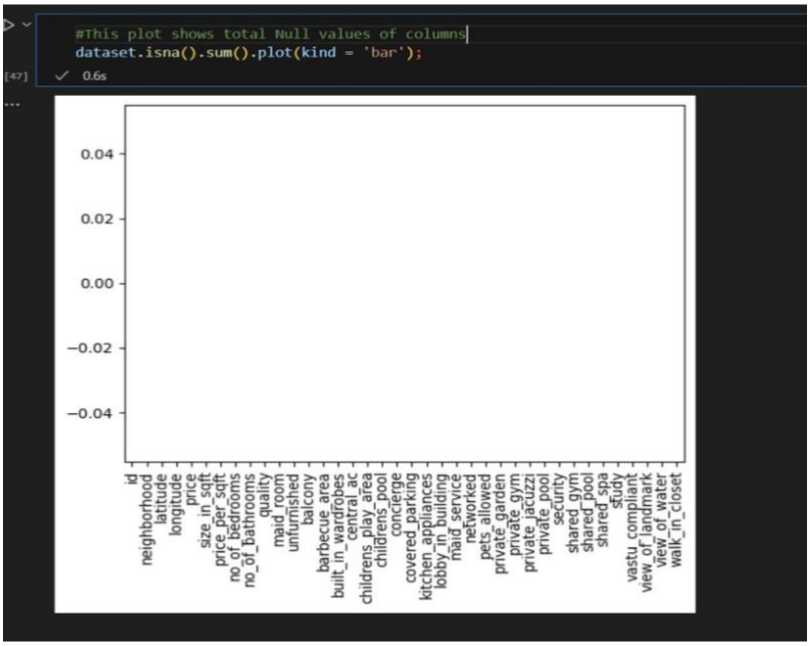

Figure 11: Null Value Visualization

Since none of the data columns contain any missing values, we proceeded with the next steps.

7.2 Conversion of data types

Since the “neighborhood” and “quality” columns are of the object data type, we have converted them to categorical values, to make it easy for us to use them in our model without any difficulties.

```python
dataset = dataset.astype({'neighborhood': 'category', 'quality': 'category'})
dataset.head()
```

Python

	id	neighborhood	latitude	longitude	price	size_in_sqft	price_per_sqft	no_of_bedrooms	no_of_bathrooms	quality	...	private_pool
0	5528049	Palm Jumeirah	25.113208	55.138932	2700000	1079	2502.32	1	2	Medium	...	False
1	6008529	Palm Jumeirah	25.106809	55.151201	2850000	1582	1801.52	2	2	Medium	...	False
2	6034542	Jumeirah Lake Towers	25.063302	55.137728	1150000	1951	589.44	3	5	Medium	...	False
3	6326063	Culture Village	25.227295	55.341761	2850000	2020	1410.89	2	3	Low	...	False
4	6356778	Palm Jumeirah	25.114275	55.139764	1729200	507	3410.65	0	1	Medium	...	False

5 rows × 38 columns

Figure 12: Data Type Conversion of ‘Neighborhood’ and ‘Quality’

The value of “Price of a square feet” was in “float” data type, and to make it easy for the user we have converted the type of the data column to “int”.

```python
dataset['price_per_sqft'] = dataset['price_per_sqft'].astype(int)
dataset.head()
```

17] ✓ 0.0s Python

	id	neighborhood	latitude	longitude	price	size_in_sqft	price_per_sqft	no_of_bedrooms	no_of_bathrooms	quality	...	private_pool
0	5528049	Palm Jumeirah	25.113208	55.138932	2700000	1079	2502	1	2	Medium	...	False
1	6008529	Palm Jumeirah	25.106809	55.151201	2850000	1582	1801	2	2	Medium	...	False
2	6034542	Jumeirah Lake Towers	25.063302	55.137728	1150000	1951	589	3	5	Medium	...	False
3	6326063	Culture Village	25.227295	55.341761	2850000	2020	1410	2	3	Low	...	False
4	6356778	Palm Jumeirah	25.114275	55.139764	1729200	507	3410	0	1	Medium	...	False

5 rows × 38 columns

Figure 13: Data Type Conversion of ‘Price per square feet"

After converting the data types,

```python
result = dataset.dtypes
print(result)
```

✓ 0.0s

```
id                         int64
neighborhood            category
latitude                 float64
longitude                float64
price                      int64
size_in_sqft               int64
price_per_sqft             int64
no_of_bedrooms             int64
no_of_bathrooms            int64
quality                 category
maid_room                   bool
unfurnished                 bool
balcony                     bool
barbecue_area               bool
built_in_wardrobes          bool
central_ac                  bool
childrens_play_area         bool
childrens_pool              bool
concierge                   bool
covered_parking             bool
kitchen_appliances          bool
lobby_in_building           bool
maid_service                bool
networked                   bool
pets_allowed                bool
...
view_of_landmark            bool
view_of_water               bool
walk_in_closet              bool
```

Figure 14 : Summary of Data Types

7.3 Removal of Unnecessary Data Columns

Here what we have done was checking what are the categories we have inside the "neighborhood" and "quality" data columns.

```
categories = dataset['neighborhood'].cat.categories
print(categories)
[7]  ✓ 0.0s

Index(['Al Barari', 'Al Barsha', 'Al Furjan', 'Al Kifaf', 'Al Quoz',
       'Al Sufouh', 'Arjan', 'Barsha Heights (Tecom)', 'Bluewaters',
       'Business Bay', 'City Walk', 'Culture Village', 'DAMAC Hills', 'DIFC',
       'Discovery Gardens', 'Downtown Dubai',
       'Dubai Creek Harbour (The Lagoons)', 'Dubai Festival City',
       'Dubai Harbour', 'Dubai Healthcare City', 'Dubai Hills Estate',
       'Dubai Land', 'Dubai Marina', 'Dubai Production City (IMPZ)',
       'Dubai Residence Complex', 'Dubai Silicon Oasis',
       'Dubai South (Dubai World Central)', 'Dubai Sports City',
       'Falcon City of Wonders', 'Green Community', 'Greens',
       'International City', 'Jebel Ali', 'Jumeirah',
       'Jumeirah Beach Residence', 'Jumeirah Golf Estates',
       'Jumeirah Lake Towers', 'Jumeirah Village Circle',
       'Jumeirah Village Triangle', 'Meydan', 'Mina Rashid', 'Mirdif',
       'Mohammed Bin Rashid City', 'Motor City', 'Mudon', 'Old Town',
       'Palm Jumeirah', 'Remraam', 'The Hills', 'The Views', 'Town Square',
       'Umm Suqeim', 'World Trade Center', 'wasl gate'],
      dtype='object')
```

Figure 15: Categories of 'Neighborhood' Variable

It is visible that the "neighborhood" column has more than 10 different categories and some of them do not contain more than 20 data records. Therefore, we have removed them from our dataset.

```
dataset['neighborhood'].value_counts()
[9]  ✓ 0.0s

Downtown Dubai                       302
Dubai Marina                         288
Jumeirah Village Circle              200
Palm Jumeirah                        178
Jumeirah Beach Residence             116
Business Bay                          97
Jumeirah Lake Towers                  70
Dubai Hills Estate                    53
The Views                             47
Jumeirah                              39
Dubai Creek Harbour (The Lagoons)     38
Mohammed Bin Rashid City              31
DIFC                                  31
Dubai Harbour                         30
Greens                                30
Motor City                            27
Town Square                           27
Dubai Sports City                     25
Al Furjan                             23
DAMAC Hills                           21
Meydan                                17
Old Town                              17
City Walk                             14
Umm Suqeim                            13
Dubai Silicon Oasis                   12
...
```

Figure 16: Record Counts for Categories of 'Neighborhood'

When it comes to the “quality” data column, there are four types of properties in the dataset.

```
categories = dataset['quality'].cat.categories
print(categories)
```

```
Index(['High', 'Low', 'Medium', 'Ultra'], dtype='object')
```

Figure 17: Categories of 'Quality" Variable

As a sub part of this data mining project, we will be predicting the quality of the property as well. First, using the regression model we will be predicting the price of the model, and then using the predicted price we will be predicting the quality of the property. Therefore, we have deleted the "quality" column from here as well.

Since we have removed the “neighborhood” column, the latitude and longitude columns are dropped as well, to avoid the model complexity.
And since the ID of the record does not contain any predictive power, that column can also be dropped.

“Price” is the response variable and therefore “price per square feet” cannot be considered as a predictor variable, therefore it can be dropped as well.

- For the regression model,

```
dataset2 = dataset.drop(['neighborhood', 'longitude', 'latitude','id'], axis=1)
dataset2.head()
```

✓ 0.0s Python

	price	size_in_sqft	price_per_sqft	no_of_bedrooms	no_of_bathrooms	quality	maid_room	unfurnished	balcony	barbecue_area	...	private_po
0	2700000	1079	2502.32	1	2	Medium	False	False	True	True	...	Fals
1	2850000	1582	1801.52	2	2	Medium	False	False	True	False	...	Fals
2	1150000	1951	589.44	3	5	Medium	True	True	True	False	...	Fals
3	2850000	2020	1410.89	2	3	Low	False	True	True	False	...	Fals
4	1729200	507	3410.65	0	1	Medium	False	False	False	False	...	Fals

5 rows × 34 columns

Figure 18: Removal of Data Columns for Regression Model

- For the decision tree classification

```
dataset2 = dataset.drop(['neighborhood', 'longitude', 'latitude','price_per_sqft','id','quality'], axis=1)
dataset2.head()
```

✓ 0.0s Python

	price	size_in_sqft	no_of_bedrooms	no_of_bathrooms	maid_room	unfurnished	balcony	barbecue_area	built_in_wardrobes	central_ac	...	priv
0	2700000	1079	1	2	False	False	True	True	False	True	...	
1	2850000	1582	2	2	False	False	True	False	True	True	...	
2	1150000	1951	3	5	True	True	True	False	True	False	...	
3	2850000	2020	2	3	False	True	True	False	False	False	...	
4	1729200	507	0	1	False	False	False	False	True	True	...	

5 rows × 32 columns

Figure 19: Removal of Data Columns for Classification Model

7.4 Label Encoding

Since we have used the "quality" column in the Decision Tree Classification, it has been encoded in the data preprocessing part of the model.

Step 1.6 - Encoding category labels to numerical values

Encoding the quality column

1 - Low 2 - Medium 3 - High 4 - Ultra

```python
from sklearn.preprocessing import LabelEncoder
label_encoder = LabelEncoder()
dataset2['quality'] = label_encoder.fit_transform(dataset2['quality'])
```

[127] Python

```python
dataset2.head()
```

[128] Python

	price	size_in_sqft	price_per_sqft	no_of_bedrooms	no_of_bathrooms	quality	maid_room	unfurnished	balcony	barbecue_area	...	private_poo
0	2700000	1079	2502.32	1	2	2	False	False	True	True	...	Fals
1	2850000	1582	1801.52	2	2	2	False	False	True	False	...	Fals
2	1150000	1951	589.44	3	5	2	True	True	True	False	...	Fals
3	2850000	2020	1410.89	2	3	1	False	True	True	False	...	Fals
4	1729200	507	3410.65	0	1	2	False	False	False	False	...	Fals

5 rows × 34 columns

Figure 20: Label Encoding for "Quality" Variable

Encoding other Boolean type columns (Yes/No) to 1s and 0s,

- For the classification model,

```python
le = LabelEncoder()
dataset_encoded = dataset2.iloc[:,6:34]

for i in dataset_encoded:
    dataset2[i] = le.fit_transform(dataset_encoded[i])

dataset2.head()
```

Python

	price	size_in_sqft	price_per_sqft	no_of_bedrooms	no_of_bathrooms	maid_room	unfurnished	balcony	barbecue_area	built_in_wardrobes	...
0	2700000	1079	2502.32	1	2	0	0	1	1	0	...
1	2850000	1582	1801.52	2	2	0	0	1	0	1	...
2	1150000	1951	589.44	3	5	1	1	1	0	1	...
3	2850000	2020	1410.89	2	3	0	1	1	0	0	...
4	1729200	507	3410.65	0	1	0	0	0	0	1	...

5 rows × 34 columns

Figure 21: Classification Model Data Label Encoding

- For the regression model,

```python
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
dataset_encoded = dataset2.iloc[:,4:33]

for i in dataset_encoded:
    dataset2[i] = le.fit_transform(dataset_encoded[i])

dataset2.head()
```

✓ 0.0s Python

	price	size_in_sqft	no_of_bedrooms	no_of_bathrooms	maid_room	unfurnished	balcony	barbecue_area	built_in_wardrobes	central_ac	...	priv
0	2700000	1079	1	2	0	0	1	1	0	1	...	
1	2850000	1582	2	2	0	0	1	0	1	1	...	
2	1150000	1951	3	5	1	1	1	0	1	0	...	
3	2850000	2020	2	3	0	1	1	0	0	0	...	
4	1729200	507	0	1	0	0	0	0	1	1	...	

5 rows × 32 columns

Figure 22: Regression Model Data Label Encoding

7.5 Rearrangement of the columns

When splitting the dataset while defining the dependent and independent variable, it is convenient to have the response variable as the last data column, therefore we have rearranged the columns of the dataset.

- For the regression model,

Step 1.7 - Rearraging columns

Since in this Regression Models, our response varibale is 'Price' we put it as the last column, so that is why we rearrange the dataset

```python
columns = [ 'size_in_sqft', 'no_of_bedrooms', 'no_of_bathrooms' ,'maid_room' ,'unfurnished' ,
            'balcony' , 'barbecue_area' , 'built_in_wardrobes', 'central_ac' ,'childrens_play_area' ,'childrens_pool' ,
             'concierge' , 'covered_parking','kitchen_appliances' ,'lobby_in_building','maid_service' ,'networked',
             'pets_allowed' ,'private_garden' ,'private_gym' ,'private_jacuzzi' ,'private_pool' ,'security' ,'shared_gym' ,
             'shared_pool' ,'shared_spa' ,'study' ,'vastu_compliant' ,'view_of_landmark' ,'view_of_water' ,'walk_in_closet' ,
             'price']
dataset2 = dataset2[columns]
```

✓ 0.0s Python

Figure 23: Rearrangement of Data Columns for Regression Model

- For the classification model,

Step 1.7 - Rearraging columns

Since in this Classification Model, our response varibale is 'Quality' we put it as the last column, so that is why we rearrange the dataset

```python
columns = ['price', 'size_in_sqft','price_per_sqft','no_of_bedrooms', 'no_of_bathrooms' ,'maid_room' ,'unfurnished' ,
            'balcony' , 'barbecue_area' , 'built_in_wardrobes', 'central_ac' ,'childrens_play_area' ,'childrens_pool' ,
             'concierge' , 'covered_parking','kitchen_appliances' ,'lobby_in_building','maid_service' ,'networked',
             'pets_allowed' ,'private_garden' ,'private_gym' ,'private_jacuzzi' ,'private_pool' ,'security' ,'shared_gym' ,
             'shared_pool' ,'shared_spa' ,'study' ,'vastu_compliant' ,'view_of_landmark' ,'view_of_water' ,'walk_in_closet' ,
             'quality']
dataset2 = dataset2[columns]
```

[189] Python

Figure 24: Rearrangement of Data Columns for Classification Model

7.6 Normalization of Data

```python
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

✓ 0.0s

Figure 25: Data Normalization/Standardization

****Please note that we have tried two predictive data mining methods and predicted both “Price” and “Quality” response variables. Therefore, we have built the two models in two separate files and done the preprocessing according to each model.**

In this report, we have stated and showed data preprocessing components of both the models.

8. Model Preparation

8.1 Classification Model

8.1.1 Decision Tree Classification Model

A decision tree classification model is a type of machine learning model that uses a tree-like structure to predict categorical values. It is a supervised learning algorithm, which means that it is trained on a dataset of known inputs and outputs. The model then learns to make predictions about new inputs based on the patterns it has learned from the training data. To train a decision tree classification model, the algorithm first splits the training data into smaller and smaller subsets based on the values of the input variables. Each split is made in order to maximize the purity of the target variable within each subset. The process continues until each subset contains only data points with the same target value, or until a certain stopping criterion is met.

To train the model we will be creating, we have split the dataset into 20% and 80% sets, with 80% of data for the training, because to predict unseen things, first, a model must learn from the provided data. Then for testing purposes we take the remaining 20%.

```python
X = dataset2.iloc[:,0:33].values
Y = dataset2.iloc[:,-1].values
```

Figure 26: Defining Independent and Dependent Variables

```python
X_train, X_test, Y_train, Y_test = train_test_split(X,Y,test_size=0.2, random_state=100)
```

Figure 27: Training and Test Dataset

Next, the following steps have been taken to build the model.

```python
#create the decision tree and fit it to the training data

model = DecisionTreeClassifier(criterion = 'gini', max_depth=None)

#max_depth is height of the tree
```

Figure 28: Decision Tree Classifier Model

```python
#fitting the model
model.fit(X_train, Y_train)
```

```
▾ DecisionTreeClassifier
DecisionTreeClassifier()
```

Figure 29: Fitting data to the Decision Tree Classifier

```python
from sklearn import tree
```

```python
column_names = dataset2.iloc[:,0:33].columns
column_names
```

```
Index(['price', 'size_in_sqft', 'price_per_sqft', 'no_of_bedrooms',
       'no_of_bathrooms', 'maid_room', 'unfurnished', 'balcony',
       'barbecue_area', 'built_in_wardrobes', 'central_ac',
       'childrens_play_area', 'childrens_pool', 'concierge', 'covered_parking',
       'kitchen_appliances', 'lobby_in_building', 'maid_service', 'networked',
       'pets_allowed', 'private_garden', 'private_gym', 'private_jacuzzi',
       'private_pool', 'security', 'shared_gym', 'shared_pool', 'shared_spa',
       'study', 'vastu_compliant', 'view_of_landmark', 'view_of_water',
       'walk_in_closet'],
      dtype='object')
```

Figure 30: Arrangement of Columns to Plot Decision Tree

After creating the Decision Tress Classification model, we have rendered a graphical version of the trained decision tree.

```python
from sklearn.tree import export_graphviz
import graphviz
```

```python
#plot the decision tree

dot_data = tree.export_graphviz(model,out_file=None, feature_names=column_names, class_names=["High","Low","Medium","Ultra"],
                                filled=True, rounded=True, special_characters=True)
graph = graphviz.Source(dot_data)
graph.render('dtree_render', view=True)
```

```
'dtree_render.pdf'
```

Figure 31: Implementation of Decision Tree Graphic

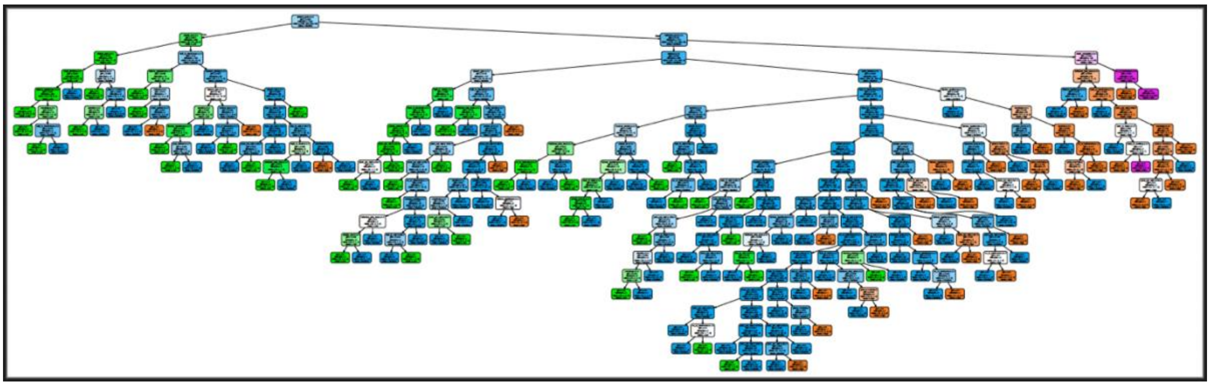

Figure 32: Rendered Decision Tree

Through the below link, you can find a clearer image of the decision tree which we had rendered: Decision Tree

8.1.2 Naïve Bayers Classifier

A Naive Bayes classifier is a type of machine learning algorithm that uses Bayes' theorem to predict the probability of an event occurring. It is a supervised learning algorithm, which means that it is trained on a dataset of known inputs and outputs. The model then learns to make predictions about new inputs based on the patterns it has learned from the training data. Naive Bayes classifiers are based on the assumption that the input features are independent of each other, given the class variable. This assumption is often not true in real-world problems, but it can be a good approximation for many problems. Naive Bayes classifiers are simple to train and interpret, and they can be used to solve a wide variety of classification and regression problems. They are particularly well-suited for problems with high-dimensional data, such as text classification and image classification.

```python
#Naive Bayers Classifier
from sklearn.naive_bayes import GaussianNB
nvb_classifier = GaussianNB()
nvb_classifier.fit(X_train, Y_train)

y_pred_nvb = nvb_classifier.predict(X_test)
print(np.concatenate((y_pred_nvb.reshape(len(y_pred_nvb),1), Y_test.reshape(len(Y_test),1)),1))
```

Figure 33: Naive Bayers Classifier

8.1.3 Logistic Regression Classifier

A logistic regression classifier is a type of machine learning algorithm used to predict the probability of a binary outcome. It is a supervised learning algorithm, which means that it is trained on a dataset of known inputs and outputs. The model then learns to make predictions about new inputs based on the patterns it has learned from the training data. Logistic regression classifiers work by fitting a logistic function to the training data. The logistic function is a sigmoid function that outputs a value between 0 and 1. The value of the logistic function represents the probability of the binary outcome occurring.

```
#Logistic Regression Classifier
from sklearn.linear_model import LogisticRegression
logistic_reg_classifier1 = LogisticRegression(random_state = 0)
logistic_reg_classifier1.fit(X_train, Y_train)

y_pred_log = logistic_reg_classifier1.predict(X_test)
print(np.concatenate((y_pred_log.reshape(len(y_pred_log),1), Y_test.reshape(len(Y_test),1)),1))
```

Figure 34: Logistic Regression Classifier

8.1.4 Random Forest Classifier

A random forest classification model is a type of machine learning algorithm that uses an ensemble of decision trees to classify data. It is a supervised learning algorithm, which means that it is trained on a dataset of known inputs and outputs. The model then learns to make predictions about new inputs based on the patterns it has learned from the training data. Random forest classification models work by constructing a large number of decision trees, each of which is trained on a random sample of the training data. The trees are also trained using a random subset of the input features. This helps to reduce the risk of overfitting and improve the generalization performance of the model. Once the trees have been trained, they are used to make predictions about new inputs by averaging the predictions of all of the trees in the forest. This process is known as bagging.

```
#Random Forest Classifier
from sklearn.ensemble import RandomForestClassifier
rand_classifier = RandomForestClassifier(n_estimators = 10, criterion = 'entropy', random_state = 0)
rand_classifier.fit(X_train, Y_train)

y_pred_rand = rand_classifier.predict(X_test)
print(np.concatenate((y_pred_rand.reshape(len(y_pred_rand),1), Y_test.reshape(len(Y_test),1)),1))
```

Figure 35: Random Forest Classifier

8.2 Regression Models

Similar to the previous model, in order to train the model we will be building, we have split the dataset into 20% and 80% sets. 80% of data is taken for training, while the remaining 20% is taken for testing purposes.

We then started building different types of regression models to find out the most accurate model to integrate with our software solution.

8.2.1 Multiple Regression Model

A multiple regression model is a statistical model that estimates the relationship between a quantitative dependent variable and two or more independent variables. It is a generalization of simple linear regression, which only considers the relationship between a dependent variable and a single independent variable.

```python
from sklearn.linear_model import LinearRegression
lin_regressor = LinearRegression()
lin_regressor.fit(X_train,Y_train)
```

```
✓ 0.1s                                                    Python

▾ LinearRegression
LinearRegression()
```

Figure 36: Multiple Linear Regressor

8.2.2 Decision Tree Regression Model

A regression tree is basically a decision tree that is used for the task of regression which can be used to predict continuous valued outputs instead of discrete outputs. Decision tree regression models are relatively simple to understand and interpret, which makes them a popular choice for many applications. They are also known for being robust to noise in the data. However, they can be prone to overfitting, which means that they can learn the training data too well and fail to generalize to new data.

```python
from sklearn.tree import DecisionTreeRegressor
tree_regression = DecisionTreeRegressor(max_depth = 5)
tree_regression.fit(X_test, Y_test)
```

```
✓ 0.0s                                                    Python

▾ DecisionTreeRegressor
DecisionTreeRegressor(max_depth=5)
```

Figure 37: Decision Tree Regressor

8.2.3 Support Vector Regression Model

SVR is different from traditional linear regression methods in that it does not try to fit a line to the data points exactly. Instead, it allows for a certain margin of error, which is known as the epsilon tube. The epsilon tube is a region around the hyperplane where the data points are considered to be within the margin of error. SVR also differs from traditional linear regression methods in that it can be used to learn non-linear relationships between the input and output variables. This is done by using a kernel function to transform the input variables into a higher-dimensional space, where it is easier to find a hyperplane that fits the data. SVR is a powerful regression algorithm that can be used to solve a wide variety of problems. It is particularly well-suited for problems where the data is noisy or where the relationship between the input and output variables is non-linear.

```
from sklearn.svm import SVR
regressor_SVR = SVR(kernel = 'rbf')
regressor_SVR.fit(X_test, Y_test)
```

✓ 0.0s

```
▾ SVR
SVR()
```

Figure 38: Support Vector Regressor

8.2.4. Random Forest Regression Model

As mentioned earlier in the classification models, the difference between the Random Forest Regression Model and the Classification model is the type of output that they predict. A random forest classifier predicts categorical values, while a random forest regressor predicts continuous values.

```
from sklearn.ensemble import RandomForestRegressor
rand_regression = RandomForestRegressor(n_estimators = 10, random_state = 0)
rand_regression.fit(X_test, Y_test)
```

✓ 0.2s

```
▾ RandomForestRegressor
RandomForestRegressor(n_estimators=10, random_state=0)
```

Figure 39: Random Forest Regressor

8.3 K-Means Clustering

The clustering step for the dataset was done as an additional phase as it wasn't our main intention to build a clustering model.
After the clustering is done, five clusters were obtained which are indicated in the scatter plot. The result can be interpreted as follows.
Cluster 1 (red) contains the most expensive properties, which are also the largest. These properties are likely to be in desirable areas and have high-end amenities. It is the smallest cluster, suggesting that there are a limited number of very expensive properties available in Dubai.
Cluster 2 (blue) contains properties that are slightly smaller and less expensive than Cluster 1. These properties are also likely to be in desirable areas but may not have as many high-end amenities. This is the largest cluster, suggesting that there is a wide range of properties available in the mid-price range.
Cluster 3 (green) contains properties that are smaller and less expensive than Cluster 2. These properties may be in less desirable areas or have fewer amenities.
Cluster 4 (cyan) contains the smallest and least expensive properties. These properties may be in even less desirable areas or have even fewer amenities.
Clusters 3 and 4 are also relatively large, suggesting that there is a good supply of affordable properties available in Dubai.

Cluster 5 (magenta) contains properties that are larger and more expensive than Cluster 4, but smaller and less expensive than Cluster 3. These properties may be in desirable areas, but may have fewer amenities or need repairs. Cluster 5 suggests that there is a good mix of affordable and more expensive properties available in this cluster.

```
from sklearn.cluster import KMeans

km = KMeans(n_clusters=5,random_state = 42)

#y_predicted = km.fit_predict(dataset.iloc[:,1:33].values)
y_means = km.fit_predict(X)
plt.ylabel('Housing Prices')
plt.legend()
plt.show()
```

Figure 40: K- Means Clustering Model

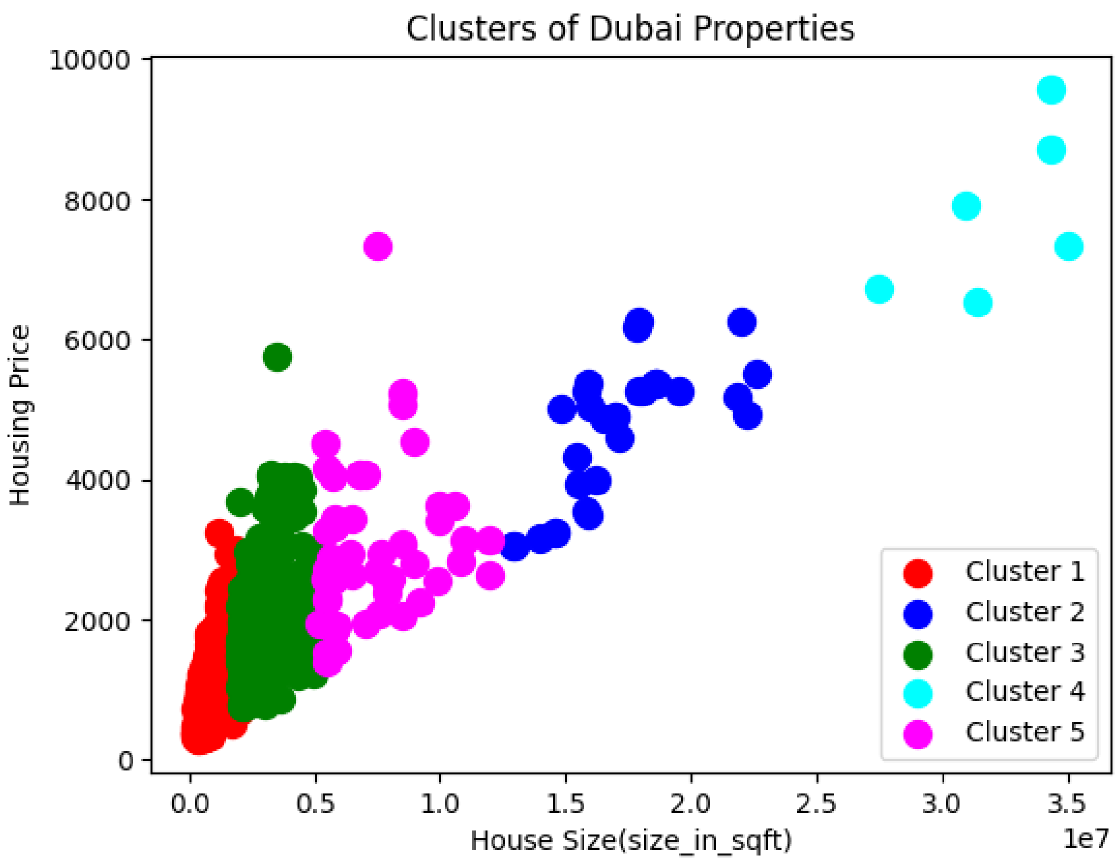

Figure 41: Clusters of Dubai Properties

This is a scatter plot of Dubai residential properties, with house size on the x-axis and housing price on the y-axis. The properties are colored according to the five clusters.

```
from sklearn.cluster import KMeans
wcss = []
for i in range(1, 20):
    kmeans = KMeans(n_clusters = i, random_state = 42)
    kmeans.fit(X)
    wcss.append(kmeans.inertia_)
plt.plot(range(1, 20), wcss)
plt.title('The Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()
```

Figure 42: Elbow Method

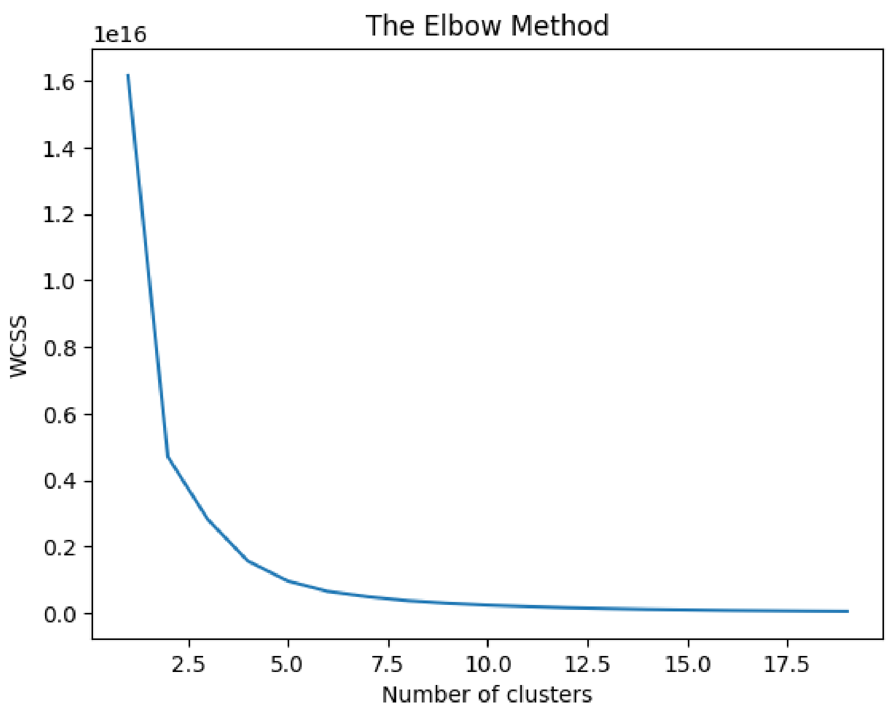

Figure 43: Elbow Method Visualization

The optimal Number of clusters can be identified using the elbow method. In the graph below, the weight sum of squares (WCSS) is plotted against the number of clusters. The number of clusters corresponding to the elbow point of the graph is considered to be the optimal number of clusters.

9. Model Comparison and Evaluation

9.1 Comparison and Evaluation of Classification Models

Since we have tried out several classification models, we have compared and evaluated all the models against each other.

Feature	Decision Tree Classifier	Naive Bayes Classifier	Logistic Regression Classifier	Random Forest Classifier
Model type	Supervised	Supervised	Supervised	Supervised
Output type	Categorical	Categorical or continuous	Probabilistic binary	Categorical or continuous
Interpretability	High	High	High	Medium
Overfitting risk	Medium	Low	Medium	Low
Accuracy	Good	Good	Good	Very good
Computational cost	Low	Low	Low	Medium
Feature	Decision Tree Classifier	Naive Bayes Classifier	Logistic Regression Classifier	Random Forest Classifier
Strengths	Simple to train and interpret, robust to noise in the data	Fast to train and predict, can handle high-dimensional data	Simple to train and interpret, robust to noise in the data	Accurate and robust to noise in the data, less prone to overfitting than other classifiers
Weaknesses	Prone to overfitting, can be difficult to interpret for complex datasets	Assumes that the input features are independent of each other, can be sensitive to the quality of the training data	Not as accurate as some other classifiers, can be susceptible to overfitting	Can be computationally expensive to train, especially for large datasets

Figure 44: Comparison of Classification Models

First, we have calculated the accuracy scores for each of the models,

Using the confusion matrix method, we have got the summarized details of the actual and predicted values of each model.

A confusion matrix is a table that summarizes the performance of a classification model. It is a two-dimensional table, with one dimension representing the actual class of an instance and the other dimension representing the predicted class.

```python
from sklearn.metrics import confusion_matrix
import seaborn as sns

cm1 = confusion_matrix(Y_test,y_pred_decisionT)
fig, ax = plt.subplots(figsize=(3,3))
ax = sns.heatmap(cm1,
                 annot=True,
                 cmap = 'Blues',
                 fmt = 'd')
plt.title("Confusion Matrix for Decision Tree Classifier")
plt.ylabel("True Label")
plt.xlabel("Predicted Label")


cm2 = confusion_matrix(Y_test,y_pred_nvb)
fig, ax = plt.subplots(figsize=(3,3))
ax = sns.heatmap(cm2,
                 annot=True,
                 cmap = 'Blues',
                 fmt = 'd')
plt.title("Confusion Matrix for Naive Bayers Classifier")
plt.ylabel("True Label")
plt.xlabel("Predicted Label")



cm3 = confusion_matrix(Y_test,y_pred_log)
fig, ax = plt.subplots(figsize=(3,3))
ax = sns.heatmap(cm3,
                 annot=True,
                 cmap = 'Blues',
                 fmt = 'd')
plt.title("Confusion Matrix for Logistic Regression Classifier")
plt.ylabel("True Label")
plt.xlabel("Predicted Label")


cm4 = confusion_matrix(Y_test,y_pred_rand)
fig, ax = plt.subplots(figsize=(3,3))
ax = sns.heatmap(cm4,
                 annot=True,
                 cmap = 'Blues',
                 fmt = 'd')
plt.title("Confusion Matrix for Random Forest Classifier ")
```

Figure 45: Implementation of Confusion Matices

- Confusion Matrix for Decision Tree Classifier

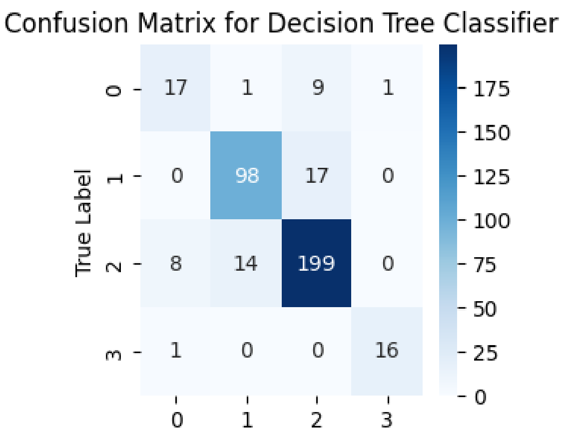

Figure 46: Confusion Matrix of Decision Tree Classifier

Accuracy = (175 + 100) / (175 + 150 + 100 + 25)
= 275 / 450
= 0.611

Precision = 175 / (175 + 150)
= 175 / 325
= 0.538

Recall = 175 / (175 + 25)
= 175 / 200
= 0.875

- Confusion Matrix for Naïve Bayers Classifier

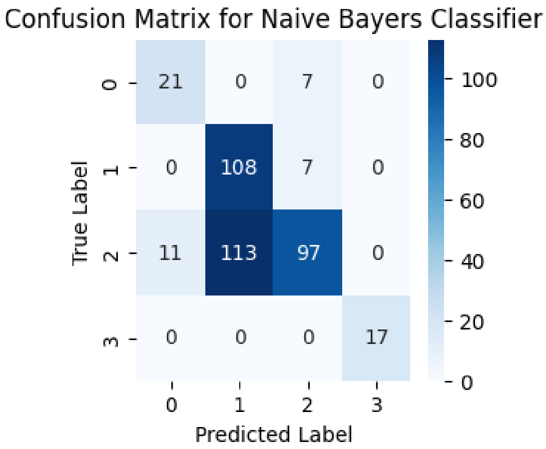

Figure 47: Confusion Matrix of Naive Bayers Classifier

Accuracy = (21 + 108) / (21 + 11 + 108 + 7)
= 129 / 147
= 0.878

Precision = 21 / (21 + 11)
= 21 / 32
= 0.656

Recall = 21 / (21 + 7)
= 21 / 28
= 0.75

- Confusion Matrix for Logistic Regression Classifier

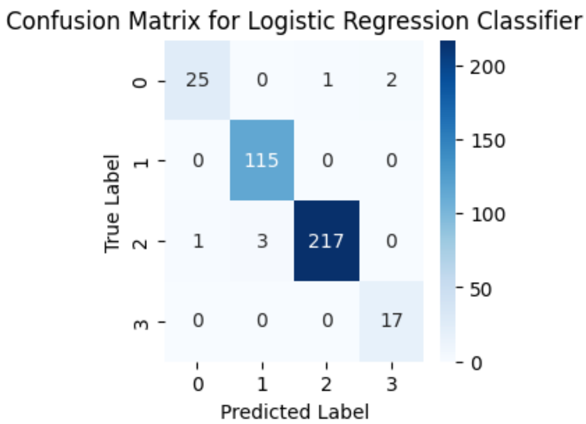

Figure 48: Confusion Matrix of Logistic Regression Classifier

Accuracy = (21 + 108) / (21 + 11 + 108 + 7)
= 129 / 147
= 0.878

Precision = 21 / (21 + 11)
= 21 / 32
= 0.656

Recall = 21 / (21 + 7)
= 21 / 28
= 0.75

- Confusion Matrix for Random Forest

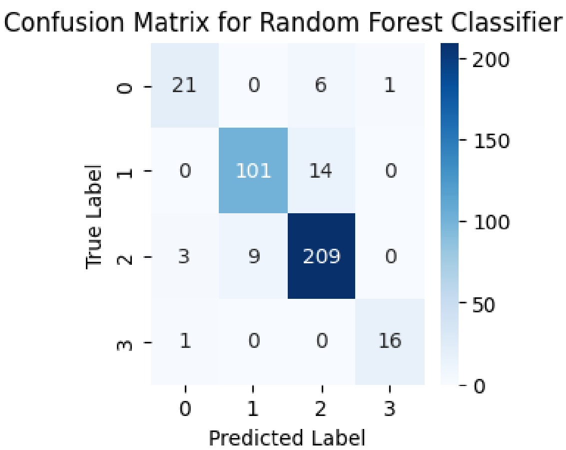

Figure 49: Confusion Matrix of Random Forest Classifier

The accuracy, precision and recall is calculated in the classification report for this model

After computing the confusion matrix, we compared the accuracy scores of each model to identify the most accurate model to integrate with our software solution.

```
from sklearn.metrics import r2_score
print("Multiple Linear Regression R2 Score",r2_score(Y_test, y_pred))
print("Decision Tree Regression R2 Score", r2_score(Y_test, y_pred_1))
print("Support Vector Regression R2 Score", r2_score(Y_test, y_pred_2))
print("Random Forest Regression R2 Score", r2_score(Y_test, y_pred_3))
```

```
Multiple Linear Regression R2 Score 0.7497157467812359
Decision Tree Regression R2 Score 0.9658626000401075
Support Vector Regression R2 Score -0.048843277633376
Random Forest Regression R2 Score 0.970507141272015
```

```
models = ["Multiple","Decision Tree","Support Vector","Random Forest"]
n = [r2_score(Y_test, y_pred),r2_score(Y_test, y_pred_1),r2_score(Y_test, y_pred_2),r2_score(Y_test, y_pred_3)]
y_pos = np.arange(len(models))
highlights = ['grey'if (x<max(n)) else 'red' for x in n]
```

Figure 50: Computing Accuracy for Each Classifier

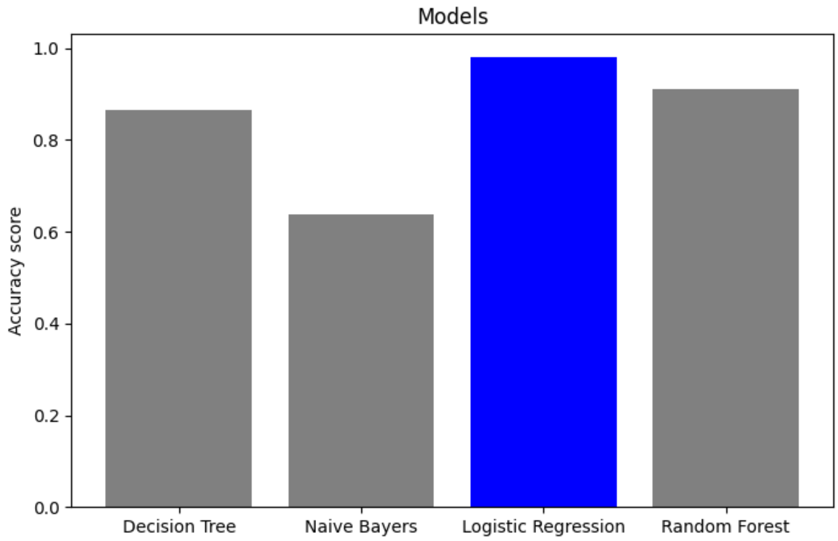

Figure 51: Comparison of Accuracy of Regression Models

According to the comparison, it is clear that the most accurate model among all is the Logistic Regression Model. Therefore, we have selected that model to integrate with the software solution.

```
from sklearn.metrics import accuracy_score,classification_report
accuracy_score(Y_test,y_pred_rand)
print(classification_report(Y_test,y_pred_rand))
```

```
              precision    recall  f1-score   support

           0       0.84      0.75      0.79        28
           1       0.92      0.88      0.90       115
           2       0.91      0.95      0.93       221
           3       0.94      0.94      0.94        17

    accuracy                           0.91       381
   macro avg       0.90      0.88      0.89       381
weighted avg       0.91      0.91      0.91       381
```

Figure 52: Classification Report

We fitted the data set using Decision Tree, Naive Bayes, Random Tree, Logistic regression classifier and gradient boost and ended up with a high value for test MSE. Therefore, then we turned into clustering as an additional part of our predictive model.

9.2 Comparison and Evaluation of Regression Models

Since we have tried out several models using the regression method, we compared and evaluated the models one by one against each model.

Characteristic	Multiple regression	Support vector regression	Random forest regression	Decision tree regression
Output type	Continuous	Continuous	Continuous	Continuous
Typical applications	Predicting house prices, predicting customer churn, predicting sales	Predicting stock prices, predicting energy demand, predicting machine failures	Predicting house prices, predicting customer churn, predicting insurance claims	Predicting customer churn, predicting fraud, predicting medical diagnoses
Strengths	Simple to train and interpret, robust to noise in the data	Can handle non-linear relationships between the input and output variables, robust to outliers	Accurate and robust to noise in the data, less prone to overfitting than other regression models	Simple to train and interpret, robust to noise in the data
Weaknesses	Not as accurate as some other regression models, can be susceptible to overfitting	Can be computationally expensive to train, especially for large datasets	Can be difficult to interpret for complex datasets	Prone to overfitting, can be difficult to interpret for complex datasets

Figure 53: Comparison of Regression Models

```
from sklearn.metrics import r2_score
print("Multiple Linear Regression R2 Score",r2_score(Y_test, y_pred))
print("Decision Tree Regression R2 Score", r2_score(Y_test, y_pred_1))
print("Support Vector Regression R2 Score", r2_score(Y_test, y_pred_2))
print("Random Forest Regression R2 Score", r2_score(Y_test, y_pred_3))
[35]
...
Multiple Linear Regression R2 Score 0.7497157467812359
Decision Tree Regression R2 Score 0.9658626000401075
Support Vector Regression R2 Score -0.04884327763733376
Random Forest Regression R2 Score 0.970507141272015

models = ["Multiple","Decision Tree","Support Vector","Random Forest"]
n = [r2_score(Y_test, y_pred),r2_score(Y_test, y_pred_1),r2_score(Y_test, y_pred_2),r2_score(Y_test, y_pred_3)]
y_pos = np.arange(len(models))
highlights = ['grey'if (x<max(n)) else 'red' for x in n]
[36]
```

Figure 54: Computing Accuracy for Each Regressor

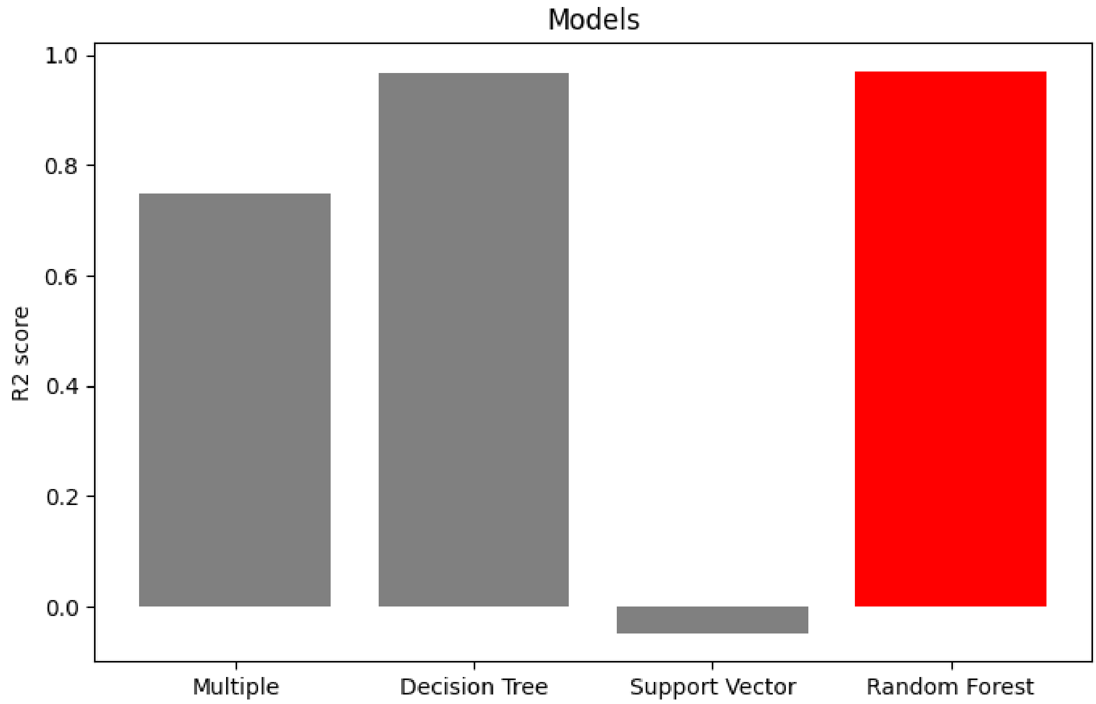

Figure 55: Comparison of Accuracy of Regression Models

According to the comparison, it is clear that the most accurate model among all is the Random Forest Regression Model. Therefore, we have selected that model to integrate with the software solution.

10. UI Implementation

Figure 56: Home Page for the Software Solution

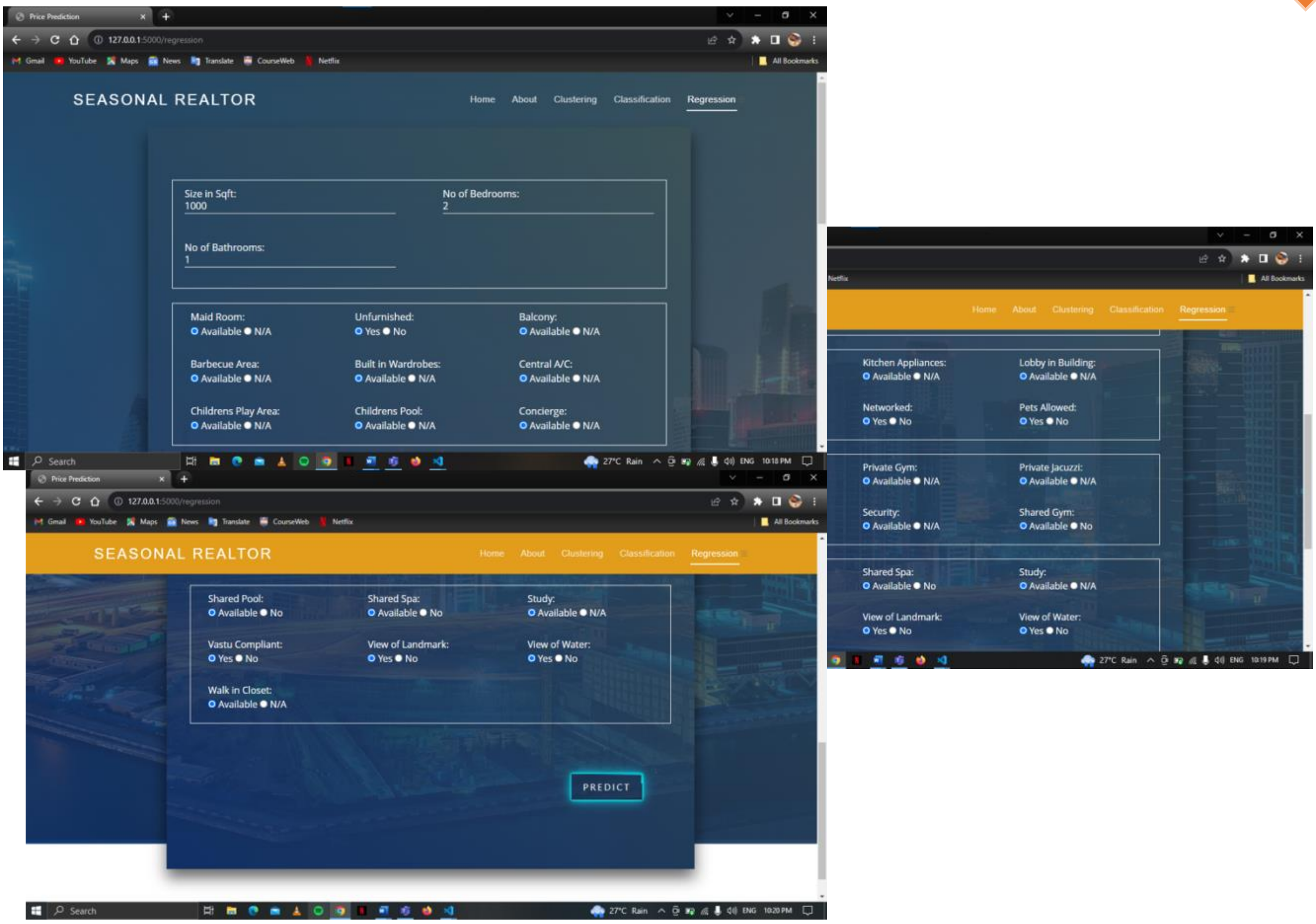

Figure 57: Form to Input Data to Regression Model

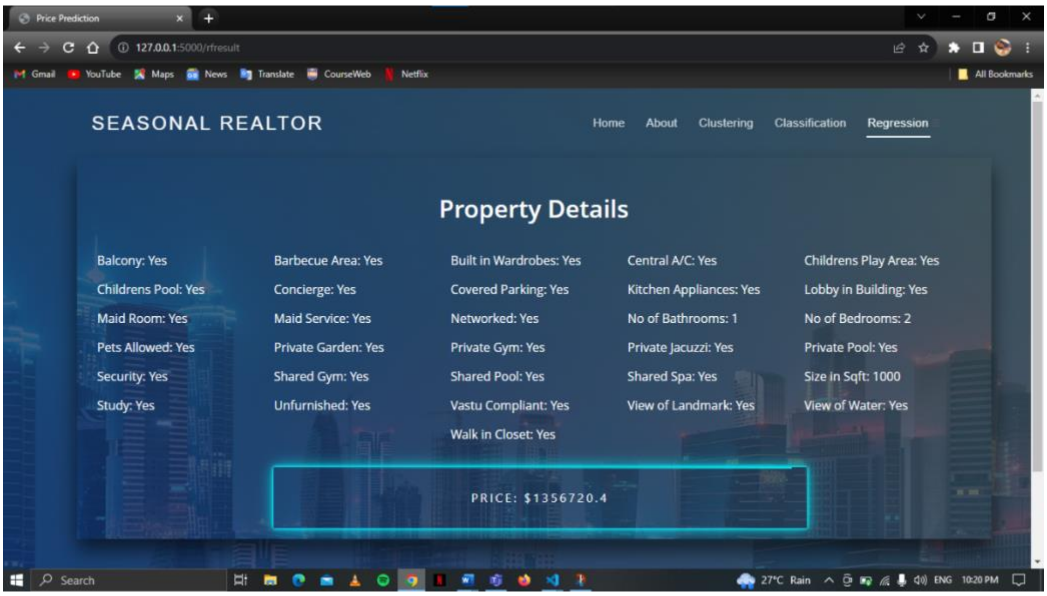

Figure 58: Regression - Predicted Result Representation in Software Solution

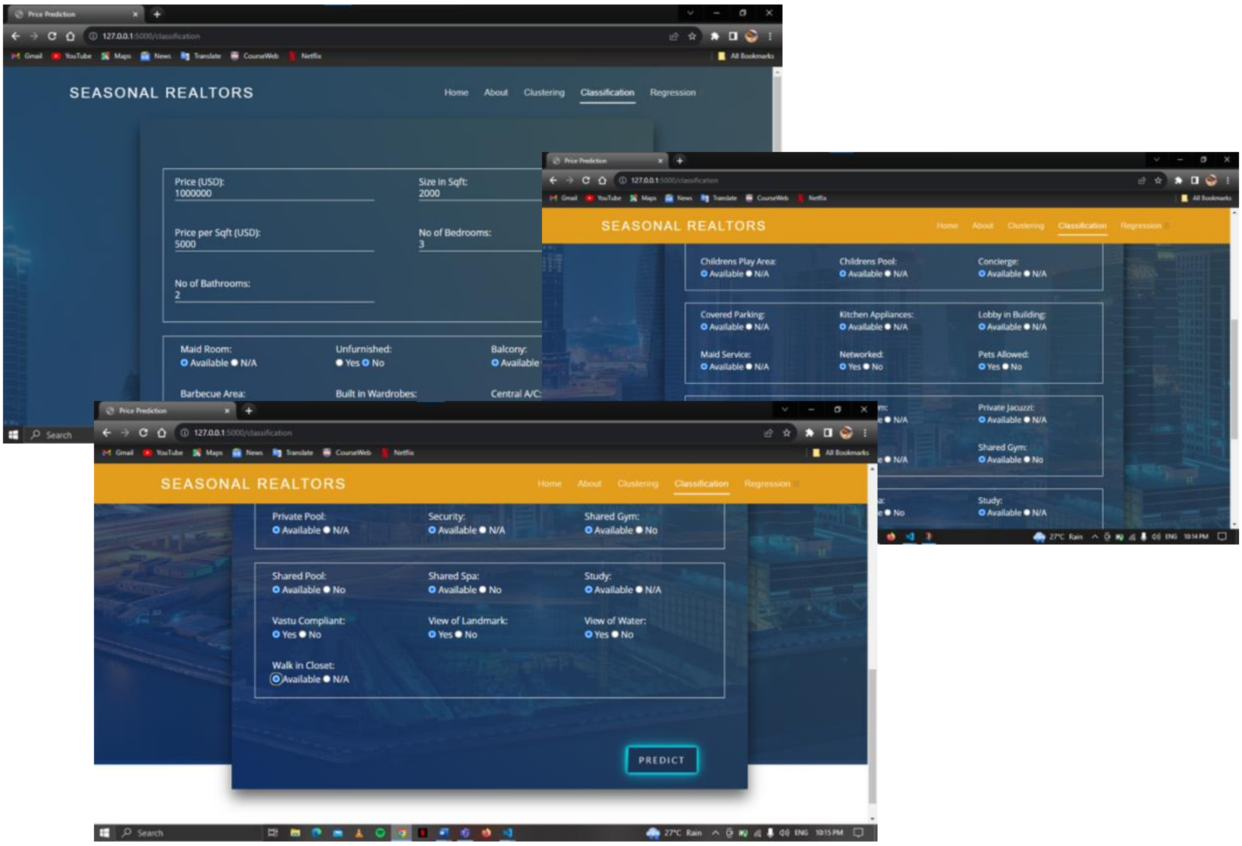

Figure 59: Form to Input Data to Classification Model

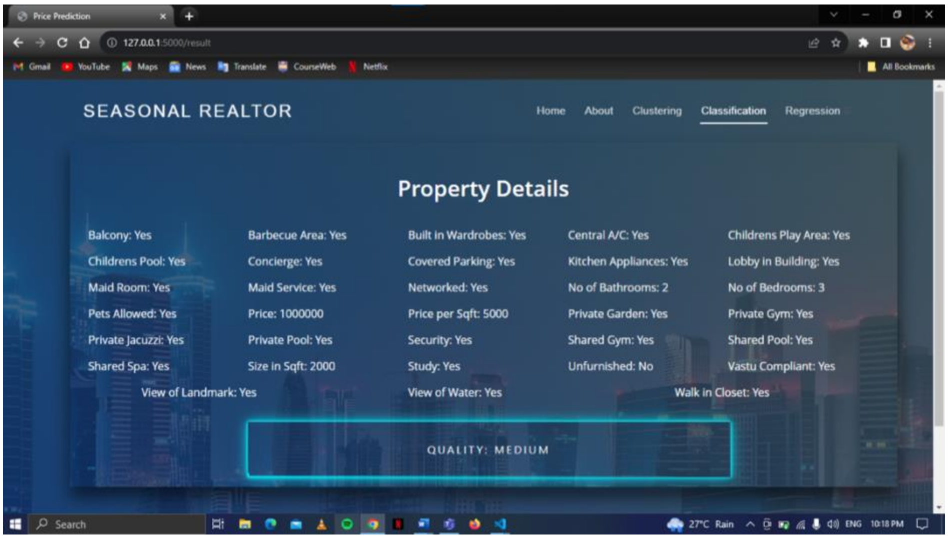

Figure 60: Classification - Predicted Result Representation in Software Solution

We have implemented a simple application with an attractive graphical interface, to provide a platform for users to interact with our in-built models. Separate forms have been designed for each model, providing space for the user to enter or select their property details as needed. With a simple click of a button, the input details are then sent into the model and the user is redirected to a separate page where the predicted values are displayed along with the user inputs for further convenience. The overall process of the application is very convenient and efficient, and does not require much technical knowledge, thus allowing users to navigate through the web application without any difficulty or confusion.

11. Test Cases

Proper testing was done to make sure that the Price predictor works up to the expected standards. The relevant test cases are listed below.

Test Scenario ID		01			
Test case description		Test the selected model to check if it is working properly or not.			
Pre-Requisite		All the data files were pre-processed.			
Serial No.	**Action**	**Inputs**	**Expected Output**	**Actual Output**	**Test Result**
1.1	Set all the value inputs and submit the form.	Source - properties_data.csv ML predicting Model – Logistic Regression. Price - 1845000 size_in_sqft - 1526 price_per_sqft - 1209.04 no_of_bedrooms - 2 no_of_bathrooms - 2 maid_room - No unfurnished - Yes balcony - Yes barbecue_area - Yes built_in_wardrobes - Yes central_ac - Yes childrens_play_area - Yes childrens_pool - No concierge - Yes covered_parking - Yes kitchen_appliances - No lobby_in_building - No maid_service - No networked - No pets_allowed - No private_garden - No private_gym - No private_jacuzzi - No private_pool - No security - No shared_gym - No shared_pool - Yes shared_spa - No study - No vastu_compliant - No view_of_landmark - No view_of_water - Yes walk_in_closet - No	Display diagnosis – Low quality	Display diagnosis – Low quality	Pass

Figure 61: Classification Test Case - 1

Test Scenario ID		02			
Test case description		Test the selected model properly working or not			
Pre-Requisite		All the data files were pre-processed			
Serial No.	**Action**	**Inputs**	**Expected Output**	**Actual Output**	**Test Result**
1.2	Set all the value inputs and submit the form.	Source - properties_data.csv ML predicting Model – Logistic Regression. Price - 2850000 size_in_sqft - 2020 price_per_sqft - 1410.89 no_of_bedrooms - 2 no_of_bathrooms - 3 maid_room - No unfurnished - Yes balcony - Yes barbecue_area - No built_in_wardrobes - No central_ac - No childrens_play_area - No childrens_pool - No concierge - Yes covered_parking - Yes kitchen_appliances - No lobby_in_building - No maid_service - No networked - No pets_allowed - Yes private_garden - No private_gym - No private_jacuzzi - No private_pool - No security - No shared_gym - No shared_pool - No shared_spa - No study - No vastu_compliant - No view_of_landmark - No view_of_water - No walk_in_closet - No	Display diagnosis – Low quality	Display diagnosis – Low quality	Pass

Figure 62: Classification Test Case - 2

Test Scenario ID		02			
Test case description		Test the selected model properly working or not			
Pre-Requisite		All the data files were pre-processed			
Serial No.	**Action**	**Inputs**	**Expected Output**	**Actual Output**	**Test Result**
1.2	Set all the value inputs and submit the form.	Source - properties_data.csv ML predicting Model – Logistic Regression. size_in_sqft - 2020 no_of_bedrooms - 2 no_of_bathrooms - 3 maid_room - No unfurnished - Yes balcony - Yes barbecue_area - No built_in_wardrobes - No central_ac - No childrens_play_area - No childrens_pool - No concierge - Yes covered_parking - Yes kitchen_appliances - No lobby_in_building - No maid_service - No networked - No pets_allowed - Yes private_garden - No private_gym - No private_jacuzzi - No private_pool - No security - No shared_gym - No shared_pool - No shared_spa - No study - No vastu_compliant - No view_of_landmark - No view_of_water - No walk_in_closet - No	Display diagnosis – 2850000	Display diagnosis – 2805088.8	Pass

Figure 63: Regression Test Case- 1

Test Scenario ID		01			
Test case description		Test the selected model properly working or not			
Pre-Requisite		All the data files were pre-processed			
Serial No.	**Action**	**Inputs**	**Expected Output**	**Actual Output**	**Test Result**
1.1	Set all the value inputs and submit the form.	Source - properties_data.csv ML predicting Model – Logistic Regression. size_in_sqft - 1526 no_of_bedrooms - 2 no_of_bathrooms - 2 maid_room - No unfurnished - Yes balcony - Yes barbecue_area - Yes built_in_wardrobes - Yes central_ac - Yes childrens_play_area - Yes childrens_pool - No concierge - Yes covered_parking - Yes kitchen_appliances - No lobby_in_building - No maid_service - No networked - No pets_allowed - No private_garden - No private_gym - No private_jacuzzi - No private_pool - No security - No shared_gym - No shared_pool - Yes shared_spa - No study - No vastu_compliant - No view_of_landmark - No view_of_water - Yes walk_in_closet - No	Display diagnosis – 1845000	Display diagnosis – 1212520.4	Fail

Figure 64: Regression Test Case -2

12. Project Structure

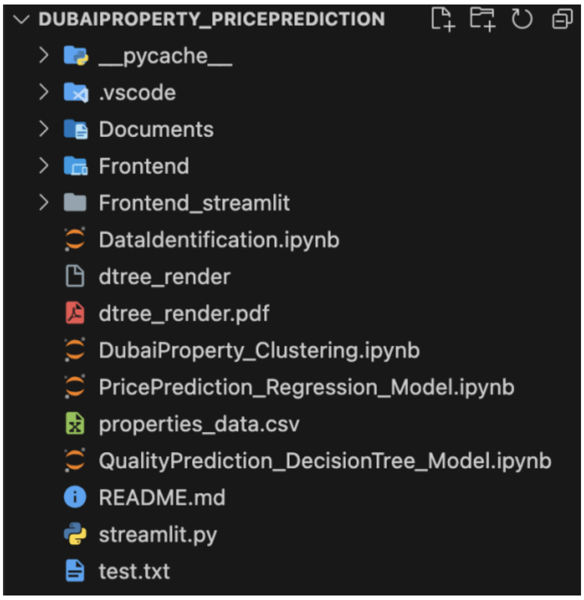

Figure 65: Project Folder Structure Inside Visual Studio Code

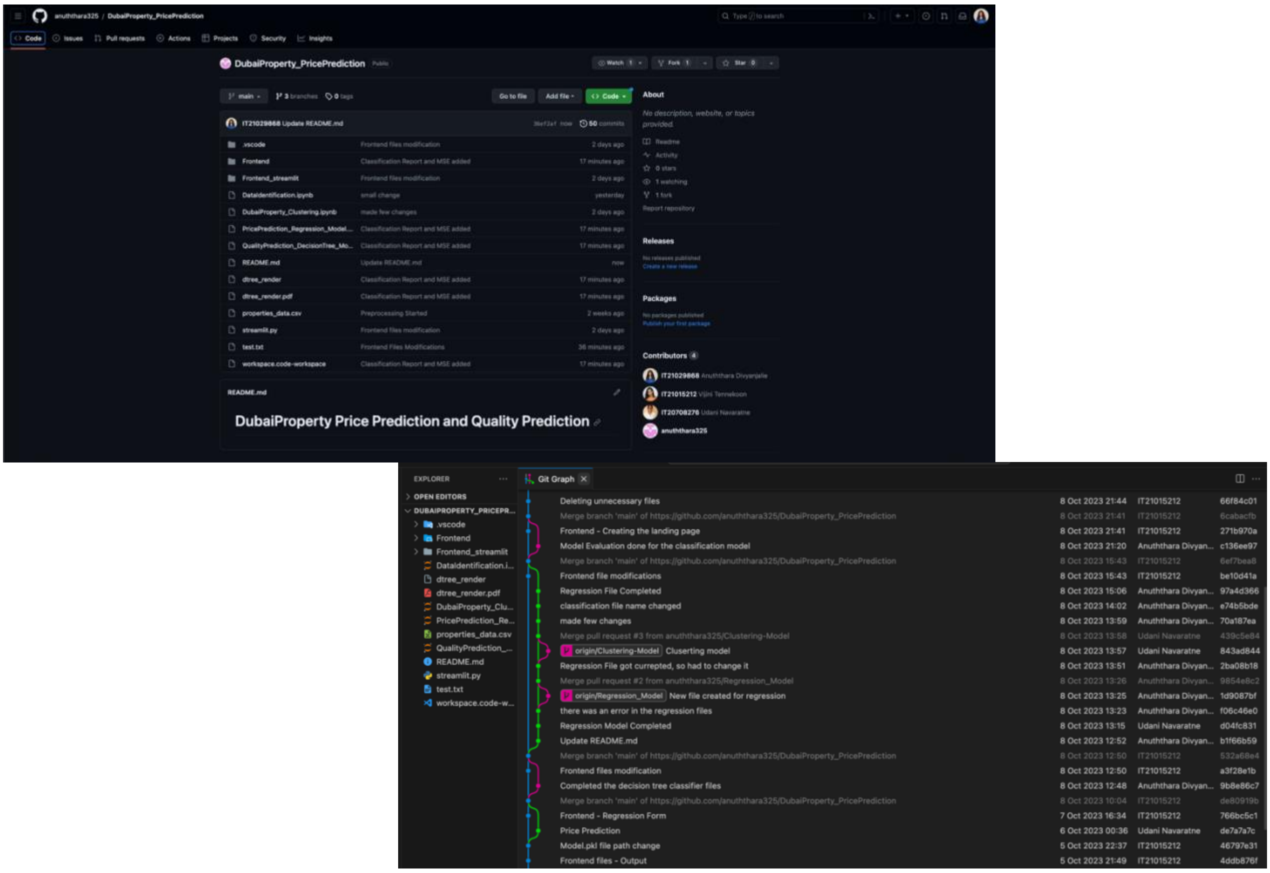

Figure 66: Version Control

13. Benefits of the proposed solutions

Dubai is one of the fastest-growing cities in the world. Yet it should be noted that the Dubai real estate market remains fairly priced compared to other major cities like London, Hong Kong, Munich, and Frankfurt.

These major cities according to the UBS Real Estate Bubble Index released for the year 2023 are overvalued and the bubble can burst at any point sinking the prices drastically (UBS Chief Investment Office, 2020). This can be seen in references graph below

High market valuations and uncertain short-term outlooks are bringing the longer-term trajectory of city housing to focus.

Yet what needs to be noted is the fact that Dubai is a real estate investment paradise because of its high rental yields.

Therefore, the solution of providing a predictive model for both quality and price of a property can benefit many.

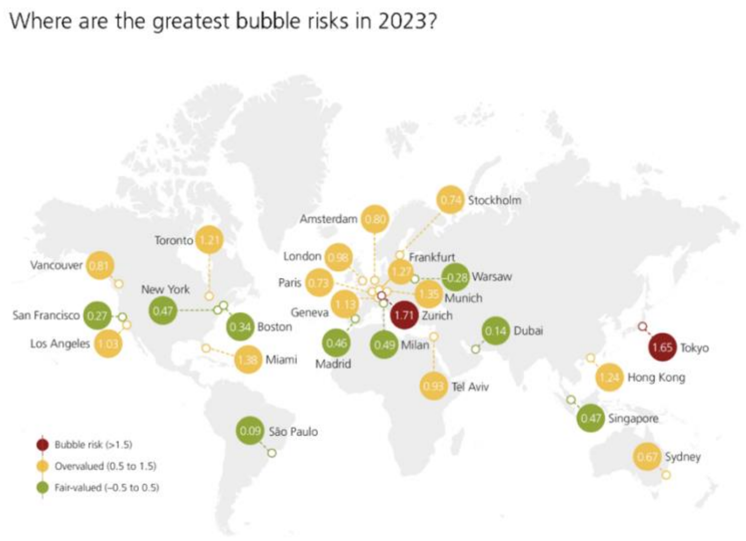

Figure 67: UBS Global Real Estate Bubble Index 2023 Graph

[Reference Link]()

Investors can leverage the predictive models for house quality and pricing to make data-driven decisions. These models offer insights into potential returns on investment, risk assessment, and long-term portfolio management. Additionally, Dubai's unique advantage of having no property tax further enhances its appeal as an investment hotspot.

These models deliver insights into prospective returns on investment, risk evaluation, and the proficient management of real estate portfolios. Predictive models not only empower investors, but also provide crucial information for policymakers and market analysts. They facilitate a comprehensive comparative analysis between Dubai and other prominent global cities, facilitating a deeper understanding of the market's relative appeal. By accentuating the long-term trajectory of Dubai's housing market, these models offer valuable insights into strategies for sustainable growth and can significantly contribute to well-informed decision-making, whether at the individual or institutional level. In summary, these models represent a potent toolkit for navigating Dubai's real estate landscape, harnessing the potential of high rental yields, and ensuring that investment choices align with enduring financial objectives in a market distinguished by its stability amid global uncertainties.

14. Conclusion

The main objective of the study was to create a data product that assists property developers, investors, or property sellers in Dubai, to identify the best price for an apartment in Dubai given their requirements. We also felt the need to build a classification model to predict the quality of the property as well. This would give the user a further understanding of the property they want to predict the price of.

15. Roles and Responsibilities

IT Number	Name	Responsibilities
IT21029868	B.D.A.D.Hettiarachchi	Implementing the model Building and designing the UI Analysing & visualization of data Testing the data Documentation
IT20708276	U.D.K.Navaratne	Implementing the model Analysing & visualization of data Building and designing the UI Testing the data Documentation
IT21088582	A.N. Elvitigala	Implementing the model Analysing & visualization of data Building and designing the UI Testing the data Documentation
IT21015212	V.L.K .Tennekoon	Implementing the model Analysing & visualization of data Building and designing the UI Testing the data Documentation

Figure 68: Table of Roles and Responsibilities

16. References

[1] S. BANERJEE, "Categories DATA SCIENCE, MACHINE LEARNING," [Online]. Available: https://www.mindsmapped.com/regression-analysis-with-python/.

[2] "A Complete Guide to Data Visualization in Python With Libraries & More," [Online]. Available: https://www.simplilearn.com/tutorials/python-tutorial/data-visualization-in-python.

[3] "Cluster Analysis in Data Mining: Applications, Methods & Requirements," [Online]. Available: https://www.upgrad.com/blog/cluster-analysis-data-mining/.

[4] "Different Methods Of Clustering," [Online]. Available: https://www.analyticsvidhya.com/blog/2016/11/an-introduction-to-clustering-and-different-methods-of-clustering/.

[5] "Random Forest Algorithm," [Online]. Available: https://www.javatpoint.com/machine-learning-random-forest-algorithm-.

[6] "How To Make a Web Application Using Flask in Python 3," [Online]. Available: https://www.digitalocean.com/community/tutorials/how-to-make-a-web-application-using-flask-in-python-3.

[7] "UBS Global Real Estate Bubble Index 2023," [Online]. Available: https://www.ubs.com/global/en/wealth-management/insights/2023/global-real-estate-bubble-index.html-.