Final Project Report

I. Definition

Domain Background

The real estate industry is one of the fastest-growing sectors in developing countries, especially in Vietnam. The price of real estate in Hanoi, the capital of Vietnam, is very expensive, which leads big challenge to possessing real estate in Hanoi. More than 50% of Hanoi's population is people who come from other cities to work and study. Most of them must rent a house or apartment to make accommodations. The rental price is affected by multiple factors, including brokers. Naturally, it becomes increasingly crucial to make use of AI and Bigdata technology to enable renters to self-estimate the price of rental houses.

Problem Statement

The rental price is affected by multiple factors, including brokers. Brokers can push the rental price higher than the actual price to get more commission. Currently, there are not any tools that can help renters estimate the price of rental houses or find the price of similar rental houses. In the capstone project, I build an AI model that can estimate the price of rental houses in Hanoi. In this scope of the capstone project, I limit rental houses to rental apartments.

Datasets and inputs

The dataset is crawled from nha.chotot.com, which is one of the biggest websites about real estate in Vietnam. I only collect news that relevant rental apartments in Hanoi. The dataset has some useful features i.e. area, no bedroom, no bathroom, location, … and detailed description which contain a lot of information.

The raw dataset contains **16916** rows, **20** columns

```
path = './data/raw_data.csv'
df = pd.read_csv(path)
df.head()
```

	Phone_Number	Poster	No_Bedroom	No_Bathroom	No_Livingroom	Area	Apartment_Type	Project	Investor	Floor	Furniture_Type	FURNITURE	Convenient
0	098285***	môi giới	3.0	2.0	1.0	80.0	tập thể	NaN	NaN	3.0	unk	['có_điều hoà', ' nóng lạnh', '...	
1	088833***	cá nhân	2.0	1.0	NaN	48.0	chung_cư	NaN	NaN	NaN	unk	[]	['trường', 'đại_họ
2	098441***	cá nhân	1.0	1.0	NaN	40.0	tập_thể chung_cư	NaN	NaN	NaN	unk	[]	['ngõ rộng
3	091552***	môi giới	1.0	1.0	NaN	30.0	studio	NaN	NaN	NaN	full	['điều_hoà', 'giường tủ', 'nóng...	['trung_t
4	091552***	môi giới	1.0	1.0	NaN	30.0	chung_cư	NaN	NaN	2.0	full	['điều_hoà', ' giường tủ', 'nón...	['xe để dưới hầm fr

5 rows × 20 columns Open in new tab

Solution statement

The plan of this project is to build a predictive model based on AI/ML to predict the price of rental apartments in Hanoi. The model is actually a regression model. The raw dataset is rental-apartment news crawled from the nha.chotot.com website. The target value of the dataset is the price of rental apartments on this website. The features to build the model are based on the features of apartments. i.e. area, no bedroom, no living room, and location,…

Benchmark model

It's always a good practice to set a benchmark while building further machine learning models. However, there aren't any AI/ML models built on my dataset. Honestly, I do not know any business metrics to validate the efficiency of AI models. So, I use the RMSE metrics for this project, the lower value the better model. I would try different models to gain as small as possible RMSE value on the test set.

In this project, tree-based models will be selected to build the predictive model due to explainability. Then, hyperparameter tuning will help to find out the optimized parameters used in the final model. In the end, It will be tested with the test set to validate that the model can result in acceptable accuracy.

Evaluation metrics

Since the project is building a regression model, I choose RMSE metrics as the model evaluation metric

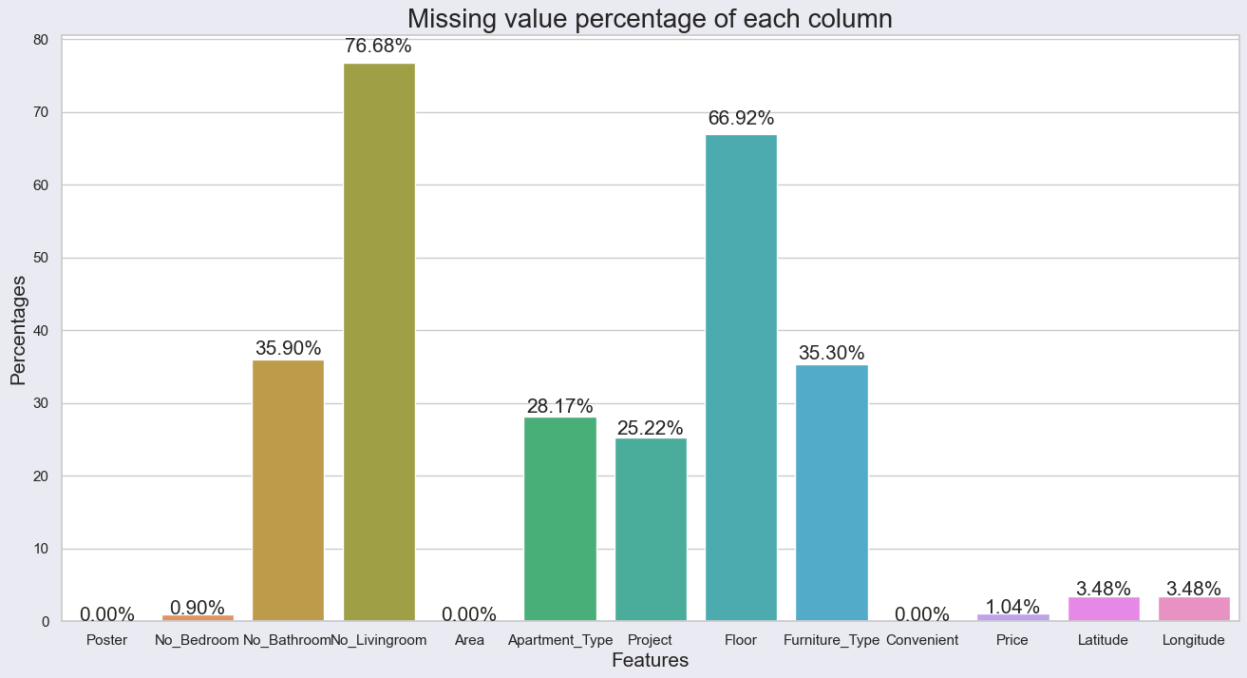
Project Design

The project is designed with the following steps:

1. Crawling data. Crawling news that relevant rental apartments in Hanoi. The Scrapy and Selenium tool is used to crawl data. The crawled data is saved in CSV file.
2. Exploratory Data Analysis. EDA is an important step in the machine learning lifecycle. The step explores the dataset which includes how many features, which type of each feature, checking the missing value, visualizing the data distribution, etc. In that way, we can have a better understanding the dataset looks like and how to select the important features for building the model.
3. Data preprocessing. There are some processes to tackle missing values, normalize categorial features, format features to specific types, etc.
4. Feature engineering. The next step will find features that are useful for building the model.
5. Building model + Tunning hyperparameter. The next step is to build a predictive model. Here we choose a decision tree model as a baseline which will help explain the feature importance better so that I can get some insight into what factors affect the price of rental apartments. I also choose a random forest model in order to towards a low RSME value in the training of the model. Then, I tweak hyperparameters to find out the best hyperparameters.
6. Testing model. Evaluate the built model on the test set.
7. Conclusion and further improvement. Review the build process and proposed advanced features that can make the model more stable in production.

II. Exploratory Data Analysis & Data Preprocess

Firstly, I visualize the proportion of missing values in columns.

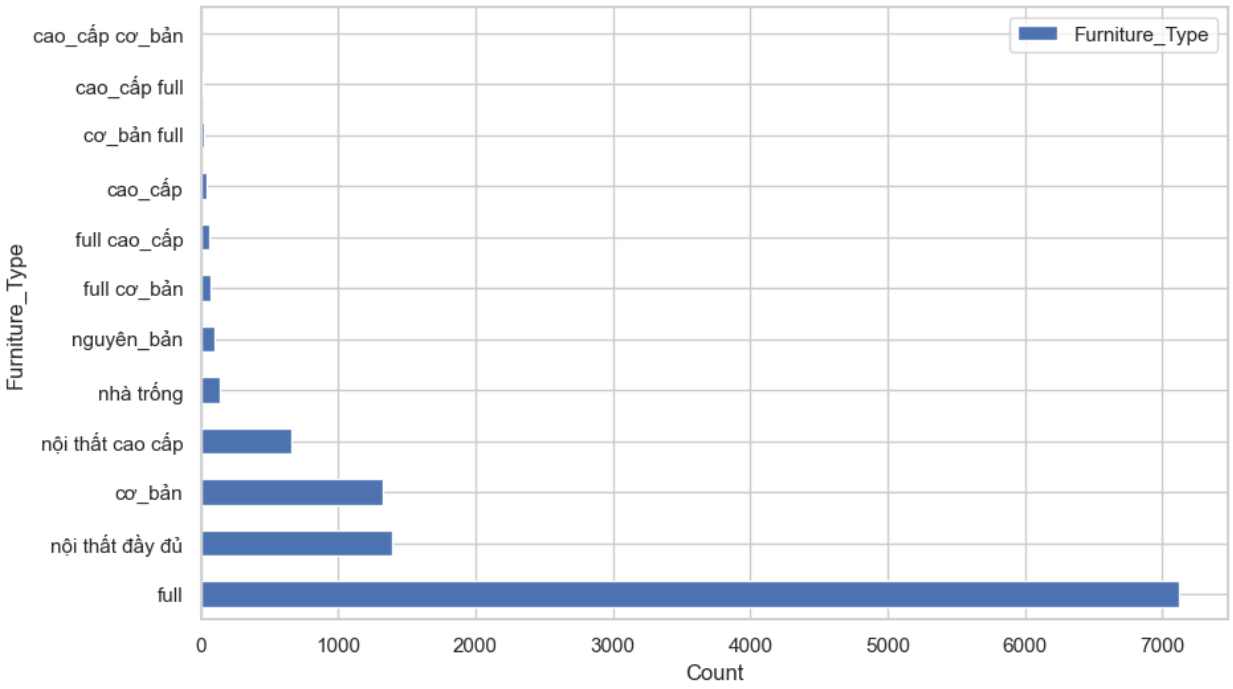


The above bar chart depicts the proportion of missing values in each column. While there are some columns having a large proportion of missing values such as No_Livingroom, Floor, No_Bathroom; Poster, Area, and Convenient feature witness a zero proportion of missing values.

Subsequently, in each feature, I will visualize to get an insight of the feature, then take account into some processes to clean data, and handle missing values, instead of the splitting project into separate steps such as EDA, Data Preprocess,...

Furniture feature

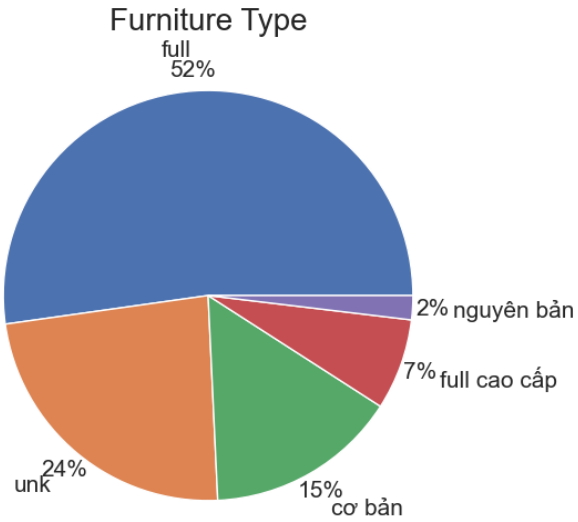
The furniture feature has 5972 rows missed the value.



The above horizontal bar chart is the frequency value of furniture type. There are 12 types of furniture, which are illustrated in the above figure. However, it has some duplicate furniture types in the dataset i.e. full cao_cấp == cao_cấp full, full cơ_bản == cơ_bản full, etc. So, I defined a consistent list of furniture types and then normalize them into it.

```
# normalize furniture type into consistent list
for i in range(df.shape[0]):
  temp = str(df.iloc[i]['Furniture_Type'])
  if temp == 'cơ_bản' or temp == 'cơ bản' or temp == 'full cơ_bản' or temp == 'full cơ bản' or temp ==
   'cơ_bản full':
      df['Furniture_Type'][i] = 'cơ bản'
  if temp == 'nhà trống':
      df['Furniture_Type'][i] = 'nguyên bản'
  if temp == 'nguyên_bản' or temp == 'nguyên bản':
      df['Furniture_Type'][i] = 'nguyên bản'
  if temp == 'nội thất cao cấp' or temp == 'full cao_cấp' or temp == 'cao_cấp' or temp == 'cao_cấp full'
   or temp == 'cao_cấp cơ_bản':
      df['Furniture_Type'][i] = 'full cao cấp'
  if temp == 'nội thất đầy đủ':
      df['Furniture_Type'][i] = 'full'
```

The raw description also contains a lot of information. So, I can extract some information about furniture type from them. Then I fill missing values by the “unk” value.



The proportion of furniture type after extraction and normalization

Area and Price Feature

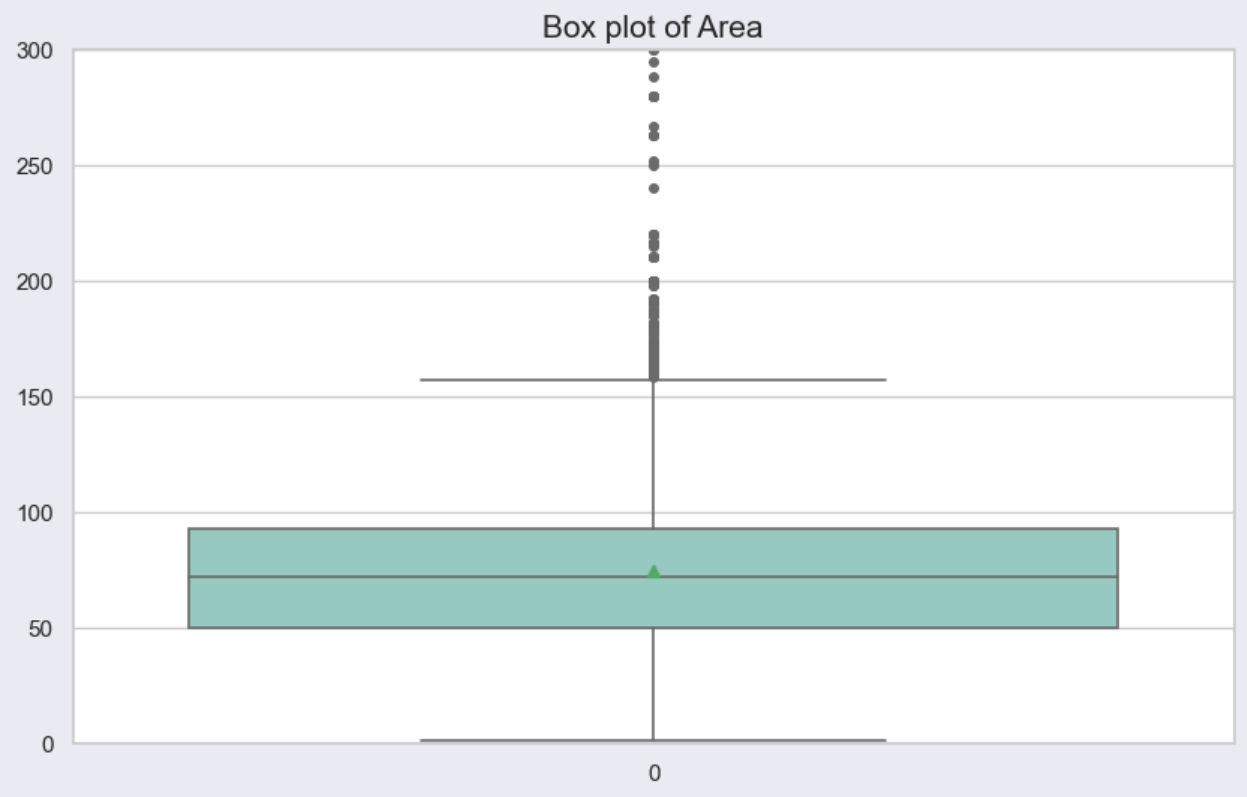
There are a lot of text values in the area and price features.

```
nan
['13.5 tr/thá', 'PRICE']
['5,5 tr/th.', 'PRICE']
[':', 'PRICE']
['10,5 tr/th', 'PRICE']
['12,5 triệu/thá', 'PRICE']
['8.5 triệu', 'PRICE']
['7.5 triệu/thá', 'PRICE']
['6.5 triệu/thá', 'PRICE']
nan
```

So, I write a function to extract numbers from text values.

```
import re
def convert_obj_to_float(df, field):
    form = "[+-]?(\d+(\.\d*)?|\.\d+)([eE][+-]?\d+)?"
    for i in range(df.shape[0]):
        if isinstance(df[field][i], str):
            temp = re.findall(form, df[field][i].replace(',','.').replace('m2','').replace(' tr ','.')
             .replace('tr','.').replace('00.000','.'))
            if not len(temp):
                df[field] = df[field].drop(index=i)
            else:
                temp_ = 0.0
                for t in temp:
                    temp_ += float(t[0])
                df[field][i] = temp_/len(temp)
        if isinstance(df[field][i], int):
            df[field][i] = float(df[field][i])
    df[field] = df[field].astype('float')
convert_obj_to_float(df, 'Price')
convert_obj_to_float(df, 'Area')
```

After using the above function to extract numbers, Price and Area features have **192** and **16** missing values, respectively.



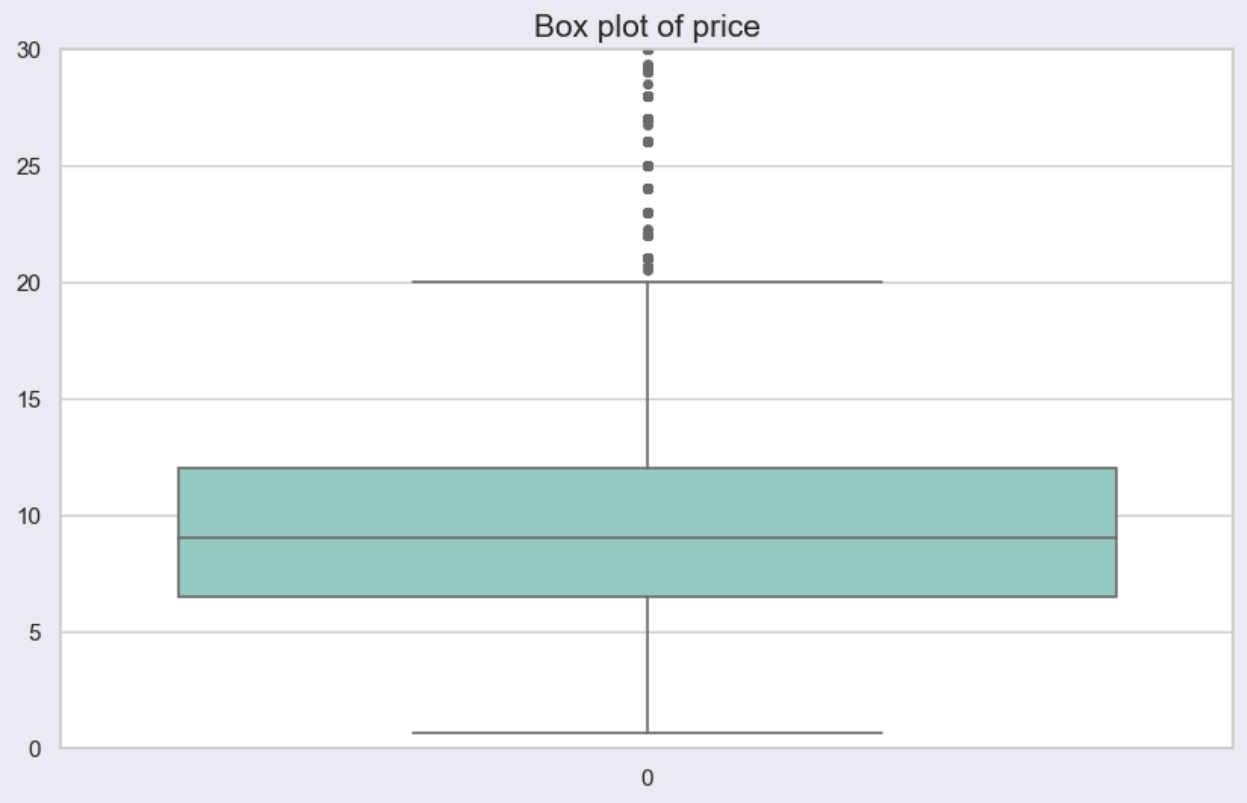
The above boxplot is a boxplot of the area feature. It illustrates that there are some outlier values which is larger than 160 m2.

```
df.query('Area > 160')
```

	Phone_Number	Poster	No_Bedroom	No_Bathroom	No_Livingroom	Area	Apartment_Type	Project	Investor	Floor
169	036789***	môi giới	4.0	3.0	NaN	182.0	chung_cư cao_cấp	dolphin plaza trần bình	NaN	NaN
343	032616***	cá nhân	3.0	3.0	NaN	220.0	chung_cư	vincom center bà triệu	NaN	10.0
372	036789***	môi giới	4.0	3.0	NaN	192.0	chung_cư	dolphin plaza	NaN	NaN
447	093396***	cá nhân	5.0	4.0	NaN	295.0	penthouse	udic westlake tây hồ	NaN	NaN
558	093333***	môi giới	2.0	1.0	1.0	445.0	mini chung_cư	NaN	NaN	NaN
708	036789***	môi giới	4.0	3.0	NaN	186.0	chung_cư cao_cấp	dolphin plaza số 6 nguyễn_hoàng.	NaN	NaN
824	098736***	môi giới	3.0	2.0	NaN	190.0	chung cư	thành công tower 57 láng hạ	NaN	NaN

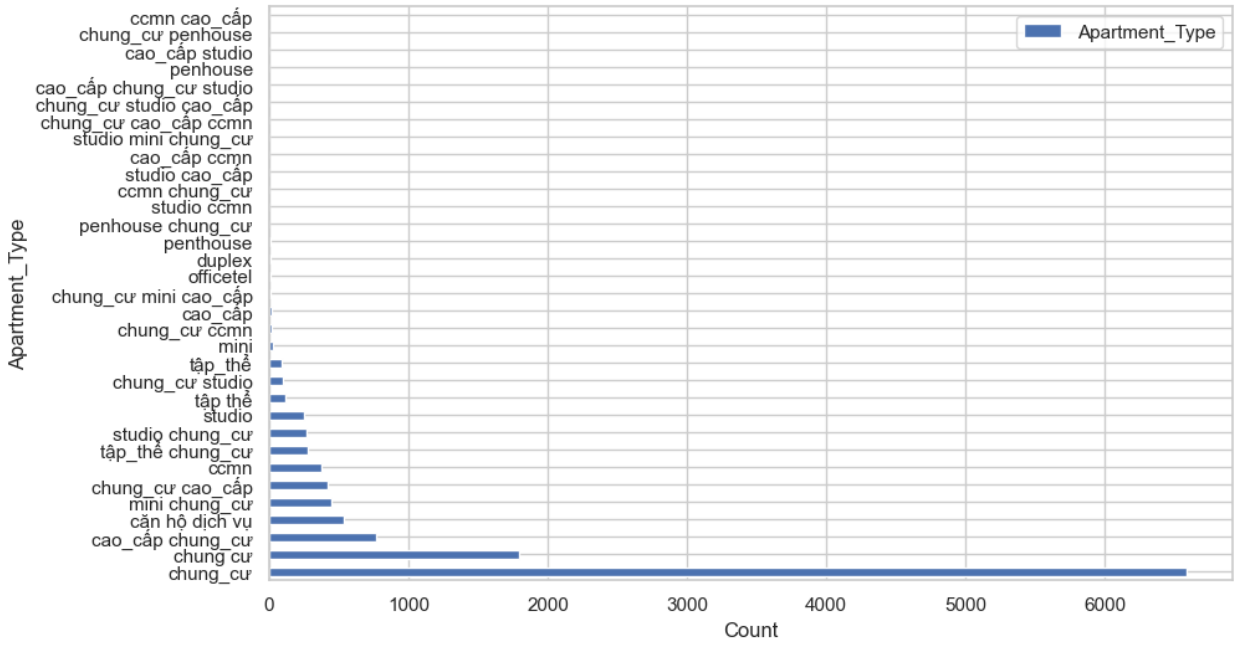
162 rows × 20 columns Open in new tab

All rows have an area that is larger than 160 m2. Apparently, some apartments have an abnormal area i.e. 445 m2



The boxplot seen above is from the pricing feature. It demonstrates that the pricing will be an outlier if it is more than 20 million Vietnamdong. However, for luxury apartments, the cost of a rental apartment greater than 20 million Vietnamdong is reasonable. I also print out those rows and find out that the price feature's outliner threshold is 50 million Vietnamdong.

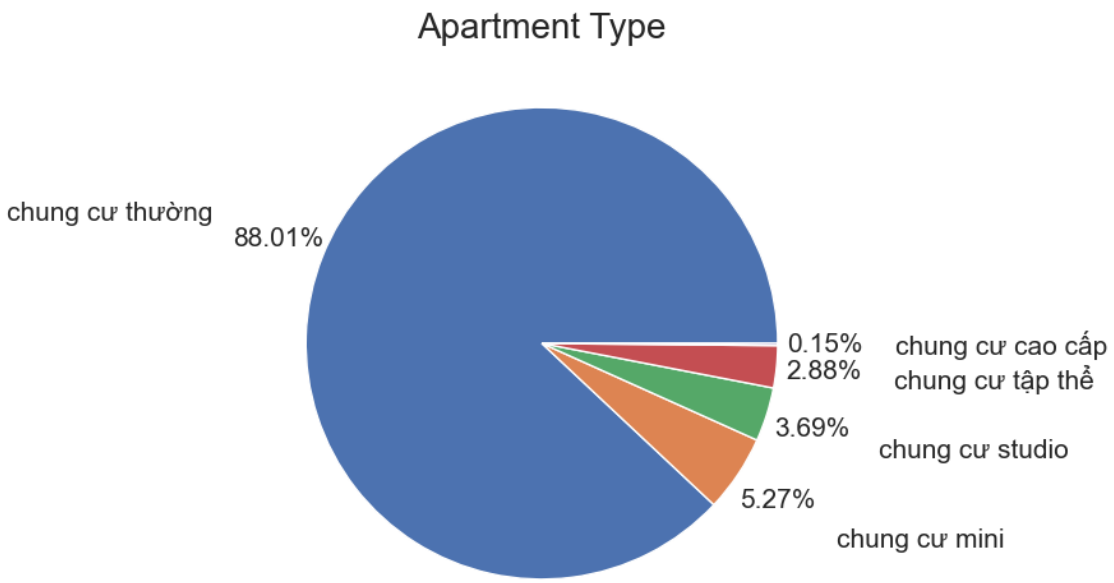
Apartment Type



The above horizontal bar chart is the frequency value of apartment type. There are many types of apartments. However, some of them are duplicates. So, I defined a consistent list of apartment types and then normalize them into it.

```
# normalize apartment type into consistent list
# chung cư cao cấp == luxury apartment
# chung cư studio == studio apartment
# chung cư mini == mini apartment
# chung cư tập thể == collective apartment
# chung cư thường == normal apartment
for i in range(df.shape[0]):
    if 'duplex' in str(df['Apartment_Type'][i]):
        df['Apartment_Type'][i] = 'chung cư cao cấp'
    elif 'officetel' in str(df['Apartment_Type'][i]):
        df['Apartment_Type'][i] = 'chung cư cao cấp'
    elif 'studio' in str(df['Apartment_Type'][i]):
        df['Apartment_Type'][i] = 'chung cư studio'
    elif 'penhouse' in str(df['Apartment_Type'][i]):
        df['Apartment_Type'][i] = 'chung cư cao cấp'
    elif 'mini' in str(df['Apartment_Type'][i]) or 'ccmn' in str(df['Apartment_Type'][i]):
        df['Apartment_Type'][i] = 'chung cư mini'
    elif 'tập_thể' in str(df['Apartment_Type'][i]) or 'tập thể' in str(df['Apartment_Type'][i]):
        df['Apartment_Type'][i] = 'chung cư tập thể'
    else:
        df['Apartment_Type'][i] = 'chung cư thường'
```

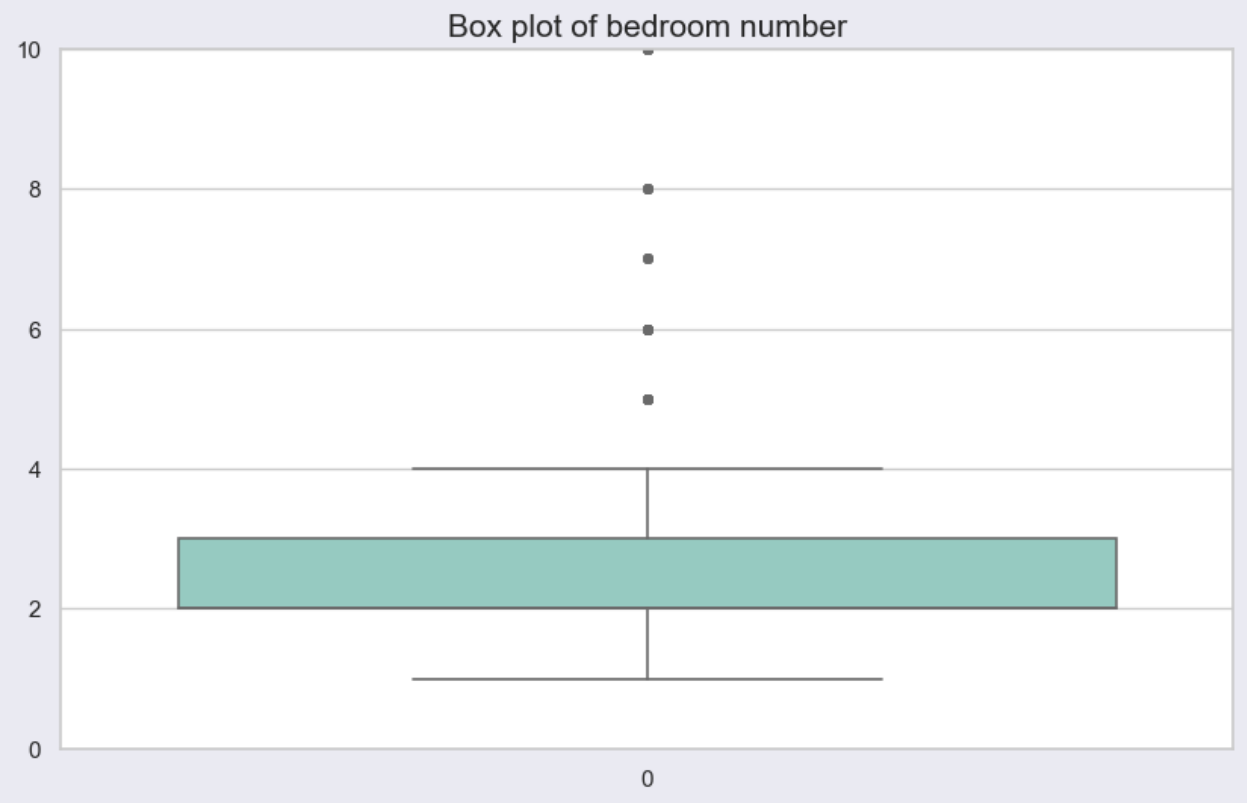
Normalization function



The proportion of apartment type after normalization

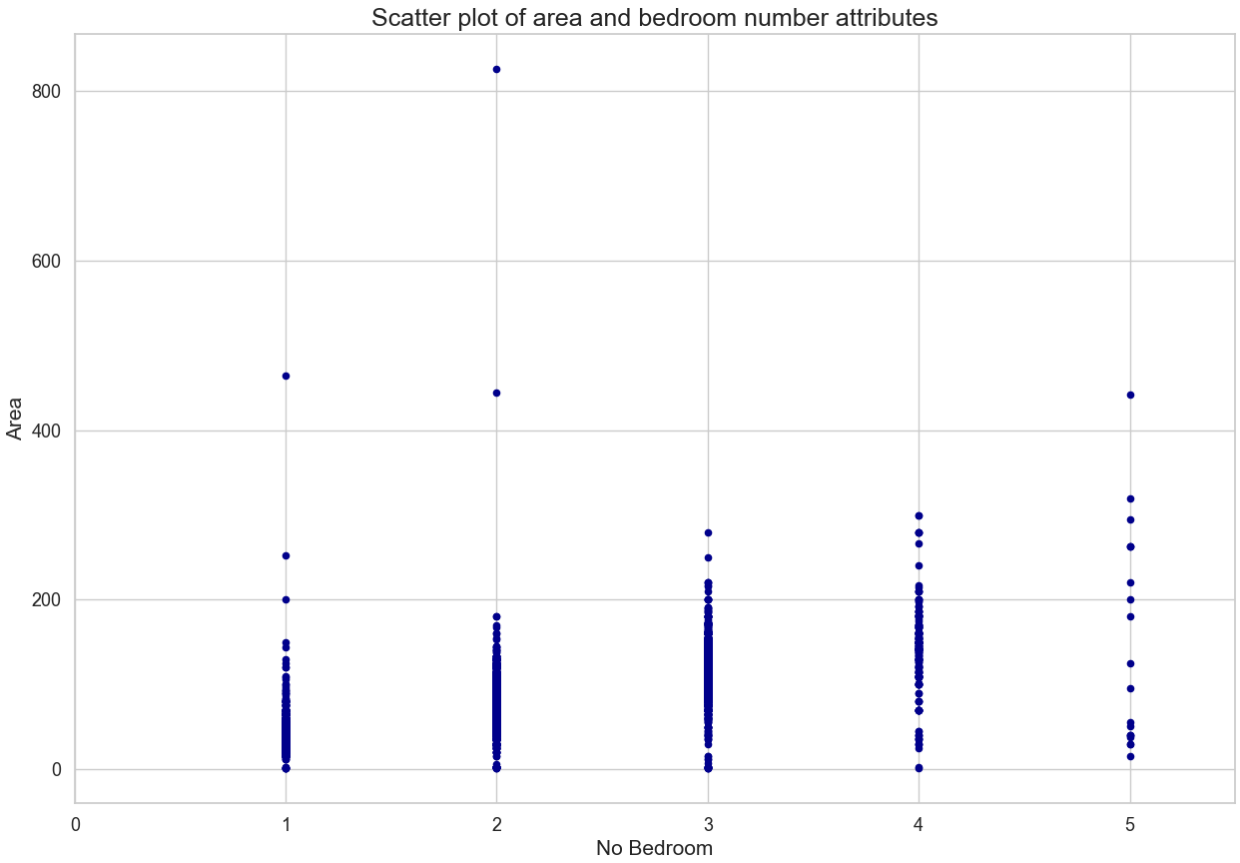
No Room Feature

In the dataset, while the No Livingroom feature has **12971** missing values; the No Bathroom and Bedroom features have a smaller proportion of missing values, **6073** and **152**, respectively.

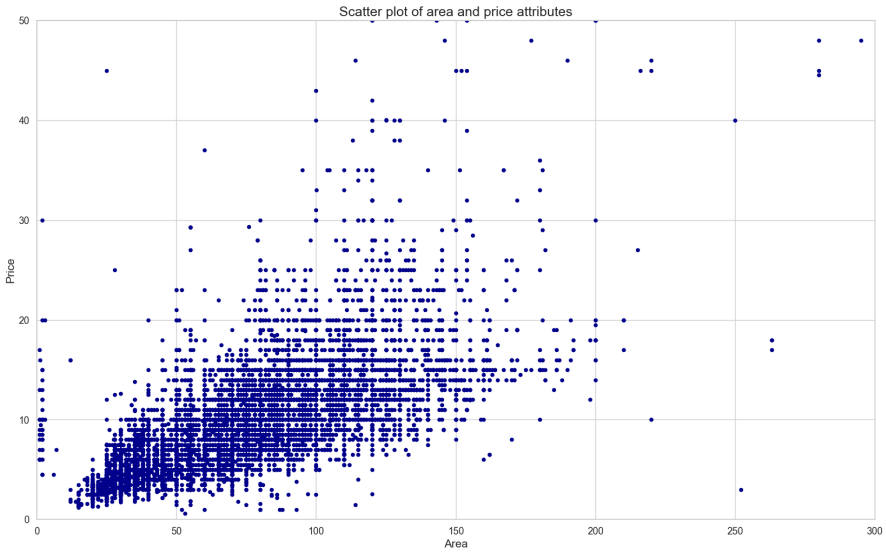


The boxplot seen above is from the no bedroom feature. It demonstrates that the no bedroom will be an outlier if it is larger than 4.

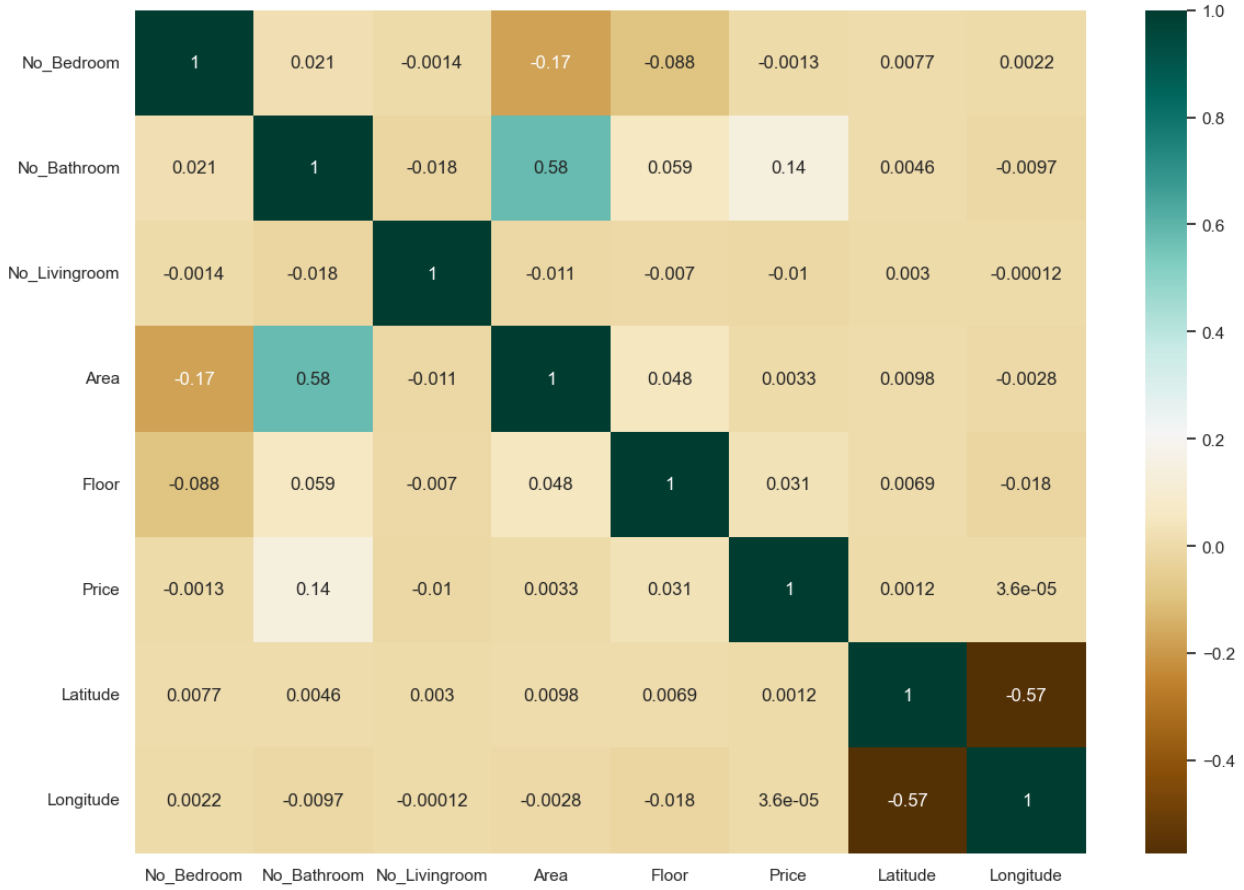
Visualize the relationship between different variables



The above scatter figure demonstrates a positive correlation between no bedroom and area.

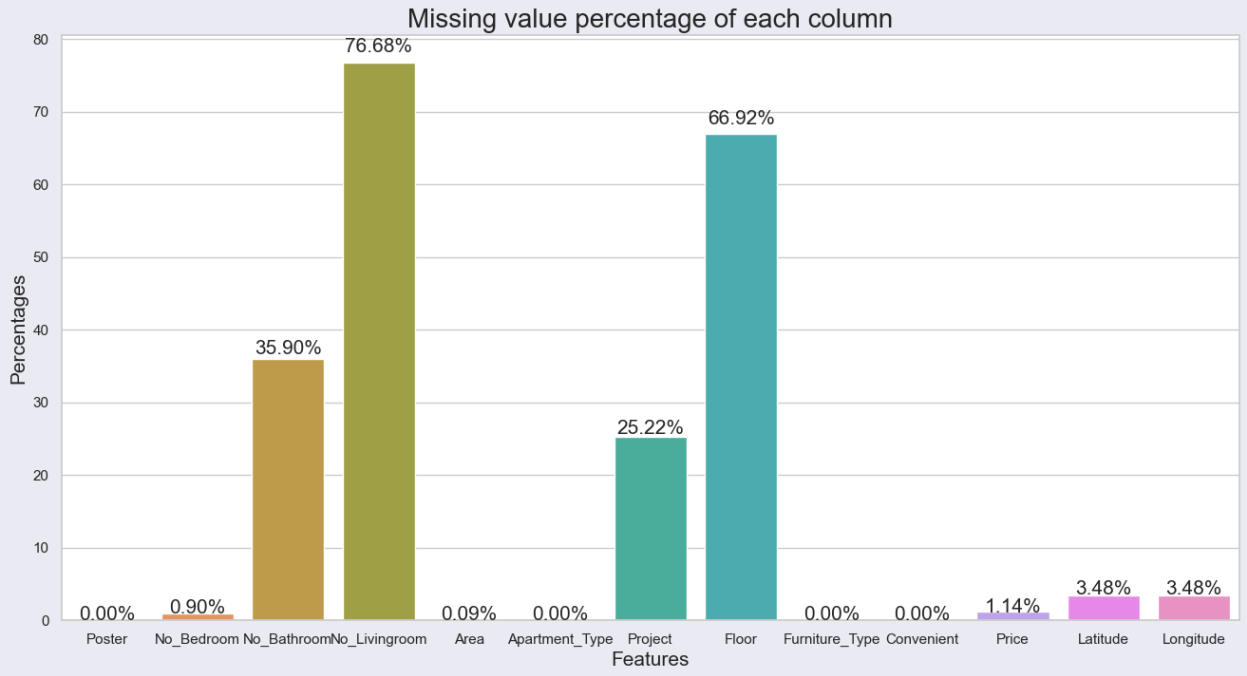


Apparently, the above scatter illustrates a positive correlation between area and rental price.



Correlation of different variables

Data cleaning



The missing-value proportion of each feature after some above steps

In this part, I will remove rows that have outliers or nan values in columns such as Area, No Bedroom, Price, Latitude, and Longitude.

```
# remove rows that have nan value in some columns such as: Area, No Bedroom, Price, Latitude, Longitude
df.dropna(axis=0, subset=['No_Bedroom', 'Area', 'Price', 'Latitude', 'Longitude'], inplace=True)
```

```
# remove outlier rows

# remove rows that have price > 50 million vietnamdong
df = df.loc[df['Price'] <= 50]

# remove rows that have no bedroom > 4
df = df[~df['No_Bedroom'].isnull()]
df = df.loc[df['No_Bedroom'] <= 4]

# remove rows that have Area not in range [15, 300]
df = df[~df['Area'].isnull()]
df = df.loc[df['Area'] <= 300]
df = df.loc[df['Area'] >= 15]
```

Finally, the dataset has **15282** rows

III. Feature Engineering

Feature engineering is an important step before building an ML model. Based on previous visualization as well as my knowledge, I will drop columns that have a large proportion of missing values or unuseful features.

```
df = df.drop(
    columns=['Phone_Number', 'Project', 'Floor', 'Raw_Description', 'Location', 'Url', 'Entities', 'Convenient', 'Furniture', 'No_Livingroom',
     'No_Bathroom'])
```

Label encoding

Encode categorical values into numerical ones.

```
df['Poster'] = df['Poster'].map(lambda x: x.strip(), na_action='ignore')
df['Poster'] = df['Poster'].map({'môi giới': 0, 'cá nhân': 1})
```

```
df['Apartment_Type'] = df['Apartment_Type'].map({'chung cư tập thể': 0, 'chung cư thường': 1, 'chung cư studio': 2, 'chung cư mini': 3, 'chung cư cao
 cấp': 4})
```

```
df['Furniture_Type'] = df['Furniture_Type'].map({'unk': 0, 'full': 1, 'cơ bản': 2, 'full cao cấp': 3, 'nguyên bản': 4})
```

```
df.head()
```

Split train, val, test set

The proportion of train, val, and test sets are 0.7, 0.1, and 0.2, respectively.

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

```
df_train, df_test = train_test_split(df, test_size=0.2, random_state=42)
df_train, df_val = train_test_split(df_train, test_size=0.1, random_state=42)
```

```
len(df_train), len(df_val), len(df_test)
```

```
(11074, 1231, 3077)
```

IV. Model Building

The decision tree and random forest will be selected to build the model due to explainability. Firstly, I experiment with the base decision tree and random forest model without any hyperparameter tuning.

```
decision_tree = DecisionTreeRegressor()
```

```
decision_tree.fit(X_train, y_train)
```

```
random_forest = RandomForestRegressor()
random_forest.fit(X_train, y_train)
```

Hyperparameters tuning

I define ranges or list values of hyperparameters, then I use random search to find the best set of hyperparameters.

The Random Forest algorithm will be used to build the model.

```python
import numpy as np

n_estimators = np.arange(20, 150, step=10)
max_features = ["auto", "sqrt", "log2"]
max_depth = list(np.arange(10, 100, step=10)) + [None]
min_samples_split = np.arange(2, 10, step=2)
min_samples_leaf = np.arange(2, 10, step=2)
bootstrap = [True, False]

param_grid = {
    "n_estimators": n_estimators,
    "max_features": max_features,
    "max_depth": max_depth,
    "min_samples_split": min_samples_split,
    "min_samples_leaf": min_samples_leaf,
    "bootstrap": bootstrap,
}
```

```python
forest = RandomForestRegressor()

random_cv = RandomizedSearchCV(
    forest, param_grid, n_iter=80, cv=5, scoring="neg_mean_squared_error", n_jobs=-1, verbose=1
)
```

V. Results

RMSE of models in the validation set

Algorithm	Hyperparameter tuning	RMSE
Decision Tree	No	3.0623
Random Forest	No	2.3014
Random Forest	Yes	**2.2805**

The above table is the experimental results of models on the validation set. It demonstrates that hyperparameter tuning will improve the result slightly and that the random forest produces better results than the decision tree.

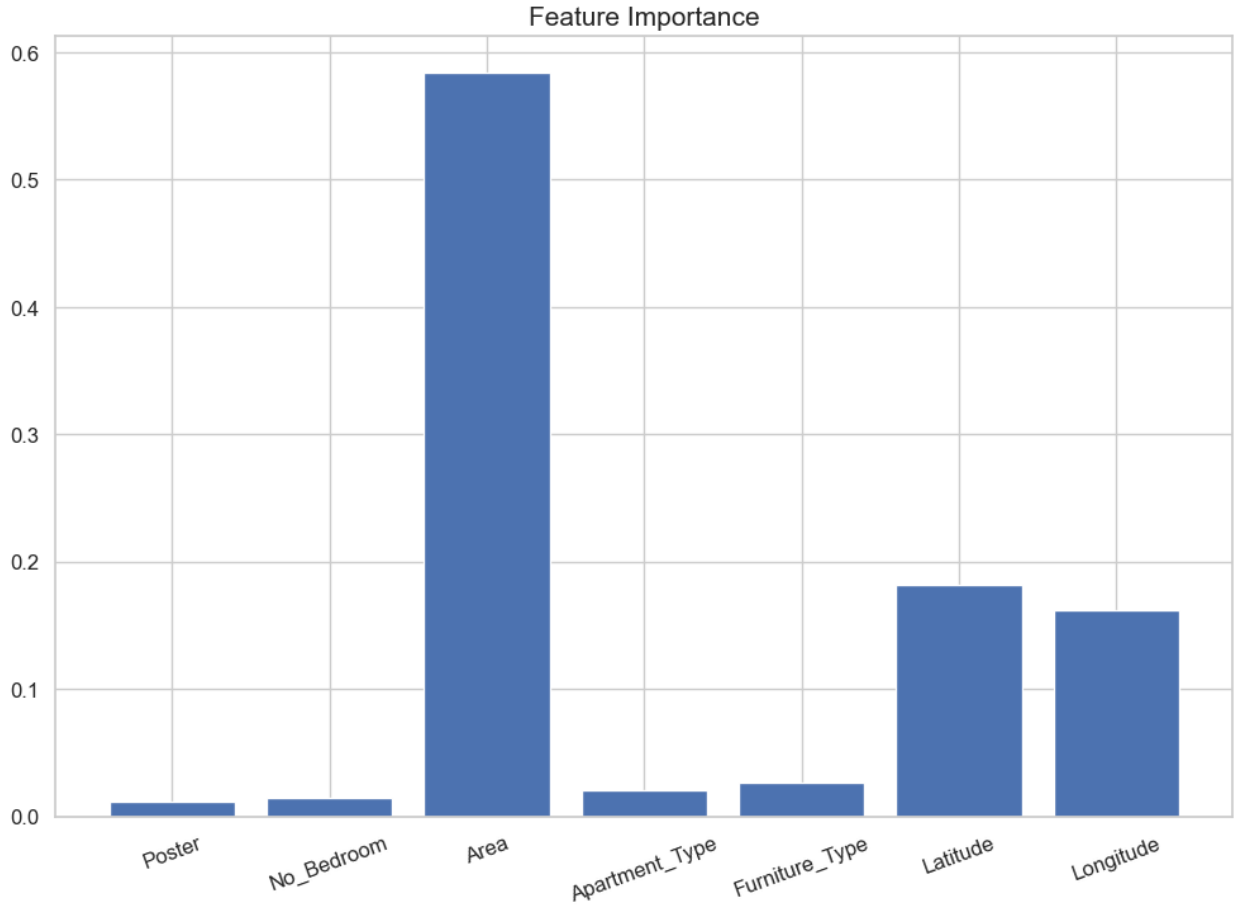
The best model gives a result of 2.6557 RMSE on the test set.

```python
y_pred = model.predict(X_test)
print("Root MSE on the test set: {:.4f}".format(mean_squared_error(y_test, y_pred) ** (1 / 2)))
```

```
Root MSE on the test set: 2.6557
```

I think it is an acceptable result.

Feature Importance



The figure shows that area feature is the most important feature which affects the price of rental apartments in Hanoi.

VI. Conclusion & Improvements

In the project, I defined the rental apartment problem in Hanoi and built a predictive model that can predict the rental price of an apartment based on several features: Poster, No_Bedroom, Area, Apartment_Type, Furniture_Type, Latitude, Longitude. The best model gives a result of 2.6557 RMSE on the test set. I also find out the area feature is the most important feature that affects the price of rental apartments.

During the AWS Machine Learning Course, I have learnt a lot of knowledge about the machine learning lifecycle as well as several AI-related services of AWS. It will be useful knowledge for my career path.

Improvements

Apply MLOps concepts to deliver models faster in production and make models more stable as well as immediately re-train models if the model's accuracy decrease.