Institute of Technology of Cambodia

Department of Applied Mathematics and Statistics (AMS)

Report paper: Condo for Sale in Cambodia

Lecturer: Mr. CHAN Sophal

Subject: Programming for Data Science

Group's members	ID
SAO Samarth	e20200084
SET Sophy	e20201576
SANG Rithpork	e20200232
SOK Sreyseey	e20201226
SONG Phalla	e20200439

2022~2023

Contents

Abstract

This project investigates the use of machine learning models to predict the price of condos for sale in Cambodia.

1. Scraping Data

Our dataset is collected from Khmer 24 website by using web scraping technique. To scrape data about **Condo for Sale in Cambodia** from Khmer 24 website, first we go to the Khmer 24 website and search for 'Condo for Sale in Cambodia', and then find the page that lists all the condos for sale. Second, we use the 'Inspect element' tool to view the HTML code for the page and then identify the elements that contain the data we want to scrape, such as the condo's price, location, number of bedrooms, number of bathrooms, posted time, size. Since data in this website is unstructured, meaning that it is not in a structured format. So, we use **regular expressions technique** to extract specific pieces of data, such as extract string text for some feature like feature Sub Location, Bedrooms, Bathrooms, and Floor from the HTML code in order to extract information that we want.

2. Data Preprocessing

2.1. Data Description

Dataset obtains 1547 rows and 8 columns. There exist 8 features.

```
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Locations     1547 non-null   object
 1   Posted        1547 non-null   object
 2   Size(m2)      1547 non-null   int64
 3   Price         1547 non-null   object
 4   Sub Location  616 non-null    object
 5   Bedrooms      781 non-null    float64
 6   Bathrooms     522 non-null    float64
 7   Floor         683 non-null    float64
dtypes: float64(3), int64(1), object(4)
```

We will provide a detailed overview of each feature that effect to condo prediction price.

- Locations or Sub Location: This feature indicates the location of the condo. It can be a specific address, a Khan or a district.
- Posted: This variable refers the date and time when the condo was listed for sale.
- Size (m^2): This feature indicates the size of the condo in square meters.
- Price: This feature indicates the price of the condo.

- Bedrooms: This feature indicates the number of bedrooms in the condo.
- Bathrooms: This feature indicates the number of bathrooms in the condo
- Floor: This feature indicates the floor number of the condo.

3. Missing Values and Duplicate Values

• Missing Values

We observed the dataset by checking missing value. There are some missing values in our dataset, such as the column Sub Location has 60.181% missing values, the column Bedrooms has 49.515% missing values, the column Bathrooms has 66.257% missing values, and the last column Floor has 55.85% missing values.

```
Locations          0
Posted             0
Size(m2)           0
Price              0
Sub Location     931
Bedrooms         766
Bathrooms       1025
Floor            864
dtype: int64
```

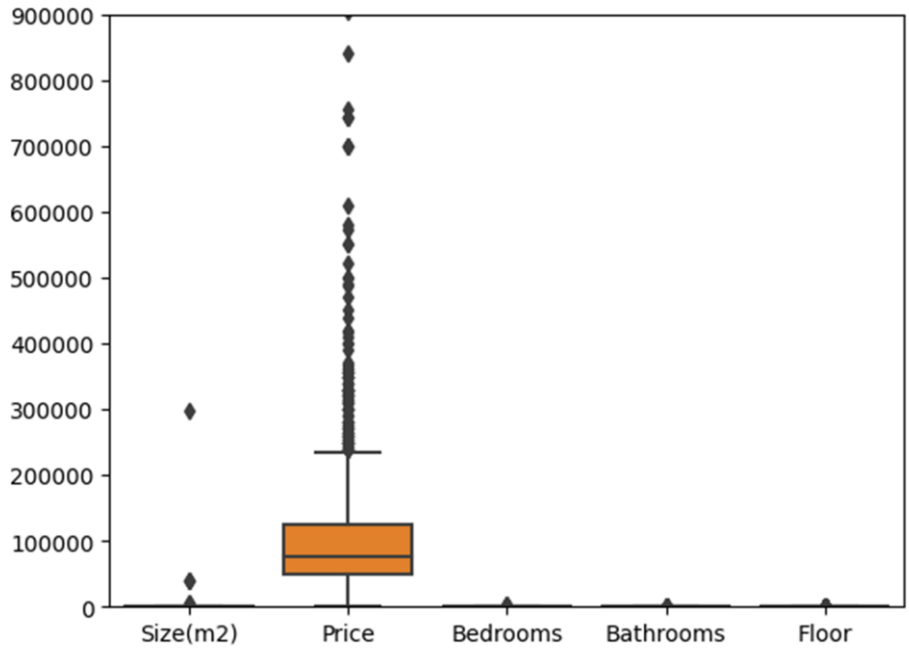
```
Sub Location 60.181 % Missing Values
Bedrooms 49.515 % Missing Values
Bathrooms 66.257 % Missing Values
Floor 55.85 % Missing Values
```

Since there are many missing values for some features, so we don't decided to delete, but we decided to fill all missing values with KNN algorithm for numerical feature like Bedrooms, Bathrooms, and Floor and for categorical feature Sub Location we fill with checking Location which Sub Location in Location that has a lot of condos.

• Duplicate Values

Since there are not many duplicate values, removing them completely will not affect our data set or further interpretation of our models.

4. Outliers

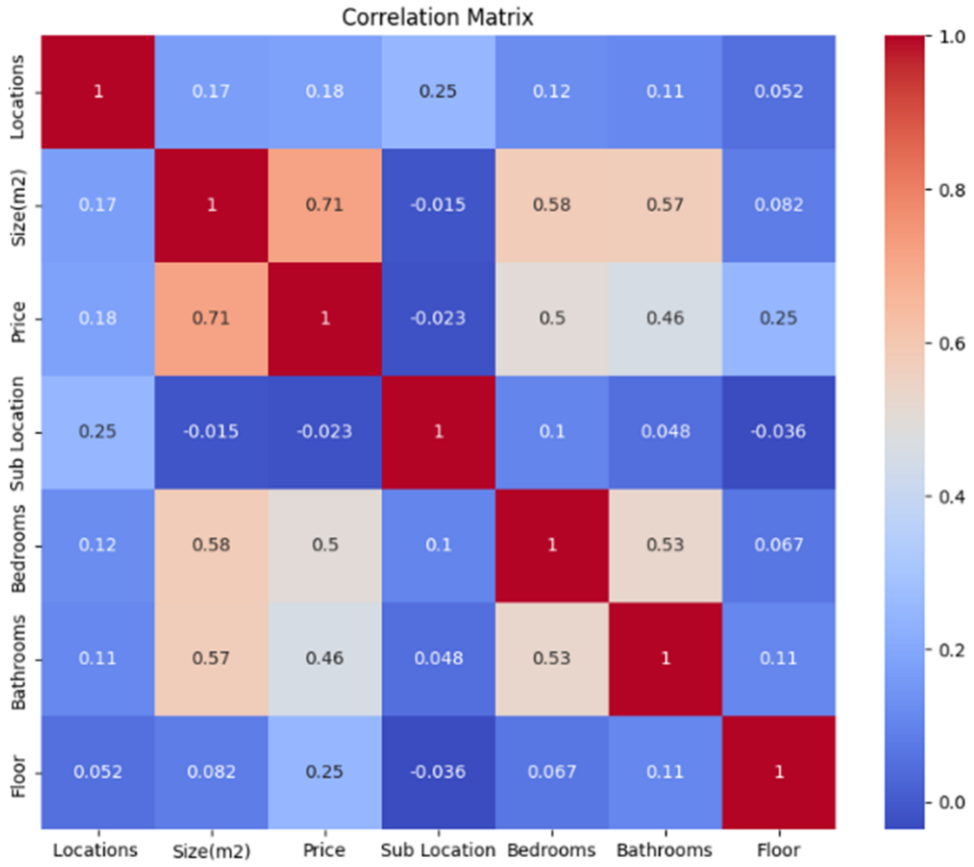


We chose to remove outliers as we have an abundance of data to spare.

5. Exploratory Data Analysis

5.1. Correlation Analysis

Since we observed the correlation matrix we see that the feature Size (m^2) is highly correlated with the feature Price. And the feature Bedrooms, Bathrooms, Floor is good correlated with the feature Price too.



5.2. Univariate Analysis

In this section, we will plot each feature with respect to its frequency. Pie charts have been selected as the plotting method. Here are some figures.

6. Feature Engineering

6.1. Feature Selection

We use Random Forest selection method to select important feature.

```
from sklearn.feature_selection import SelectFromModel
from sklearn.ensemble import RandomForestRegressor
from catboost import CatBoostRegressor

X = df_new3[['Locations ', 'Size(m2)','Sub Location', 'Bedrooms',
       'Bathrooms', 'Floor']]
y = df_new3['Price']

model = RandomForestRegressor()
selector = SelectFromModel(model)
X_new = selector.fit_transform(X, y)

selected_features = X.columns[selector.get_support()]
correlation_selection = df_new3[selected_features.tolist() + ['Price']].corr()
print(correlation_selection)
```

```
          Size(m2)     Floor     Price
Size(m2)  1.000000  0.081549  0.713162
Floor     0.081549  1.000000  0.248376
Price     0.713162  0.248376  1.000000
```

And then we do dummy method and scale data to select model to train and testing.

7. Model Building

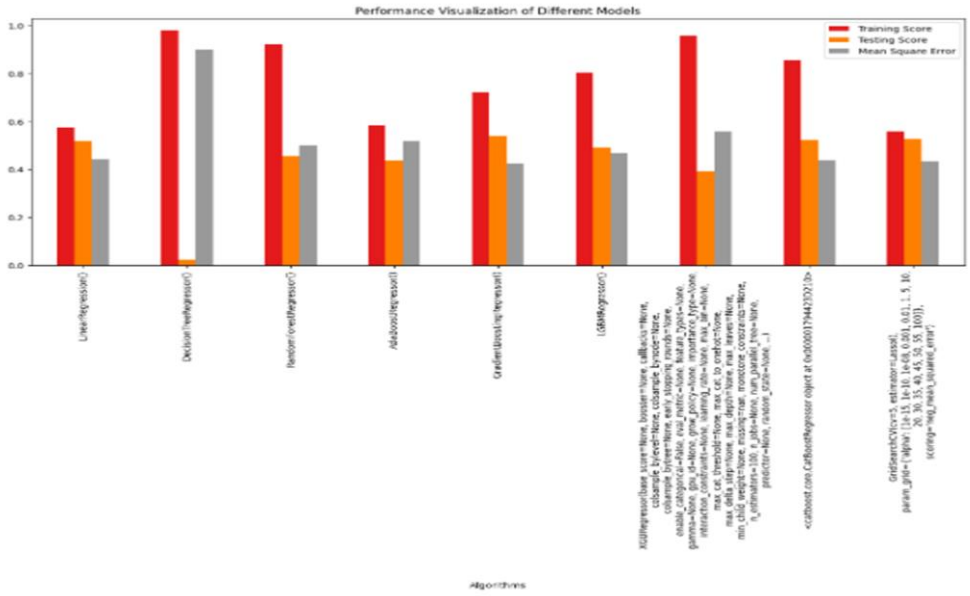
We use 9 models in model building sections:

1. Linear-Regression Model
2. Decision-Tree-Regressor Model
3. Random-Forest-Regressor Model
4. Ada-Boost-Regressor Model
5. Gradient-Boosting-Regressor Model
6. LGMB Regressor Model
7. XGB-Regressor Model
8. CatBoost-Regressor Model
9. Lasso-Regressor Model

8. Results

All models performance comparison.

	Algorithms	Training Score	Testing Score	Mean Square Error
0	LinearRegression()	0.574740	0.518607	0.444076
1	DecisionTreeRegressor()	0.980405	0.025413	0.899039
2	RandomForestRegressor()	0.922320	0.457235	0.500692
3	AdaBoostRegressor()	0.583103	0.437667	0.518743
4	GradientBoostingRegressor()	0.721175	0.539468	0.424833
5	LGBMRegressor()	0.803285	0.492251	0.468390
6	XGBRegressor(base_score=None, booster=None, ca...	0.957994	0.393075	0.559878
7	<catboost.core.CatBoostRegressor object at 0x0...	0.856320	0.524492	0.438648
8	GridSearchCV(cv=5, estimator=Lasso(),\n ...	0.559526	0.528684	0.434781



9. Conclusion

Considering the performance all models above, the best model to choose depends on the specific requirements and priorities.

If the emphasis is on achieving the highest testing score, the Gradient Boosting Regressor (0.5395) or CatBoost Regressor (0.5245) might be good options.

If reducing mean square error is a priority, the Gradient Boosting Regressor (0.4248) or Lasso (0.4348) could be considered.

It's important to note that these conclusions are based solely on the provided performance metrics and there might be additional factors to consider, such as computational efficiency, interpretability, and the specific requirements of the application.