RAYAT SHIKSHAN SANSTHA'S

SADGURU GADGE MAHARAJ COLLEGE, KARAD

(An Autonomous College)

DEPARTMENT OF STATISTICS

PROJECT TITLE

"House Price Prediction Using Machine Learning"

Submitted by,

Mr. Shambho Satish Nangare

(M.Sc. II)

Roll no: 23

Project Guide,

Mrs. A.S.Patil

Department of statistics
Sadguru Gadge Maharaj College, Karad.

CERTIFICATE

This is to certified that the case study report entitled “House Price Prediction Using Machine Learning” being submitted by Mr. Shambho Satish Nangare as partial fulfilment for the award of degree of M.Sc. II in statistics of Sadguru Gadge Maharaj College, Karad record of benefited work carried out by her supervision and guidance.

To the best of my knowledge and belief, the matter presented in the project report is original and has not been submitted elsewhere for any other purpose.

Place: Karad

Date:

Sr.no	Roll. No	Name	Signature
1	23	Shambho Satish Nangare	

Teacher In-Charge   Examiner   PG Co-ordinate   Head

Department of Statistics

ACKNOWLEDGEMENT

I have great pleasure in presenting this report of successful completion of my project viz. House Price Forecasting Using Machine Learning.

I take this opportunity to express my great sense of gratitude to my Guide Mrs.A.S.Patil of Statistics Department, S.G.M. College, Karad. For granting me permission to undertake this project for their constant encouragement, guidance and inspiration without which I could not have completed this task.

I would like to extend my sincere thanks to Mrs. Mahajan S.V. (Head Department of Statistics), Dr. Mrs. Patil S.P. and Mrs.A.S.Patil for their guidance and kind appreciation in this project study.

Your sincerely,

Nangare Shambho Satish

M.Sc.-II

Department of statistics

INDEX

INTRODUCTION

Real Estate Property is not only the basic need of a man but today it also represents the richness and prestige of a person. Investment in the Real Estate generally seems to be profitable because their property values do not decline rapidly. Changes in House price can affect various household investors, bankers, policy makers and many. Sales forecasting has always been a very significant area to concentrate upon. Manual infestation of being able to predict House Prices could lead to drastic errors leading to poor management of the organization and most importantly would be time consuming, which is something not desirable in today's expedited world. A major part of the global economy relies upon the business sectors, which are expected to produce appropriate quantities of products to meet the overall needs. The forecasting process can be used for many purposes, including: predicting the future demand of the products or service and predicting how much of the product will be sold in a given period.

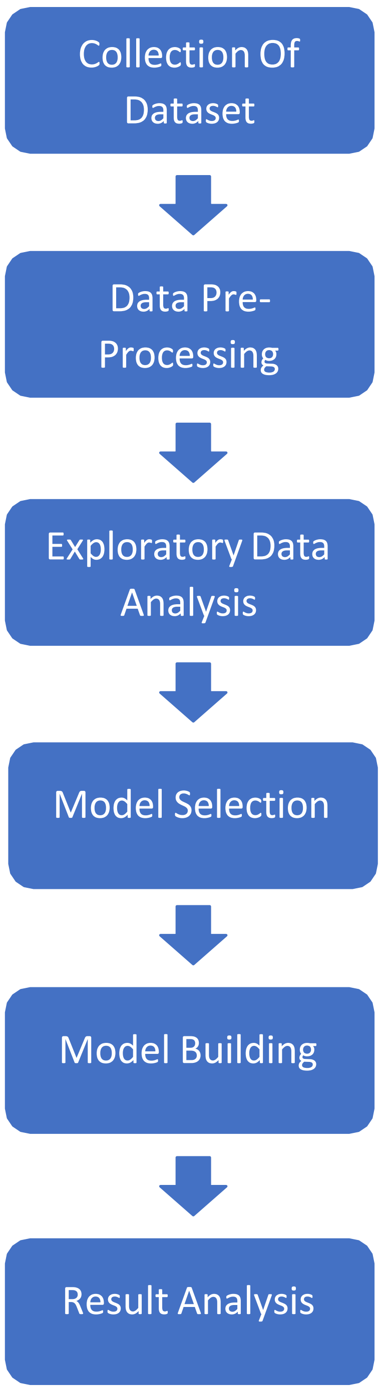
In our paper we have proposed the machine learning algorithms towards the data collected across various property aggregators across India. The objective here is to predict the price of Houses in India using three different algorithms and then comparing them to see which one gives a more accurate result based on some key features gathered from the raw data we have. Accurately predicting house prices can be a daunting task. Analysis and exploration of the collected data have also been done to gain a complete insight of the data. Analysis of the data would help the business organizations to make a probabilistic decision at each important stage of marketing strategy.

OBJECTIVE OF THE STUDY

- Construct classification models for data about House price.
- To provide proper house price to the customers.
- To eliminate need of real estate agent to gain information regarding house prices.
- To investigate the effectiveness of different machine learning algorithm to classify House prices and compare their performance.

MEHODOLOGY

The methodology of predicting the price of House proceeds by obtaining & preprocessing the dataset, then applying various algorithms/regression techniques to find out the best suitable algorithm for the project.



TOOLS AND TECHNIQUE USED

- Techniques :-

  Machine learning classification model
  1) Decision Tree.
  2) K-Nearest Neighbours.
  3) Random Forest.

- Tools :-

  Microsoft Excel.
  Python (Jupyter Notebook).

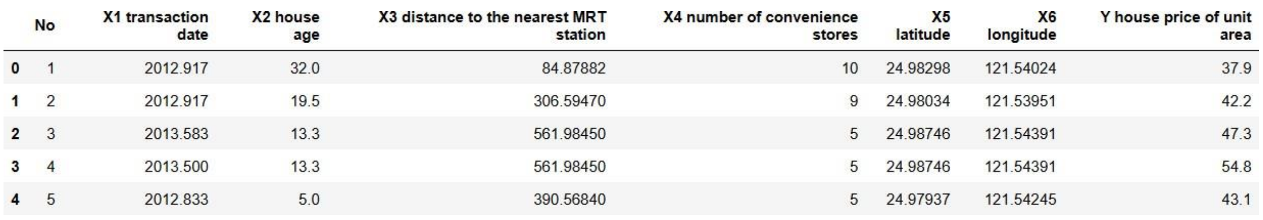
DATA DESCRIPTION

We have collected the secondary data from the website Kaggle which was raw and needed to be cleaned.

- Understanding the data:-

  - No - Transaction number identifier (record index).
  - X1 Transaction date - Date of the transaction, expressed in the number of months.
  - X2 House age - Age of the house in years.
  - X3 Distance to the nearest MRT station - Distance to the nearest subway station in meters.
  - X4 Number of convenience stores - Number of convenience stores nearby.
  - X5 Latitude - Latitude where the property is located.
  - X6 longitude – Longitude at which the property is located.
  - Y House price of unit area – House price per unit area, measured per ping (unit area equivalent to approximately 3.3 square meters).

- The following table shows the first five rows from the dataset.



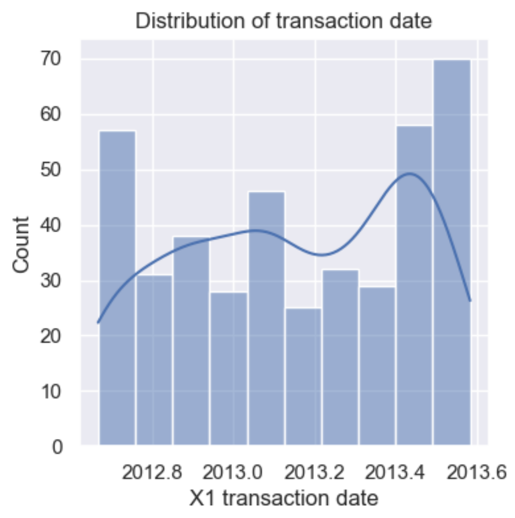
	No	X1 transaction date	X2 house age	X3 distance to the nearest MRT station	X4 number of convenience stores	X5 latitude	X6 longitude	Y house price of unit area
0	1	2012.917	32.0	84.87882	10	24.98298	121.54024	37.9
1	2	2012.917	19.5	306.59470	9	24.98034	121.53951	42.2
2	3	2013.583	13.3	561.98450	5	24.98746	121.54391	47.3
3	4	2013.500	13.3	561.98450	5	24.98746	121.54391	54.8
4	5	2012.833	5.0	390.56840	5	24.97937	121.54245	43.1

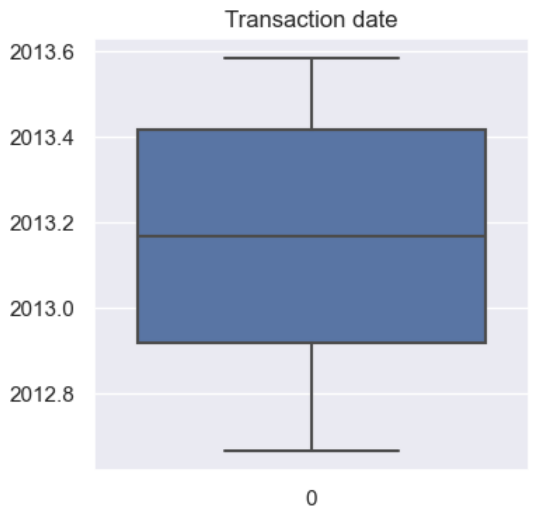
EXPLORATORY DATA ANALYSIS

Before doing actual analysis and model building, we will do Exploratory Data Analysis (EDA) to analyse the data using visual techniques. It is used to discover trends, patterns, or to check assumptions with the help of statistical summary and graphical representations.

➢ Univariate Analysis

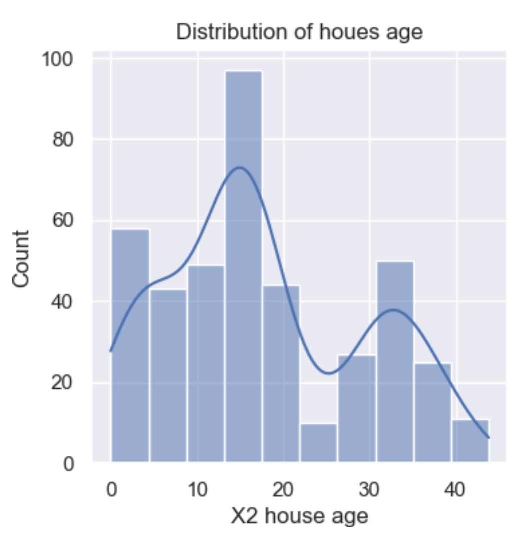
- Transaction Date

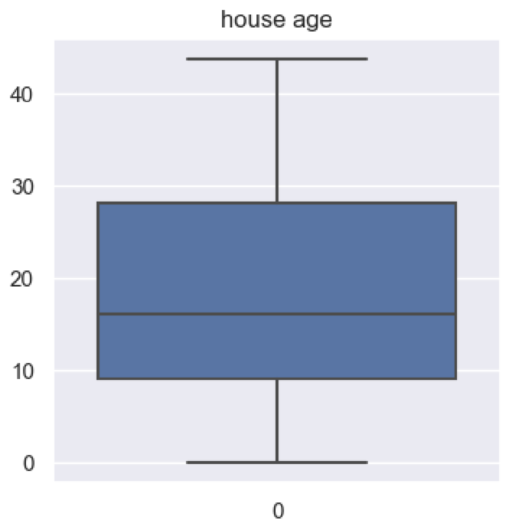




The Boxplot shows that there are no outliers present in transaction date column.

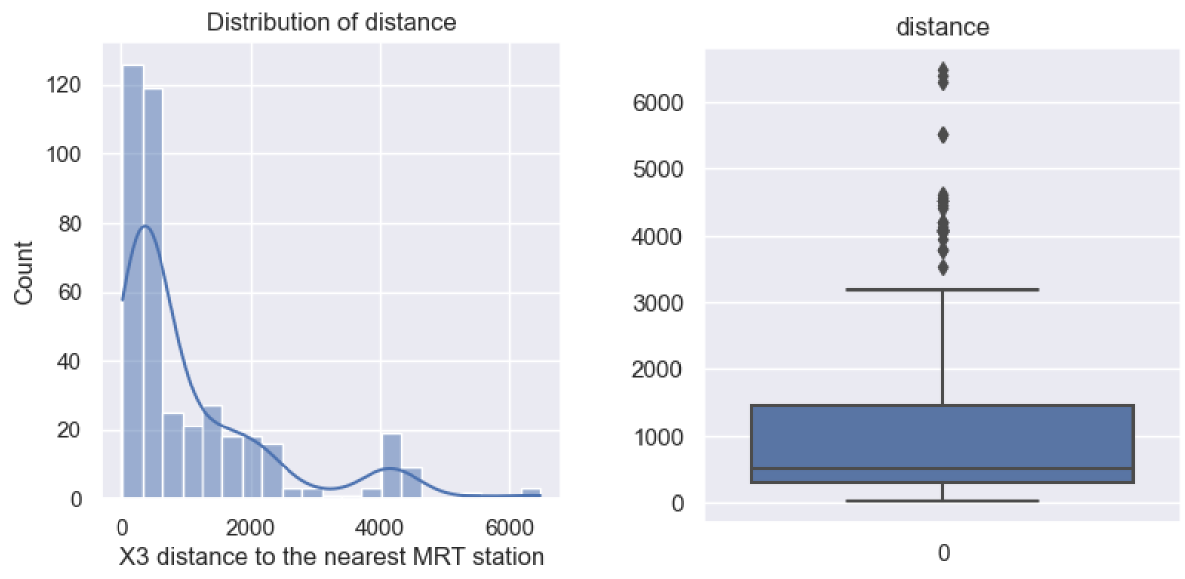
- House Age





The above boxplot shows that there are no outliers present in house age column.

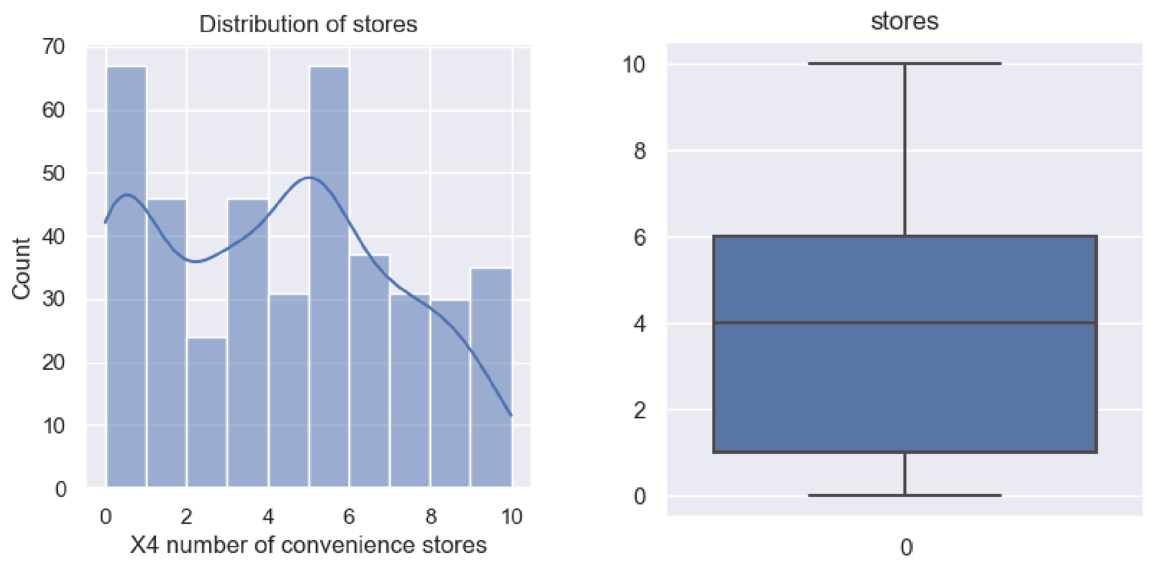
- Distance to the nearest MRT station



The above graph shows that the distribution appears left-skewed.

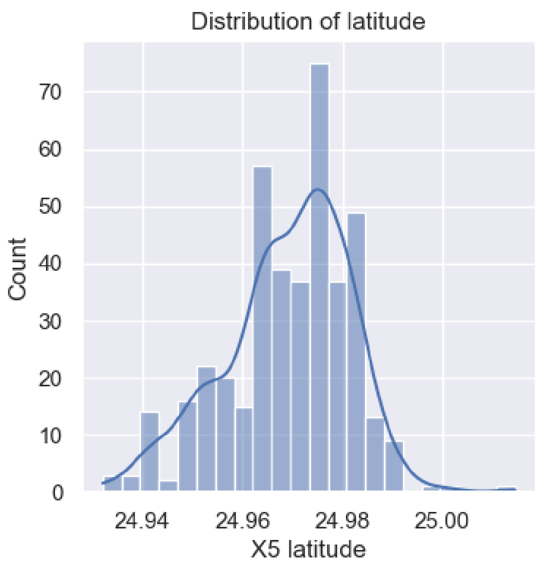
The above boxplot shows that there are some outliers presents in the distance to the nearest MRT station column.

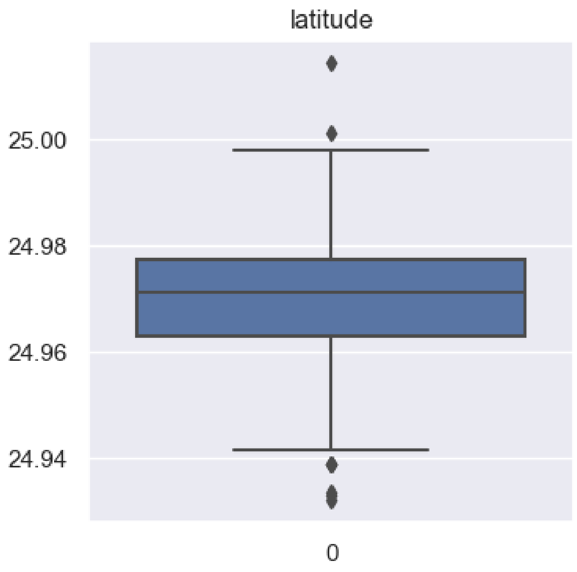
- Number of convenience stores



The above boxplot shows that there are no outliers present in number of convenience stores column.

- Latitude

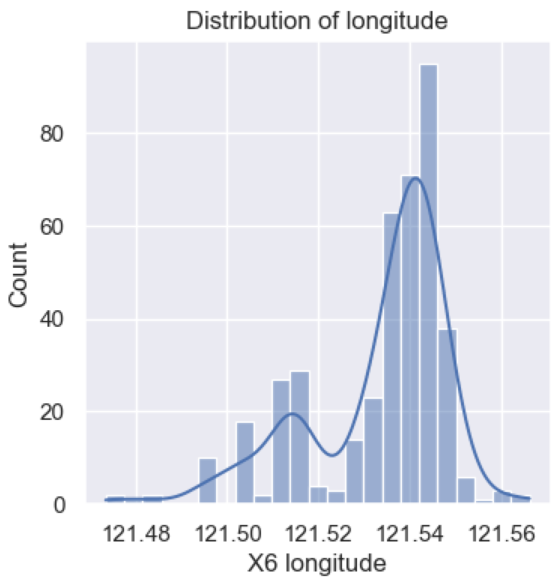


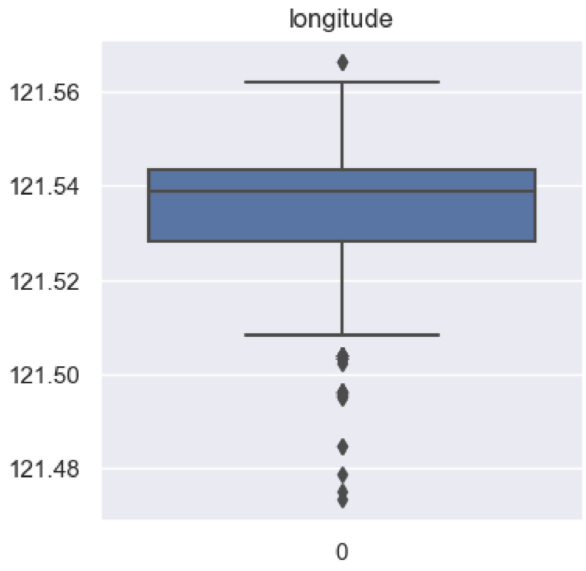


This graph indicates the latitude data follows a roughly normal distribution, though there may be slight skew or irregularity in the shape.

The above boxplot shows that there are some outliers present in the latitude column.

- Longitude



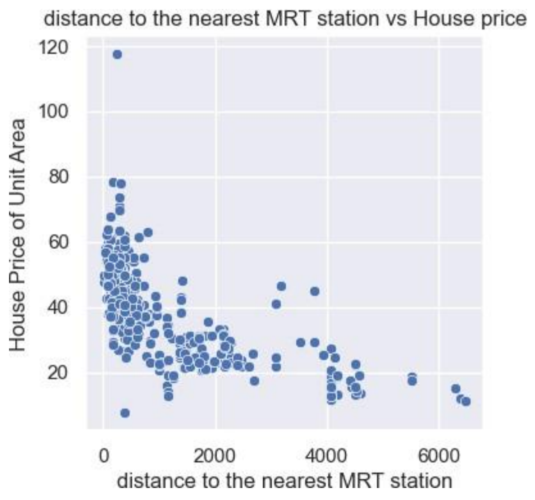


The above graph shows that the distribution appears right-skewed.

The above boxplot shows that there are some outliers present in longitude column.

➢ Bivariate Analysis

➢ Distance to nearest MRT station vs House price



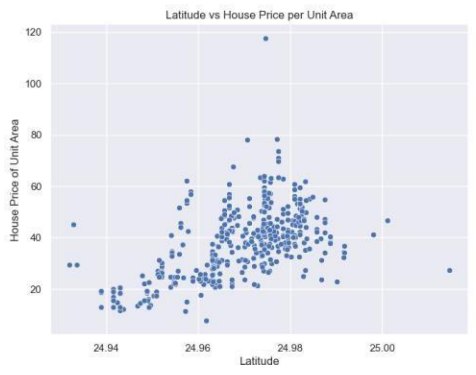
As the distance from the nearest MRT station increases, the house price tends to decreases.

Properties closer to the MRT station (within the first 1000 meters) generally have much higher prices, indicating that proximately to public transportation significantly boosts property value.

Beyond 2000-3000 meters, the house prices stabilize at lower levels, with fewer high-price outliers.

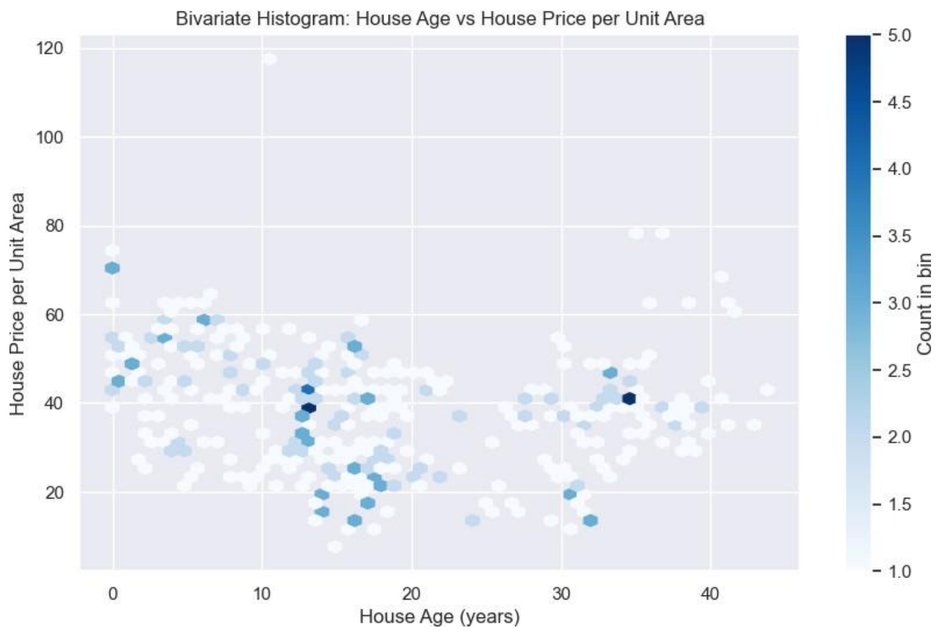
This suggests that access to public transport, particularly MRT stations, is a key factor influencing property prices.

➢ Longitude vs House price



The plot suggest that longitude has a localized impact on house prices, with certain regions (likely between 121.52 and 121.54) having higher concentrations of mid to high-priced properties. However, there are also outliers with extremely high prices, possibly indicating luxury properties in specific locations.

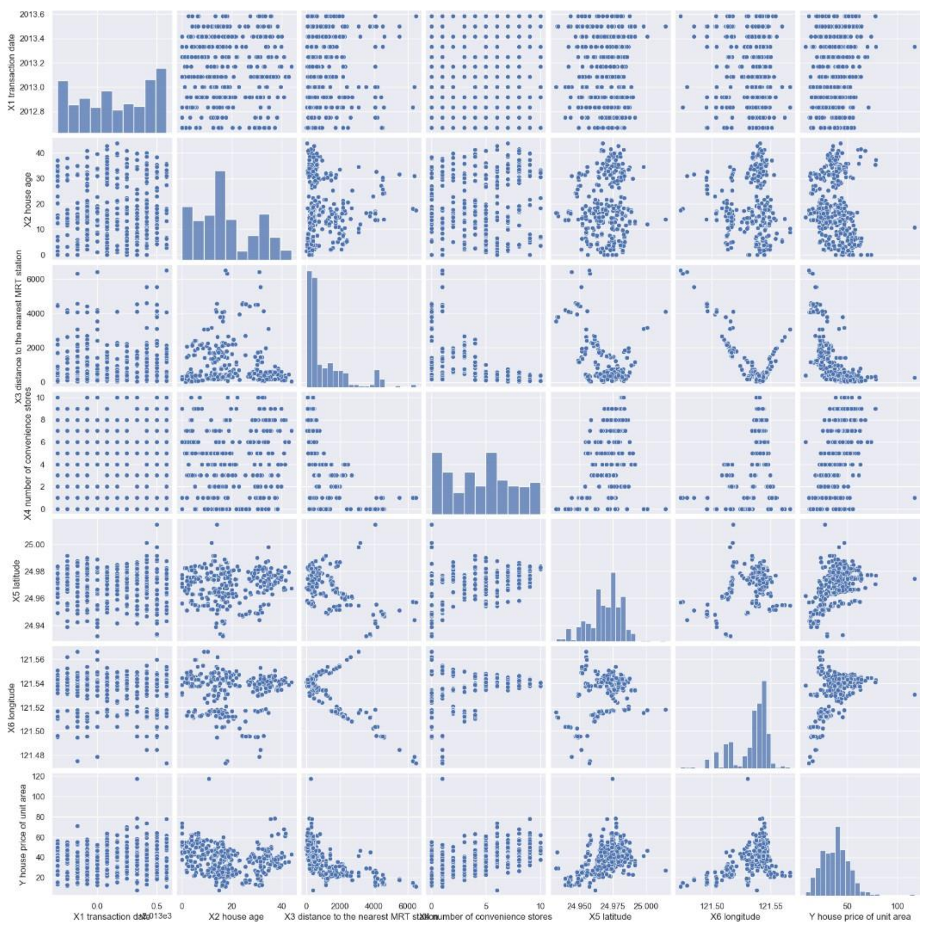
➢ House age vs House price



For older houses (above 30 years), the prices per unit area seem to hover between 20 and 50, showing the possible stabilization in price over time.

This plot suggests that house age might not be a dominant predictor of price per unit area, but houses in the 5-15 years range seem to dominant the market in terms of quantity. Additional factors (like location, amenities or demand) may also be influencing house prices.

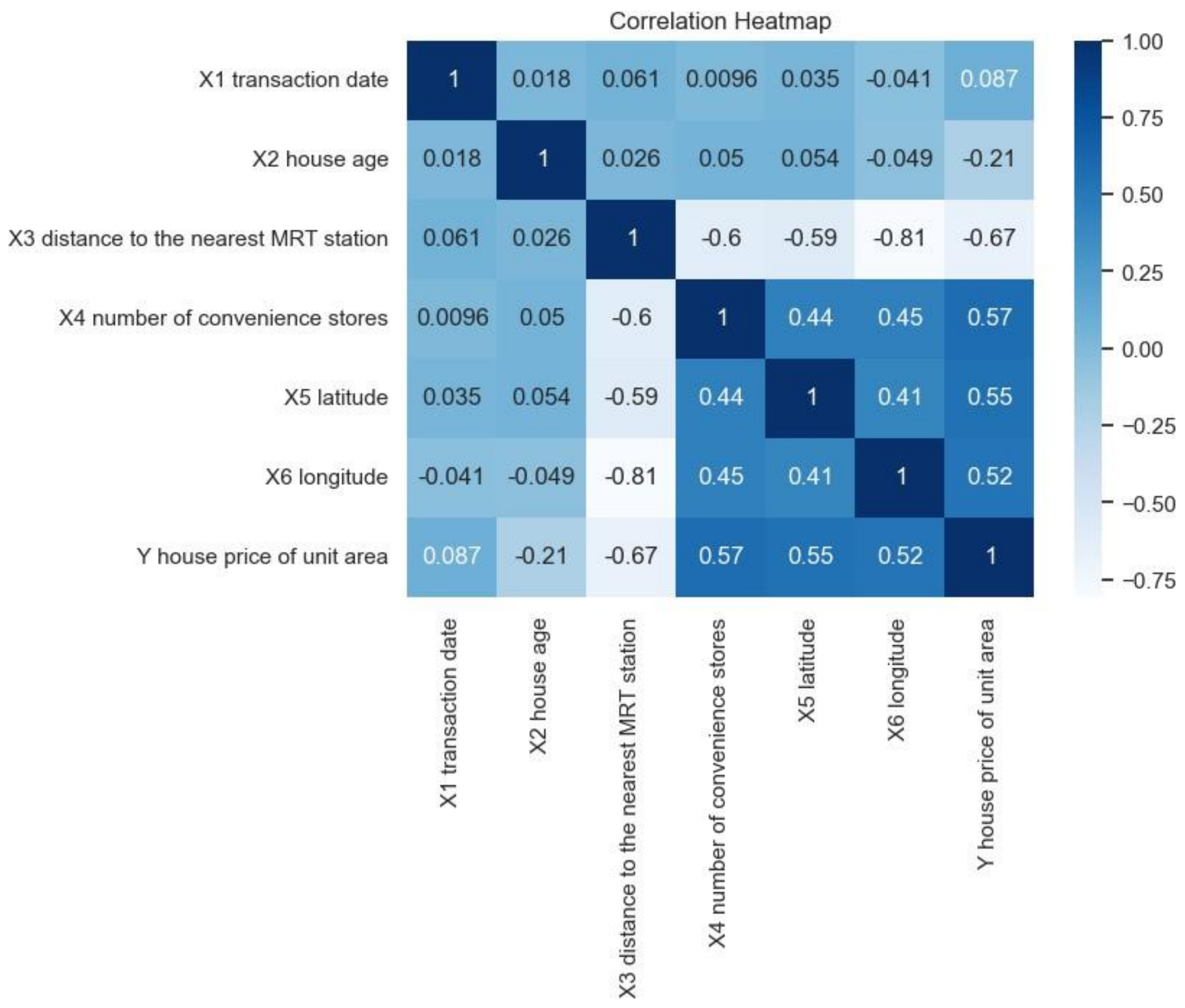
We use the seaborn library to visualize the data through a pairwise plot and identify which variables best fit our model.



We identified that the model does not fit directly to the linear regression model, however we can already see variables that are useful to us such as: Latitude and Longitude.

We then use the heat map from the seaborn library to identify the relationship we have in our variables.

➢ Correlation plot



We identified that the variables with which it has the best relationship are: number of convenience stores, latitude and longitude.

However, we not that in order to calculate the price of the house, variables such as: age of the house and distance to the nearest subway station are also important: the age of the house and the distance to the nearest subway station.

TRAIN – TEST SPLIT

The train-test split is a technique for evaluating the performance of a machine learning algorithm. It is used for classification or regression problems for any Supervised machine Learning Algorithm. The procedure involves taking a dataset and dividing it into two subsets. The first one is the Training set which is used to fit the model and the second is not used to train but to make predictions known as the testing set.

- Train Dataset: It is used to fit the machine learning model.
- Test Dataset: It is used to evaluate the fitted machine learning model.
- Using train_test_split, the data is split into training and validation sets (x_train, x_test, y_test) with a test size of 20% and a random seed of 42.

MODEL BUIELDING

Model building is a pivotal phase in any data-driven project, where theoretical constructs meet practical application. It involves the creation and refinement of mathematical or computational representations that capture patterns and relationships within the data. This process transforms raw data into actionable insights, guiding decision-making and predicting future outcomes. Through meticulous experimentation and validation, models evolve, adapting to the intricacies of the data and the nuances of the problem at hand. From simple linear regressions to complex deep learning architectures, each model is crafted with precision, aiming to uncover hidden insights and empower organizations to make informed decisions. In this journey from data to knowledge, model building serves as a beacon of intelligence, illuminating the path towards understanding and discovery.

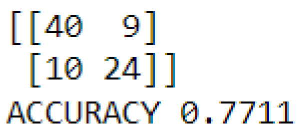
The models that are used for prediction are:

- Decision Tree
- Random Forest
- k-Nearest Neighbours

➢ Decision Tree

Decision trees are powerful and intuitive models widely used in machine learning for both classification and regression tasks. At their core, decision trees mimic human decision-making processes by recursively partitioning the data into subsets based on the value of predictor variables. Each split in the tree represents a decision point, where the algorithm selects the feature that best separates the data into homogeneous groups. This hierarchical structure results in a tree-like model that is easy to interpret and visualize, making decision trees invaluable for gaining insights into the underlying patterns in the data. With their simplicity, flexibility, and interpretability, decision trees are a foundational tool in the machine learning toolbox, empowering analysts and data scientists to uncover actionable insights and make informed decisions.

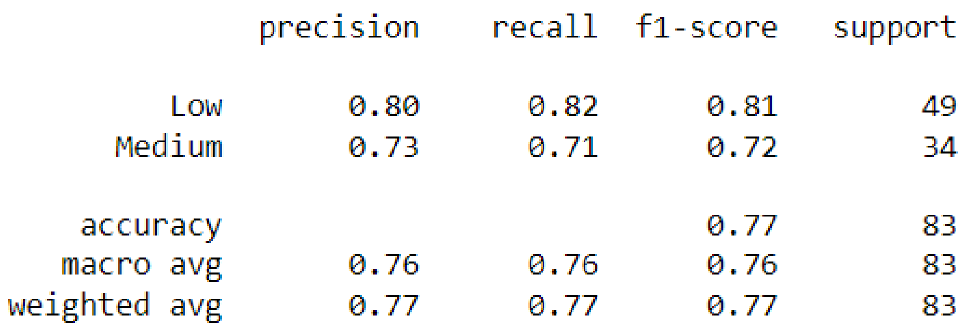
Accuracy of model: 77.11%
Confusion matrix:



```
[[40  9]
 [10 24]]
ACCURACY 0.7711
```

□ **True Positive (TP)** = 24 (bottom-right cell): Correctly predicted positive class
□ **False Positive (FP)** = 9 (top-right cell): Incorrectly predicted positive class
□ **False Negative (FN)** = 10 (bottom-left cell): Incorrectly predicted negative class
□ **True Negative (TN)** = 40 (top-left cell): Correctly predicted negative class

Classification Report:



```
              precision    recall  f1-score   support

         Low       0.80      0.82      0.81        49
      Medium       0.73      0.71      0.72        34

    accuracy                           0.77        83
   macro avg       0.76      0.76      0.76        83
weighted avg       0.77      0.77      0.77        83
```

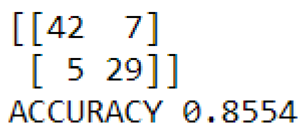
Conclusion:

- **Accuracy (77.11%)** is decent, indicating the model performs reasonably well overall, but it doesn't give a complete picture, especially if the classes are imbalanced.
- The **Precision** (72.7%) and **Recall** (70.6%) are relatively close to each other, meaning the model is fairly balanced in terms of both minimizing false positives and false negatives.
- The **F1-Score** of approximately 0.716 shows that the model has a fairly good balance between precision and recall.
- The **Specificity** of 81.6% indicates that the model is good at identifying negative cases, which is important in scenarios where false positives should be minimized.

➢ Random Forest

Random Forest is a versatile and powerful ensemble learning technique that leverages the collective wisdom of multiple decision trees to make robust predictions. By constructing a multitude of decision trees using random subsets of the training data and features, Random Forest mitigates overfitting and improves generalization performance. Each tree in the forest independently learns from a different subset of the data, making predictions based on a combination of the individual tree predictions. This ensemble approach results in a highly accurate and stable model that is resilient to noise and outliers. With its ability to handle high dimensional data, capture complex relationships, and provide feature importance rankings, Random Forest has become a go-to choice for a wide range of classification and regression tasks in both research and industry settings.

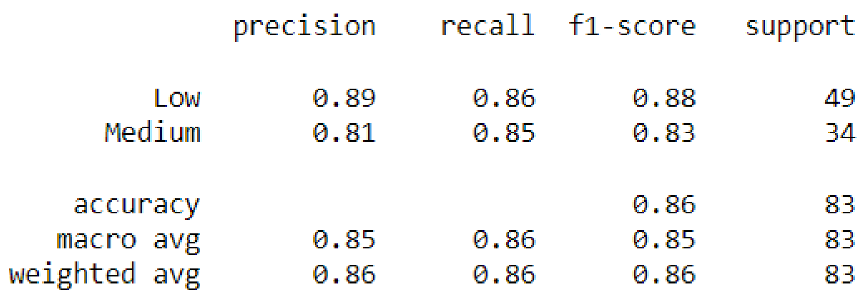
Accuracy of model: 85.54%
Confusion matrix:



```
[[42  7]
 [ 5 29]]
ACCURACY 0.8554
```

❒ **True Positive (TP)** = 29 (bottom-right cell): Correctly predicted positive class
❒ **False Positive (FP)** = 7 (top-right cell): Incorrectly predicted positive class
❒ **False Negative (FN)** = 5 (bottom-left cell): Incorrectly predicted negative class
❒ **True Negative (TN)** = 42 (top-left cell): Correctly predicted negative class

Classification report:



```
              precision    recall  f1-score   support

         Low       0.89      0.86      0.88        49
      Medium       0.81      0.85      0.83        34

    accuracy                           0.86        83
   macro avg       0.85      0.86      0.85        83
weighted avg       0.86      0.86      0.86        83
```

Conclusion:

- **Accuracy (85.4%)** is high, suggesting the model performs well overall, correctly classifying most instances.
- **Precision (80.6%)** is strong, meaning the model is fairly reliable when predicting positive cases.
- **Recall (85.3%)** is even better, indicating the model is very good at identifying positive cases and minimizing false negatives.
- **F1-Score (82.9%)** is balanced and shows a good trade-off between precision and recall.
- **Specificity (85.7%)** indicates the model also effectively identifies negative cases, with minimal false positives.

➢ K-Nearest Neighbours

k-Nearest Neighbours uses a distance metric to find the k most similar observations, in this case, use the model from the Scikit Learning Libraries that use the Minkowski distance. It is a generalization of the Euclidean distance and the Manhattan distance. The program "House price forecasting k-NN neighbours.py" in Python implement this algorithm follows the same methodological structure that previous models. The advantage of using this method in this project is there is no assumption about the distribution of the variables. Therefore, the skewness distribution of the variables would not affect the results of the models. However, it is affected by the outlier's observation in the dataset that was shown in the previous chapters. The challenge in this model is found the right values of k, a number of neighbours, that would use to train the model.

Accuracy of model: 81.93%
Confusion matrix:

```
[[42  7]
 [ 8 26]]
ACCURACY 0.8193
```

□ **True Positives (TP)** = 26: The number of instances that are actually positive and were correctly classified as positive.

□ **False Positives (FP)** = 7: The number of instances that are actually negative but were incorrectly classified as positive.

□ **False Negatives (FN)** = 8: The number of instances that are actually positive but were incorrectly classified as negative.

□ **True Negatives (TN)** = 42: The number of instances that are actually negative and were correctly classified as negative.

Classification report:

```
              precision    recall  f1-score   support

         Low       0.84      0.86      0.85        49
      Medium       0.79      0.76      0.78        34

    accuracy                           0.82        83
   macro avg       0.81      0.81      0.81        83
weighted avg       0.82      0.82      0.82        83
```

Conclusion:

- **Accuracy (81.93%)**: The model performs reasonably well overall, correctly classifying about **82%** of instances.
- **Precision (78.79%)**: When the model predicts a positive class, it's fairly reliable (**78.8%)** of those predictions are correct), but there is room for improvement, especially in reducing false positives.
- **Recall (76.47%)**: The model does a good job identifying positive instances, but it misses about **23.5%** of the positive cases. This could be a concern in cases where false negatives are costly (e.g., in medical diagnoses or fraud detection).
- **F1-Score (77.61%)**: The model strikes a good balance between precision and recall, which suggests it is doing a reasonable job in both avoiding false positives and false negatives.
- **Specificity (85.71%)**: The model is very good at correctly identifying negative instances, with very few false positives, which is often important when false positives are costly.

CONCLUSION

- The distribution of House age is skewed to the right, this suggests that the real estate market is primarily driven by newer properties, which are more commonly available. However, the presence of a long tail on the right side implies that there are a few older houses, which might represent outliers or special cases in the market.
- Here accuracy of model Decision Tree, Random Forest, K-NN are 77.11%, 85.54%, 81.91% respectively.
- Comparing accuracy of three models, we conclude that Random Forest model perform well with accuracy 85.54% which is approximately 86%

FUTURE SCOPE

The future scope of House price forecasting is poised for significant advancement, driven by a combination of emerging technologies, data science, and macroeconomic trends. As the House market becomes increasingly data-driven, several factors are expected to shape the future of price forecasting:

Deep Learning Models: Future forecasting tools will likely leverage deep learning algorithms that can analyse vast amounts of unstructured data like news articles, social media sentiment, and economic indicators to predict price trends with higher accuracy.

Real-Time Data Collection: With the rise of the Internet of Things (IoT) and smart cities, real-time data on traffic, infrastructure projects, local crime rates, and demographic trends will provide more granular insights for forecasting.

Decentralized Property Records: Blockchain can be used for transparent property transaction histories, making it easier to track price evolution, ownership transfers, and other factors influencing price forecasting.

Tailored Predictions for Individuals: AI-driven platforms could offer highly personalized insights for home buyers, investors, and developers. These tools might forecast price changes based on individual criteria, such as family size, lifestyle preferences, or investment goals.

Climate Change Predictions: As climate change increasingly affects the House market, forecasting models will need to incorporate environmental risks such as flooding, wildfires, or droughts that may affect property values.

The future of House price forecasting is likely to be characterized by greater precision, personalization, and real-time adaptability. By integrating advanced technologies like AI, machine learning, blockchain, and big data, the industry will evolve into a more dynamic and efficient market, providing more reliable predictions for investors, homebuyers, and developers alike. However, ensuring data quality and addressing ethical and security concerns will be key to harnessing the full potential of these advancements.

REFERENCES

- **"Data Science for House" by Sandeep M. V.**
- A great resource for understanding how data science methodologies, including machine learning and statistical analysis, can be applied to predict housing prices and evaluate the House market.

- **"Predicting House Prices with Machine Learning Algorithms"**
    - o **Authors**: Nathaniel J. Schwartz and Andrew F. Kandel.
    - o This paper explores the use of various machine learning algorithms, such as linear regression, random forests, and neural networks, for predicting housing prices.

- **"A Survey of Predictive Modelling for Housing Price Forecasting"**
    - o **Authors**: J. M. Gupta and R. S. Goh.
    - o This survey paper reviews various methods, including statistical techniques and machine learning approaches, for forecasting housing prices.

- Kaggle - Housing Price Prediction Datasets.

- UCI Machine Learning Repository - House Dataset.

- Towards Data Science Articles.

APPENDIX

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import matplotlib.style
%matplotlib inline
import seaborn as sns; sns.set() # for plot styling
from scipy import stats
plt.rcParams['figure.figsize']=[15,8]
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from sklearn.model_selection import RandomizedSearchCV
from scipy.cluster.hierarchy import dendrogram,linkage,fcluster
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
from sklearn.cluster import KMeans
from sklearn.tree import DecisionTreeRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.linear_model import LinearRegression, Lasso, Ridge
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn import metrics,model_selection
from sklearn.linear_model import LogisticRegression
from sklearn import tree
from sklearn.linear_model import LinearRegression,Ridge,Lasso
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import RandomForestClassifier
from sklearn.neural_network import MLPRegressor
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
import warnings
warnings.filterwarnings('ignore')
```

Loading Dataset

```
df = pd.read_csv('final project dataset.csv')

#Exploratory data analysis

df.head()

df.tail()
```

If The dataset contain 'NaN', 'N/A', 'NA', 'n/a', 'n.a.', 'N#A', 'n#a', '?' values .We need to replace with Null.

```
df.replace(['NaN', 'N/A', 'NA', 'n/a', 'n.a.', 'N#A', 'n#a', '?' ],'other',inplace=True)

df.info()

df.shape

df.drop(columns=['No'],inplace=True)
```

summary statistics of numeric variable:

```
df.describe().round(2).T
```

summary statistics for all variables:

```
df.describe(include="all").round(2).T

numeric_vars = df.select_dtypes(include=['int64','float64']).columns.tolist()
categorical_vars = df.select_dtypes(include=['object']).columns.tolist()
```

```
print('numeric variables: ',numeric_vars)
print('categorical variables: ',categorical_vars)

numeric variables:  ['X1 transaction date', 'X2 house age', 'X3 distance to the nearest MRT station',
'X4 number of convenience stores', 'X5 latitude', 'X6 longitude', 'Y house price of unit area']
categorical variables:  []

# Count the number of categorical and numerical variables
categorical_count = df.select_dtypes(include='object').shape[1]
numerical_count = df.select_dtypes(exclude='object').shape[1]

print(f"Number of categorical variables: {categorical_count}")
print(f"Number of numerical variables: {numerical_count}")

Number of categorical variables: 0
Number of numerical variables: 7

# Handling missing values
# Imputing missing values with the mean for continuous variables and mode for categorical variables
for col in df.columns:
    if df[col].dtype == 'object':
        df[col].fillna(df[col].mode()[0], inplace=True)
    else:
        df[col].fillna(df[col].mean(),  inplace=True)

# Checking for missing values before imputation
missing_values = df.isnull().sum()

# Rechecking for missing values after imputation
missing_values_after = df.isnull().sum()

missing_df =  df.isnull().sum().to_frame().rename(columns={0:"Total No. of Missing Values"})
missing_df["% of Missing Values"] = round((missing_df["Total No. of Missing Values"]/len( df))*100,2)
missing_df
```

UNIVARIATE ANALYSIS

Distribution of variables

```
plt.figure(figsize=[4,4
                   ])
sns.histplot(df['X1 transaction date'],kde=True)
plt.title('Distribution of transaction date')
plt.show()

plt.figure(figsize=[4,4])
sns.histplot(df['X1 transaction date'],kde=True)
plt.title('Distribution of transaction date')
plt.show()

plt.figure(figsize=[4,4])
sns.histplot(df['X3 distance to the nearest MRT station'],kde=True)
plt.title('Distribution of distance')
plt.show()

plt.figure(figsize=[4,4])
sns.histplot(df['X4 number of convenience stores'],kde=True)
plt.title('Distribution of stores')
plt.show()

plt.figure(figsize=[4,4])
sns.histplot(df['X5 latitude'],kde=True)
plt.title('Distribution of latitude')
plt.show()

plt.figure(figsize=[4,4])
sns.histplot(df['X6 longitude'],kde=True)
plt.title('Distribution of longitude')
plt.show()

plt.figure(figsize=[4,4])
sns.histplot(df['Y house price of unit area'],kde=True)
plt.title('Distribution of house price')
plt.show()
```

Box plot

```
df.columns

Index(['X1 transaction date', 'X2 house age',
       'X3 distance to the nearest MRT station',
       'X4 number of convenience stores', 'X5 latitude', 'X6 longitude',
       'Y house price of unit area'],
      dtype='object')
```

```python
plt.figure(figsize=[4,4])
sns.boxplot(df["X1 transaction date"])
plt.title('Transaction date')
plt.show()

plt.figure(figsize=[4,4])
sns.boxplot(df['X2 house age'])
plt.title('house age')
plt.show()

plt.figure(figsize=[4,4])
sns.boxplot(df['X3 distance to the nearest MRT station'])
plt.title('distance')
plt.show()

plt.figure(figsize=[4,4])
sns.boxplot(df['X4 number of convenience stores'])
plt.title('stores')
plt.show()

plt.figure(figsize=[4,4])
sns.boxplot(df['X5 latitude'])
plt.title('latitude')
plt.show()

plt.figure(figsize=[4,4])
sns.boxplot(df['X6 longitude'])
plt.title('longitude')
plt.show()

plt.figure(figsize=[4,4])
sns.boxplot(df['Y house price of unit area'])
plt.title('house price')
plt.show()

plt.figure(figsize=[4,4])
sns.boxplot(df)
plt.title('house price')
plt.show()

# Correlation matrix

# Select only the numeric columns from the DataFrame
numeric_columns = df.select_dtypes(include=['number'])

# Calculate the correlation matrix
correlation_matrix = numeric_columns.corr()

# Create a heatmap to visualize the correlations
plt.figure(figsize=(10, 6))
sns.heatmap(correlation_matrix, annot=True, cmap='Blues', fmt=".2f", linewidths=0.5)
plt.title('Correlation Matrix')
```

```
Text(0.5, 1.0, 'Correlation Matrix')
```

```python
x = df.drop('Y house price of unit area',axis=1)
y = df['Y house price of unit area']

x_train,x_test,y_train,y_test = train_test_split(x,y,test_size = 0.2,shuffle = True,random_state=42)

# Print the shapes of the resulting sets
print("Training set shape:", x_train.shape, y_train.shape)
print("Testing set shape:", x_test.shape, y_test.shape)
```

```
Training set shape: (331, 6) (331,)
Testing set shape: (83, 6) (83,)
```

```python
def evaluate_model(true,predicted):
    mse = mean_squared_error(true, predicted)
    mae = mean_absolute_error(true,predicted)
    rmse = np.sqrt(mse)
    r2_square = r2_score(true,predicted)
    return mae,rmse,r2_square

models = {
            "Linear Regression" : LinearRegression(),
            "Lasso" : Lasso(),
            "Ridge" : Ridge(),
            "k-Neighbors Regression" : KNeighborsRegressor(),
            "Decision Tree" : DecisionTreeRegressor(),
            "Random Forest Regressor" : RandomForestRegressor(n_estimators = 100, random_state = 0),
            "AdaBoost Regressor" :AdaBoostRegressor(),


score_text=""
for i in range(len(models)):
            model = list(models.values())[i]
            model.fit(x_train,y_train)

            #Make prediction:
            y_train_pred = model.predict(x_train)
            y_test_pred = model.predict(x_test)

            #Evaluate Train and Test dataset :

            model_train_mae, model_train_rmse, model_train_r2 = evaluate_model(y_train, y_train_pred)
            model_test_mae, model_test_rmse, model_test_r2 = evaluate_model(y_test, y_test_pred)
            model_name = list(models.keys())[i]

            print(model_name)

            print("Model Performance for Training set :")

            print('Root Mean Squared Error :',model_train_rmse)
            print("Mean Absolute Error : ", model_train_mae)
            print("R2 Score : ", model_train_r2)

            print("----------------------------------------------------")

            print("Model Performance for Testing set :")

            print('Root Mean Squared Error : ', model_test_rmse)
            print('Mean Absolute Error :  ',{model_test_mae})
            print('R2 Score : ', {model_test_r2})
            print()


price_bins = [0, 40,80,120]  # Example bin edges (low, medium, high price)
price_labels = ['Low', 'Medium', 'High']  # Labels for the bins

# Discretize the `Y house price of unit area` into categories
df['Price_Category'] = pd.cut(df['Y house price of unit area'], bins=price_bins, labels=price_labels,
include_lowest=True)

# Step 2: Split data into features and target variable
X = df[['X1 transaction date', 'X2 house age', 'X3 distance to the nearest MRT station',

        'X4 number of convenience stores', 'X5 latitude', 'X6 longitude']]  # Features
y = df['Price_Category']  # Discretized target variable (Price Category)

# Step 3: Train a classifier (e.g., Decision Tree Classifier)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
clf = DecisionTreeClassifier(random_state=0)
clf.fit(X_train, y_train)

# Step 4: Make predictions
y_pred = clf.predict(X_test)

# Step 5: Generate the confusion matrix
cm = confusion_matrix(y_test, y_pred, labels=price_labels)

# Print the confusion matrix
print("Confusion Matrix:")
print(cm)

x_pred = clf.predict(X_train)
cm = confusion_matrix(y_train, x_pred)
print(cm)
print("ACCURACY {:.4f}".format(accuracy_score(y_train,  x_pred)))

#Prediction for testing data
y_pred = clf.predict(X_test)
cm = confusion_matrix(y_test, y_pred)
print(cm)
print("ACCURACY {:.4f}".format(accuracy_score(y_test,  y_pred)))

from sklearn.tree import plot_tree
```

```python
print(classification_report(y_test,  y_pred))

plt.figure(figsize=(20, 10))
plot_tree(clf, filled=True, rounded=True, fontsize=10)
plt.show()

# Random forest

model_rf =RandomForestClassifier(n_estimators=10,max_depth=4,min_samples_leaf=8, criterion="entropy",
random_state=0)
model_rf.fit(X_train, y_train)
# Calculate training set accuracy
x_pred_rf = model_rf.predict(X_train)
cm = confusion_matrix(y_train, x_pred_rf)
print(cm)
print("ACCURACY {:.4f}".format(accuracy_score(y_train,  x_pred_rf)))

y_pred1  =  model_rf.predict(X_test)
cm = confusion_matrix(y_test,y_pred1)
print(cm)
print("ACCURACY {:.4f}".format(accuracy_score(y_test, y_pred1)))

print(classification_report(y_test, y_pred1))

knn = KNeighborsClassifier(n_neighbors=3)  # You can adjust n_neighbors
knn.fit(X_train, y_train)

KNeighborsClassifier(n_neighbors=3)

x_pred_knn = knn.predict(X_train)
cm = confusion_matrix(y_train, x_pred_knn)
print(cm)
print("ACCURACY {:.4f}".format(accuracy_score(y_train, x_pred_knn)))

print(classification_report(y_test, y_pred_knn))
```