To predict the price of a house depending upon the size of house

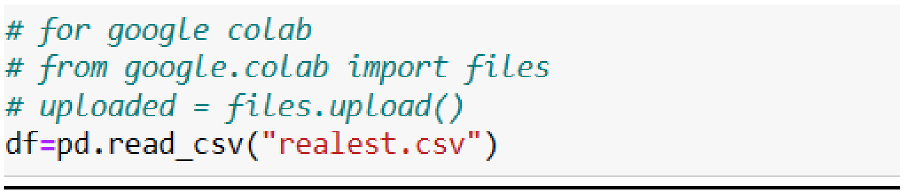
Name: J.Amrutha sai
Date: 19-06-2023
Machine learning assignment

Introduction

In this assignment, I aim to predict house prices based on their sizes. I have taken a dataset containing information about the cost and size of 26 houses. In this, I am utilizing linear regression to train a model on the accepted data (This dataset used for this analysis has two variables: Price and Size).

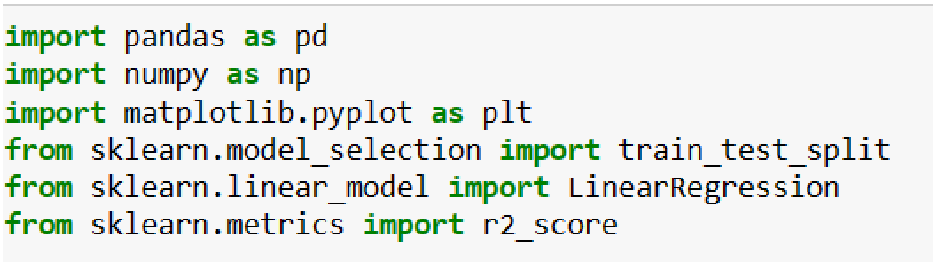
Data Preprocessing

By using the **'pandas'** library I have loaded the dataset from the CSV file



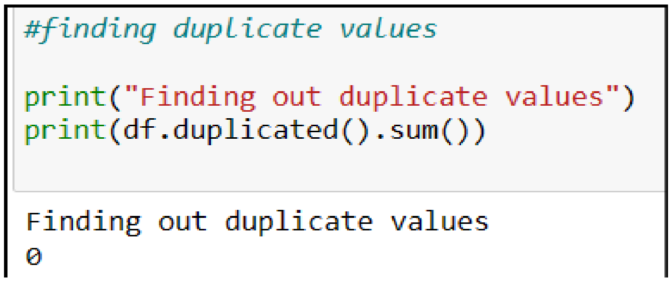
```
# for google colab
# from google.colab import files
# uploaded = files.upload()
df=pd.read_csv("realest.csv")
```

Below are the libraries used in this assignment



```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
```

Also checked for missing values and duplicate values

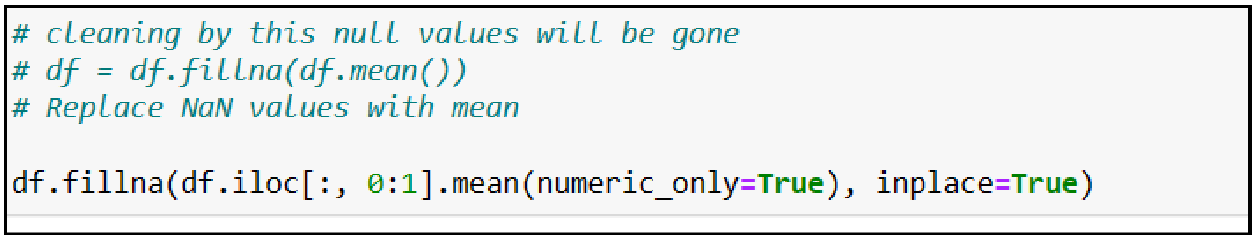


```
#finding duplicate values

print("Finding out duplicate values")
print(df.duplicated().sum())
```

```
Finding out duplicate values
0
```

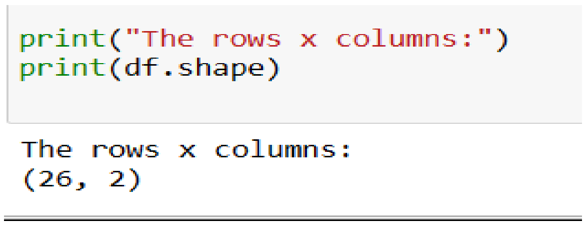
There are **no duplicate** values in the dataset
Here we will fill the cells which have ‘NAN’ with the mean



```
# cleaning by this null values will be gone
# df = df.fillna(df.mean())
# Replace NaN values with mean

df.fillna(df.iloc[:, 0:1].mean(numeric_only=True), inplace=True)
```

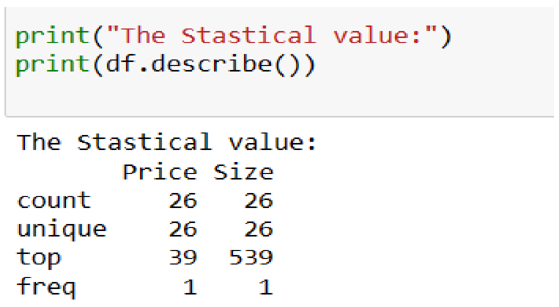
And after all these the final dataset contained 26 rows and 2 columns



```
print("The rows x columns:")
print(df.shape)
```

```
The rows x columns:
(26, 2)
```

<u>Descriptive Statistical</u>



```
print("The Stastical value:")
print(df.describe())
```

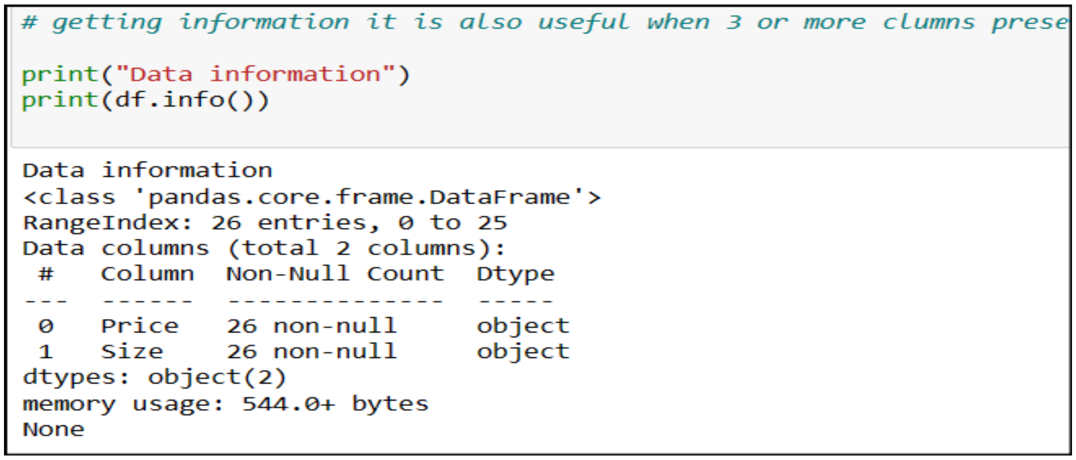
```
The Stastical value:
       Price Size
count     26   26
unique    26   26
top       39  539
freq       1    1
```

In order to know about the distribution and variability of the dataset, a statistical analysis is performed.

Count(Total no of size values in the dataset), Unique(no of distinct size values in the dataset), Top(the most frequent price value in the dataset), and Frequency(no of occurrences of the top price value in the dataset.
Based on the code snippet above we can observe that there are 26 price values in a dataset with 26 unique prices. The size of the columns is 26 for both price and size. And since unique=count the frequency size value is 1 occurring 1 time.

Data Analysis

Here performing an analysis such that if there are any necessary data preprocessing steps required will be known.

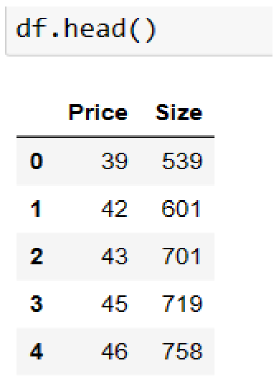


```
# getting information it is also useful when 3 or more clumns prese

print("Data information")
print(df.info())
```

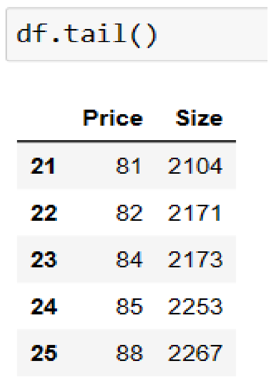
```
Data information
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 26 entries, 0 to 25
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Price   26 non-null     object
 1   Size    26 non-null     object
dtypes: object(2)
memory usage: 544.0+ bytes
None
```

We can see by the above code snippet that we got the information on null object/missing values, and **datatypes** as both are objects(int/float), memory usage, and we can also see the Range indexes for the rows.
Here no missing value/ non-null is shown we can ensure that there are no null objects in the dataset and now to check if the data is loaded correctly or not df. head() and df. tail() is used. The **head()** function is used to give us a glimpse of the data for checking top columns, The **tail()** function is also for the same purpose but it inspects the bottom rows of the dataset and verifies that data have been read completely.



```
df.head()
```

	Price	Size
0	39	539
1	42	601
2	43	701
3	45	719
4	46	758



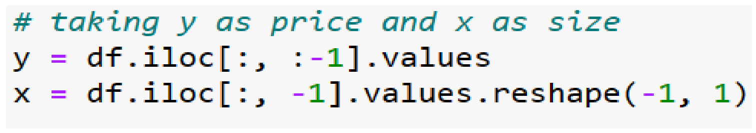
```
df.tail()
```

	Price	Size
21	81	2104
22	82	2171
23	84	2173
24	85	2253
25	88	2267

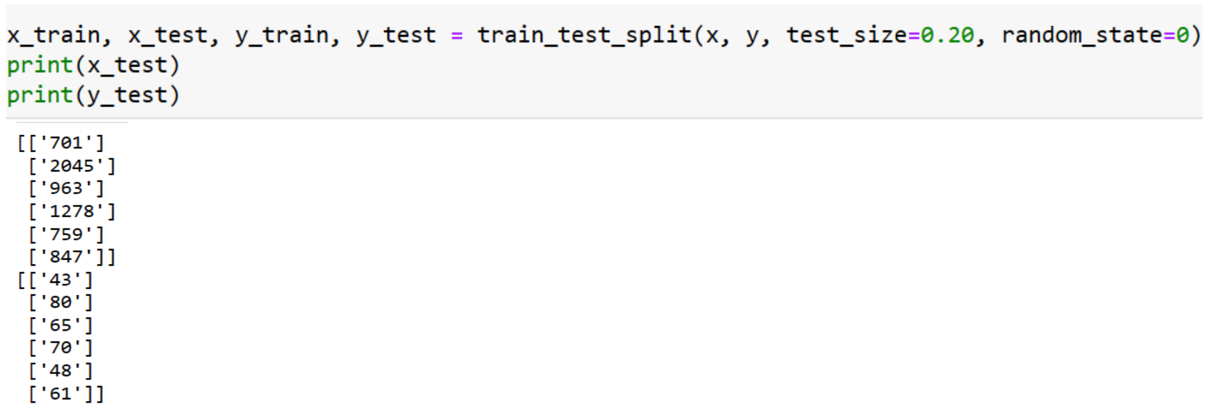
Since the indices are from 0 to 25 and we can see that dataset is loaded correctly.

Model Training and Evaluation

For the prediction of the house prices based on size, I have chosen the Linear regression model. As it takes a linear relationship between the input variables and the dependent variable(Price). x and y values are taken as size and price respectively



```
# taking y as price and x as size
y = df.iloc[:, :-1].values
x = df.iloc[:, -1].values.reshape(-1, 1)
```

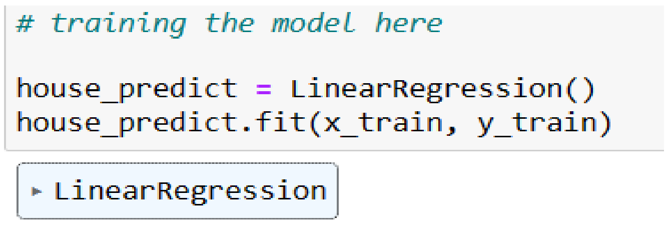


```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.20, random_state=0)
print(x_test)
print(y_test)
```

```
[['701']
 ['2045']
 ['963']
 ['1278']
 ['759']
 ['847']]
[['43']
 ['80']
 ['65']
 ['70']
 ['48']
 ['61']]
```

At the beginning of the model training process, we first split the dataset into training and test sets (as shown in the above code snippet). In this, we use a function called **‘*train_test_split*’** taken from a library scikit-learn(sklearn.model_selection) to randomly split data, as we can see **test_size = 0.2** this means that 20% of the data is for the test, and 80% of the data for the training.

Next to perform a model training using ‘x_train’ we call an **instance of the Linear Regression model** and used the **fit method**, passing Price and training data. Then the model will able to learn the relationship between the price and size(shown in the code snippet below)



```
# training the model here

house_predict = LinearRegression()
house_predict.fit(x_train, y_train)
```

```
▸ LinearRegression
```

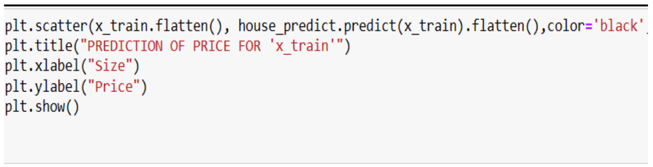
During the training, we generally use the default parameters of the Linear Regression Model (LinearRegression())

Model Evaluation

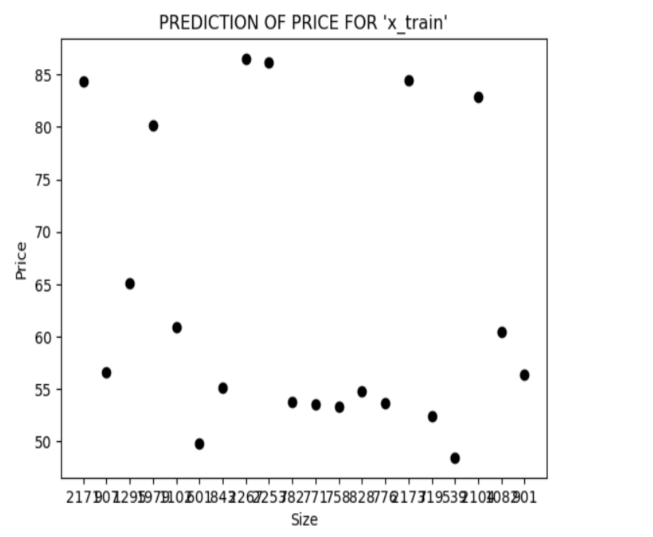
For evaluating the performance of the Linear Regression Model we used a function called **‘r2_score**’ from taken from a library scikit-learn(sklearn. metrics). It evaluates the proportion of the variance in the target variable/dependent variable(price) that is predictable from the input variable(size).
We calculate it by actually passing the actual prices from the test dataset and predicted prices(predicted from the trained model).

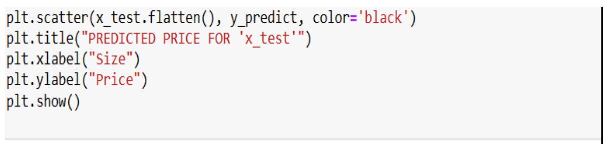
The R-squared score ranges from 0 to 1. Where 1 is a perfect fit. Based on these we can say that the higher R-squared value suggests that the model captures a significant portion of the price variation based on the size

For future analysis, we considered the predicted prices with the actual prices from the testing set by plotting a scattered plot to visualize the predicted prices and actual prices

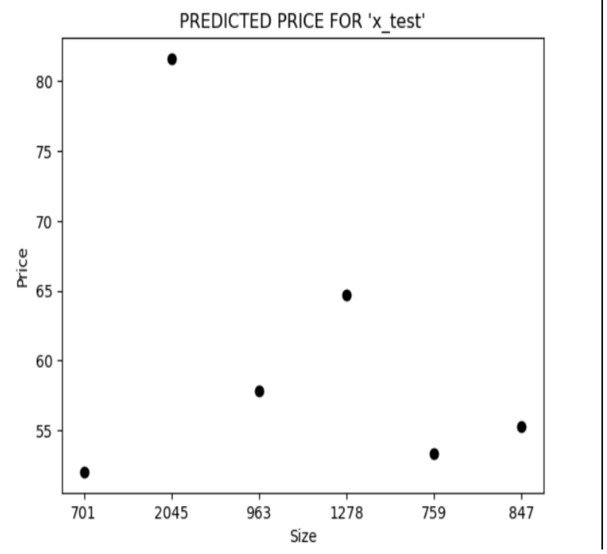


```
plt.scatter(x_train.flatten(), house_predict.predict(x_train).flatten(),color='black'
plt.title("PREDICTION OF PRICE FOR 'x_train'")
plt.xlabel("Size")
plt.ylabel("Price")
plt.show()
```



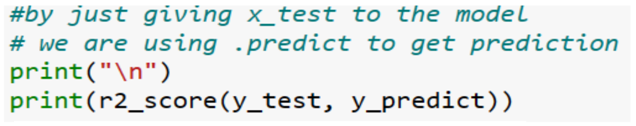


```
plt.scatter(x_test.flatten(), y_predict, color='black')
plt.title("PREDICTED PRICE FOR 'x_test'")
plt.xlabel("Size")
plt.ylabel("Price")
plt.show()
```



Here above are the Predicted prices plots for training data and test data(taken plots **randomly** not in an order).

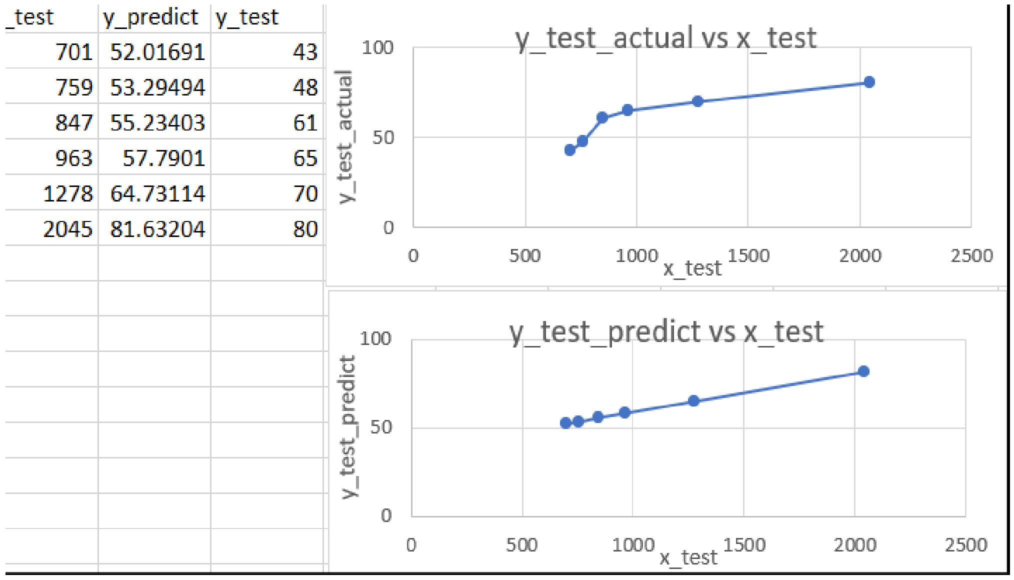
Results and Declaration



```
#by just giving x_test to the model
# we are using .predict to get prediction
print("\n")
print(r2_score(y_test, y_predict))
```

```
0.7633711389075191
```

Here we can see that the R-squared score is 0.763, which shows that it can explain **approx 76%** of the variance in the house prices based on their sizes.



_test	y_predict	y_test
701	52.01691	43
759	53.29494	48
847	55.23403	61
963	57.7901	65
1278	64.73114	70
2045	81.63204	80

Fig-1

Here in the graphs(Fig-1 and Fig-2) and we can see that y-predict has a **linear relationship** with size and

price(since we have to use the linear regression model we can compare with the actual price)
We can clearly see the straight line for predicted values.

The **actual price graphs are not linear** because there are many factors that affect the price of the house like no of a bedroom, Lot area, Location…etc so the in real life the size is not only one variable that affects the price

x_train	y_predict	y_train
539	48.44723	39
601	49.8134	42
719	52.41354	45
758	53.27291	46
771	53.55936	50
776	53.66954	53
782	53.80175	57
828	54.81536	59
843	55.14589	60
901	56.42392	63
907	56.55613	64
1082	60.41227	67
1102	60.85297	69
1295	65.10574	72
1979	80.17772	73
2104	82.93211	81
2171	84.40846	82
2173	84.45253	84
2253	86.21533	85
2267	86.52382	88

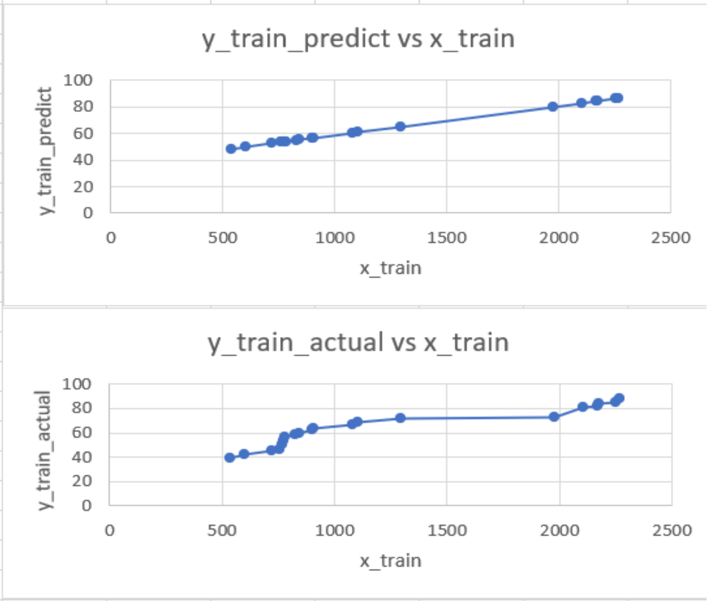
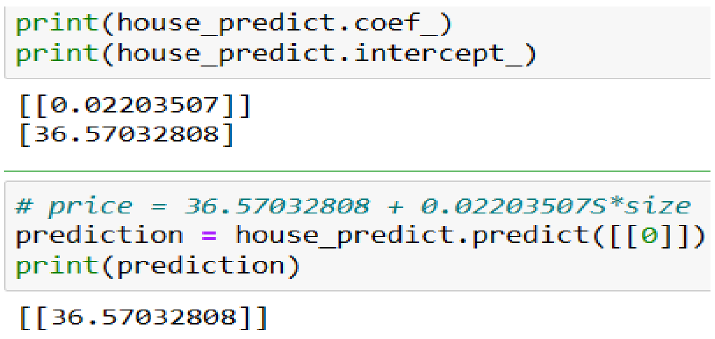


Fig-2



```
print(house_predict.coef_)
print(house_predict.intercept_)
```

```
[[0.02203507]]
[36.57032808]
```

```
# price = 36.57032808 + 0.022035075*size
prediction = house_predict.predict([[0]])
print(prediction)
```

```
[[36.57032808]]
```

We can also know the linear equation of the y-predict graph using '.coef' and '.intercept'

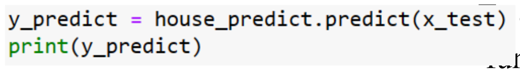
Price = (.intercept(value) + .coef(value) * size)

Conclusion

In conclusion, we successfully built a linear regression model to predict house prices based on their sizes. The model achieved a satisfactory performance with an R-squared score of 0.73 and with a linear equation of ***price = 36.57032808 + 0.022035075* size.*** However, it's important to note that other factors not considered in this analysis may also influence house prices

Appendix

‘df. dropna(subset=['Price'],inplace=True)’ can also use this line of code to remove NAN rows in the 'Price’ column.



```
y_predict = house_predict.predict(x_test)
print(y_predict)
```

define the y_predict we should use the nction “.predict()”(if it is not used we will get an error called Linear Regression is not callable).

REFERENCES

https://pandas.pydata.org/docs/
https://scikit-learn.org/stable/documentation.html
https://matplotlib.org/stable/contents.html
https://www.kaggle.com/code/shubhamc003/chicago-house-price-model-building
https://www.kaggle.com/datasets/tawfikelmetwally/chicago-house-price
https://www.coincent.ai/student-board/lms