LINEAR REGRESSION

NUMERICAL COMPUTING

40030510	MELIKA ZAMANI	Winter 2023

PROJECT OUTLINE

we are given a .csv file which contains a real dataset of house price of unit area. in the given dataset we see different factors effect on the price a house . now we are going to do some analysis on the dataset using correlation of each factor with another and also the correlation of the each factor and the price of the house . at the end we get the linear regression of the dataset and predict the price of a house and compare it with the actual price. in every steps some graphs is provided for a better understanding.

IMPORT LIBRARIES

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

IMPORT DATASET + CLEANING

```
data = pd.read_csv('Real estate.csv')
# Remove any rows with missing values
data.dropna(inplace=True)
# Remove any duplicate rows
data.drop_duplicates(inplace=True)
```

DATA OVERVIEW

```
print(data.head())
print(data.info())
print(data.describe())
print(data.columns) #shows the columns
```

```
# Select the relevant columns
X = data[cols].values
```

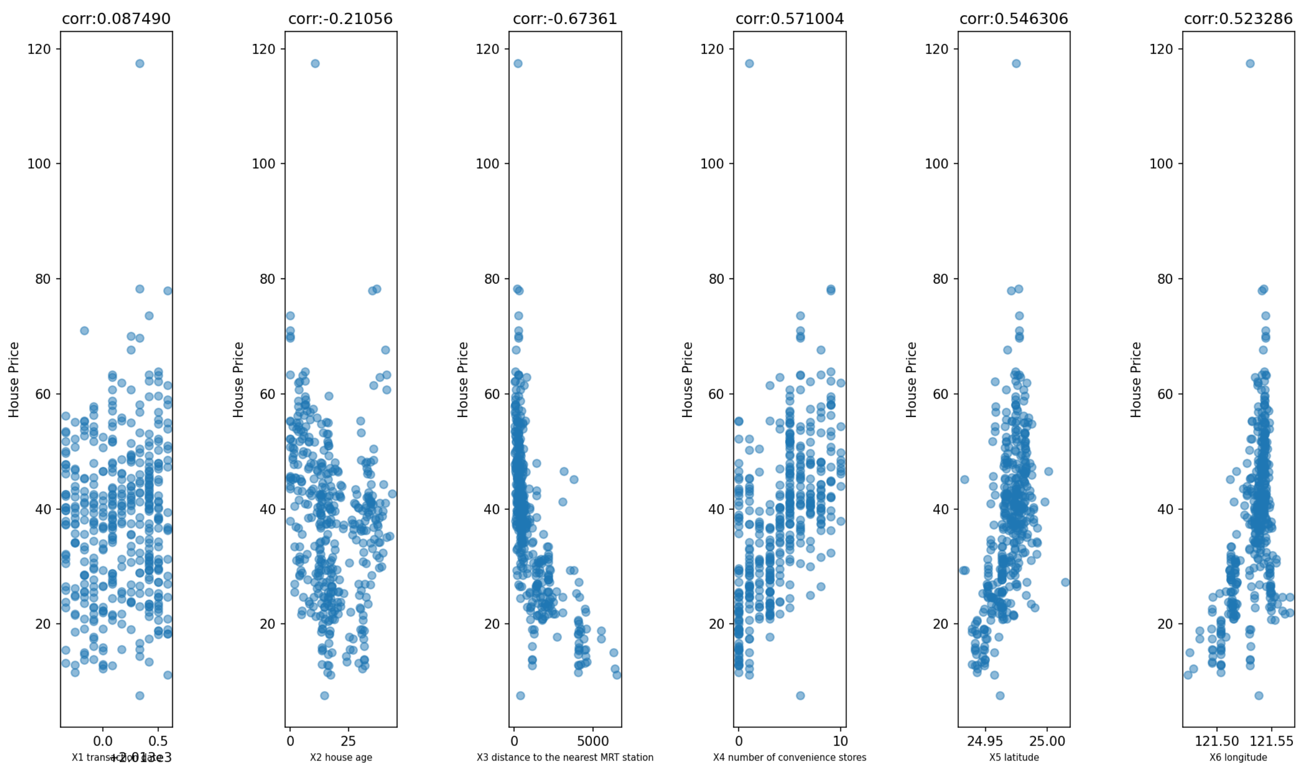
```
   No  X1 transaction date  ...  X6 longitude  Y house price of unit area
0   1             2012.917  ...     121.54024                        37.9
1   2             2012.917  ...     121.53951                        42.2
2   3             2013.583  ...     121.54391                        47.3
3   4             2013.500  ...     121.54391                        54.8
4   5             2012.833  ...     121.54245                        43.1

[5 rows x 8 columns]
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 414 entries, 0 to 413
Data columns (total 8 columns):
 #   Column                                  Non-Null Count  Dtype
---  ------                                  --------------  -----
 0   No                                      414 non-null    int64
 1   X1 transaction date                     414 non-null    float64
 2   X2 house age                            414 non-null    float64
 3   X3 distance to the nearest MRT station  414 non-null    float64
 4   X4 number of convenience stores         414 non-null    int64
```

```python
y = data['Y house price of unit area'].values


cols = ['X1 transaction date', 'X2 house age', 'X3 distance to the nearest MRT station'
    , 'X4 number of convenience stores', 'X5 latitude', 'X6 longitude']
```

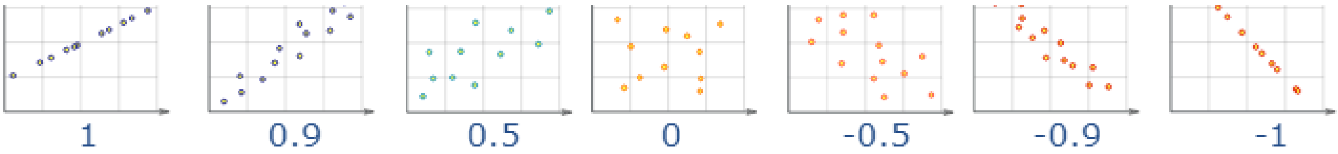
```
 5   X5 latitude                            414 non-null    float64
 6   X6 longitude                           414 non-null    float64
 7   Y house price of unit area             414 non-null    float64
dtypes: float64(6), int64(2)
memory usage: 26.0 KB
None
               No  ...  Y house price of unit area
count  414.000000  ...                  414.000000
mean   207.500000  ...                   37.980193
std    119.655756  ...                   13.606488
min      1.000000  ...                    7.600000
25%    104.250000  ...                   27.700000
50%    207.500000  ...                   38.450000
75%    310.750000  ...                   46.600000
max    414.000000  ...                  117.500000

[8 rows x 8 columns]
Index(['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'],
      dtype='object')
```

Xi AND Y CORRELATION

```python
# showing different relations with x columns and y before regression
fig, axs = plt.subplots(1, X.shape[1], figsize=(14, 6))
fig.subplots_adjust(wspace=1)
for i in range(X.shape[1]):
    axs[i].scatter(X[:, i], y, alpha=0.5)
    axs[i].set_xlabel(data.columns.array[i + 1], fontsize=7)
    axs[i].set_ylabel('House Price')
    axs[i].set_title("corr:" + str(data[cols[i]].corr(data['Y house price of unit area']))[:8])
plt.show()
```



at the top of each diagram you see the correlation result which is calculated using ρ (X,Y) = cov (X,Y) / σX.σY formula. we get the following information using correlation number.



it means that if the correlation is the closest to 1 it has the most effect on the pricing ; on the other hand if the correlation is closest to -1 is hast negative impact on the pricing.

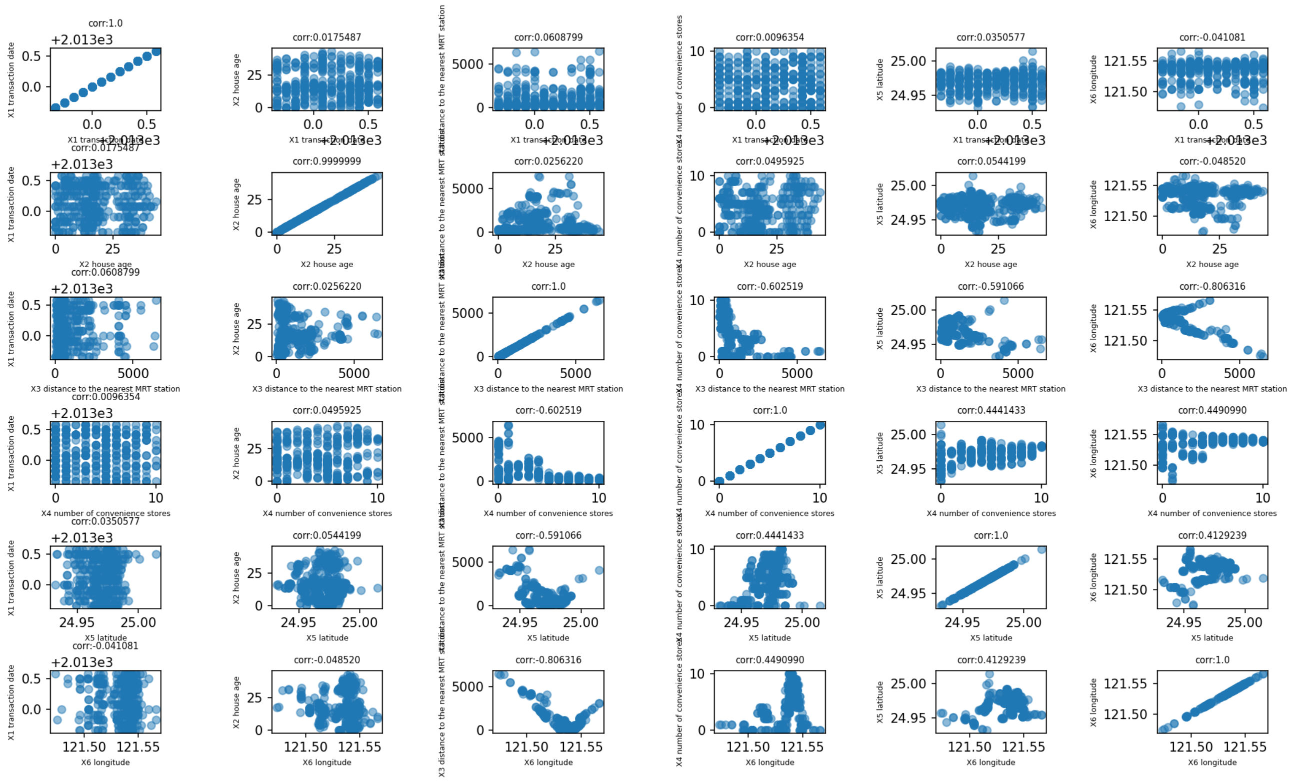
Xi	Correlation	Status	Notes
X1 transaction date	0.087	LOW POSITIVE	it has a very low effect .we can say it has no correlation
X2 house age	−0.210	LOW NEGATIVE	the older the house the cheaper it gets
X3 distance to the nearest MRT station	−0.673	LOW NEGATIVE	the more the distance the lesser the price
X4 number of convenience stores	0.571	LOW POSITIVE	the more the stores the more expensive the house
X5 latitude	0.546	LOW POSITIVE	the more the latitude the more expensive the house
X6 longitude	0.523	LOW POSITIVE	the more the longitude the more expensive the house

LOW NEGATIVE · HIGH NEGATIVE · LOW POSITIVE · HIGH POSITIVE

Xis CORRELATION

```
# correlation between columns before regression
fig, axs = plt.subplots(5, 5, figsize=(15, 15))
fig.subplots_adjust(hspace=1)
fig.subplots_adjust(wspace=1)
for i in range(5):
    colCount = 0
    for j in range(6):
```

```
        if j > i :
            x_col1 = cols[i]
            x_col2 = cols[j]
            axs[i, colCount].scatter(data[x_col1], data[x_col2], alpha=0.5)
            axs[i, colCount].set_xlabel(x_col1, fontsize=6)
            axs[i, colCount].set_ylabel(x_col2, fontsize=6)
            axs[i, colCount].set_title("corr:" + str(data[x_col1].corr(data[x_col2]))[:9], fontsize=7)
            colCount += 1
plt.show()
```



as you see in the above picture we have shown the correlation between different factors to see if they have any relation with each other. some has none correlation and we can't get a conclusion from the diagram. by the way the main diameter shows the correlation of each factor with itself so it has a correlation of 1.

SPILITTING DATA

```
#Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

TRAIN LINEAT REGRESSION

```
# Train the linear regression model on the training data
reg = LinearRegression().fit(X_train, y_train)
```

MAKE PREDICTIONS

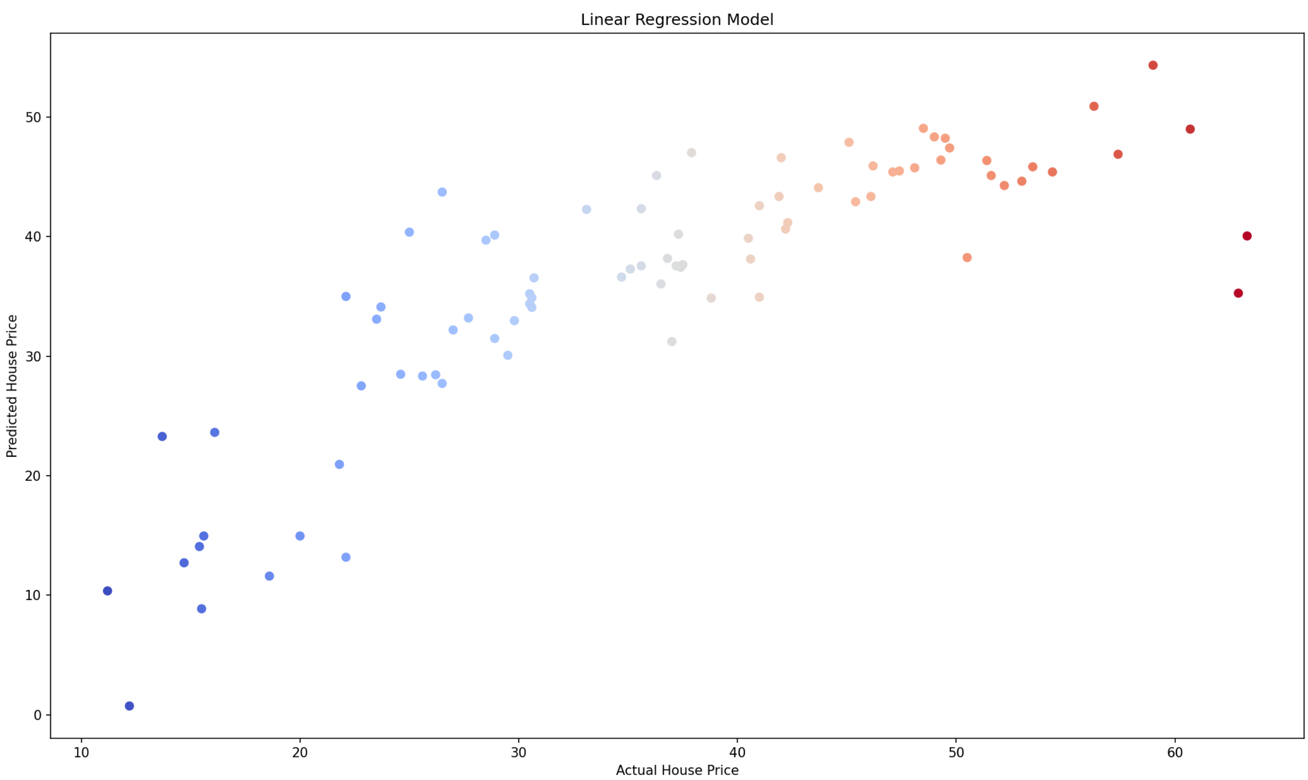
```
# Make predictions on the testing data
y_pred = reg.predict(X_test)
```

MEAN SQUARED & R-SQUARED

```
# Calculate the mean squared error and R-squared score
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
```

PLOTING RESULT

```
colors = y_test / np.max(y_test)  # normalize the actual house prices to [0, 1]
plt.scatter(y_test, y_pred, c=colors, cmap='coolwarm')
plt.xlabel('Actual House Price')
plt.ylabel('Predicted House Price')
plt.title('Linear Regression Model')
plt.show()


# Print the results
print("Coefficients: ", reg.coef_)
print("Intercept: ", reg.intercept_)
print("Mean squared error: {:.2f}".format(mse))
print("R-squared score: {:.2f}".format(r2))
```



as the diagram shows the correlation between the predicted price and the actual price is between 0 and 1 so it means that we had predicted the prices good and the error is not too big.

CONCLUSION

Linear regression is a simple but powerful technique that can be used for a wide range of prediction tasks. In this project, we worked with real-world datasets and developed a linear regression approach to predict related concepts.