Computer Engineering Department

BIM 309 – Artificial Intelligence

House Price Prediction with Linear Regression

Prepared By
Necip Şükrü Aşık

Date: 09.01.2022

“Linear regression was developed in the field of statistics and is studied as a model for understanding the relationship between input and output numerical variables, but has been borrowed by machine learning. It is both a statistical algorithm and a machine learning algorithm.”[2]

In this assignment, I tried to predict how house prices change based on features using a dataset of house prices and home features by using Linear Regression.

Libraries I used:

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

I transferred the file to the dataset with read_csv, then I split independent variable to x and dependent variable to y :

```
dataset = pd.read_csv('data/prices.csv')
x = dataset[['lot_area','living_area','num_floors','num_bedrooms',
             'num_bathrooms','waterfront','year_built','year_renovated']]
y = dataset['price']
```

Index	price
0	221900
1	538000
2	180000
3	604000
4	510000
5	1225000
6	257500
7	291850
8	229500
9	323000
10	662500
11	468000

Şekil 1.Y variables

Index	lot_area	living_area	num_floors	ıum_bedrooms	ıum_bathroom	waterfront
0	5650	1180	1	3	1	0
1	7242	2570	2	3	2.25	0
2	10000	770	1	2	1	0
3	5000	1960	1	4	3	0
4	8080	1680	1	3	2	0
5	101930	5420	1	4	4.5	0
6	6819	1715	2	3	2.25	0
7	9711	1060	1	3	1.5	0
8	7470	1780	1	3	1	0
9	6560	1890	2	3	2.5	0
10	9796	3560	1	3	2.5	0
11	6000	1160	1	2	1	0
12	19901	1430	1.5	3	1	0

Şekil 2. X variables

After writing the independent and dependent variable, I allocated 20% of the data to the test and 80% to training . I used “random_state=0” for take same sequence each time to see results clearly. I called "LinearRegression()" as "lr". Then I built the model by taking the "X_train" and "Y_train" :

```
X_train, X_test, Y_train, Y_test = train_test_split(x, y, test_size=0.2, random_state=0)

regressor = LinearRegression()
regressor.fit(X_train, Y_train)
```

Index	price
5268	495000
16909	635000
16123	382500
12181	382500
12617	670000
19024	1001000
5063	1100000
9888	713000
2774	416000
3197	401000
10315	402000
13059	255000

Şekil 3. Y_train

Index	price
17384	297000
722	1578000
2680	562100
18754	631500
14554	780000
16227	485000
6631	340000
19813	335606
3367	425000
21372	490000
3268	732000
20961	389700

Şekil 4. Y_test

Index	lot_area	living_area	num_floors	um_bedrooms	um_bathroom	waterfront
17384	1650	1430	3	2	1.5	0
722	51836	4670	2	4	3.25	0
2680	3700	1440	1	2	0.75	0
18754	2640	1130	1	2	1	0
14554	9603	3180	2	4	2.5	0
16227	3436	1650	2	3	2.5	0
6631	28000	1720	1	3	2.75	0
19813	4600	2538	2	3	2.5	0
3367	5440	2460	2	4	2.5	0
21372	2975	4460	3	5	3.5	0
3268	11300	2360	1	4	1.75	0
20961	3581	1720	2	3	2.5	0
21456	1049	1020	3	2	1.5	0

Şekil 5. X_test

Index	lot_area	living_area	num_floors	ıum_bedrooms	ıum_bathroom	waterfront	
5268	5510	1570	1	3	1	0	19
16909	11000	1780	1	3	2.5	0	19
16123	9862	1090	1	3	1.5	0	19
12181	7079	2210	2	4	2.5	0	19
12617	4763	1800	2	3	2.5	0	19
19024	8000	3100	1.5	4	2	0	19
5063	60123	5070	2	4	3.75	0	20
9888	4000	1180	1.5	1	1	0	19
2774	5372	1800	2	3	2.5	0	19
3197	12523	3010	1	4	1.75	0	19
10315	7620	2770	1	5	2.75	0	19
13059	8528	1240	1	3	1.5	0	19
17081	4000	1990	1	2	1.75	0	19

Şekil 6. X_train

The machine does not know the “y_test” values. I give the machine "x_test" and find the prediction of "y_test". I combine the “y_test” (true values) with the predicted values with DataFrame so I can compare easily:

```
prediction = regressor.predict(X_test)
act_pred = pd.DataFrame({'Actual': Y_test, 'Predicted': prediction})
```

Index	Actual	Predicted
17384	297000	385431
722	1578000	1.29183e+06
2680	562100	498648
18754	631500	382516
14554	780000	776521
16227	485000	384773
6631	340000	500011
19813	335606	613658
3367	425000	563255
21372	490000	1.17106e+06
3268	732000	522205
20961	389700	380152

Şekil 7. act_pred

*“**R-squared** gives you the percentage variation in y explained by x-variables. **The mean squared error** tells you how close a regression line is to a set of points. It does this by taking the distances from the points to the regression line (these distances are the “errors”) and squaring them. **Root Mean Square Error (RMSE)** is the standard deviation of the residuals (prediction errors). Residuals are a measure of how far from the regression line data points are; RMSE is a measure of how spread out these residuals are. In other words, it tells you how concentrated the data is around the line of best fit.”[1]*

I find the errors with “y_test” and “predictions” values so I can see how accurate the predictions are :

```
print(act_pred)
print('R2:',r2_score(act_pred["Actual"],act_pred["Predicted"]))
print('MSE:', metrics.mean_squared_error(Y_test, prediction))
print('RMSE:', np.sqrt(metrics.mean_squared_error(Y_test, prediction)))
```

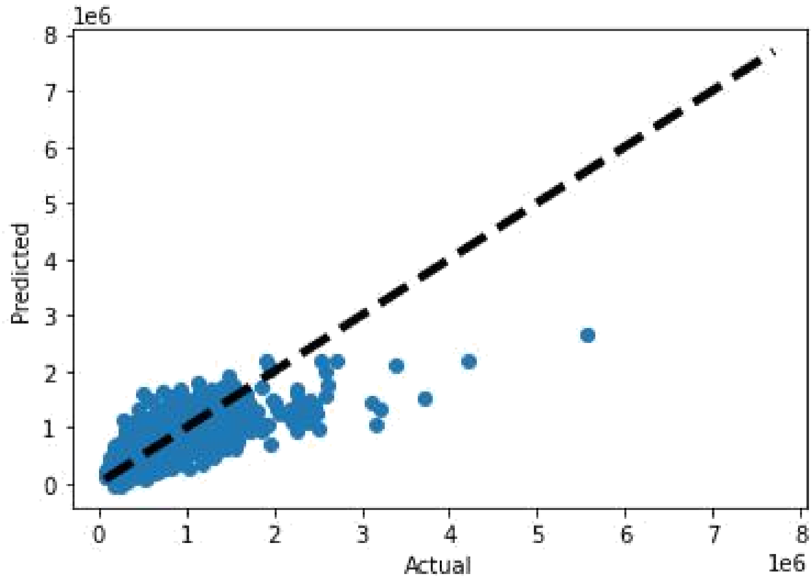
```
        Actual     Predicted
17384   297000  3.854307e+05
722    1578000  1.291834e+06
2680    562100  4.986484e+05
18754   631500  3.825159e+05
14554   780000  7.765211e+05
...        ...           ...
5427    844000  8.236249e+05
16547   335500  2.714028e+05
4585    369950  3.498274e+05
17762   300000  2.916120e+05
16323   575950  4.220961e+05

[4323 rows x 2 columns]
R2: 0.5624779645691738
MSE: 52031988364.79344
RMSE: 228105.21336609876
```

Şekil 8. Results of R-Squared, MSE, RMSE

Plots:

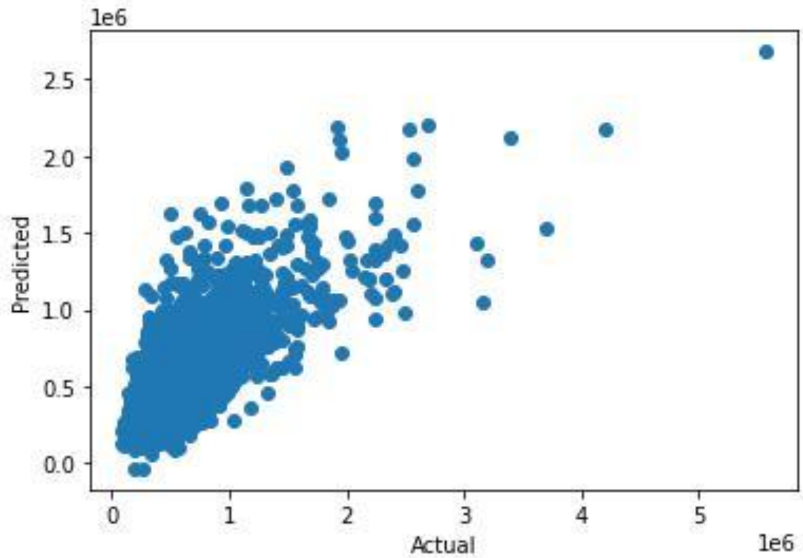
I used the variable "ax" for single a Axes. I showed the distribution with “act_pred[“Actual”] “ and “act_pred[“Prediction”]”. Line 3 draws a linear line from start to finish for easy interpretation :

```
1. fig,ax = plt.subplots()
2. ax.scatter(act_pred["Actual"],act_pred["Predicted"])
3. ax.plot([y.min(), y.max()], [y.min(), y.max()], 'k')
4. ax.set_xlabel('Actual')
5. ax.set_ylabel('Predicted')
6. plt.show()
7.
```



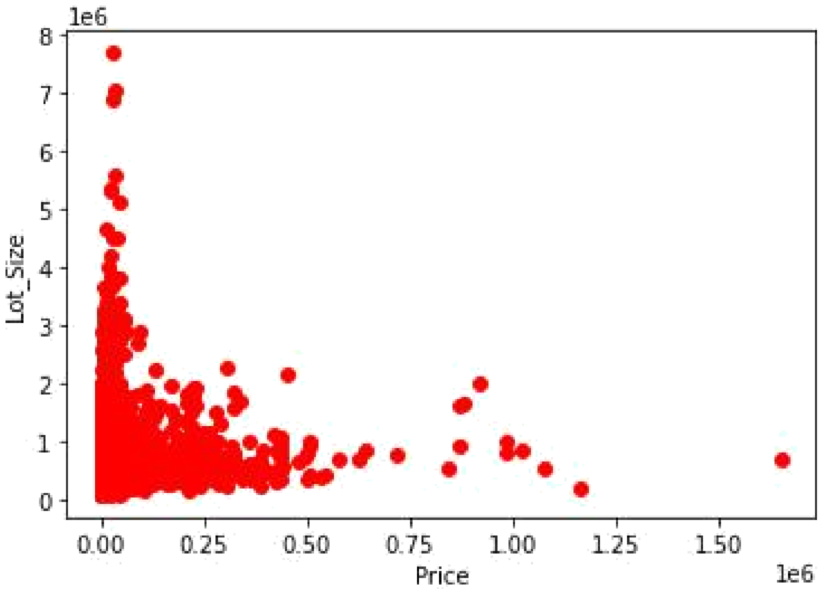
Graph without line :

```
plt.scatter(act_pred["Actual"], act_pred["Predicted"]);
plt.xlabel('Actual')
plt.ylabel('Predicted')
plt.show()
```

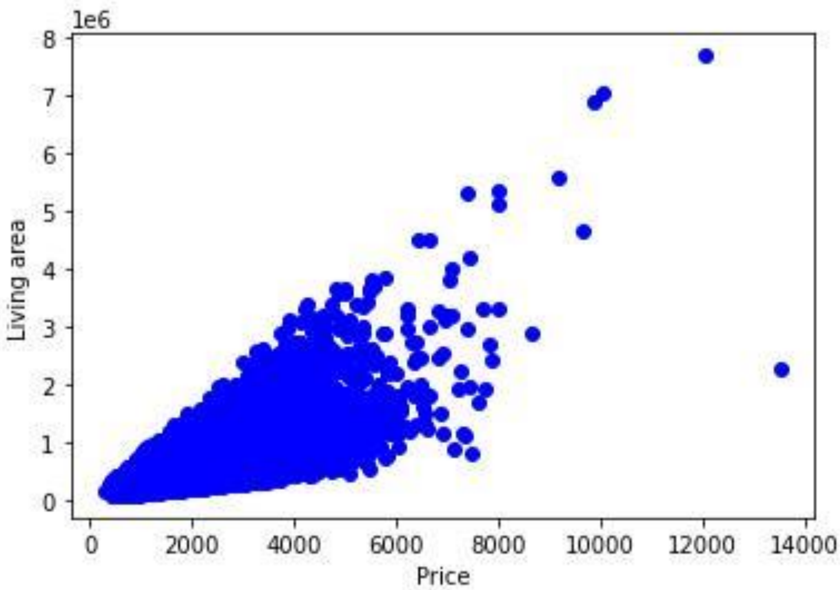


Graphs of features by price :

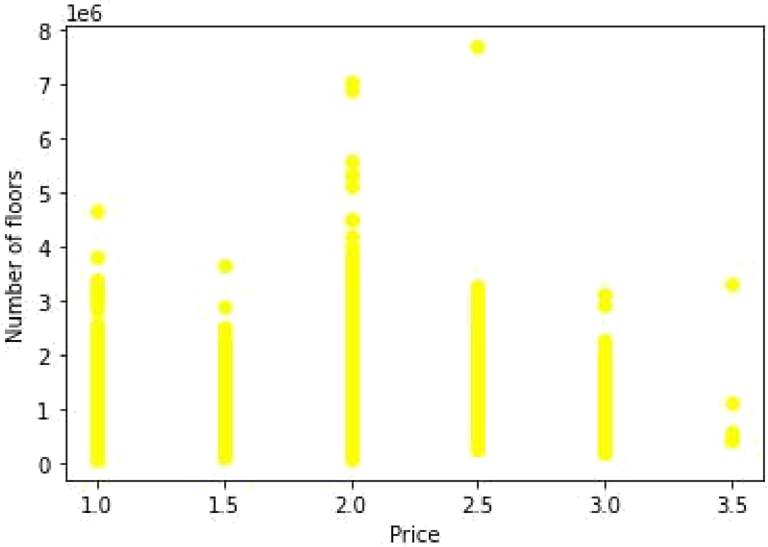
```
plt.scatter(x["lot_area"],y,color='red')
plt.xlabel('Price')
plt.ylabel('Lot_Size')
plt.show()
```



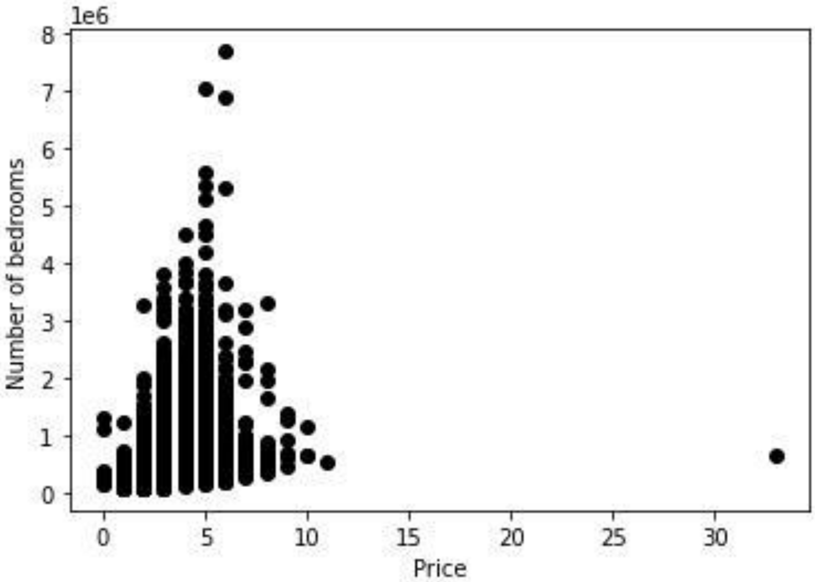
Şekil 9. Graph of"lot_area" vs "price"



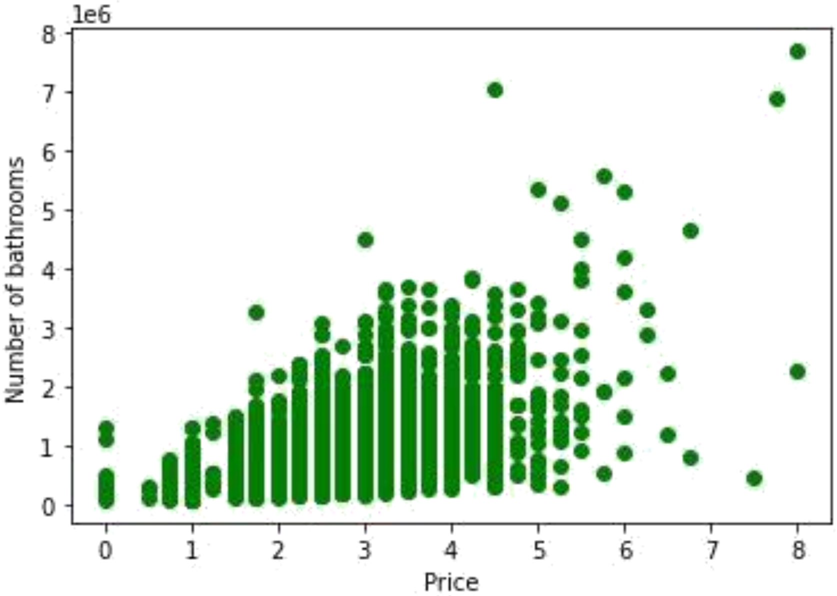
Şekil 10.Graph of "living_area" vs "price"



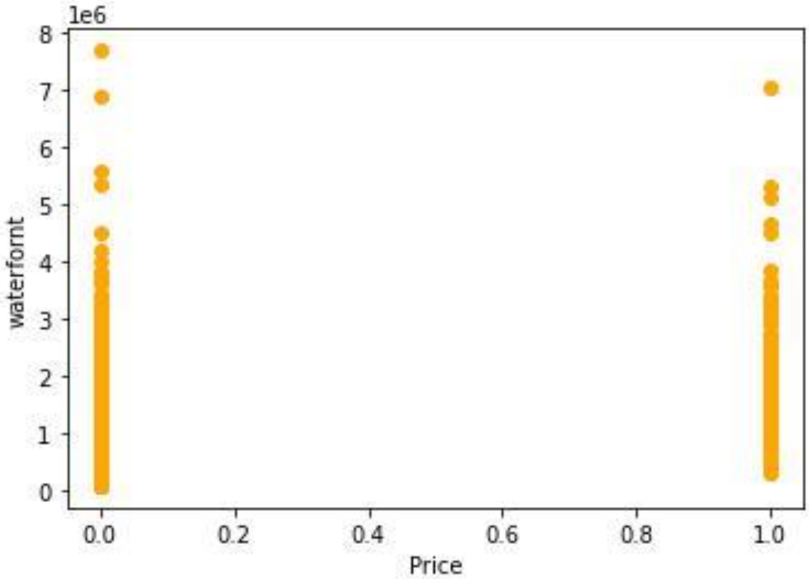
Şekil 11. Graph of "num_floors" vs "price"



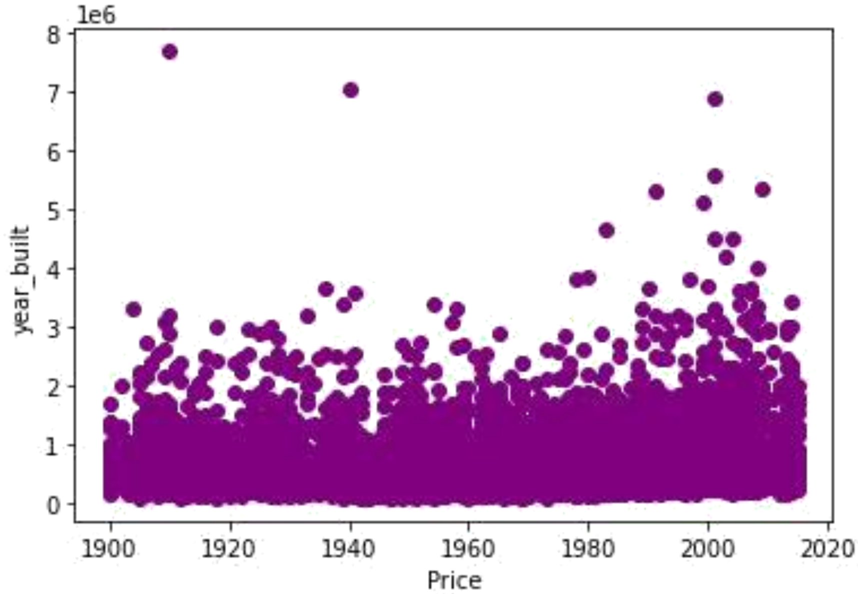
Şekil 12. Graph of "num_bedrooms" vs "price"



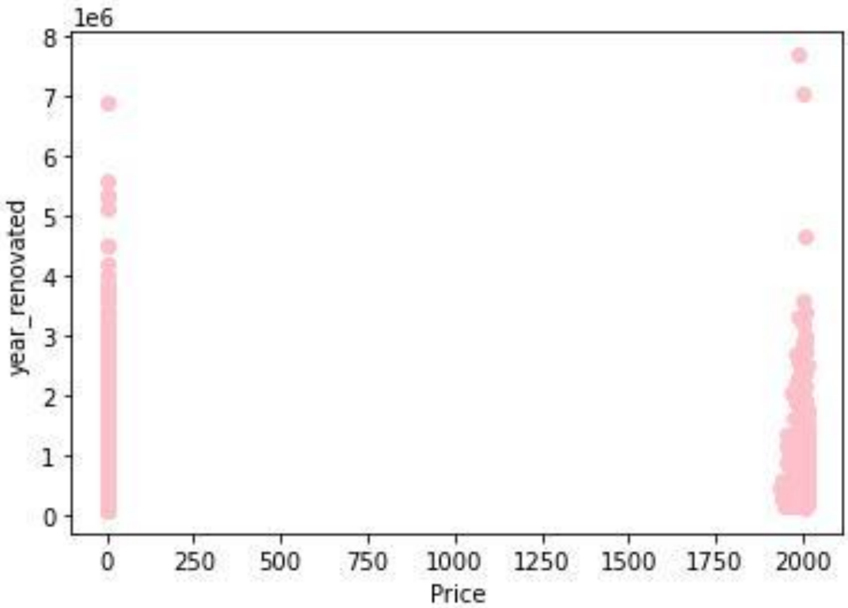
Şekil 13. Graph of "num_bathrooms" vs "price"



Şekil 14. Graph of "waterfront" vs "price"



Şekil 15. Graph of "year_built" vs "price"



Şekil 16. Graph of "year_renovated" vs "price"

References:

[1] https://www.statisticshowto.com/

[2] https://machinelearningmastery.com/linear-regression-for-machine-learning/