House Price Prediction Model

06.12.2022

Tayfun Kok

My Goal?

- Developing a machine learning model to predict the price of a house based on its characteristics such as location, age, number of rooms, etc.

- Implementing linear regression model.

Use Case

- No bank wants to lend more than the real value of the house
- The bank does not immediately give you the home loan you want to take out even if your credit score is sufficient.
- The bank first assesses the real value of the house.
- Then gives a home loan according to the real value.

Use Case

- The bank can use the model developed in this project to calculate the real value of the house the customer wants to buy

- For example, this model works like this

➢ The customer enters the characteristics of the house such as location, number of rooms, age, etc. into the application.

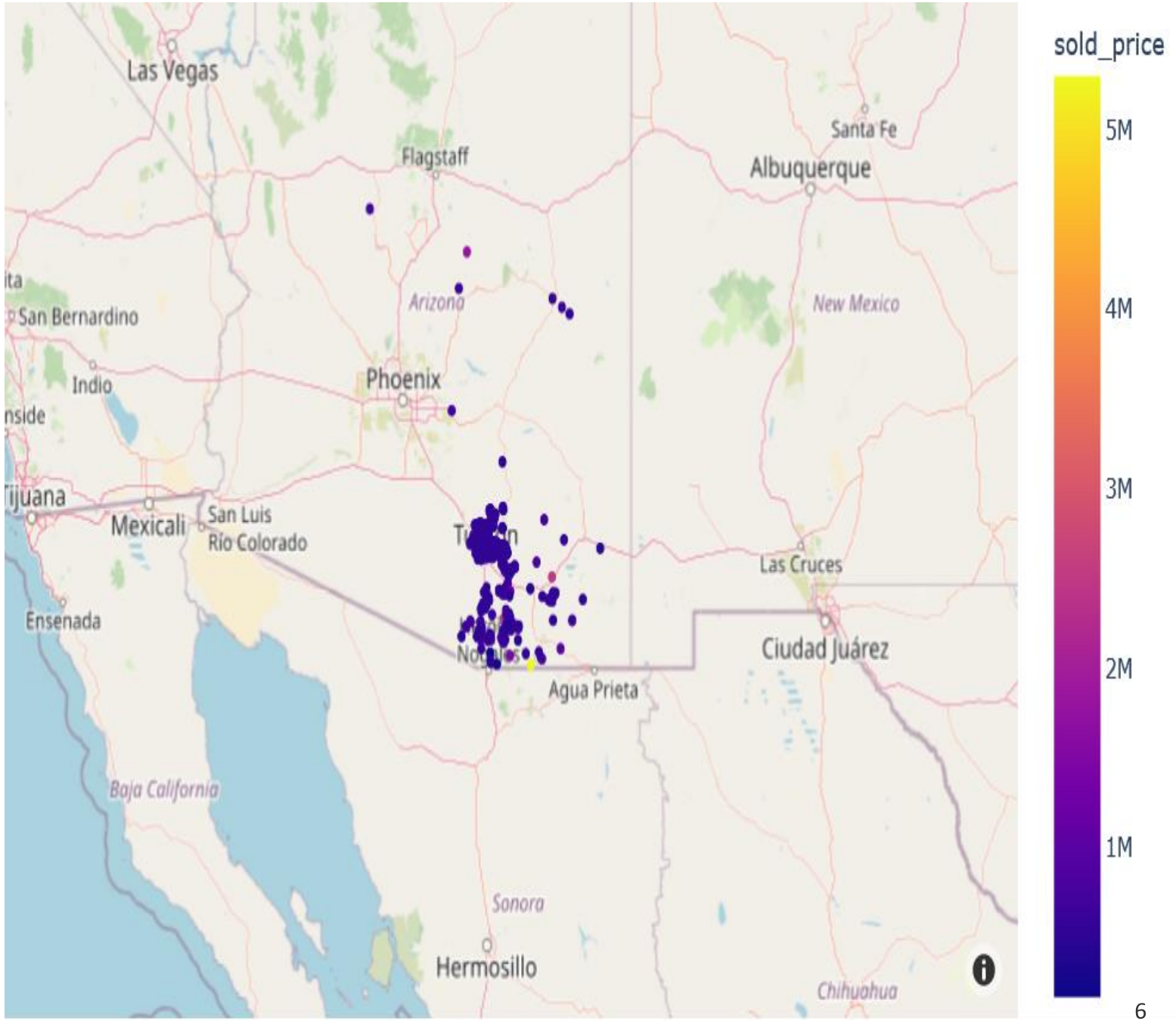
➢ Then the application tells the customer how much loan they can get from the bank for this house.

Recognizing and Understanding Data

- The shape of the dataset : (5000,16)
- Longitudes : vertical lines that measure east or west of the meridian in Greenwich, England.
- Latitudes : horizontal lines that measure distance north or south of the equator
- "lot_acres" column contains 10 null values also there are some empty values needed to be filled
- There were outliers and messy values needed to be cleaned.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 16 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   MLS               5000 non-null   int64
 1   sold_price        5000 non-null   float64
 2   zipcode           5000 non-null   int64
 3   longitude         5000 non-null   float64
 4   latitude          5000 non-null   float64
 5   lot_acres         4990 non-null   float64
 6   taxes             5000 non-null   float64
 7   year_built        5000 non-null   int64
 8   bedrooms          5000 non-null   int64
 9   bathrooms         5000 non-null   object
 10  sqrt_ft           5000 non-null   object
 11  garage            5000 non-null   object
 12  kitchen_features  5000 non-null   object
 13  fireplaces        5000 non-null   object
 14  floor_covering    5000 non-null   object
 15  HOA               5000 non-null   object
dtypes: float64(5), int64(4), object(7)
memory usage: 625.1+ KB
```

Distribution of the houses on the map : Before cleaned the data

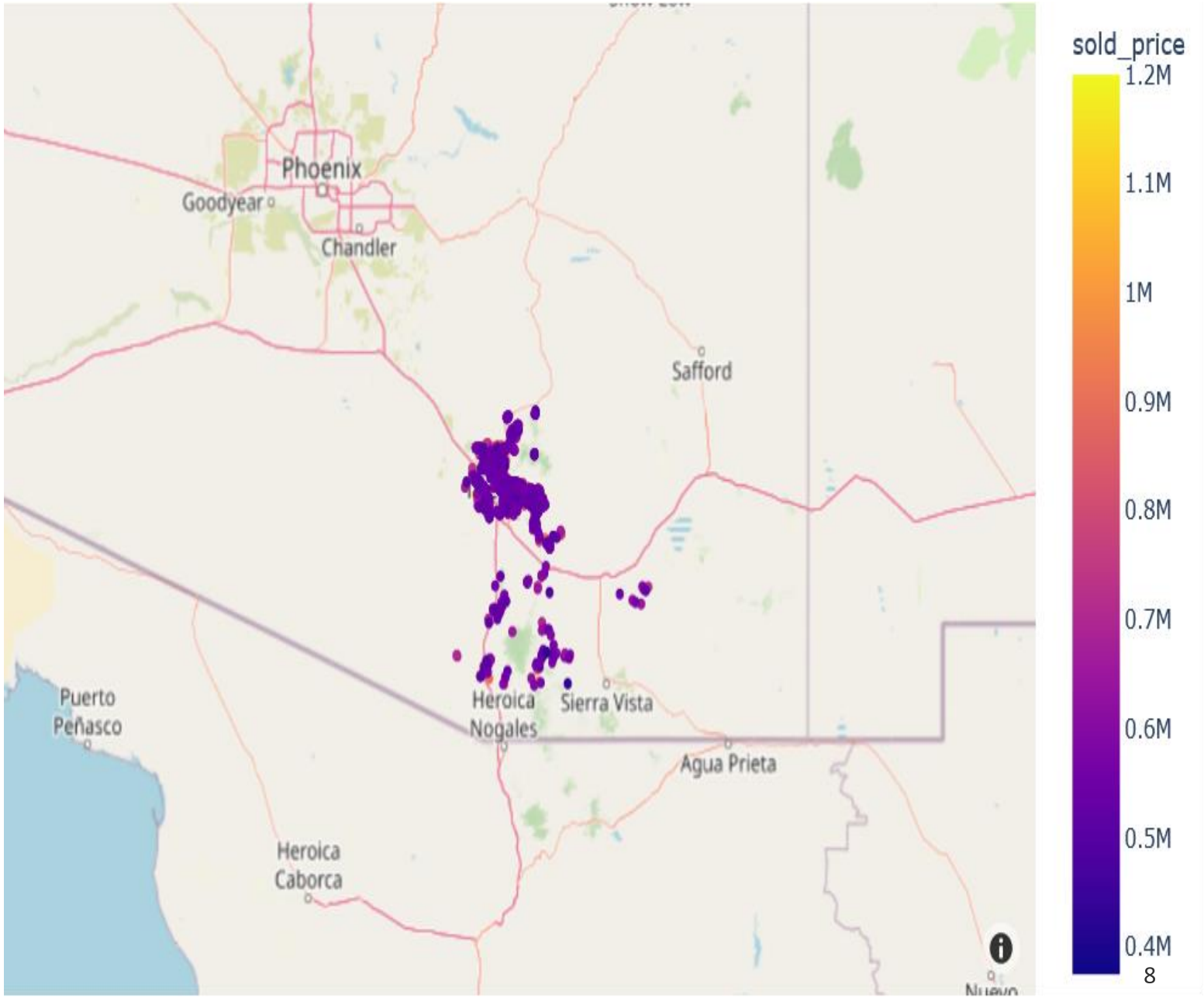


The Cleaned Data

- Dropped "MLS", "kitchen_features" and "floor_covering" columns.
- Filled null and empty values with suitable values.
- Cleaned Outliers and messy values.
- Created "age" column based on 2019.
- The shape of the cleaned data : (4282,13)

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 4282 entries, 217 to 4998
Data columns (total 13 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   sold_price   4282 non-null   float64
 1   zipcode      4282 non-null   int64  
 2   longitude    4282 non-null   float64
 3   latitude     4282 non-null   float64
 4   lot_acres    4282 non-null   float64
 5   taxes        4282 non-null   float64
 6   bedrooms     4282 non-null   int64  
 7   bathrooms    4282 non-null   float64
 8   sqrt_ft      4282 non-null   float64
 9   garage       4282 non-null   float64
 10  fireplaces   4282 non-null   int64  
 11  HOA          4282 non-null   float64
 12  age          4282 non-null   int64  
dtypes: float64(9), int64(4)
memory usage: 468.3 KB
```

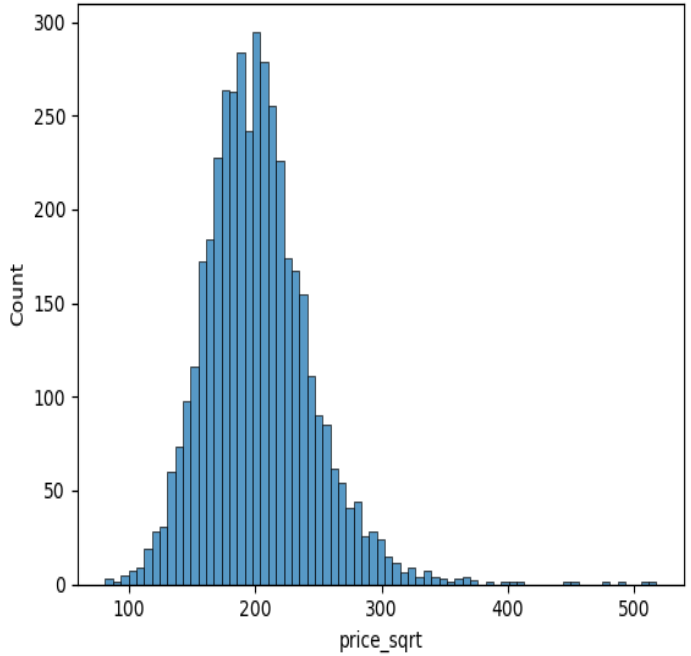
Distribution of the houses on the map : After cleaned the data



Feature Engineering :

- Created "price_sqrt" column
- Price_sqrt=sold_price/sqrt_ft
- price_sqrt : distribution between 100 and 300

price_sqrt
213.013168
265.588915
271.125169
340.136054
276.580247
264.534884
240.269101
311.284047
277.200277
287.838810



Feature Engineering :

- Divided it into ten equal groups according to the values of price_sqrt
- Numbered these groups from 1 to 10.
- For example:

  81-153 -> 1

  53-168 -> 2

  .

  .

  253-517 -> 10

1	430
10	429
7	428
9	428
5	428
6	428
3	428
8	428
4	428
2	427

price_sqrt	category
213.013168	7
265.588915	10
271.125169	10
340.136054	10
276.580247	10
264.534884	10
240.269101	9
311.284047	10
277.200277	10
287.838810	10

Classification : KNN

- Accuracy : 0.87
- K=3

```
2  :  0.992760392340028
3  :  0.8727230266230733
4  :  0.7372723026623074
5  :  0.6270434376459598
6  :  0.5558150397010743
7  :  0.5114432508173751
8  :  0.4885567491826249
9  :  0.4677720691265764
10  :  0.4586641756188697
```

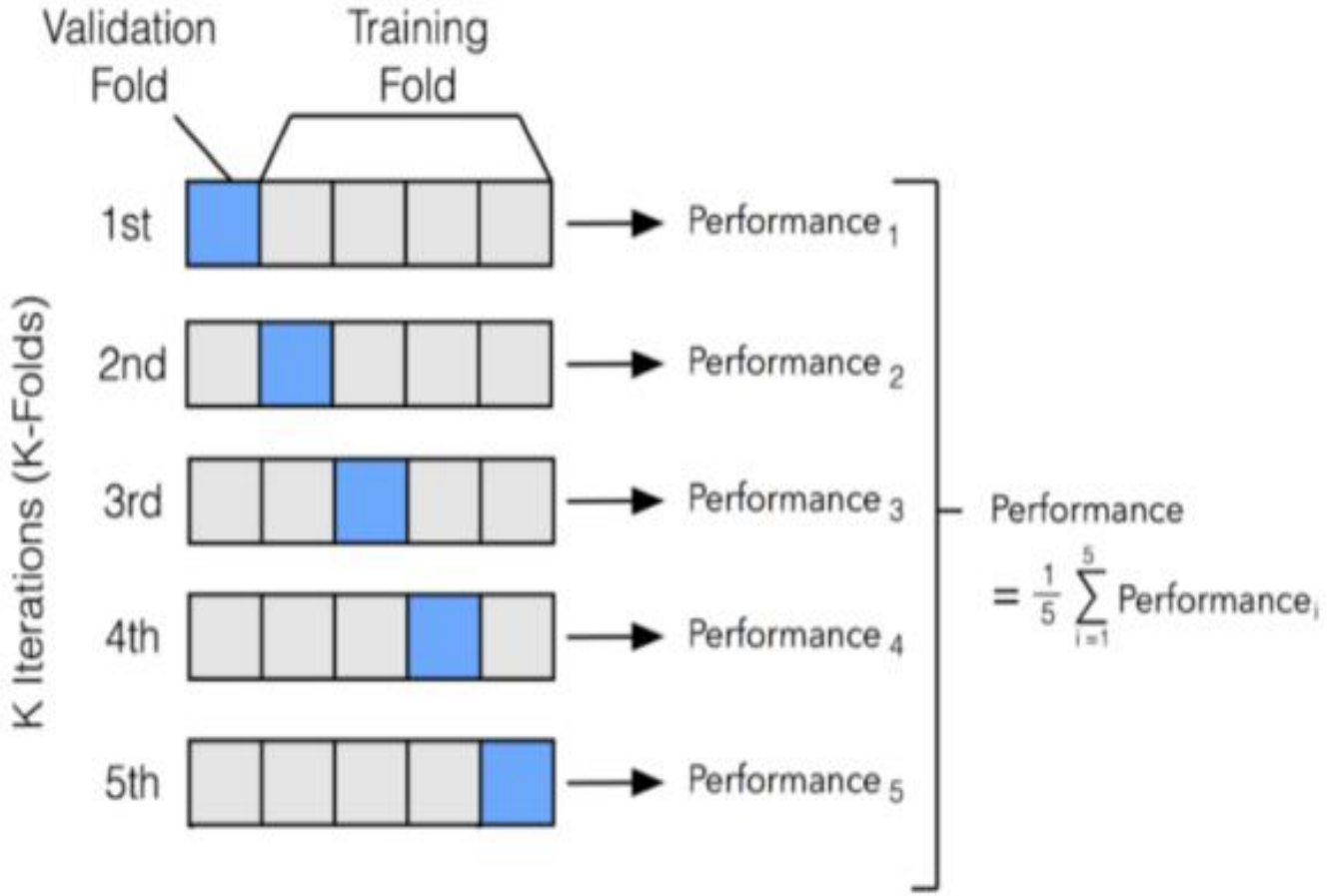
	longitude	latitude	category	Prediction
999	-110.908102	32.322269	3	3.0
2153	-110.954881	32.313543	5	5.0
2396	-110.678130	32.100513	3	3.0
492	-110.990511	32.465389	8	8.0
930	-110.860817	32.544615	7	4.0

Model Building : Linear Regression

- Split the data as train and test data.
- Train size : (3853, 10)
-
- Test Size : (429, 10)
- Normalized the data.
- Trained the model with the train data.
-

	sold_price	lot_acres	taxes	bedrooms	bathrooms	sqrt_ft	garage	fireplaces	HOA	age	category
0	1100000.0	1.39	19156.00	4	4.0	5164.0	3.0	4	153.0	18	7
1	1150000.0	11.49	10716.00	4	5.0	4330.0	3.0	3	0.0	12	10
2	1200000.0	1.21	10610.26	4	4.0	4426.0	3.0	0	117.0	17	10
3	1200000.0	7.11	7324.90	3	3.0	3528.0	2.0	1	50.0	47	10
4	1120150.0	2.98	13573.00	4	6.0	4050.0	4.0	2	149.0	12	10

Cross Validation

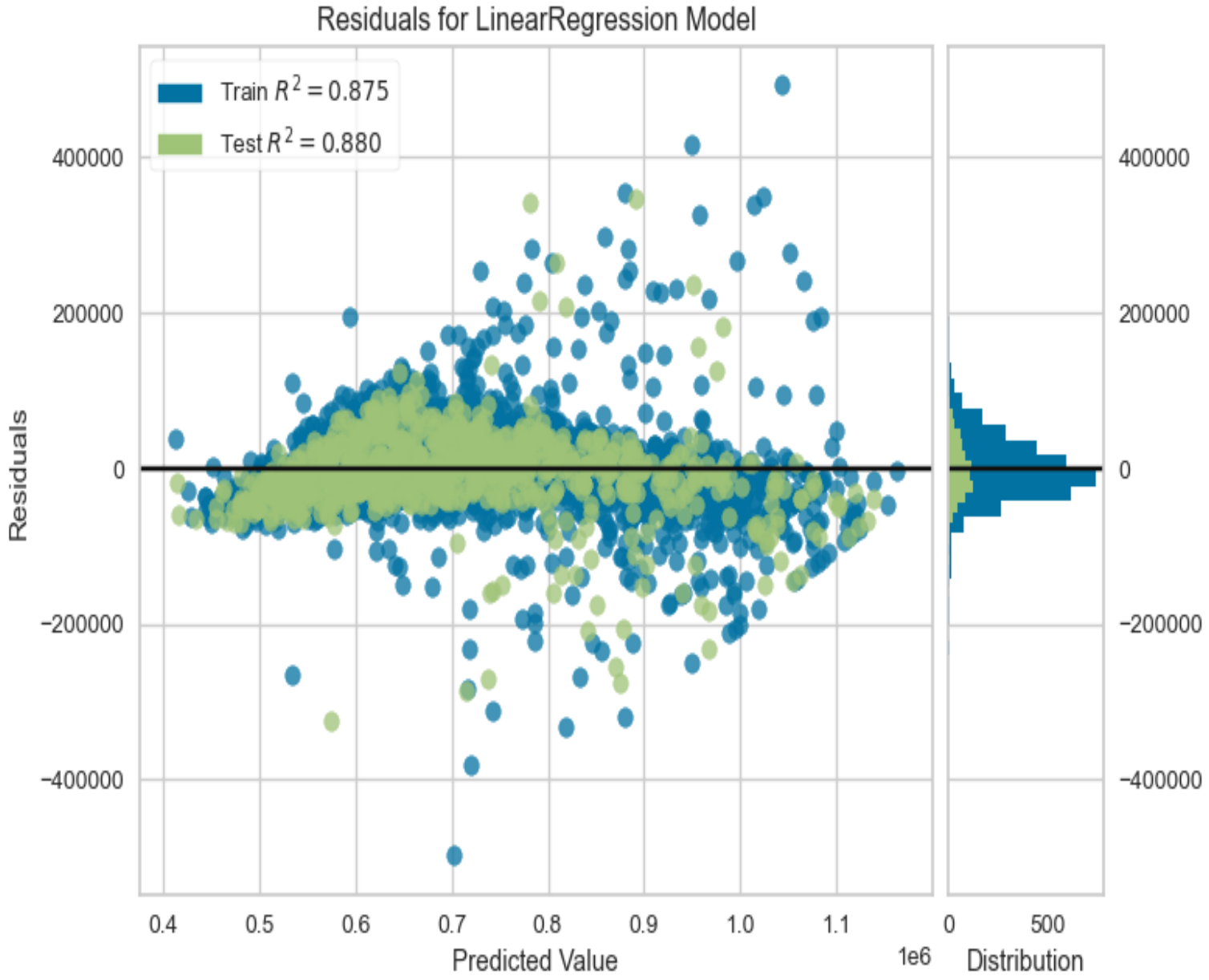


The Model Performance

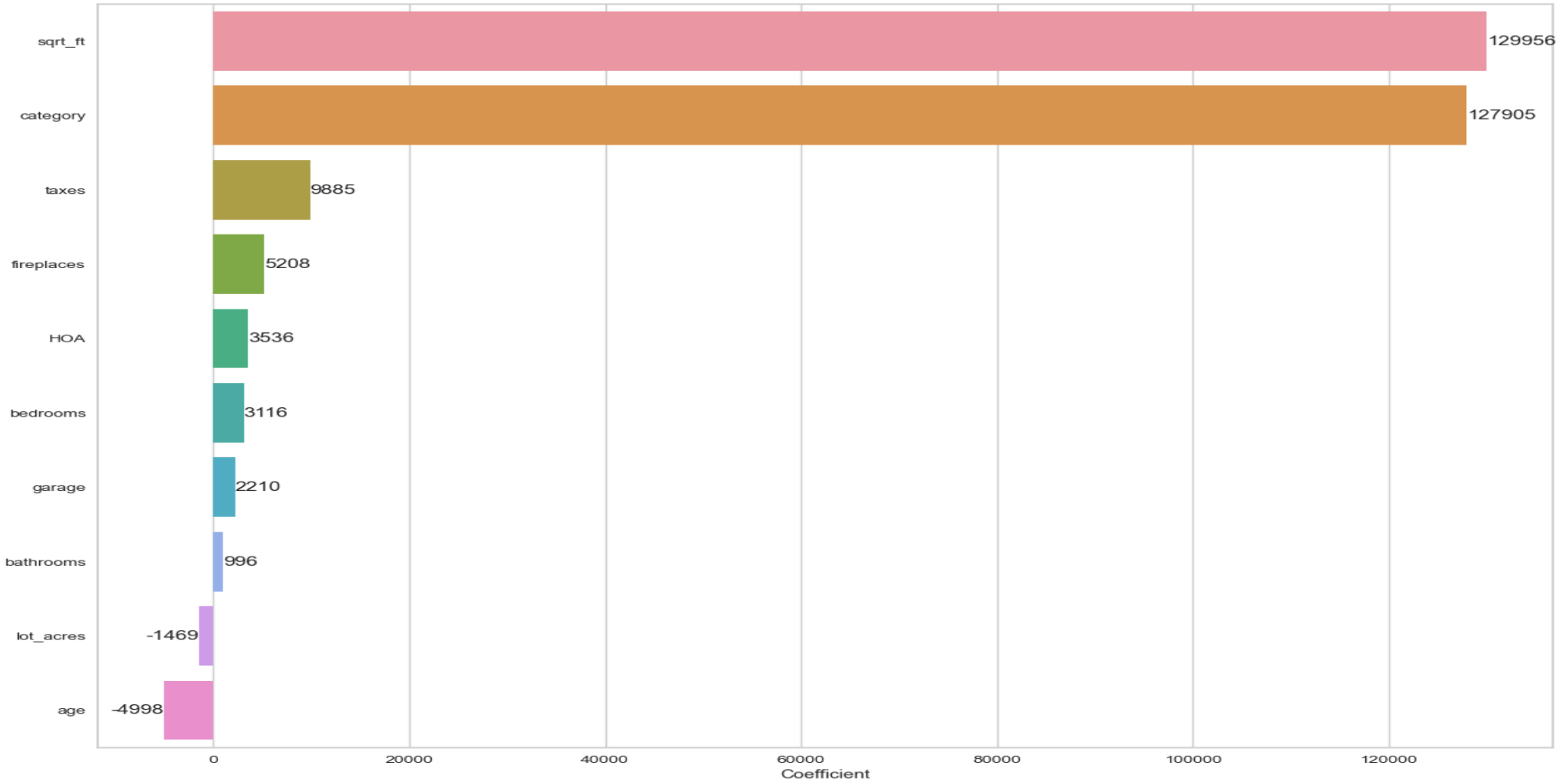
- R2 score of the train data : 0.875
- R2 score of the cross validation : 0.874
- R2 score of the test data: 0.88
- Train and validation score are almost the same so there is not overfitting.

test_score
0.871261
0.886177
0.894673
0.858241
0.859653

Distribution of the Residuals



Coefficients



Prediction

	longitude	latitude	lot_acres	taxes	bedrooms	bathrooms	sqrt_ft	garage	fireplaces	HOA	age	Price
1182	-110.947012	32.398618	0.65	5096.00	3	4.0	3927.0	2.0	2	60.0	13	775000.0
4038	-110.981701	32.387787	0.65	4945.49	5	5.0	3300.0	2.0	1	7.0	50	530000.0
2610	-111.092245	32.461711	0.59	7237.00	3	3.0	3052.0	2.0	2	76.0	14	575000.0
2017	-110.851361	32.311857	1.00	4658.71	3	4.0	3933.0	2.0	3	37.0	34	675000.0
644	-110.849321	32.320880	0.51	7280.24	4	3.0	3498.0	3.0	2	117.0	24	870000.0
2294	-110.853531	32.289969	0.80	6333.47	3	2.0	2964.0	2.0	1	130.0	33	659200.0
3524	-110.715703	32.247349	3.33	4732.00	3	4.0	3058.0	3.0	1	25.0	26	544366.0
3464	-110.925499	32.334862	0.83	4032.99	2	2.0	1925.0	2.0	2	6.0	27	575000.0
3088	-110.914905	32.304747	1.41	5097.29	3	3.0	2740.0	2.0	2	0.0	39	590000.0
2087	-110.925584	32.462017	1.00	6411.72	4	3.0	3533.0	3.0	2	0.0	13	655000.0

Prediction : Category

- Predicted categories with KNN.
- Drop longitude and Latitude columns and add category column to the sample data.

	longitude	latitude	Category
1182	-110.947012	32.398618	5.0
4038	-110.981701	32.387787	2.0
2610	-111.092245	32.461711	4.0
2017	-110.851361	32.311857	3.0
644	-110.849321	32.320880	9.0
2294	-110.853531	32.289969	8.0
3524	-110.715703	32.247349	3.0
3464	-110.925499	32.334862	10.0
3088	-110.914905	32.304747	7.0
2087	-110.925584	32.462017	4.0

Prediction

	lot_acres	taxes	bedrooms	bathrooms	sqrt_ft	garage	fireplaces	HOA	age	Category
1182	0.65	5096.00	3	4.0	3927.0	2.0	2	60.0	13	5.0
4038	0.65	4945.49	5	5.0	3300.0	2.0	1	7.0	50	2.0
2610	0.59	7237.00	3	3.0	3052.0	2.0	2	76.0	14	4.0
2017	1.00	4658.71	3	4.0	3933.0	2.0	3	37.0	34	3.0
644	0.51	7280.24	4	3.0	3498.0	3.0	2	117.0	24	9.0
2294	0.80	6333.47	3	2.0	2964.0	2.0	1	130.0	33	8.0
3524	3.33	4732.00	3	4.0	3058.0	3.0	1	25.0	26	3.0
3464	0.83	4032.99	2	2.0	1925.0	2.0	2	6.0	27	10.0
3088	1.41	5097.29	3	3.0	2740.0	2.0	2	0.0	39	7.0
2087	1.00	6411.72	4	3.0	3533.0	3.0	2	0.0	13	4.0

Prediction : Price

	lot_acres	taxes	bedrooms	bathrooms	sqrt_ft	garage	fireplaces	HOA	age	Category	Price	Prediction
1182	0.65	5096.00	3	4.0	3927.0	2.0	2	60.0	13	5.0	775000.0	737634.004123
4038	0.65	4945.49	5	5.0	3300.0	2.0	1	7.0	50	2.0	530000.0	486913.651799
2610	0.59	7237.00	3	3.0	3052.0	2.0	2	76.0	14	4.0	575000.0	552296.839704
2017	1.00	4658.71	3	4.0	3933.0	2.0	3	37.0	34	3.0	675000.0	642769.777007
644	0.51	7280.24	4	3.0	3498.0	3.0	2	117.0	24	9.0	870000.0	854752.218043
2294	0.80	6333.47	3	2.0	2964.0	2.0	1	130.0	33	8.0	659200.0	703247.084080
3524	3.33	4732.00	3	4.0	3058.0	3.0	1	25.0	26	3.0	544366.0	493676.393543
3464	0.83	4032.99	2	2.0	1925.0	2.0	2	6.0	27	10.0	575000.0	607956.431375
3088	1.41	5097.29	3	3.0	2740.0	2.0	2	0.0	39	7.0	590000.0	614551.020126
2087	1.00	6411.72	4	3.0	3533.0	3.0	2	0.0	13	4.0	655000.0	631502.865722

Conclusion

- Achieved a performance of 88 percent with linear regression.
- Means that our model knows house prices with a margin of error of 12 percent.
- For example, if the real price of the house is 1.000.000, the model estimates it between 880.000 and 1.120.000.
- This result is not bad but better performance can be achieved with bagging and boosting models like random forest and XGBoost.


Thank you for listening...