Housing Price Prediction

Team Polaris: David Bartlett, Kyndall Kelly, Rufin Perez, Kristina Swanson, Toby Wong

Proposal

Overview

Prediction of Housing Prices based on different features including area, bedrooms, bathrooms, stories, mainroad, guestroom, basement, hot water heating, air conditioning, parking, preferential area, and furnishing.

Goals

1. Train a machine learning model using supervised learning to be able to input information on a house and have the model output a price.
2. Create an analysis based on the model in a professional presentation

SQL Database

Query   Query History

```sql
CREATE TABLE Housing(
    price NUMERIC,
    area NUMERIC,
    bedrooms INTEGER,
    bathrooms INTEGER,
    stories INTEGER,
    parking INTEGER,
    mainroad_yes BOOLEAN,
    guestroom_yes BOOLEAN,
    basement_yes BOOLEAN,
    hotwaterheating_yes BOOLEAN,
    airconditioning_yes BOOLEAN,
    prefarea_yes BOOLEAN,
    furnishingstatus_semi_furnished BOOLEAN,
    furnishingstatus_unfurnished BOOLEAN
);
```

Data Output   Messages   Notifications

```
CREATE TABLE

Query returned successfully in 55 msec.
```

Project4/postgres@PostgreSQL 16

No limit

Data Output   Messages   Notifications

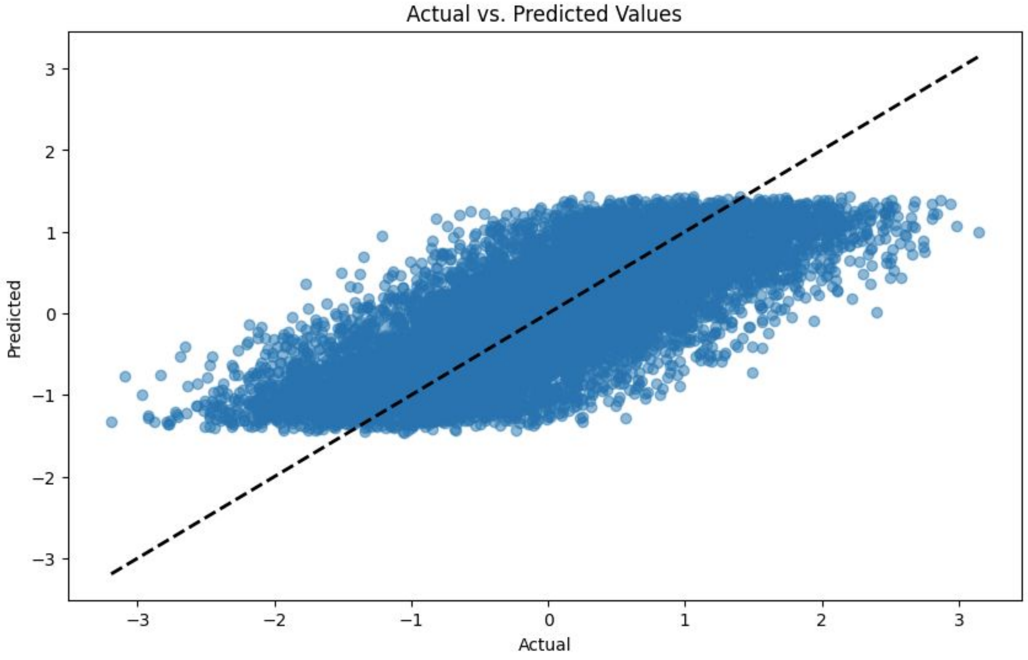
	price integer	area integer	bedrooms integer	bathrooms integer	stories integer	parking integer	mainroad_yes boolean	guestroom_yes boolean	basement_yes boolean	hotwaterheating_yes boolean	airconditioning_yes boolean	prefarea_yes boolean	furnishingstatus_semi_furnished boolean	furnishing boolean
1	13300000	7420	4	2	3	2	true	false	false	false	true	true	false	false
2	12250000	8960	4	4	4	3	true	false	false	false	true	false	false	false
3	12250000	9960	3	2	2	2	true	false	true	false	false	true	true	false
4	12215000	7500	4	2	2	3	true	false	true	false	true	true	false	false
5	11410000	7420	4	1	2	2	true	true	true	false	true	false	false	false
6	10850000	7500	3	3	1	2	true	false	true	false	true	true	true	false
7	10150000	8580	4	3	4	2	true	false	false	false	true	true	true	false
8	10150000	16200	5	3	2	0	true	false	false	false	false	false	false	true
9	9870000	8100	4	1	2	2	true	true	true	false	true	true	false	false
10	9800000	5750	3	2	4	1	true	true	false	false	true	true	false	true
11	9800000	13200	3	1	2	2	true	false	true	false	true	true	false	false
12	9681000	6000	4	3	2	2	true	true	true	true	false	false	true	false
13	9310000	6550	4	2	2	1	true	false	false	false	true	true	true	false
14	9240000	3500	4	2	2	2	true	false	false	true	false	false	false	false
15	9240000	7800	3	2	2	0	true	false	false	false	false	true	true	false
16	9100000	6000	4	1	2	2	true	false	true	false	false	false	true	false
17	9100000	6600	4	2	2	1	true	true	true	false	true	true	false	true
18	8960000	8500	3	2	4	2	true	false	false	false	true	false	false	false
19	8890000	4600	3	2	2	2	true	true	false	false	true	false	false	false
20	8855000	6420	3	2	2	1	true	false	false	false	true	true	true	false
21	8750000	4320	3	1	2	2	true	false	true	true	false	false	true	false
22	8680000	7155	3	2	1	2	true	true	true	false	true	false	false	true
23	8645000	8050	3	1	1	1	true	true	true	false	true	false	false	false
24	8645000	4560	3	2	2	1	true	true	true	false	true	false	false	false
25	8575000	8800	3	2	2	2	true	false	false	false	true	false	false	false
26	8540000	6540	4	2	2	2	true	true	true	false	true	true	false	false
27	8463000	6000	3	2	4	0	true	true	true	false	true	true	true	false
28	8400000	8875	3	1	1	1	true	false	false	false	false	false	true	false
29	8400000	7950	5	2	2	2	true	false	true	true	false	false	false	true
30	8400000	5500	4	2	2	1	true	false	true	false	true	true	true	false

Total rows: 1000 of 80545   Query complete 00:00:00.206   Ln 1, Col 22

Early Prediction Model

Linear Regression

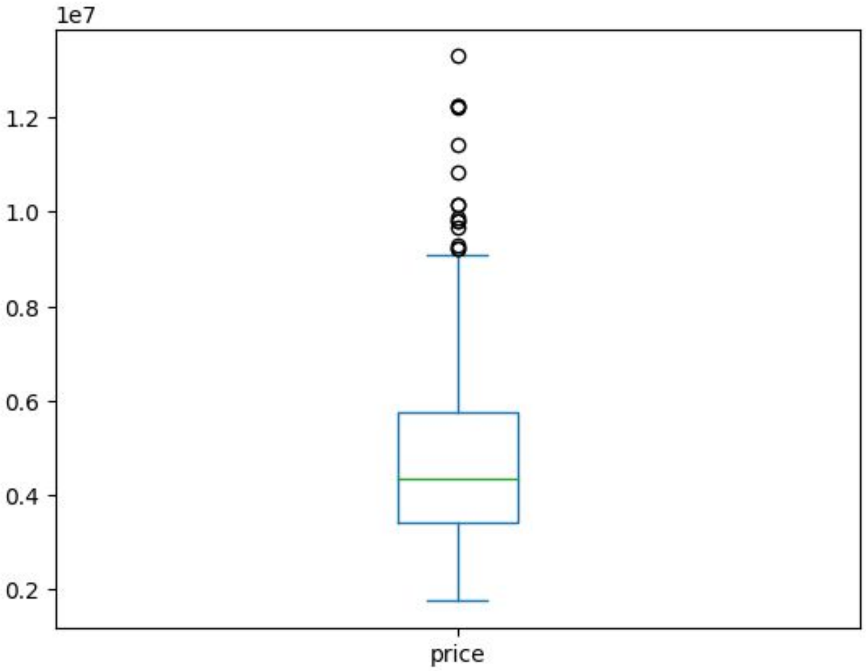
```
Model Performance:
Mean Absolute Error (MAE): 0.5178566489840383
Mean Squared Error (MSE): 0.4202267139221315
R-squared (R2): 0.5755628630306234
```

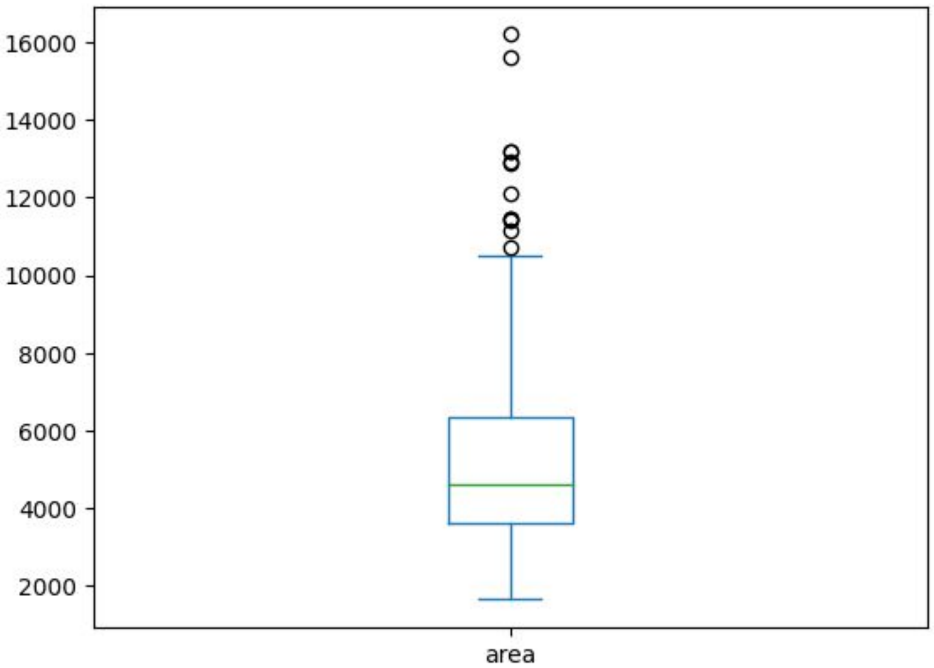


Additional Prediction Models

	MAE	MSE	R2
Random Forest Regression	1.028543e+06	1.952886e+12	0.613640
Ridge Regression	9.697663e+05	1.755104e+12	0.652769
Lasso Regression	9.700434e+05	1.754321e+12	0.652924
Elastic Net Regression	9.665317e+05	1.786560e+12	0.646546
Linear Regression	9.700434e+05	1.754319e+12	0.652924

Outliers


1e7
1.2
1.0
0.8
0.6
0.4
0.2
price


16000
14000
12000
10000
8000
6000
4000
2000
area

dummyData.py

We used “dummyData.py” to generate additional rows based on the existing parameters of the original dataset, to keep the data proportioned and varied, including reducing the impact of outliers.

We then concatenated the original ‘df’ and ‘dummy_df’ which increased the dataset from 545 rows to 80,545 rows.

```python
import random

def generate_dummy_data(n):
    dummy_data = []
    gen_data = {
        "price": 175000,
        "area": 1650,
        "bedrooms": [1,2,3,4,5],
        "bathrooms": [1,2,3,4],
        "stories": [1,2,3,4],
        "mainroad": ["Yes", "No"],
        "guestroom": ["Yes", "No"],
        "basement": ["Yes", "No"],
        "hotwaterheating": ["Yes", "No"],
        "airconditioning": ["Yes", "No"],
        "parking": [0,1,2,3,4],
        "prefarea": ["Yes", "No"],
    }

    # loop through n times and randomly generate data by selecting from gen_data
    for _ in range(n):
        house = {}
        for key, value in gen_data.items():

            if key == "price":
                house[key] = random.randint(175000, 1500000)
            elif key == "area":
                house[key] = random.randint(1650, 17000)
            else:
                house[key] = random.choice(value)

        dummy_data.append(house)

    return dummy_data
```

Exploring Other Models

MODEL	R^2 SCORE (%)
Ridge Regression	81.61
Lasso Regression	81.61
Linear Regression	81.61
Elastic Net Regression	82.17
Ada Boost Regression	93.43
Gradient Boost Regressor	96.05
Random Forest Regression	96.06

Decision Tree Regression Model

Using original data:

```
Decision Tree Regression Model
R^2 accuracy score: 0.5127835163775086
```

Using dummy Data:

```
Decision Tree Regression Model
R^2 accuracy score: 0.9249935545705076
```

Value Prediction

- Predictions with custom inputs

```
example = [[7420, 4, 2, 3, 2, 1, 0, 0, 0, 1, 1, 0, 0]]
scaled_example = scaler.transform(example)
y_pred = model.predict(scaled_example)
print("Predicted Housing price:", y_pred[0])
```

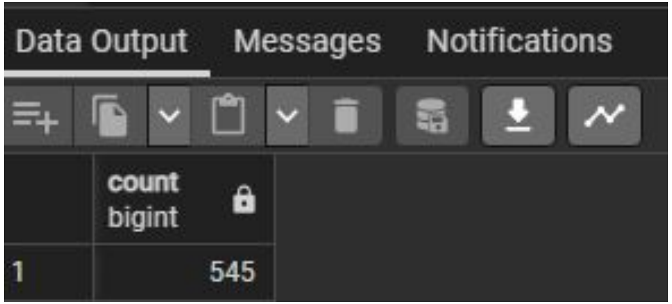
```
Predicted Housing price: 13300000.0
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439:
  warnings.warn(
```

```
example1 = [[8960, 4, 4, 4, 3, 1, 0, 0, 0, 1, 0, 0, 0]]
scaled_example1 = scaler.transform(example1)
y_pred1 = model.predict(scaled_example1)
print("Predicted Housing price:", y_pred1[0])
```

```
Predicted Housing price: 12250000.0
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439:
  warnings.warn(
```

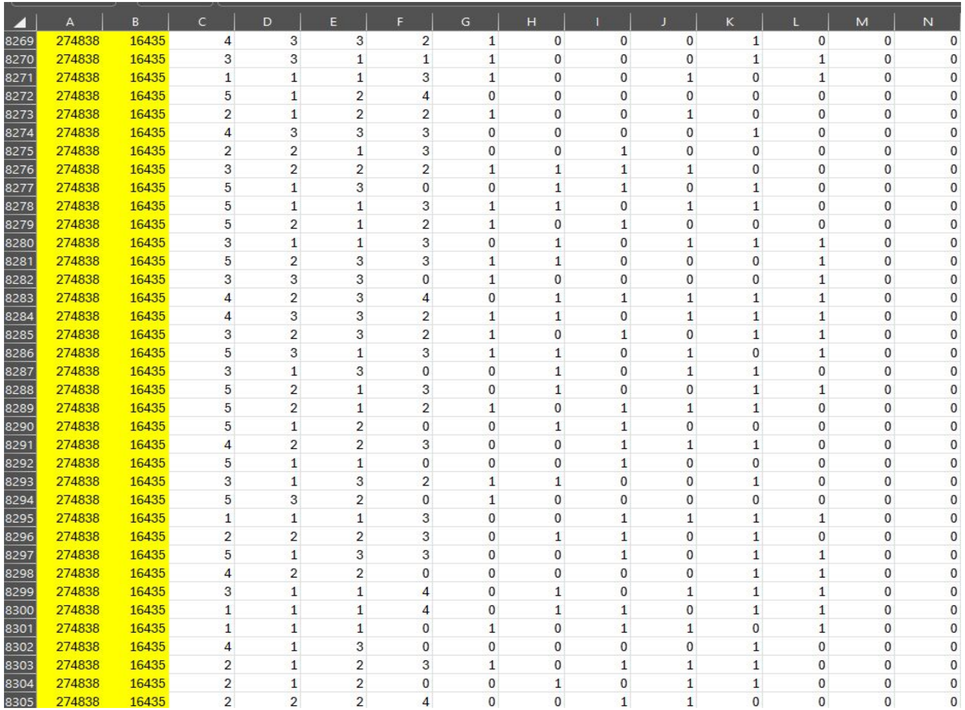
Last-second Discovery

Dataset size



	count bigint
1	545

Dummy Data Issues



	A	B	C	D	E	F	G	H	I	J	K	L	M	N
8269	274838	16435	4	3	3	2	1	0	0	0	1	0	0	0
8270	274838	16435	3	3	1	1	1	0	0	0	1	1	0	0
8271	274838	16435	1	1	1	3	1	0	0	1	0	1	0	0
8272	274838	16435	5	1	2	4	0	0	0	0	0	0	0	0
8273	274838	16435	2	1	2	2	1	0	0	1	0	0	0	0
8274	274838	16435	4	3	3	3	0	0	0	0	1	0	0	0
8275	274838	16435	2	2	1	3	0	0	1	0	0	0	0	0
8276	274838	16435	3	2	2	2	1	1	1	1	0	0	0	0
8277	274838	16435	5	1	3	0	0	1	1	0	1	0	0	0
8278	274838	16435	5	1	1	3	1	1	0	1	1	0	0	0
8279	274838	16435	5	2	1	2	1	0	1	0	0	0	0	0
8280	274838	16435	3	1	1	3	0	1	0	1	1	1	0	0
8281	274838	16435	5	2	3	3	1	1	0	0	0	1	0	0
8282	274838	16435	3	3	3	0	1	0	0	0	0	1	0	0
8283	274838	16435	4	2	3	4	0	1	1	1	1	1	0	0
8284	274838	16435	4	3	3	2	1	1	0	1	1	1	0	0
8285	274838	16435	3	2	3	2	1	0	1	0	1	1	0	0
8286	274838	16435	5	3	1	3	1	1	0	1	0	1	0	0
8287	274838	16435	3	1	3	0	0	1	0	1	1	0	0	0
8288	274838	16435	5	2	1	3	0	1	0	0	1	1	0	0
8289	274838	16435	5	2	1	2	1	0	1	1	1	0	0	0
8290	274838	16435	5	1	2	0	0	1	1	0	0	0	0	0
8291	274838	16435	4	2	2	3	0	0	1	1	1	0	0	0
8292	274838	16435	5	1	1	0	0	0	1	0	0	0	0	0
8293	274838	16435	3	1	3	2	1	1	0	0	1	0	0	0
8294	274838	16435	5	3	2	0	1	0	0	0	0	0	0	0
8295	274838	16435	1	1	1	3	0	0	1	1	1	1	0	0
8296	274838	16435	2	2	2	3	0	1	1	0	1	0	0	0
8297	274838	16435	5	1	3	3	0	0	1	0	1	1	0	0
8298	274838	16435	4	2	2	0	0	0	0	0	1	1	0	0
8299	274838	16435	3	1	1	4	0	1	0	1	1	1	0	0
8300	274838	16435	1	1	1	4	0	1	1	0	1	1	0	0
8301	274838	16435	1	1	1	0	1	0	1	1	0	1	0	0
8302	274838	16435	4	1	3	0	0	0	0	0	1	0	0	0
8303	274838	16435	2	1	2	3	1	0	1	1	1	0	0	0
8304	274838	16435	2	1	2	0	0	1	0	1	1	0	0	0
8305	274838	16435	2	2	2	4	0	0	1	1	0	0	0	0

Solution

	MAE	MSE	R2
Ridge Regression	158533.276754	4.596695e+10	0.808472
Lasso Regression	158533.370861	4.596730e+10	0.808470
Elastic Net Regression	158483.030049	4.583445e+10	0.809024
Linear Regression	158515.523412	4.594670e+10	0.808556

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
538	1960000	3420	5	1	2	0	0	1	0	0	1	0	0	1
539	1890000	1700	3	1	2	0	0	0	1	0	1	0	0	1
540	1890000	3649	2	1	1	0	0	0	1	0	1	0	0	1
541	1855000	2990	2	1	1	1	0	1	0	0	1	0	0	1
542	1820000	3000	2	1	1	2	0	0	1	0	1	0	0	0
543	1767150	2400	3	1	1	0	0	1	0	0	1	0	0	1
544	1750000	3620	2	1	1	0	0	0	1	0	1	0	0	1
545	1750000	2910	3	1	1	0	0	1	0	0	1	0	0	1
546	1750000	3850	3	1	2	0	0	0	1	0	1	0	0	1
547	575000	9000	2	2	1	4	0	0	0	1	0	0	1	0
548	200000	4500	2	2	2	0	0	0	0	1	0	0	1	0
549	575000	12500	4	1	1	4	1	0	0	0	0	0	1	0
550	475000	15000	3	3	3	1	0	0	0	1	0	0	0	0
551	525000	7500	3	3	3	1	0	0	0	1	0	0	0	0
552	30000	2500	4	2	4	4	0	0	0	1	0	0	0	0
553	475000	9000	5	1	2	0	1	0	0	0	0	0	0	0
554	275000	2000	1	1	1	0	1	0	0	1	0	0	0	0
555	550000	12500	2	1	2	4	1	0	0	1	0	0	1	0
556	550000	11500	2	2	2	2	1	0	0	0	0	0	1	0
557	475000	15000	5	4	2	4	1	0	0	0	0	0	0	0
558	475000	1650	1	1	1	2	0	0	0	0	0	0	1	0
559	500000	6500	2	2	1	0	1	0	0	1	0	0	0	0
560	275000	12000	4	2	1	2	0	0	0	1	0	0	1	0
561	600000	1650	4	3	4	1	1	0	0	1	0	0	0	0
562	325000	9500	2	2	2	4	1	0	0	1	0	0	1	0
563	600000	14000	2	2	2	0	1	0	0	1	0	0	0	0
564	475000	7000	3	3	1	1	0	0	0	0	0	0	0	0
565	475000	6500	1	1	1	4	1	0	0	0	0	0	0	0
566	250000	2500	5	1	1	2	1	0	0	1	0	0	0	0
567	325000	12500	1	1	1	3	0	0	0	1	0	0	1	0
568	200000	7500	4	3	3	0	0	0	0	0	0	0	1	0
569	200000	14500	2	2	2	0	0	0	0	1	0	0	1	0
570	30000	4500	2	2	2	2	1	0	0	1	0	0	1	0
571	575000	12000	2	2	2	4	1	0	0	0	0	0	0	0

By making adjustments to “dummyData.py”, the initial problem in the columns for “price” and “area” were able to be resolved by tuning the “dummyData” generator for each “price” and “area” to make the values proportioned with the original dataset.

Sources

Housing Price Prediction by Harish Kumardatalab
https://www.kaggle.com/datasets/harishkumardatalab/housing-price-prediction

https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html

https://deepai.org/machine-learning-glossary-and-terms/dummy-variable#:~:text=Dummy%20variables%20are%20binary%20