d7nxpi1kc

January 28, 2025

1 Task 1 - House Price Prediction

[1]:
```
# import the necc. lib.

import pandas as pd  # for reading the data, data manipulation
import numpy as np   # for numerical computations
import matplotlib.pyplot as plt  # for data visualization
import seaborn as sns  # for data visualization
from sklearn.model_selection import train_test_split  # for train & test the␣
 ↪model
```

[2]:
```
# using historical dataset (housing dataset)

df = pd.read_csv('/content/Housing.csv')
df.head()
```

[2]:
```
      price  area  bedrooms  bathrooms  stories mainroad guestroom basement  \
0  13300000  7420         4          2        3      yes        no       no
1  12250000  8960         4          4        4      yes        no       no
2  12250000  9960         3          2        2      yes        no      yes
3  12215000  7500         4          2        2      yes        no      yes
4  11410000  7420         4          1        2      yes       yes      yes

  hotwaterheating airconditioning  parking prefarea furnishingstatus
0              no             yes        2      yes        furnished
1              no             yes        3       no        furnished
2              no              no        2      yes   semi-furnished
3              no             yes        3      yes        furnished
4              no             yes        2       no        furnished
```

[3]:
```
# check the null values
df.isnull().sum()
```

[3]:
```
price               0
area                0
bedrooms            0
bathrooms           0
```

```
stories             0
mainroad            0
guestroom           0
basement            0
hotwaterheating     0
airconditioning     0
parking             0
prefarea            0
furnishingstatus    0
dtype: int64
```

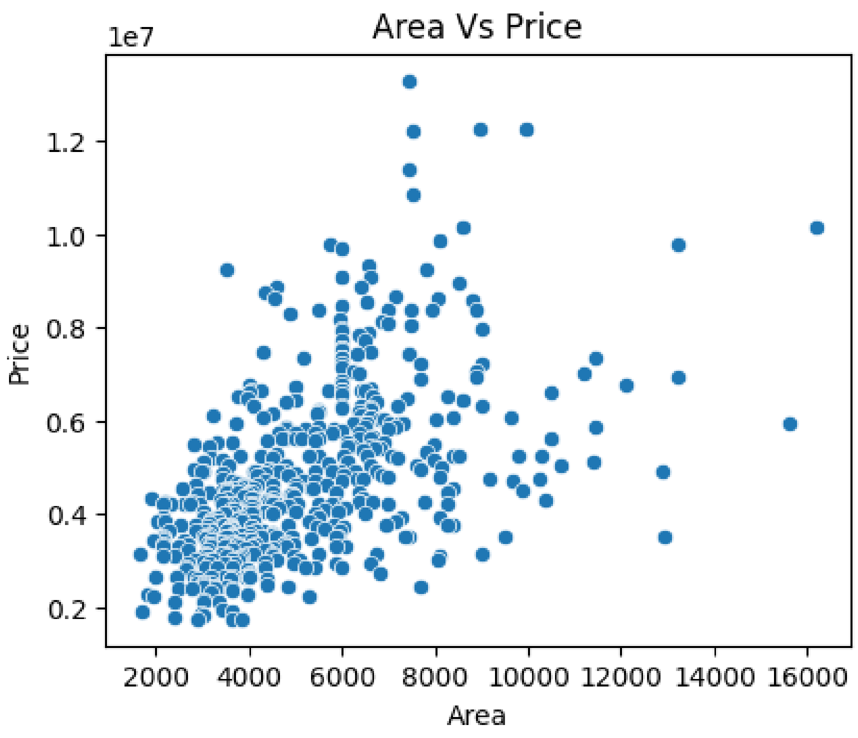
[ ]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 545 entries, 0 to 544
Data columns (total 13 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   price             545 non-null    int64
 1   area              545 non-null    int64
 2   bedrooms          545 non-null    int64
 3   bathrooms         545 non-null    int64
 4   stories           545 non-null    int64
 5   mainroad          545 non-null    object
 6   guestroom         545 non-null    object
 7   basement          545 non-null    object
 8   hotwaterheating   545 non-null    object
 9   airconditioning   545 non-null    object
 10  parking           545 non-null    int64
 11  prefarea          545 non-null    object
 12  furnishingstatus  545 non-null    object
dtypes: int64(6), object(7)
memory usage: 55.5+ KB
```

[ ]:
```
correlation = df['price'].corr(df['area'])
correlation
```

[ ]: 0.5359973457780798

[ ]:
```
# plot the scatter plot

plt.figure(figsize = (5,4))
sns.scatterplot(x = 'area', y = 'price', data = df)
plt.xlabel('Area')
plt.ylabel('Price')
plt.title('Area Vs Price')
plt.show()
```



1.0.1 there is a positive correlation in the above scatterplot.

1.0.2 indicating that as the area of house increases, its price tends to increase as well.

1.0.3 there are some outliers present in it,representing large houses(area) with the relatively low prices.

1.1 Label Encoding

```
[4]: enc_col = ['mainroad', 'guestroom', 'basement', 'hotwaterheating',␣
      ↪'airconditioning', 'prefarea', 'furnishingstatus']
```

```
[5]: from sklearn.preprocessing import LabelEncoder

     label_encoders = {}
     for col in enc_col:
       le = LabelEncoder()
       df[col] = le.fit_transform(df[col])
       label_encoders[col] = le

     df.head()
```

```
[5]:      price  area  bedrooms  bathrooms  stories  mainroad  guestroom  \
     0  13300000  7420         4          2        3         1          0
```

```
1  12250000  8960         4          4        4          1          0
2  12250000  9960         3          2        2          1          0
3  12215000  7500         4          2        2          1          0
4  11410000  7420         4          1        2          1          1

   basement  hotwaterheating  airconditioning  parking  prefarea  \
0         0                0                1        2         1
1         0                0                1        3         0
2         1                0                0        2         1
3         1                0                1        3         1
4         1                0                1        2         0

   furnishingstatus
0                 0
1                 0
2                 1
3                 0
4                 0
```

2 Data Preparation

[6]:
```
X = df.drop('price', axis = 1)
X
```

[6]:
```
     area  bedrooms  bathrooms  stories  mainroad  guestroom  basement  \
0    7420         4          2        3         1          0         0
1    8960         4          4        4         1          0         0
2    9960         3          2        2         1          0         1
3    7500         4          2        2         1          0         1
4    7420         4          1        2         1          1         1
..    …        …         …       …        …         …
540  3000         2          1        1         1          0         1
541  2400         3          1        1         0          0         0
542  3620         2          1        1         1          0         0
543  2910         3          1        1         0          0         0
544  3850         3          1        2         1          0         0

     hotwaterheating  airconditioning  parking  prefarea  furnishingstatus
0                  0                1        2         1                 0
1                  0                1        3         0                 0
2                  0                0        2         1                 1
3                  0                1        3         1                 0
4                  0                1        2         0                 0
..               …              …       …       …                …
540                0                0        2         0                 2
541                0                0        0         0                 1
```

```
542                 0                 0        0          0                  2
543                 0                 0        0          0                  0
544                 0                 0        0          0                  2

[545 rows x 12 columns]
```

```
[8]: y = df['price']
     y
```

```
[8]: 0       13300000
     1       12250000
     2       12250000
     3       12215000
     4       11410000
               …
     540      1820000
     541      1767150
     542      1750000
     543      1750000
     544      1750000
     Name: price, Length: 545, dtype: int64
```

```
[ ]:
```

```
[9]: # train_test
     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.4,␣
      ↪random_state = 47)
     X_train
```

```
[9]:      area  bedrooms  bathrooms  stories  mainroad  guestroom  basement  \
     336  8080         3          1        1         1          0         0
     74   4040         3          1        2         1          0         1
     121  7231         3          1        2         1          1         1
     311  6060         2          1        1         1          0         1
     299  7000         3          1        1         1          0         0
     ..    …         …          …        …         …          …         …
     59   6000         3          2        4         1          1         0
     23   4560         3          2        2         1          1         1
     264  4900         2          1        2         1          0         1
     327  6480         3          1        2         0          0         0
     135  6000         3          2        4         1          0         0

          hotwaterheating  airconditioning  parking  prefarea  furnishingstatus
     336                0                1        2         0                 1
     74                 1                0        1         0                 0
     121                0                1        0         1                 1
     311                0                0        1         0                 1
```

```
299                  0                0        3          0                 0
..                 …               …       …        …                 …
59                   0                1        1          0                 0
23                   0                1        1          0                 0
264                  0                0        0          0                 1
327                  0                1        1          0                 1
135                  0                1        0          0                 2

[327 rows x 12 columns]
```

[ ]:
```
y_test
```

[ ]:
```
316   -0.378188
77     1.007785
360   -0.565482
90     0.895409
493   -1.052446
          …
395   -0.677858
425   -0.752776
195    0.108775
452   -0.865152
154    0.408445
Name: price, Length: 164, dtype: float64
```

3 Standardizing Data

[ ]:
```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
col_stand =␣
 ↪['price',        'area',        'bedrooms',        'bathrooms',        'stories',␣
 ↪'parking']
df[col_stand] = scaler.fit_transform(df[col_stand])

df[col_stand]
```

[ ]:
```
        price      area  bedrooms  bathrooms   stories   parking
0    4.566365  1.046726  1.403419   1.421812  1.378217  1.517692
1    4.004484  1.757010  1.403419   5.405809  2.532024  2.679409
2    4.004484  2.218232  0.047278   1.421812  0.224410  1.517692
3    3.985755  1.083624  1.403419   1.421812  0.224410  2.679409
4    3.554979  1.046726  1.403419  -0.570187  0.224410  1.517692
..        …         …         …          …         …         …
540 -1.576868 -0.991879 -1.308863  -0.570187 -0.929397  1.517692
541 -1.605149 -1.268613  0.047278  -0.570187 -0.929397 -0.805741
```

```
542 -1.614327 -0.705921 -1.308863  -0.570187 -0.929397 -0.805741
543 -1.614327 -1.033389  0.047278  -0.570187 -0.929397 -0.805741
544 -1.614327 -0.599839  0.047278  -0.570187  0.224410 -0.805741

[545 rows x 6 columns]
```

[ ]:
```
X.shape
```

[ ]:
```
(545, 12)
```

[ ]:
```
# linear model

from sklearn.linear_model import LinearRegression
lr = LinearRegression()
```

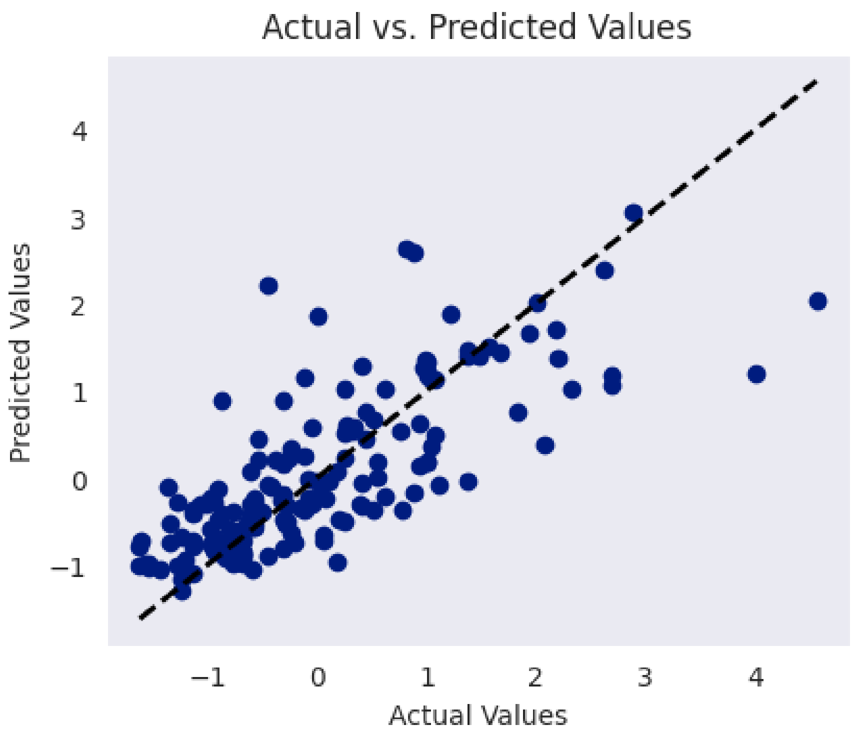
[ ]:
```
# fit the model
lr.fit(X_train, y_train)
lr
```

[ ]:
```
LinearRegression()
```

[ ]:
```
# predict the model

y_pred = lr.predict(X_test)
y_pred.flatten()
```

[ ]:
```
array([ 3693278.67108634,  3310537.49083126,  9159321.06094269,
        4776028.73343608,  4587055.31980847,  2846571.1152851 ,
        2276850.73855951,  7805919.35857575,  3051405.72138166,
        4468566.33192995,  5239287.07339578,  5256288.01294352,
        6868254.72666885,  7246402.56225509,  6320916.14821895,
        5025028.75986915,  4994941.97443227,  5006140.32022727,
        4295770.45983834,  2655699.04986355,  8272625.12851925,
        6582516.88276331,  3677469.09973206,  3855138.91656847,
        6402519.92730603,  5370553.92436425,  4585232.2213038 ,
        3942408.96867674,  4056143.66370808,  5890319.90023746,
        6419128.09399607,  7258160.35918025,  8185653.5767131 ,
        6548024.31837718,  5877283.74826159,  2116390.64560591,
        5119008.16230867,  5355690.6358595 , 10294295.57337128,
        3832760.75766472,  4950291.11682416,  3236337.67808746,
        8237492.93795545,  3866276.41675041,  7367617.58012689,
        4878668.927726  ,  2762263.44629577,  3374782.94272235,
        4422920.70108534,  3142025.24408873,  6664382.18227005,
        5607259.0580322 ,  3153867.7937742 ,  3144934.73138666,
        2634000.75611623,  5197830.25748175,  4060529.32231758,
        6392715.87157609,  7132426.26092044,  5453251.8884254 ,
        6963854.84067544,  7857130.62169011,  4354996.12526923,
        3253490.27403302,  2767409.05888393,  6369319.3510568 ,
        2842845.14177625,  4538352.58683404,  4538610.19213993,
        4022911.60234949,  3243331.03146477,  2464094.05464148,
        6342163.00098637,  4414959.38970703,  4477219.31067176,
        6306462.89560644,  2422363.8903199 ,  4159229.10413038,
        5896596.87327683,  8778240.81397403,  2679979.50141088,
        9656164.89654824,  3558628.02585547,  7122402.53008726,
        3611012.90293998,  4312954.85993405,  5065118.74418853,
        4339711.30790303,  4626690.77263596,  2653260.92456147,
        3297384.68428999,  3830157.12254246,  5842696.04846098,
        3662172.36539243,  4647146.42440728,  5227914.10483159,
        3530986.75168859,  3924764.39546365,  7000770.91325849,
        2434373.23889837,  2823828.73990585,  2912457.11639877,
        2535753.17899219,  3733113.0930588 ,  5407444.6448636 ,
        3958043.01867656,  3282852.69209254,  3525046.50798955,
        7252973.14305926,  7608347.28302647,  4038102.22461713,
        4482314.96171455,  4082662.76722499,  2635847.02124685,
        3342613.5104194 ,  6758866.24134691,  7676566.99692004,
        4072369.01536932,  4238039.36719006,  3650533.25211133,
        4329444.55509804,  4822875.82175822,  3281448.15716463,
        2351395.19119149,  5295109.19506267,  2608731.45013929,
        2729063.36971581,  3249866.99746995,  6397371.28724071,
        3604836.10016704,  7933734.59399701,  4396069.59487062,
        3887809.02755679,  3554607.11356822,  4870112.11793283,
        1974388.53759887,  3584518.18233098,  2619333.16706622,
        2576375.34690541,  6443543.25052519,  4527038.66058401,
        3497198.00708147,  2127076.56040927,  5396649.2042872 ,
        3048970.54217674,  8111130.49471667,  3191771.75273962,
        6268423.75315395,  5114823.8404434 ,  4026892.68348448,
        3489519.99111911,  5464270.40013635,  5604758.40247326,
        2812732.16308546,  5449901.78633903,  5686350.35391185,
        4475203.76436014,  2578902.27750311,  2822918.19951409,
        3270326.37462202,  6511192.54636049,  5790953.15540896,
        4435404.01726702,  4000993.22558845,  3737647.93534409,
        2792236.63465817,  6206165.87026382,  6263068.72259876,
        5554608.95489808,  4515729.39269953,  6926110.77748154,
        3003044.06121554,  4265059.61351993,  4373997.71286652,
        4273498.68137133,  5558509.7796334 ,  5859103.70223329,
        3494537.05957325,  3377412.34637454,  3471987.874607  ,
        3179137.09975115,  3438334.73782183,  5941871.25550317,
        2483889.6870298 ,  5527589.41264809,  4342494.88971335,
        3030209.84104415,  5373081.29847654,  3143676.73886431,
        2918089.17000674,  5573297.00390846,  3326782.65294954,
        3427270.29286135,  3414911.11609942,  6736833.04947317,
        2363153.29294358,  2815967.86446703,  2921057.83080714,
        2573848.41630772,  7840669.72617658,  2991034.86326354,
        5582842.1656501 ,  4361212.8147027 ,  2907096.0342881 ,
        3014420.72168166,  6396912.75183353,  4289108.21209267,
        2163908.33242595,  4108987.49580576,  5263777.85896857,
        6891227.74364997,  2858378.56040093,  5341235.85143379,
        4028398.18176226,  4883163.98343911,  6452046.47628847,
        5103267.64215395,  6180377.44268683])
```

[ ]:
```
plt.figure(figsize=(5, 4))
plt.scatter(y_test, y_pred)  # y_test: actual, y_pred: predicted
plt.xlabel('Actual Values')
plt.ylabel('Predicted Values')
plt.title('Actual vs. Predicted Values')
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], 'k--', lw=2) ␣
 ↪# Diagonal line for reference
plt.show()
```



There is a positive correlation in the above plot, indicating as actual values are increasing, predicted values are increasing well.

There are few outliers also.

4 Evaluation

```
[ ]: from sklearn.metrics import r2_score, mean_absolute_error, mean_squared_error

     R2_Score = r2_score(y_test, y_pred)
     MAE = mean_absolute_error(y_test, y_pred)
     MSE = mean_squared_error(y_test, y_pred)

     print("R2_Score : " , R2_Score)
     print("MAE : ", MAE)
     print("MSE : ", MSE)
```

```
R2_Score :  0.6591665511958988
MAE :  854935.0026136297
MSE :  1248964183909.7803
```

5 Hyperparameter Tuning

```
[ ]: # hypertuning gridsearch

     from sklearn.model_selection import GridSearchCV
     from sklearn.linear_model import Lasso

     param_grid = {'alpha' : [0.001, 0.01, 0.1, 1, 10, 100]}
     lasso = Lasso()
     gs_lasso = GridSearchCV(lasso, param_grid, cv = 5, scoring =␣
      ↪'neg_mean_squared_error')
     gs_lasso.fit(X_train, y_train)
```

```
[ ]: GridSearchCV(cv=5, estimator=Lasso(),
                  param_grid={'alpha': [0.001, 0.01, 0.1, 1, 10, 100]},
                  scoring='neg_mean_squared_error')
```

```
[ ]: gs_lasso_predict = gs_lasso.predict(X_test)

     # evaluation
     print('MSE' , mean_squared_error(y_test, gs_lasso_predict))
     print('R2_Score', r2_score(y_test, gs_lasso_predict))
```

```
MSE 0.4410223812072118
R2_Score 0.6423675391170636
```

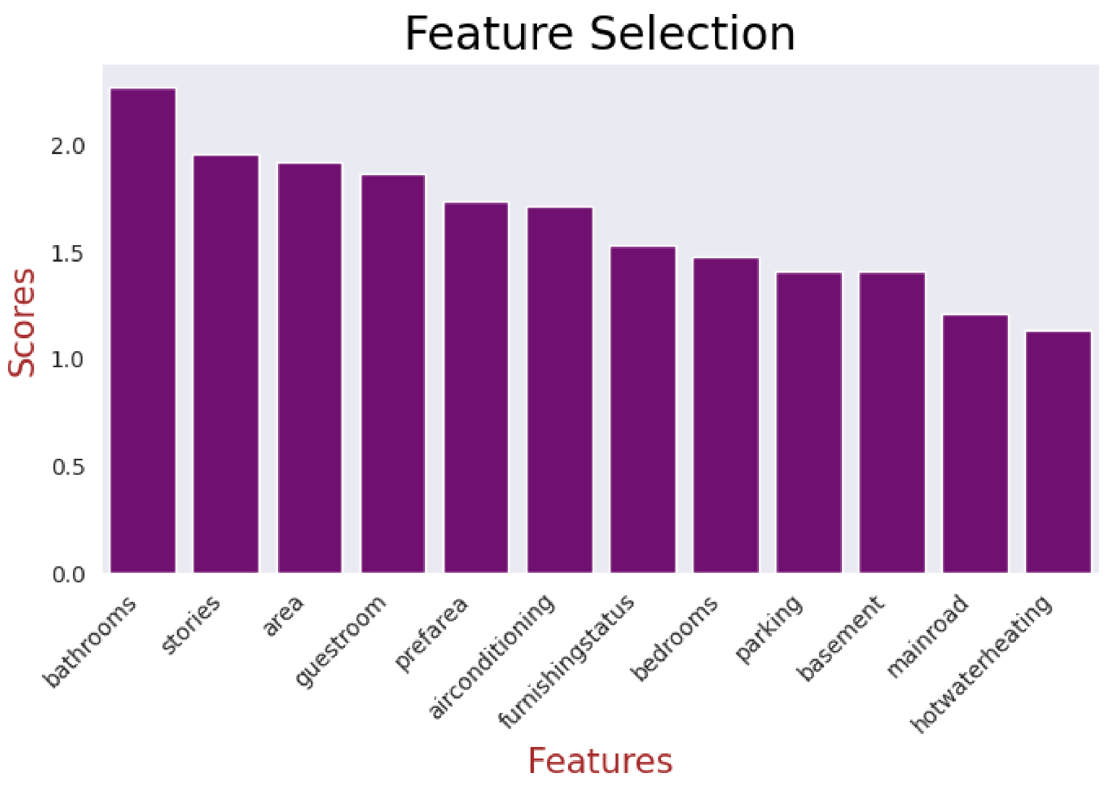
5.0.1 From the Gridsearch lasso, there is a slighest difference in the values of r2_score, and mse

6 Feature Importance

```
[ ]: # using RFE method for the feature selection

     from sklearn.feature_selection import SelectKBest, f_classif

     X = df.drop('price', axis = 1)
     y = df['price']

     selector = SelectKBest(f_classif, k = 5)
     X_new = selector.fit_transform(X, y)

     selector_features = X.columns[selector.get_support()]

     #get the scores
     features_scores = selector.scores_

     # create dataframe
     feature_df = pd.DataFrame({'Feature' : X.columns, 'Score' : features_scores})
     feature_df
```

```
[ ]:              Feature     Score
     0                area  1.917156
     1            bedrooms  1.477442
     2           bathrooms  2.266249
     3             stories  1.958777
     4            mainroad  1.208116
     5           guestroom  1.862660
     6            basement  1.411466
     7     hotwaterheating  1.135258
     8     airconditioning  1.709469
     9             parking  1.412568
     10           prefarea  1.734748
     11   furnishingstatus  1.529607
```

```
[ ]: sort_df = feature_df.sort_values(by = ['Score'], ascending = False)

     plt.figure(figsize = (7,5))
     sns.set_style('dark')
     sns.barplot(x = 'Feature', y = 'Score', data = sort_df, color = 'purple')
     plt.title('Feature Selection', color='black', size=20)
     plt.xlabel('Features', color='brown', size=15)
     plt.ylabel('Scores', color='brown', size=15)
     plt.xticks(rotation=45, ha='right')
     plt.tight_layout()
     plt.show()
```



7 Conclusion

This project successfully developed a house price prediction model using a Linear Regression algorithm.

The model's performance was evaluated using R-squared score(0.65) , Mean Squared Error (1248964183909), and Mean Absolute Error (854935), indicating a reasonable level of prediction accuracy.

Feature engineering and data preprocessing techniques improved model performance, leading to a higher R-squared score and lower MSE and MAE values.

Feature Selection using the SelectKBest method indicates that 'bathrooms', 'stories', and 'area' are among the most crucial features in predicting house prices. These features have the highest scores according to the f_classif scoring function, suggesting a strong relationship with the target variable

Hyperparameter tuning and feature selection further enhanced predictive capabilities.

This project provides a base for future research and development in house price prediction.

```
[ ]:
```