Machine Learning on Housing Dataset

Dependencies

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns

# Set the option to display all columns
pd.set_option('display.max_columns', None)
```

Initial Data Exploration

Importing

```
df = pd.read_csv("Datasets/Housing.csv")

df.head()

           id             date     price  bedrooms  bathrooms
sqft_living  \
0  7229300521  20141013T000000  231300.0         2       1.00
1180
1  6414100192  20141209T000000  538000.0         3       2.25
2570
2  5631500400  20150225T000000  180000.0         2       1.00
770
3  2487200875  20141209T000000  604000.0         4       3.00
1960
4  1954400510  20150218T000000  510000.0         3       2.00
1680

   sqft_lot  floors  waterfront  view  condition  grade  sqft_above  \
0      5650     1.0           0     0          3      7        1180
1      7242     2.0           0     0          3      7        2170
2     10000     1.0           0     0          3      6         770
3      5000     1.0           0     0          5      7        1050
4      8080     1.0           0     0          3      8        1680

   sqft_basement  yr_built  yr_renovated  zipcode      lat     long  \
0              0      1955             0    98178  47.5112 -122.257
1            400      1951          1991    98125  47.7210 -122.319
2              0      1933             0    98028  47.7379 -122.233
3            910      1965             0    98136  47.5208 -122.393
4              0      1987             0    98074  47.6168 -122.045
```

```
   sqft_living15  sqft_lot15
0           1340        5650
1           1690        7639
2           2720        8062
3           1360        5000
4           1800        7503
```

Shape

```
shape = df.shape
print("No. of rows: ",shape[0])
print("No. of cols: ",shape[1])
```

```
No. of rows:  21613
No. of cols:  21
```

Info

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21613 entries, 0 to 21612
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   id             21613 non-null  int64  
 1   date           21613 non-null  object 
 2   price          21613 non-null  float64
 3   bedrooms       21613 non-null  int64  
 4   bathrooms      21613 non-null  float64
 5   sqft_living    21613 non-null  int64  
 6   sqft_lot       21613 non-null  int64  
 7   floors         21613 non-null  float64
 8   waterfront     21613 non-null  int64  
 9   view           21613 non-null  int64  
 10  condition      21613 non-null  int64  
 11  grade          21613 non-null  int64  
 12  sqft_above     21613 non-null  int64  
 13  sqft_basement  21613 non-null  int64  
 14  yr_built       21613 non-null  int64  
 15  yr_renovated   21613 non-null  int64  
 16  zipcode        21613 non-null  int64  
 17  lat            21613 non-null  float64
 18  long           21613 non-null  float64
 19  sqft_living15  21613 non-null  int64  
 20  sqft_lot15     21613 non-null  int64  
dtypes: float64(5), int64(15), object(1)
memory usage: 3.5+ MB
```

Our target variable is Price for this Data set

```
print("Our target varibale: \n", df['price'])
```

```
Our target varibale:
 0        231300.0
1        538000.0
2        180000.0
3        604000.0
4        510000.0
           ...
21608    360000.0
21609    400000.0
21610    402101.0
21611    400000.0
21612    325000.0
Name: price, Length: 21613, dtype: float64
```

Features(Columns)

```
for col in df.columns:
    print(col, end=", ")
```

```
id, date, price, bedrooms, bathrooms, sqft_living, sqft_lot, floors,
waterfront, view, condition, grade, sqft_above, sqft_basement,
yr_built, yr_renovated, zipcode, lat, long, sqft_living15, sqft_lot15,
```

Data Cleaning

As there are no missing and duplicates, then we directly jump to detecting outliers

Converting each float into 2 decimal places

```
df['lat'] = round(df['lat'],2)
df['long'] = round(df['long'],2)
```

Convert to datetime

```
df['date'] = pd.to_datetime(df['date'])
df.head()
```

```
           id       date     price  bedrooms  bathrooms
sqft_living  \
0  7229300521 2014-10-13  231300.0         2       1.00         1180

1  6414100192 2014-12-09  538000.0         3       2.25         2570

2  5631500400 2015-02-25  180000.0         2       1.00          770
```

```
3  2487200875 2014-12-09  604000.0         4       3.00         1960

4  1954400510 2015-02-18  510000.0         3       2.00         1680


   sqft_lot  floors  waterfront  view  condition  grade  sqft_above  \
0      5650     1.0           0     0          3      7        1180
1      7242     2.0           0     0          3      7        2170
2     10000     1.0           0     0          3      6         770
3      5000     1.0           0     0          5      7        1050
4      8080     1.0           0     0          3      8        1680

   sqft_basement  yr_built  yr_renovated  zipcode   lat    long  \
0              0      1955             0    98178 47.51 -122.26
1            400      1951          1991    98125 47.72 -122.32
2              0      1933             0    98028 47.74 -122.23
3            910      1965             0    98136 47.52 -122.39
4              0      1987             0    98074 47.62 -122.04

   sqft_living15  sqft_lot15
0           1340        5650
1           1690        7639
2           2720        8062
3           1360        5000
4           1800        7503
```

Outliers

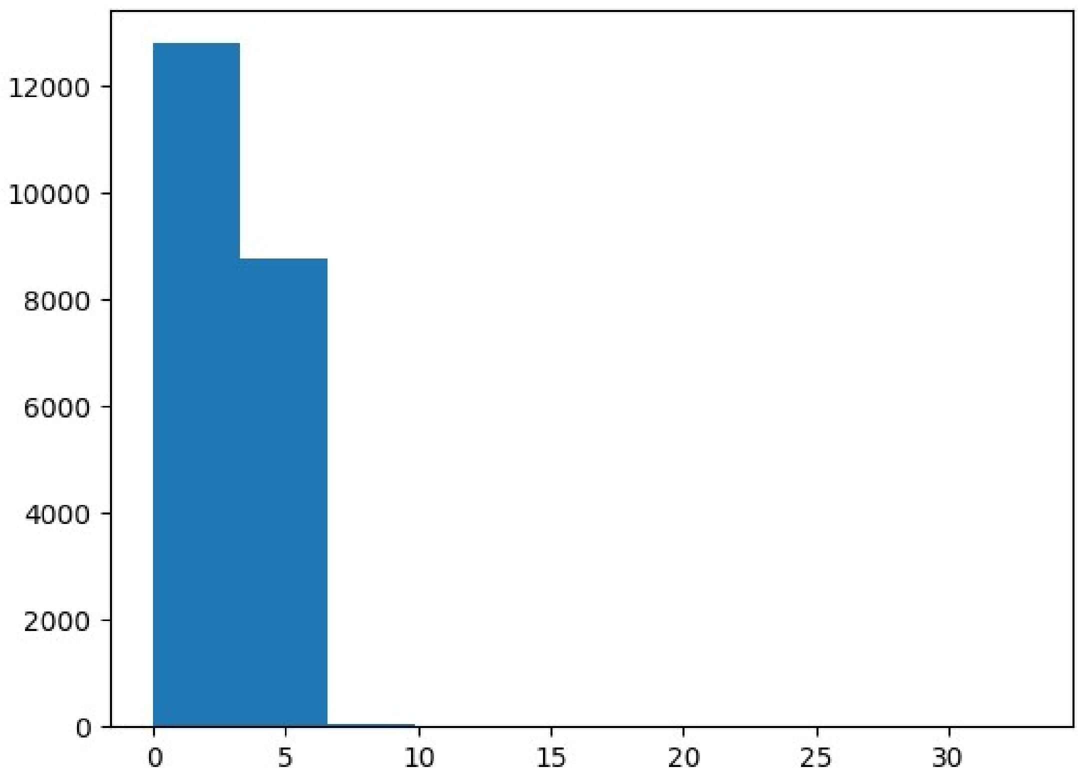
We can see that no. of bedrooms ranging from 0-10 is more but x axis still shows max xlim till 30 that means there are houses that have more than 10

```
df[df['bedrooms'] >= 10]
```

```
               id        date      price  bedrooms  bathrooms
sqft_living  \
8757   1773100755 2014-08-21   520000.0        11       3.00
3000
13314   627300145 2014-08-14  1148000.0        10       5.25
4590
15161  5566100170 2014-10-29   650000.0        10       2.00
3610
15870  2402100895 2014-06-25   640000.0        33       1.75
1620
19254  8812401450 2014-12-29   660000.0        10       3.00
2920

       sqft_lot  floors  waterfront  view  condition  grade
sqft_above  \
8757       4960     2.0           0     0          3      7
2400
```

```
13314     10920     1.0           0     2          3      9
2500
15161     11914     2.0           0     0          4      7
3010
15870      6000     1.0           0     0          5      7
1040
19254      3745     2.0           0     0          4      7
1860

       sqft_basement  yr_built  yr_renovated  zipcode    lat
long  \
8757             600      1918          1999    98106  47.56 -122.36

13314           2090      2008             0    98004  47.59 -122.11

15161            600      1958             0    98006  47.57 -122.18

15870            580      1947             0    98103  47.69 -122.33

19254           1060      1913             0    98105  47.66 -122.32


       sqft_living15  sqft_lot15
8757            1420        4960
13314           2730       10400
15161           2040       11914
15870           1330        4700
19254           1810        3745
```

```
plt.hist(df['bedrooms'])
plt.show()
```



Now, if we put this dataset in linear regression it might give more importance to houses having more than 30 bedrooms, that is problematic for us

so we have to remove it

```
loc = df.loc[df['bedrooms'] >= 10].index
cleanData = df.drop(loc,axis=0)

cleanData.loc[df['bedrooms'] >= 10]

Empty DataFrame
Columns: [id, date, price, bedrooms, bathrooms, sqft_living, sqft_lot,
floors, waterfront, view, condition, grade, sqft_above, sqft_basement,
yr_built, yr_renovated, zipcode, lat, long, sqft_living15, sqft_lot15]
Index: []
```

Data cleaning done

Feature Engineering

1. How much sqft does 1 bedroom consist of

```
cleanData['sqft_per_bedroom'] =
round(cleanData['sqft_living']/cleanData['bedrooms'],2)
cleanData
```

```
                 id       date     price  bedrooms  bathrooms  
sqft_living  \
0      7229300521 2014-10-13  231300.0         2       1.00         
1180   
1      6414100192 2014-12-09  538000.0         3       2.25         
2570   
2      5631500400 2015-02-25  180000.0         2       1.00          
770   
3      2487200875 2014-12-09  604000.0         4       3.00         
1960   
4      1954400510 2015-02-18  510000.0         3       2.00         
1680   
...           ...        ...       ...       ...        ...          .
..   
21608   263000018 2014-05-21  360000.0         3       2.50         
1530   
21609  6600060120 2015-02-23  400000.0         4       2.50         
2310   
21610  1523300141 2014-06-23  402101.0         2       0.75         
1020   
21611   291310100 2015-01-16  400000.0         3       2.50         
1600   
21612  1523300157 2014-10-15  325000.0         2       0.75         
1020   

       sqft_lot  floors  waterfront  view  condition  grade  
sqft_above  \
0          5650     1.0           0     0          3      7        
1180   
1          7242     2.0           0     0          3      7        
2170   
2         10000     1.0           0     0          3      6         
770   
3          5000     1.0           0     0          5      7        
1050   
4          8080     1.0           0     0          3      8        
1680   
...         ...     ...         ...   ...        ...    ...         ..
.   
21608      1131     3.0           0     0          3      8        
1530   
21609      5813     2.0           0     0          3      8        
2310   
21610      1350     2.0           0     0          3      7        
1020   
21611      2388     2.0           0     0          3      8        
1600   
21612      1076     2.0           0     0          3      7        
1020   
```

```
       sqft_basement  yr_built  yr_renovated  zipcode    lat
long  \
0                  0      1955             0    98178  47.51 -122.26

1                400      1951          1991    98125  47.72 -122.32

2                  0      1933             0    98028  47.74 -122.23

3                910      1965             0    98136  47.52 -122.39

4                  0      1987             0    98074  47.62 -122.04

...              ...       ...           ...      ...    ...     ...

21608              0      2009             0    98103  47.70 -122.35

21609              0      2014             0    98146  47.51 -122.36

21610              0      2009             0    98144  47.59 -122.30

21611              0      2004             0    98027  47.53 -122.07

21612              0      2008             0    98144  47.59 -122.30


       sqft_living15  sqft_lot15  sqft_per_bedroom
0               1340        5650            590.00
1               1690        7639            856.67
2               2720        8062            385.00
3               1360        5000            490.00
4               1800        7503            560.00
...              ...         ...               ...
21608           1530        1509            510.00
21609           1830        7200            577.50
21610           1020        2007            510.00
21611           1410        1287            533.33
21612           1020        1357            510.00

[21608 rows x 22 columns]
```

1. Year a house was sold

```
cleanData['sale_year'] = cleanData['date'].dt.year

cleanData.head()

           id       date     price  bedrooms  bathrooms
sqft_living  \
0  7229300521 2014-10-13  231300.0         2       1.00         1180
```

```
1   6414100192 2014-12-09   538000.0          3        2.25          2570

2   5631500400 2015-02-25   180000.0          2        1.00           770

3   2487200875 2014-12-09   604000.0          4        3.00          1960

4   1954400510 2015-02-18   510000.0          3        2.00          1680


   sqft_lot  floors  waterfront  view  condition  grade  sqft_above  \
0      5650     1.0           0     0          3      7        1180
1      7242     2.0           0     0          3      7        2170
2     10000     1.0           0     0          3      6         770
3      5000     1.0           0     0          5      7        1050
4      8080     1.0           0     0          3      8        1680

   sqft_basement  yr_built  yr_renovated  zipcode   lat    long  \
0              0      1955             0    98178 47.51 -122.26
1            400      1951          1991    98125 47.72 -122.32
2              0      1933             0    98028 47.74 -122.23
3            910      1965             0    98136 47.52 -122.39
4              0      1987             0    98074 47.62 -122.04

   sqft_living15  sqft_lot15  sqft_per_bedroom  sale_year
0           1340        5650            590.00       2014
1           1690        7639            856.67       2014
2           2720        8062            385.00       2015
3           1360        5000            490.00       2014
4           1800        7503            560.00       2015
```

1. At what age a house get sold(Effective Age)

effective_age = sale_year - max(yr_built, yr_renovated)

```
cleanData['effective_age'] = cleanData['sale_year'] -
cleanData[['yr_built','yr_renovated']].max(axis=1)

cleanData.head()

           id        date     price  bedrooms  bathrooms
sqft_living  \
0   7229300521 2014-10-13   231300.0          2        1.00          1180

1   6414100192 2014-12-09   538000.0          3        2.25          2570

2   5631500400 2015-02-25   180000.0          2        1.00           770

3   2487200875 2014-12-09   604000.0          4        3.00          1960

4   1954400510 2015-02-18   510000.0          3        2.00          1680
```

```
   sqft_lot  floors  waterfront  view  condition  grade  sqft_above  \
0      5650     1.0           0     0          3      7        1180
1      7242     2.0           0     0          3      7        2170
2     10000     1.0           0     0          3      6         770
3      5000     1.0           0     0          5      7        1050
4      8080     1.0           0     0          3      8        1680

   sqft_basement  yr_built  yr_renovated  zipcode    lat    long  \
0              0      1955             0    98178  47.51 -122.26
1            400      1951          1991    98125  47.72 -122.32
2              0      1933             0    98028  47.74 -122.23
3            910      1965             0    98136  47.52 -122.39
4              0      1987             0    98074  47.62 -122.04

   sqft_living15  sqft_lot15  sqft_per_bedroom  sale_year
effective_age
0           1340        5650            590.00       2014
59
1           1690        7639            856.67       2014
23
2           2720        8062            385.00       2015
82
3           1360        5000            490.00       2014
49
4           1800        7503            560.00       2015
28
```

Feature Engineering Completed

Train - Test Split

```
from sklearn.preprocessing import StandardScaler

scalerModel = StandardScaler()
column = ['price', 'bedrooms', 'bathrooms', 'sqft_living', 'sqft_lot',
'floors', 'waterfront', 'view', 'condition', 'grade', 'sqft_above',
'sqft_basement', 'yr_built', 'yr_renovated', 'zipcode', 'lat', 'long',
'sqft_living15', 'sqft_lot15', 'sqft_per_bedroom', 'sale_year',
'effective_age']
# Ensure numeric dtype for these columns (if some are strings) and
coerce errors to NaN
cleanData[column] = cleanData[column].apply(pd.to_numeric,
errors='coerce')
# Replace infinite values with NaN then drop rows that have NaN in any
of the selected columns
cleanData.replace([np.inf, -np.inf], np.nan, inplace=True)
cleanData.dropna(subset=column, inplace=True)
# Now scale
```

```
scaledData = scalerModel.fit_transform(cleanData[column])
scaled_df = pd.DataFrame(scaledData, columns=column)

scaled_df

          price  bedrooms  bathrooms  sqft_living  sqft_lot    floors
\
0     -0.841176 -1.522906  -1.450250    -0.980237 -0.228263 -0.915375

1     -0.005787 -0.411440   0.175113     0.533893 -0.189822  0.937599

2     -0.980907 -1.522906  -1.450250    -1.426851 -0.123226 -0.915375

3      0.173983  0.700025   1.150331    -0.130582 -0.243959 -0.915375

4     -0.082054 -0.411440  -0.149959    -0.435586 -0.169588 -0.915375

...         ...       ...        ...          ...       ...       ...

21590 -0.490623 -0.411440   0.500186    -0.598981 -0.337381  2.790574

21591 -0.381671  0.700025   0.500186     0.250674 -0.224328  0.937599

21592 -0.375949 -1.522906  -1.775322    -1.154525 -0.332093  0.937599

21593 -0.381671 -0.411440   0.500186    -0.522730 -0.307029  0.937599

21594 -0.585956 -1.522906  -1.775322    -1.154525 -0.338709  0.937599


       waterfront      view  condition     grade  sqft_above
sqft_basement  \
0       -0.087209 -0.305647  -0.629767 -0.560052   -0.734899      -
0.658845
1       -0.087209 -0.305647  -0.629767 -0.560052    0.461035
0.245157
2       -0.087209 -0.305647  -0.629767 -1.411825   -1.230184      -
0.658845
3       -0.087209 -0.305647   2.444682 -0.560052   -0.891940
1.397759
4       -0.087209 -0.305647  -0.629767  0.291721   -0.130892      -
0.658845
...           ...       ...        ...       ...         ...
...
21590   -0.087209 -0.305647  -0.629767  0.291721   -0.312094      -
0.658845
21591   -0.087209 -0.305647  -0.629767  0.291721    0.630157      -
0.658845
21592   -0.087209 -0.305647  -0.629767 -0.560052   -0.928181      -
0.658845
21593   -0.087209 -0.305647  -0.629767  0.291721   -0.227533      -
```

```
0.658845
21594   -0.087209 -0.305647  -0.629767 -0.560052   -0.928181      -
0.658845

       yr_built  yr_renovated   zipcode       lat      long
sqft_living15  \
0     -0.544857     -0.210100  1.869721 -0.361118 -0.327321      -
0.943510
1     -0.681050      4.747334  0.879268  1.154159 -0.753549      -
0.432755
2     -1.293917     -0.210100 -0.933448  1.298471 -0.114207
1.070322
3     -0.204376     -0.210100  1.084833 -0.288962 -1.250815      -
0.914324
4      0.544684     -0.210100 -0.073810  0.432599  1.235516      -
0.272232
...         ...           ...       ...       ...       ...
...
21590  1.293743     -0.210100  0.468136  1.009847 -0.966663      -
0.666243
21591  1.463984     -0.210100  1.271711 -0.361118 -1.037701      -
0.228453
21592  1.293743     -0.210100  1.234336  0.216131 -0.611473      -
1.410485
21593  1.123502     -0.210100 -0.952136 -0.216806  1.022402      -
0.841359
21594  1.259695     -0.210100  1.234336  0.216131 -0.611473      -
1.410485

       sqft_lot15  sqft_per_bedroom  sale_year  effective_age
0       -0.260734         -0.130837  -0.690787       0.626776
1       -0.187813          1.104714  -0.690787      -0.622706
2       -0.172304         -1.080655   1.447623       1.425056
3       -0.284565         -0.594163  -0.690787       0.279697
4       -0.192799         -0.269835   1.447623      -0.449167
...           ...               ...        ...            ...
21590   -0.412554         -0.501498  -0.690787      -1.247448
21591   -0.203907         -0.188753   1.447623      -1.386279
21592   -0.394296         -0.501498  -0.690787      -1.247448
21593   -0.420693         -0.393404   1.447623      -1.039201
21594   -0.418127         -0.501498  -0.690787      -1.212740

[21595 rows x 22 columns]

from sklearn.model_selection import train_test_split
# Split features and target
X = scaled_df.drop('price', axis=1)
y = scaled_df['price']
X_train_scaled, X_test_scaled, y_train, y_test = train_test_split(X,
```

```
y, test_size=0.2, random_state=42)
X_train_scaled.shape, X_test_scaled.shape, y_train.shape, y_test.shape
```

```
((17276, 21), (4319, 21), (17276,), (4319,))
```

Model Selection and Model Tuning

For this project we are choosing Linear Regression Model

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

# Train the model on the training data
model.fit(X_train_scaled, y_train)
```

```
LinearRegression()
```

```
# Make predictions on the test data
predictions = model.predict(X_test_scaled)
```

Model Evaluation

```
from sklearn.metrics import mean_squared_error, r2_score

mse = mean_squared_error(y_test, predictions)
r2 = r2_score(y_test, predictions)

print(f"Mean Squared Error (MSE): {mse}")
print(f"R-squared (R²): {r2}")
```

```
Mean Squared Error (MSE): 0.2917052908444622
R-squared (R²): 0.7182757120224226
```