
Real Estate House Predictions - Seattle, WA
created by Jaouad

Set environment and libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from scipy import stats
import statsmodels.api as sms
import statsmodels.formula.api as smf
from IPython.display import Image
import seaborn as sns
from sklearn import preprocessing
from sklearn.preprocessing import StandardScaler
%matplotlib inline
```

```python
# Read data in dataframe

df = pd.read_csv("King_County_House_prices_dataset.csv")
df.head()
```

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	...	grade
0	7129300520	10/13/2014	221900.0	3	1.00	1180	5650	1.0	NaN	0.0	...	7
1	6414100192	12/9/2014	538000.0	3	2.25	2570	7242	2.0	0.0	0.0	...	7
2	5631500400	2/25/2015	180000.0	2	1.00	770	10000	1.0	0.0	0.0	...	6
3	2487200875	12/9/2014	604000.0	4	3.00	1960	5000	1.0	0.0	0.0	...	7
4	1954400510	2/18/2015	510000.0	3	2.00	1680	8080	1.0	0.0	0.0	...	8

5 rows × 21 columns

```python
# Delete columns not needed

df.drop(["id", "lat", "long", "sqft_basement"], axis=1, inplace=True)
```

```python
# Convent date in readable format

df["date"] = pd.to_datetime(df["date"])
df["Month"] = df["date"].apply(lambda date: date.month)
df["Year"] = df["date"].apply(lambda date: date.year)

df.drop("date", axis=1, inplace =True)

df.tail(10)
```

	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	grade	sqft_above
21587	507250.0	3	2.50	2270	5536	2.0	NaN	0.0	3	8	2270
21588	429000.0	3	2.00	1490	1126	3.0	0.0	0.0	3	8	1490
21589	610685.0	4	2.50	2520	6023	2.0	0.0	NaN	3	9	2520
21590	1010000.0	4	3.50	3510	7200	2.0	0.0	0.0	3	9	2600
21591	475000.0	3	2.50	1310	1294	2.0	0.0	0.0	3	8	1180
21592	360000.0	3	2.50	1530	1131	3.0	0.0	0.0	3	8	1530
21593	400000.0	4	2.50	2310	5813	2.0	0.0	0.0	3	8	2310
21594	402101.0	2	0.75	1020	1350	2.0	0.0	0.0	3	7	1020
21595	400000.0	3	2.50	1600	2388	2.0	NaN	0.0	3	8	1600
21596	325000.0	2	0.75	1020	1076	2.0	0.0	0.0	3	7	1020

```python
# Show size of the dataset

df.shape
```

```
(21597, 18)
```

```python
# Show overview of the data set

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21597 entries, 0 to 21596
Data columns (total 18 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   price          21597 non-null  float64
 1   bedrooms       21597 non-null  int64  
 2   bathrooms      21597 non-null  float64
 3   sqft_living    21597 non-null  int64  
 4   sqft_lot       21597 non-null  int64  
 5   floors         21597 non-null  float64
 6   waterfront     19221 non-null  float64
 7   view           21534 non-null  float64
 8   condition      21597 non-null  int64  
 9   grade          21597 non-null  int64  
 10  sqft_above     21597 non-null  int64  
 11  yr_built       21597 non-null  int64  
 12  yr_renovated   17755 non-null  float64
 13  zipcode        21597 non-null  int64  
 14  sqft_living15  21597 non-null  int64  
 15  sqft_lot15     21597 non-null  int64  
 16  Month          21597 non-null  int64  
 17  Year           21597 non-null  int64  
dtypes: float64(6), int64(12)
memory usage: 3.0 MB
```

```python
# Check if zeros are contained
df.isnull().values.any()
```

```
False
```

```python
# View data description and see if there are any outliers

df.describe()
```

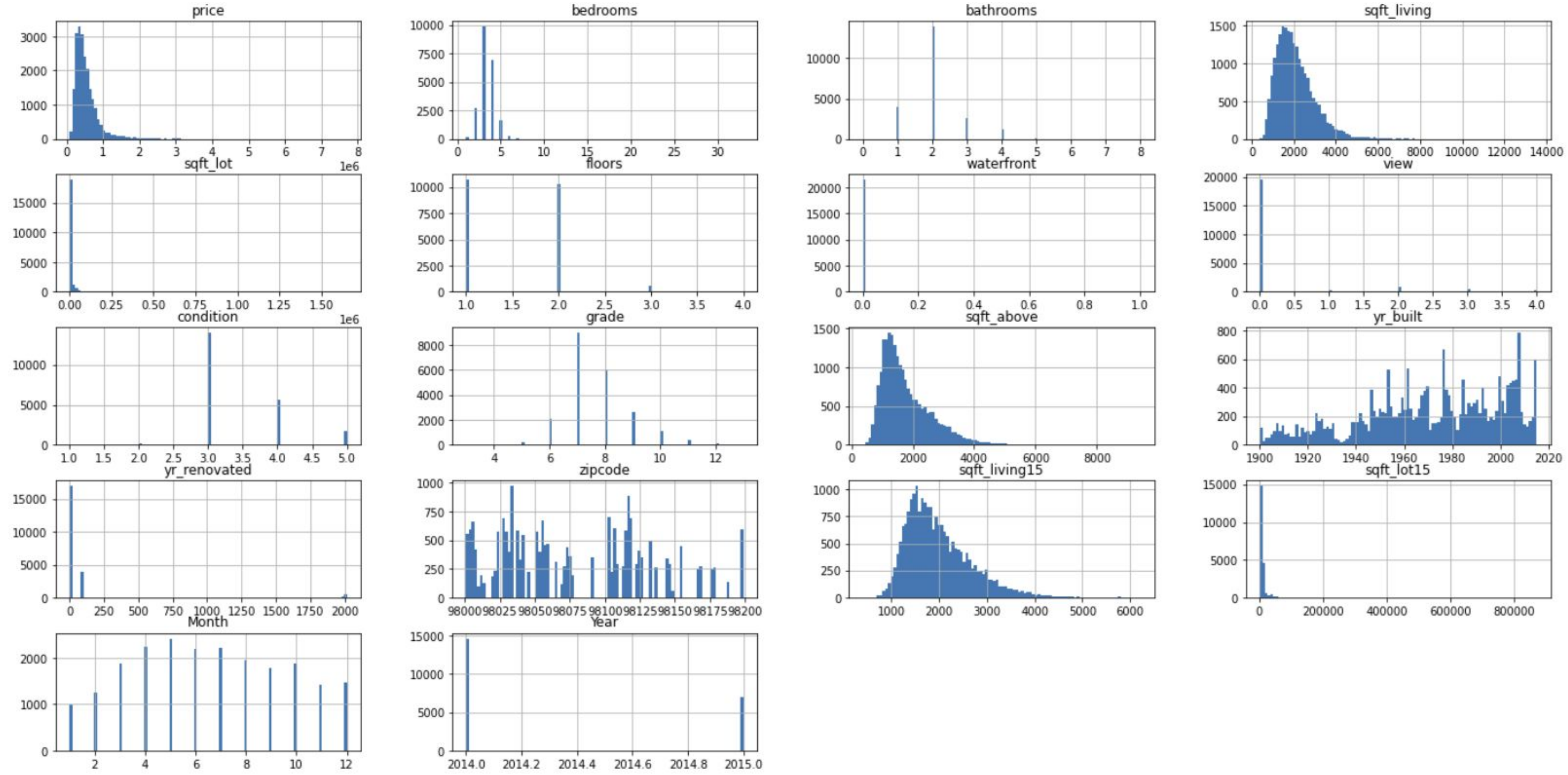
	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view
count	2.159700e+04	21597.000000	21597.000000	21597.000000	2.159700e+04	21597.000000	19221.000000	21534.000000
mean	5.402966e+05	3.373200	2.115826	2080.321850	1.509941e+04	1.494096	0.007596	0.233863
std	3.673681e+05	0.926299	0.768984	918.106125	4.141264e+04	0.539683	0.086825	0.765686
min	7.800000e+04	1.000000	0.500000	370.000000	5.200000e+02	1.000000	0.000000	0.000000
25%	3.220000e+05	3.000000	1.750000	1430.000000	5.040000e+03	1.000000	0.000000	0.000000
50%	4.500000e+05	3.000000	2.250000	1910.000000	7.618000e+03	1.500000	0.000000	0.000000
75%	6.450000e+05	4.000000	2.500000	2550.000000	1.068500e+04	2.000000	0.000000	0.000000
max	7.700000e+06	33.000000	8.000000	13540.000000	1.651359e+06	3.500000	1.000000	4.000000

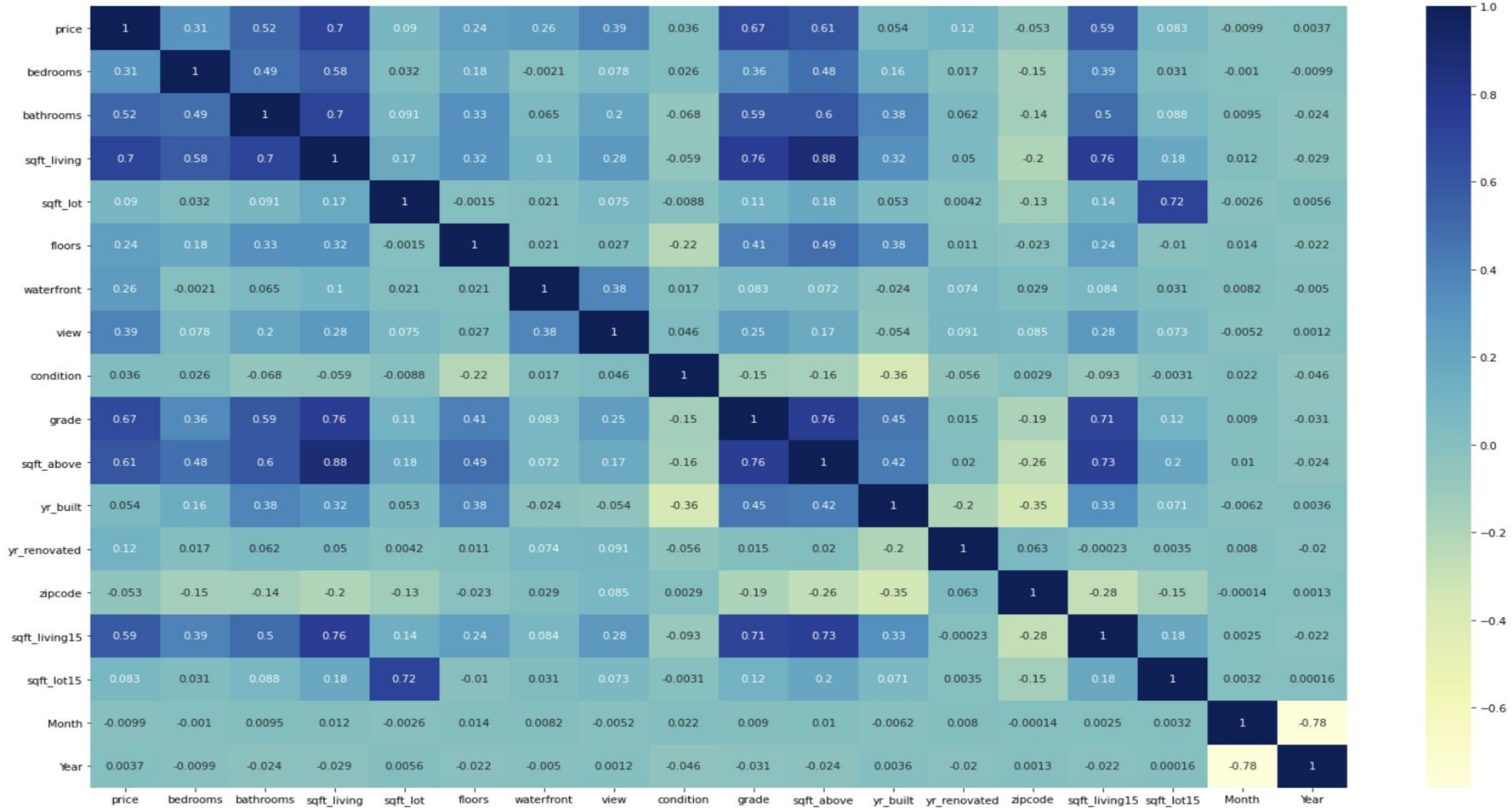
```python
# Replace the missing data, with the mean value

df['waterfront'].fillna((df['waterfront'].mean()), inplace=True)
df['yr_renovated'].fillna((df['yr_renovated'].mean()), inplace=True)
df['view'].fillna((df['view'].mean()), inplace=True)
df.dropna(inplace=True)
df.head()
```

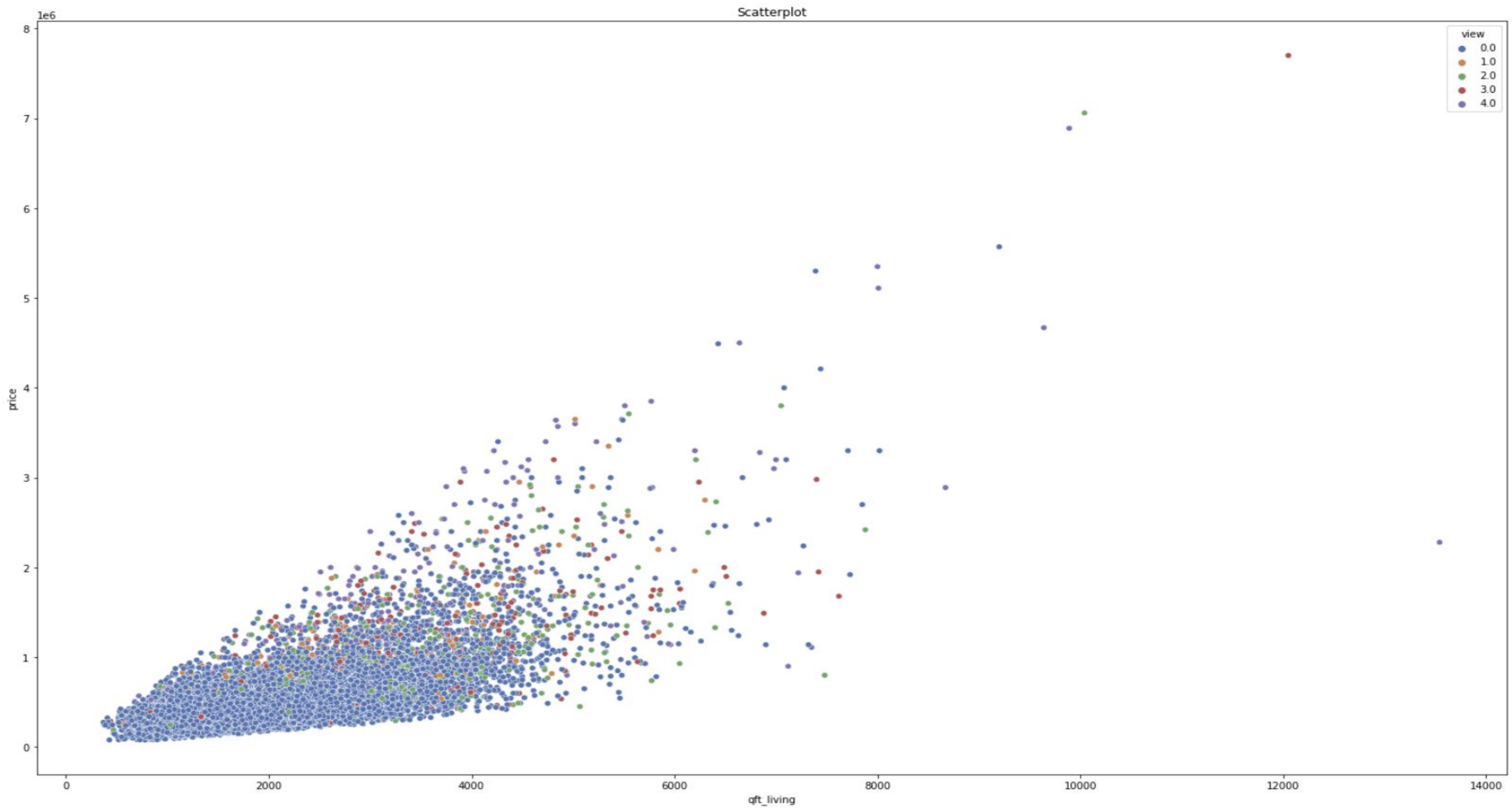
	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	grade	sqft_above	yr_built
0	221900.0	3	1.00	1180	5650	1.0	0.007596	0.0	3	7	1180	1955
1	538000.0	3	2.25	2570	7242	2.0	0.000000	0.0	3	7	2170	1951
2	180000.0	2	1.00	770	10000	1.0	0.000000	0.0	3	6	770	1933
3	604000.0	4	3.00	1960	5000	1.0	0.000000	0.0	5	7	1050	1965
4	510000.0	3	2.00	1680	8080	1.0	0.000000	0.0	3	8	1680	1987

```python
# Convert values to integer with float

df['waterfront'] = np.round(df['waterfront'])
df['bathrooms'] = np.round(df['bathrooms'])
df['yr_renovated'] = np.round(df['yr_renovated'])

df['floors'] = np.round(df['floors'])
df['view'] = np.round(df['view'])
df.head(3)
```

	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	grade	sqft_above	yr_built
0	221900.0	3	1.0	1180	5650	1.0	0.0	0.0	3	7	1180	1955
1	538000.0	3	2.0	2570	7242	2.0	0.0	0.0	3	7	2170	1951
2	180000.0	2	1.0	770	10000	1.0	0.0	0.0	3	6	770	1933


price
bedrooms
bathrooms
sqft_living
sqft_lot
floors
waterfront
view
condition
grade
sqft_above
yr_built
yr_renovated
zipcode
sqft_living15
sqft_lot15
Month
Year
1e6
1e6


price 1 0.31 0.52 0.7 0.09 0.24 0.26 0.39 0.036 0.67 0.61 0.054 0.12 -0.053 0.59 0.083 -0.0099 0.0037
bedrooms 0.31 1 0.49 0.58 0.032 0.18 -0.0021 0.078 0.026 0.36 0.48 0.16 0.017 -0.15 0.39 0.031 -0.001 -0.0099
bathrooms 0.52 0.49 1 0.7 0.091 0.33 0.065 0.2 -0.068 0.59 0.6 0.38 0.062 -0.14 0.5 0.088 0.0095 -0.024
sqft_living 0.7 0.58 0.7 1 0.17 0.32 0.1 0.28 -0.059 0.76 0.88 0.32 0.05 -0.2 0.76 0.18 0.012 -0.029
sqft_lot 0.09 0.032 0.091 0.17 1 -0.0015 0.021 0.075 -0.0088 0.11 0.18 0.053 0.0042 -0.13 0.14 0.72 -0.0026 0.0056
floors 0.24 0.18 0.33 0.32 -0.0015 1 0.021 0.027 -0.22 0.41 0.49 0.38 0.011 -0.023 0.24 -0.01 0.014 -0.022
waterfront 0.26 -0.0021 0.065 0.1 0.021 0.021 1 0.38 0.017 0.083 0.072 -0.024 0.074 0.029 0.084 0.031 0.0082 -0.005
view 0.39 0.078 0.2 0.28 0.075 0.027 0.38 1 0.046 0.25 0.17 -0.054 0.091 0.085 0.28 0.073 -0.0052 0.0012
condition 0.036 0.026 -0.068 -0.059 -0.0088 -0.22 0.017 0.046 1 -0.15 -0.16 -0.36 -0.056 0.0029 -0.093 -0.0031 0.022 -0.046
grade 0.67 0.36 0.59 0.76 0.11 0.41 0.083 0.25 -0.15 1 0.76 0.45 0.015 -0.19 0.71 0.12 0.009 -0.031
sqft_above 0.61 0.48 0.6 0.88 0.18 0.49 0.072 0.17 -0.16 0.76 1 0.42 0.02 -0.26 0.73 0.2 0.01 -0.024
yr_built 0.054 0.16 0.38 0.32 0.053 0.38 -0.024 -0.054 -0.36 0.45 0.42 1 -0.2 -0.35 0.33 0.071 -0.0062 0.0036
yr_renovated 0.12 0.017 0.062 0.05 0.0042 0.011 0.074 0.091 -0.056 0.015 0.02 -0.2 1 0.063 -0.00023 0.0035 0.008 -0.02
zipcode -0.053 -0.15 -0.14 -0.2 -0.13 -0.023 0.029 0.085 0.0029 -0.19 -0.26 -0.35 0.063 1 -0.28 -0.15 -0.00014 0.0013
sqft_living15 0.59 0.39 0.5 0.76 0.14 0.24 0.084 0.28 -0.093 0.71 0.73 0.33 -0.00023 -0.28 1 0.18 0.0025 -0.022
sqft_lot15 0.083 0.031 0.088 0.18 0.72 -0.01 0.031 0.073 -0.0031 0.12 0.2 0.071 0.0035 -0.15 0.18 1 0.0032 0.00016
Month -0.0099 -0.001 0.0095 0.012 -0.0026 0.014 0.0082 -0.0052 0.022 0.009 0.01 -0.0062 0.008 -0.00014 0.0025 0.0032 1 -0.78
Year 0.0037 -0.0099 -0.024 -0.029 0.0056 -0.022 -0.005 0.0012 -0.046 -0.031 -0.024 0.0036 -0.02 0.0013 -0.022 0.00016 -0.78 1
price bedrooms bathrooms sqft_living sqft_lot floors waterfront view condition grade sqft_above yr_built yr_renovated zipcode sqft_living15 sqft_lot15 Month Year
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6


Scatterplot
1e6
price
qft_living
view
0.0
1.0
2.0
3.0
4.0
0
1
2
3
4
5
6
7
8
0
2000
4000
6000
8000
10000
12000
14000

price ~ sqft_living

OLS Regression Results

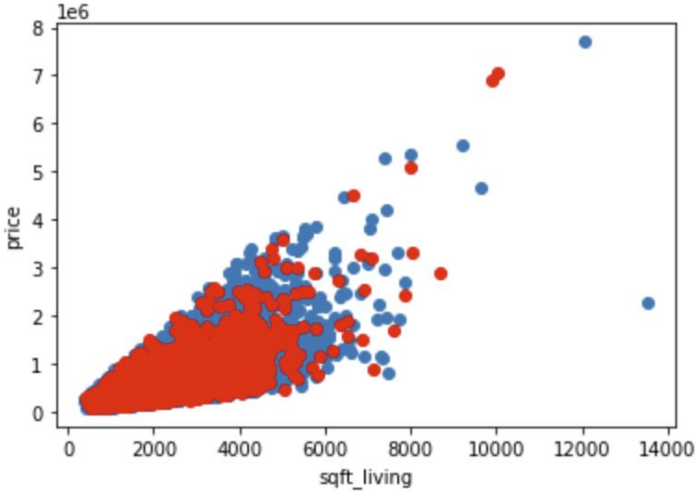
Dep. Variable:	price	**R-squared:**	0.493
Model:	OLS	**Adj. R-squared:**	0.493
Method:	Least Squares	**F-statistic:**	2.097e+04
Date:	Sun, 06 Jun 2021	**Prob (F-statistic):**	0.00
Time:	13:52:08	**Log-Likelihood:**	-3.0006e+05
No. Observations:	21597	**AIC:**	6.001e+05
Df Residuals:	21595	**BIC:**	6.001e+05
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-4.399e+04	4410.023	-9.975	0.000	-5.26e+04	-3.53e+04
sqft_living	280.8630	1.939	144.819	0.000	277.062	284.664

Omnibus:	14801.942	**Durbin-Watson:**	1.982
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	542662.604
Skew:	2.820	**Prob(JB):**	0.00
Kurtosis:	26.901	**Cond. No.**	5.63e+03

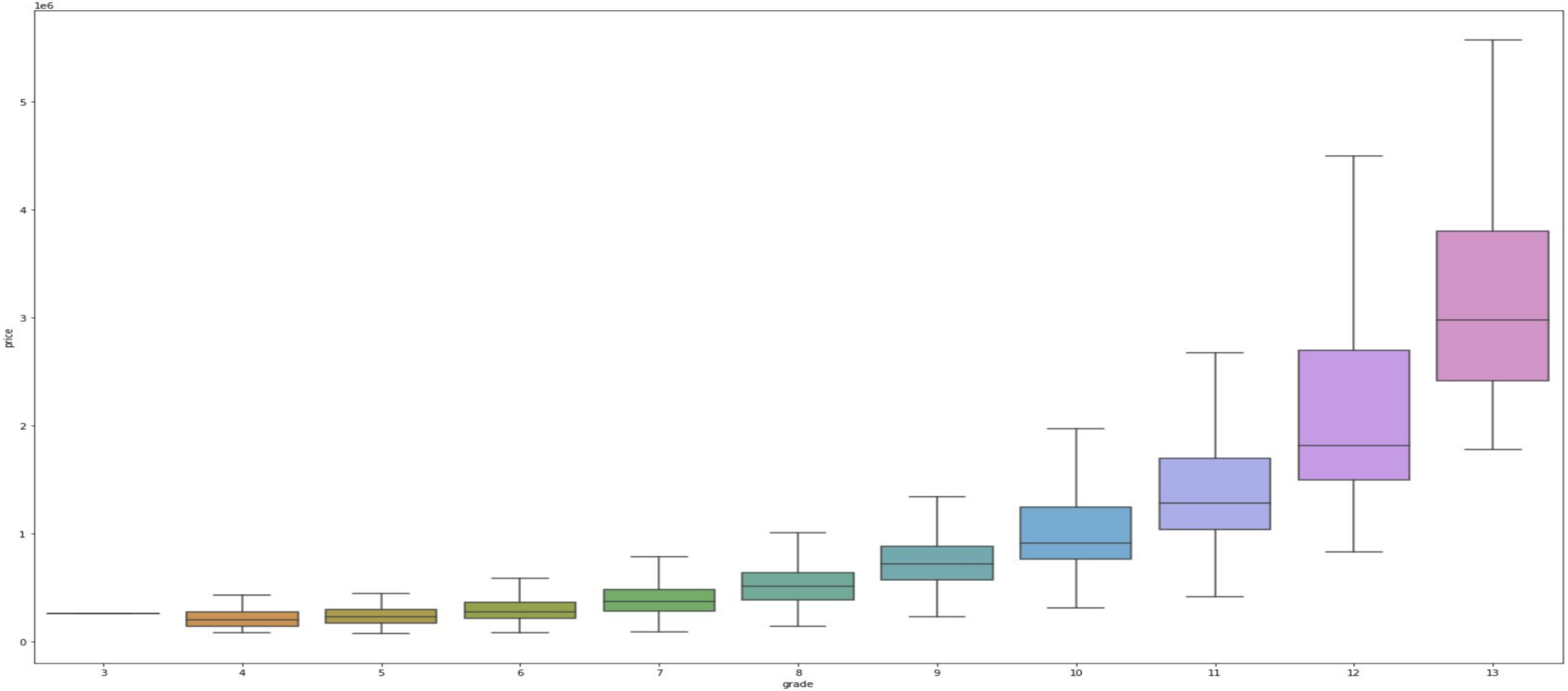
```python
# We split our data into training and test data

X = df[['sqft_living']].values
y = df['price'].values

X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=42, test_size=0.25)
```

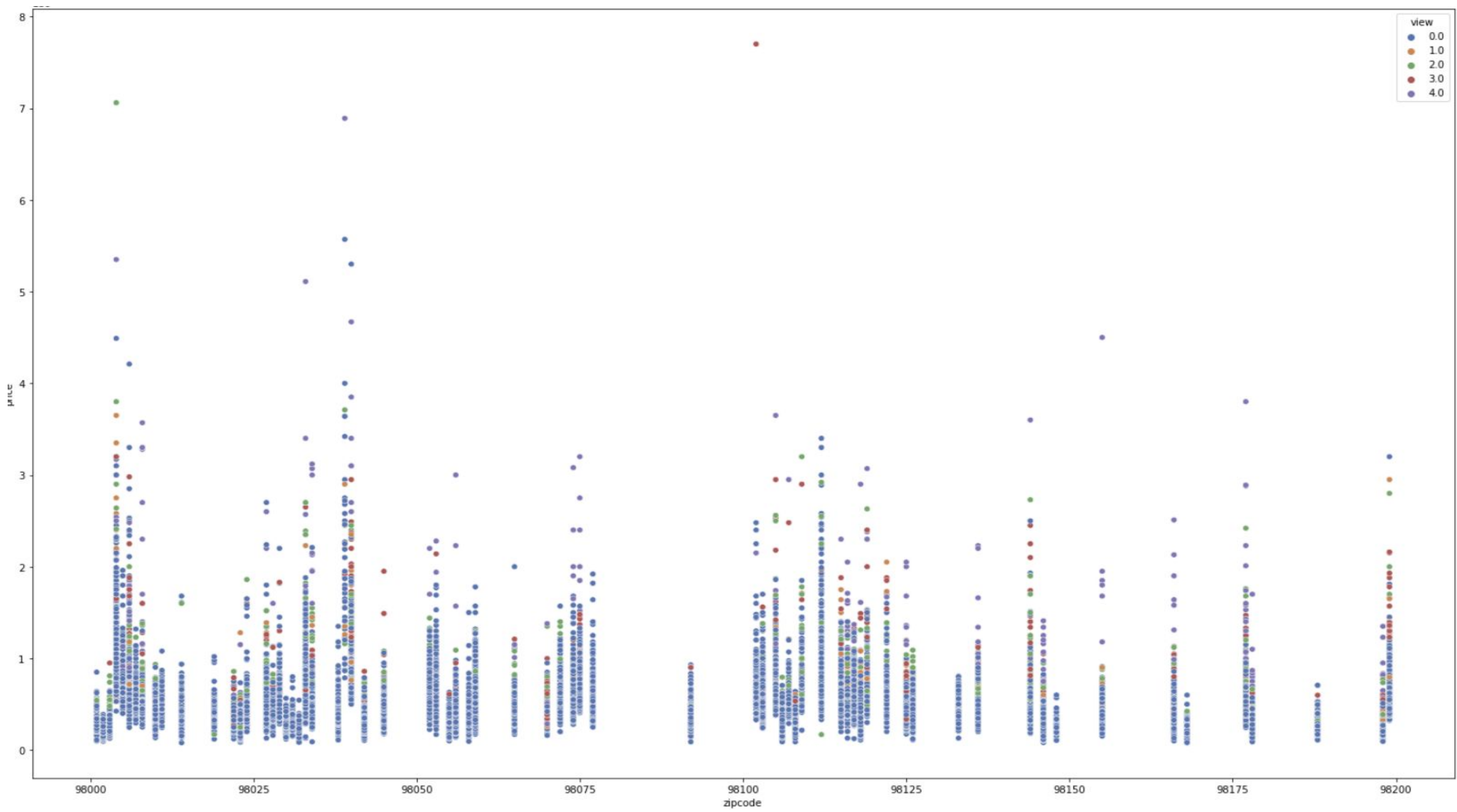
```python
# How good is my prediction

plt.scatter(X_train, y_train)
plt.scatter(X_test, y_test, color="red")
plt.xlabel("sqft_living" )
plt.ylabel("price")
plt.show()
```



```python
# The second strongest correlation between independent and target varibale plots

fig, ax = plt.subplots(figsize=(25,15))
sns.boxplot(x='grade',y='price',data=df,showfliers=False, ax=ax)
```

```
<AxesSubplot:xlabel='grade', ylabel='price'>
```





view
0.0
1.0
2.0
3.0
4.0
0
1
2
3
4
5
6
7
8
98000
98025
98050
98075
98100
98125
98150
98175
98200
zipcode

```python
df = pd.get_dummies(df, columns=["zipcode"])
df.head()
```

	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	grade	...	zipcode_98146
0	221900.0	3	1.0	1180	5650	1.0	0.0	0.0	3	7	...	0
1	538000.0	3	2.0	2570	7242	2.0	0.0	0.0	3	7	...	0
2	180000.0	2	1.0	770	10000	1.0	0.0	0.0	3	6	...	0
3	604000.0	4	3.0	1960	5000	1.0	0.0	0.0	5	7	...	0
4	510000.0	3	2.0	1680	8080	1.0	0.0	0.0	3	8	...	0

```python
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=42, test_size=0.25)
```

```python
model = LinearRegression()
model.fit(X_train, y_train)
#print("Intercept: " +str(model.intercept_))
#print("Coef: " +str(model.coef_))

print("R2_Score: " + str(model.score(X_test, y_test)))
```

```
R2_Score: 0.7674287742096019
```

```python
from sklearn.metrics import mean_squared_error
import math
print("MSE: " + str(mean_squared_error(y_test, y_test_pred)))
print("RMSE: " + str(math.sqrt(mean_squared_error(y_test, y_test_pred))))
```

```
MSE: 31908704848.957275
RMSE: 178630.07823140334
```

```python
df.head(4)
```

	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	grade	...	zipcode_98146
0	221900.0	3	1.0	1180	5650	1.0	0.0	0.0	3	7	...	0
1	538000.0	3	2.0	2570	7242	2.0	0.0	0.0	3	7	...	0
2	180000.0	2	1.0	770	10000	1.0	0.0	0.0	3	6	...	0
3	604000.0	4	3.0	1960	5000	1.0	0.0	0.0	5	7	...	0

4 rows × 87 columns

```python
y_test_pred = model.predict(X_test)
y_test_pred[0]
```

```
117539.74891492724
```

The difference amounts to 104360.00€

```python
from sklearn.preprocessing import PolynomialFeatures

pf = PolynomialFeatures()
pf.fit(X_train)

X_train_transformed = pf.transform(X_train)
X_test_transformed = pf.transform(X_test)

model = model.fit(X_train_transformed, y_train)
print(model.score(X_test_transformed, y_test))
```

```
0.8616663800874034
```

```python
y_test_pred = model.predict(X_test_transformed)
y_test_pred[0]
```

```
212651.70591182058
```

The difference amounts to 9248.00€