Previous   Next

Project Scenario

In this assignment, you are a Data Analyst working at a Real Estate Investment Trust. The Trust would like to start investing in Residential real estate. You are tasked with determining the market price of a house given a set of features. You will analyze and predict housing prices using attributes or features such as square footage, number of bedrooms, number of floors, and so on. A template notebook is provided in the lab; your job is to complete the ten questions. Some hints to the questions are given in the template notebook.

Dataset Used in this Assignment

The dataset contains house sale prices for King County, which includes Seattle. It includes homes sold between May 2014 and May 2015. It was taken from here. It was also slightly modified for the purposes of this course.

For this project, you will utilize JupyterLab running on the Cloud in Skills Network Labs environment.

Notebook URL: Alternatively, you can work on your local machine or any other environment of choice, by downloading this link : Notebook link House Sales

Launcher   House_Sales_in_King_Count_   +

Markdown   Python (Pyodide)

[11]:
```
df.dtypes
```

[11]:
```
Unnamed: 0         int64
id                 int64
date              object
price            float64
bedrooms         float64
bathrooms        float64
sqft_living        int64
sqft_lot           int64
floors           float64
waterfront         int64
view               int64
condition          int64
grade              int64
sqft_above         int64
sqft_basement      int64
yr_built           int64
yr_renovated       int64
zipcode            int64
lat              float64
long             float64
sqft_living15      int64
sqft_lot15         int64
dtype: object
```

We use the method describe to obtain a statistical summary of the dataframe.

Launcher | House_Sales_in_King_Count_ | +

Markdown | Python (Pyodide)

[19]:
```python
df.drop('id',axis=1,inplace=True)
df.drop('Unnamed: 0',axis=1,inplace=True)
```

[20]:
```python
df.describe()
```

[20]:

	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	c
count	2.161300e+04	21600.000000	21603.000000	21613.000000	2.161300e+04	21613.000000	21613.000000	21613.000000	2161
mean	5.400881e+05	3.372870	2.115736	2079.899736	1.510697e+04	1.494309	0.007542	0.234303	
std	3.671272e+05	0.926657	0.768996	918.440897	4.142051e+04	0.539989	0.086517	0.766318	
min	7.500000e+04	1.000000	0.500000	290.000000	5.200000e+02	1.000000	0.000000	0.000000	
25%	3.219500e+05	3.000000	1.750000	1427.000000	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	1.068800e+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	

We can see we have missing values for the columns `bedrooms` and `bathrooms`

[ ]:
```python
print("number of NaN values for the column bedrooms :", df['bedrooms'].isnull().sum())
print("number of NaN values for the column bathrooms :", df['bathrooms'].isnull().sum())
```

Launcher | House_Sales_in_King_Count_ | +

Markdown | Python (Pyodide)

Question 3

Use the method `value_counts` to count the number of houses with unique floor values, use the method `.to_frame()` to convert it to a dataframe.

[26]:
```python
df['floors'].value_counts().to_frame()
```

[26]:

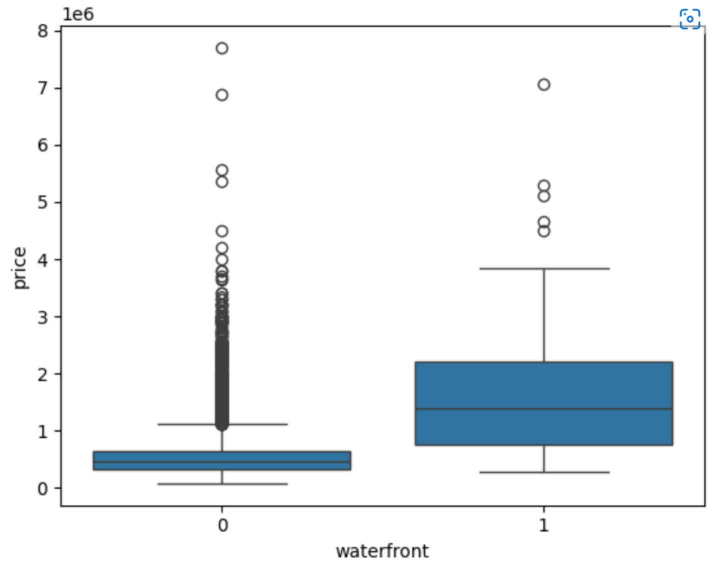
	floors
1.0	10680
2.0	8241
1.5	1910
3.0	613
2.5	161
3.5	8

Question 4

Use the function `boxplot` in the seaborn library to determine whether houses with a waterfront view or without a waterfront view have more price outliers.

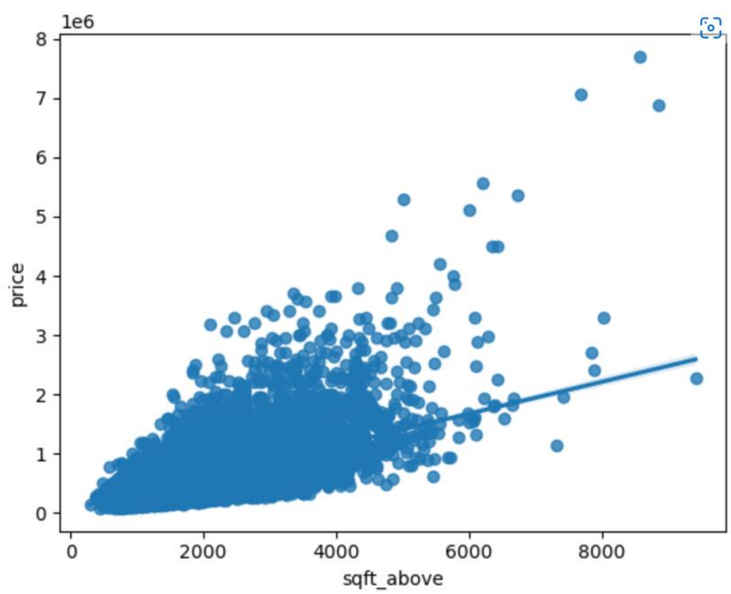
Launcher | House_Sales_in_King_Count_ | Python (Pyodide)

[27]:
```
boxplot=sns.boxplot(x='waterfront',y='price',data=df)
```



Launcher | House_Sales_in_King_Count_ | Python (Pyodide)

[28]:
```
regplot=sns.regplot(x='sqft_above',y='price',data=df)
```



Launcher | House_Sales_in_King_Count_

Markdown | Python (Pyodide)

We can use the Pandas method `corr()` to find the feature other than price that is most correlated with price.

[29]:
```
df.corr()['price'].sort_values()
```

[29]:
```
zipcode         -0.053203
long             0.021626
condition        0.036362
yr_built         0.054012
sqft_lot15       0.082447
sqft_lot         0.089661
yr_renovated     0.126434
floors           0.256794
waterfront       0.266369
lat              0.307003
bedrooms         0.308797
sqft_basement    0.323816
view             0.397293
bathrooms        0.525738
sqft_living15    0.585379
sqft_above       0.605567
grade            0.667434
sqft_living      0.702035
price            1.000000
Name: price, dtype: float64
```

Module 4: Model Development

Launcher | House_Sales_in_King_Count_

Markdown | Python (Pyodide)

Question 6

Fit a linear regression model to predict the `'price'` using the feature `'sqft_living'` then calculate the R^2. Take a screenshot of your code and the value of the R^2.

[31]:
```
x1=df[['sqft_living']]
y1=df['price']
lr1=LinearRegression()
lr1.fit(x1,y1)
lr1.score(x1,y1)
```

[31]: 0.4928532179037931

Question 7

Fit a linear regression model to predict the `'price'` using the list of features:

[ ]:
```
features =["floors", "waterfront","lat" ,"bedrooms" ,"sqft_basement" ,"view" ,"bathrooms","sqft_living15","sqft_abov
```

Then calculate the R^2. Take a screenshot of your code.

[ ]:

Launcher × | House_Sales_in_King_Count_ ● | +

Markdown | Python (Pyodide)

Question 7

Fit a linear regression model to predict the `'price'` using the list of features:

[33]:
```
features =df[["floors", "waterfront","lat" ,"bedrooms" ,"sqft_basement" ,"view" ,"bathrooms","sqft_living15","sqft_a
lr2=LinearRegression()
lr2.fit(features,y1)
```

[33]:
```
▾ LinearRegression
LinearRegression()
```

Then calculate the R^2. Take a screenshot of your code.

[34]:
```
lr2.score(features,y1)
```

[34]: 0.6576950629068081

This will help with Question 8

Create a list of tuples, the first element in the tuple contains the name of the estimator:

`'scale'`

Launcher × | House_Sales_in_King_Count_ × | +

Markdown | Python (Pyodide)

[35]:
```
Input=[('scale',StandardScaler()),('polynomial', PolynomialFeatures(include_bias=False)),('model',LinearRegression()
```

Question 8

Use the list to create a pipeline object to predict the 'price', fit the object using the features in the list `features`, and calculate the R^2.

[36]:
```
pipe=Pipeline(Input)
pipe.fit(features,y1)
pipe.score(features,y1)
```

[36]: 0.7512786321941719

Module 5: Model Evaluation and Refinement

Import the necessary modules:

[ ]:
```
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import train_test_split
print("done")
```

Launcher | House_Sales_in_King_Count_

Markdown — Python (Pyodide)

```python
print("number of test samples:", x_test.shape[0])
print("number of training samples:",x_train.shape[0])
```

```
number of test samples: 3242
number of training samples: 18371
```

Question 9

Create and fit a Ridge regression object using the training data, set the regularization parameter to 0.1, and calculate the R^2 using the test data.

[32]:
```python
from sklearn.linear_model import Ridge
```

[35]:
```python
rr=Ridge(alpha=0.1)
rr.fit(x_train,y_train)
rr.score(x_train,y_train)
```

[35]: 0.6594378534950235

Question 10

Perform a second order polynomial transform on both the training data and testing data. Create and fit a Ridge regression object using the training data, set the regularisation parameter to 0.1, and calculate the R^2 utilising the test data provided. Take a screenshot of your code and the R^2.

Launcher | House_Sales_in_King_Count_

Markdown — Python (Pyodide)

[35]: 0.6594378534950235

Question 10

Perform a second order polynomial transform on both the training data and testing data. Create and fit a Ridge regression object using the training data, set the regularisation parameter to 0.1, and calculate the R^2 utilising the test data provided. Take a screenshot of your code and the R^2.

[36]:
```python
pt=PolynomialFeatures(degree=2)
x_train_pt=pt.fit_transform(x_train)
x_test_pt=pt.fit_transform(x_test)
rr1=Ridge(alpha=0.1)
rr1.fit(x_train_pt,y_train)
rr1.score(x_test_pt,y_test)
```

[36]: 0.7002744263350642