

Instructions

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. This is a template notebook; your job is to complete the ten questions. Some hints to the questions are given.

As you are completing this notebook, take and save the **screenshots** of the final outputs of your solutions (e.g., final charts, tables, calculation results etc.). They will need to be shared in the following Peer Review section of the Final Project module.

About the Dataset

This 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.

Variable	Description
id	A notation for a house
date	Date house was sold
price	Price is prediction target
bedrooms	Number of bedrooms
bathrooms	Number of bathrooms
sqft_living	Square footage of the home
sqft_lot	Square footage of the lot
floors	Total floors (levels) in house
waterfront	House which has a view to a waterfront
view	Has been viewed
condition	How good the condition is overall
grade	overall grade given to the housing unit, based on King County grading system
sqft_above	Square footage of house apart from basement
sqft_basement	Square footage of the basement

Variable	Description
sement	
yr_built	Built Year
yr_renovated	Year when house was renovated
zipcode	Zip code
lat	Latitude coordinate
long	Longitude coordinate
sqft_living15	Living room area in 2015(implies-- some renovations) This might or might not have affected the lotsize area
sqft_lot15	LotSize area in 2015(implies-- some renovations)

Import the required libraries

```
# All Libraries required for this lab are listed below. The libraries
pre-installed on Skills Network Labs are commented.
# !mamba install -qy pandas==1.3.4 numpy==1.21.4 seaborn==0.9.0
matplotlib==3.5.0 scikit-learn==0.20.1
# Note: If your environment doesn't support "!mamba install", use "!
pip install"

# Surpress warnings:
def warn(*args, **kwargs):
    pass
import warnings
warnings.warn = warn

#!pip install -U scikit-learn

import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler, PolynomialFeatures
from sklearn.linear_model import LinearRegression
%matplotlib inline
```

Module 1: Importing Data Sets

Download the dataset by running the cell below.

```
import piplite
await piplite.install('seaborn')

from pyodide.http import pyfetch

async def download(url, filename):
    response = await pyfetch(url)
    if response.status == 200:
        with open(filename, "wb") as f:
            f.write(await response.bytes())

filepath='https://cf-courses-data.s3.us.cloud-object-
storage.appdomain.cloud/IBMDeveloperSkillsNetwork-DA0101EN-
SkillsNetwork/labs/FinalModule_Coursera/data/kc_house_data_NaN.csv'

await download(filepath, "housing.csv")
file_name="housing.csv"
```

Load the csv:

```
df = pd.read_csv(file_name)
```

Note: This version of the lab is working on JupyterLite, which requires the dataset to be downloaded to the interface. While working on the downloaded version of this notebook on their local machines(Jupyter Anaconda), the learners can simply **skip the steps above**, and simply use the URL directly in the `pandas.read_csv()` function. You can uncomment and run the statements in the cell below.

```
#filepath='https://cf-courses-data.s3.us.cloud-object-
storage.appdomain.cloud/IBMDeveloperSkillsNetwork-DA0101EN-
SkillsNetwork/labs/FinalModule_Coursera/data/kc_house_data_NaN.csv'
#df = pd.read_csv(filepath, header=None)
```

We use the method head to display the first 5 columns of the dataframe.

```
df.head()
```

	Unnamed: 0	id	date	price	bedrooms
bathrooms \					
0	0	7129300520	20141013T000000	221900.0	3.0
1.00					
1	1	6414100192	20141209T000000	538000.0	3.0
2.25					
2	2	5631500400	20150225T000000	180000.0	2.0
1.00					
3	3	2487200875	20141209T000000	604000.0	4.0
3.00					
4	4	1954400510	20150218T000000	510000.0	3.0
2.00					

	sqft_living	sqft_lot	floors	waterfront	...	grade	
sqft_above \							
0	1180	5650	1.0	0	...	7	1180
1	2570	7242	2.0	0	...	7	2170
2	770	10000	1.0	0	...	6	770
3	1960	5000	1.0	0	...	7	1050
4	1680	8080	1.0	0	...	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

[5 rows x 22 columns]

Question 1

Display the data types of each column using the function `dtypes`. Take a screenshot of your code and output. You will need to submit the screenshot for the final project.

```
df.dtypes
```

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.

```

df.describe()

```

	Unnamed: 0	id	price	bedrooms
bathrooms \				
count	21613.000000	2.161300e+04	2.161300e+04	21600.000000
mean	10806.000000	4.580302e+09	5.400881e+05	3.372870
std	6239.28002	2.876566e+09	3.671272e+05	0.926657
min	0.000000	1.000102e+06	7.500000e+04	1.000000
25%	5403.000000	2.123049e+09	3.219500e+05	3.000000
50%	10806.000000	3.904930e+09	4.500000e+05	3.000000
75%	16209.000000	7.308900e+09	6.450000e+05	4.000000
max	21612.000000	9.900000e+09	7.700000e+06	33.000000
sqft_living				
count	21613.000000	2.161300e+04	21613.000000	21613.000000
mean	2079.899736	1.510697e+04	1.494309	0.007542
std	918.440897	4.142051e+04	0.539989	0.086517
min	290.000000	5.200000e+02	1.000000	0.000000
25%	1427.000000	5.040000e+03	1.000000	0.000000
50%	1910.000000	7.618000e+03	1.500000	0.000000
75%	2550.000000	1.068800e+04	2.000000	0.000000
max				

max	13540.000000	1.651359e+06	3.500000	1.000000
4.000000				

	...	grade	sqft_above	sqft_basement	yr_built	\
count	...	21613.000000	21613.000000	21613.000000	21613.000000	
mean	...	7.656873	1788.390691	291.509045	1971.005136	
std	...	1.175459	828.090978	442.575043	29.373411	
min	...	1.000000	290.000000	0.000000	1900.000000	
25%	...	7.000000	1190.000000	0.000000	1951.000000	
50%	...	7.000000	1560.000000	0.000000	1975.000000	
75%	...	8.000000	2210.000000	560.000000	1997.000000	
max	...	13.000000	9410.000000	4820.000000	2015.000000	

	yr_renovated	zipcode	lat	long
sqft_living15	\			
count	21613.000000	21613.000000	21613.000000	21613.000000
21613.000000				
mean	84.402258	98077.939805	47.560053	-122.213896
1986.552492				
std	401.679240	53.505026	0.138564	0.140828
685.391304				
min	0.000000	98001.000000	47.155900	-122.519000
399.000000				
25%	0.000000	98033.000000	47.471000	-122.328000
1490.000000				
50%	0.000000	98065.000000	47.571800	-122.230000
1840.000000				
75%	0.000000	98118.000000	47.678000	-122.125000
2360.000000				
max	2015.000000	98199.000000	47.777600	-121.315000
6210.000000				

	sqft_lot15
count	21613.000000
mean	12768.455652
std	27304.179631
min	651.000000
25%	5100.000000
50%	7620.000000
75%	10083.000000
max	871200.000000

[8 rows x 21 columns]

Module 2: Data Wrangling

Question 2

Drop the columns "id" and "Unnamed: 0" from axis 1 using the method drop(), then use the method describe() to obtain a statistical summary of the data. Make sure the inplace parameter is set to True. Take a screenshot of your code and output. You will need to submit the screenshot for the final project.

```
df.drop(["id", "Unnamed: 0"], axis = 1, inplace = True)
df.describe()
```

	price	bedrooms	bathrooms	sqft_living
sqft_lot \				
count	2.161300e+04	21600.000000	21603.000000	21613.000000
2.161300e+04				
mean	5.400881e+05	3.372870	2.115736	2079.899736
1.510697e+04				
std	3.671272e+05	0.926657	0.768996	918.440897
4.142051e+04				
min	7.500000e+04	1.000000	0.500000	290.000000
5.200000e+02				
25%	3.219500e+05	3.000000	1.750000	1427.000000
5.040000e+03				
50%	4.500000e+05	3.000000	2.250000	1910.000000
7.618000e+03				
75%	6.450000e+05	4.000000	2.500000	2550.000000
1.068800e+04				
max	7.700000e+06	33.000000	8.000000	13540.000000
1.651359e+06				
	floors	waterfront	view	condition
grade \				
count	21613.000000	21613.000000	21613.000000	21613.000000
21613.000000				
mean	1.494309	0.007542	0.234303	3.409430
7.656873				
std	0.539989	0.086517	0.766318	0.650743
1.175459				
min	1.000000	0.000000	0.000000	1.000000
1.000000				
25%	1.000000	0.000000	0.000000	3.000000
7.000000				
50%	1.500000	0.000000	0.000000	3.000000
7.000000				
75%	2.000000	0.000000	0.000000	4.000000
8.000000				
max	3.500000	1.000000	4.000000	5.000000
13.000000				

	sqft_above	sqft_basement	yr_built	yr_renovated
zipcode \				
count	21613.000000	21613.000000	21613.000000	21613.000000
mean	1788.390691	291.509045	1971.005136	84.402258
std	828.090978	442.575043	29.373411	401.679240
min	290.000000	0.000000	1900.000000	0.000000
25%	1190.000000	0.000000	1951.000000	0.000000
50%	1560.000000	0.000000	1975.000000	0.000000
75%	2210.000000	560.000000	1997.000000	0.000000
max	9410.000000	4820.000000	2015.000000	2015.000000

	lat	long	sqft_living15	sqft_lot15
count	21613.000000	21613.000000	21613.000000	21613.000000
mean	47.560053	-122.213896	1986.552492	12768.455652
std	0.138564	0.140828	685.391304	27304.179631
min	47.155900	-122.519000	399.000000	651.000000
25%	47.471000	-122.328000	1490.000000	5100.000000
50%	47.571800	-122.230000	1840.000000	7620.000000
75%	47.678000	-122.125000	2360.000000	10083.000000
max	47.777600	-121.315000	6210.000000	871200.000000

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

```
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())
```

```
number of NaN values for the column bedrooms : 13
number of NaN values for the column bathrooms : 10
```

We can replace the missing values of the column 'bedrooms' with the mean of the column 'bedrooms' using the method replace(). Don't forget to set the inplace parameter to True

```
mean=df['bedrooms'].mean()
df['bedrooms'].replace(np.nan,mean, inplace=True)
```

We also replace the missing values of the column 'bathrooms' with the mean of the column 'bathrooms' using the method replace(). Don't forget to set the inplace parameter to True


```
mean=df['bathrooms'].mean()
df['bathrooms'].replace(np.nan,mean, inplace=True)

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())

number of NaN values for the column bedrooms : 0
number of NaN values for the column bathrooms : 0
```

Module 3: Exploratory Data Analysis

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 data frame. Take a screenshot of your code and output. You will need to submit the screenshot for the final project.

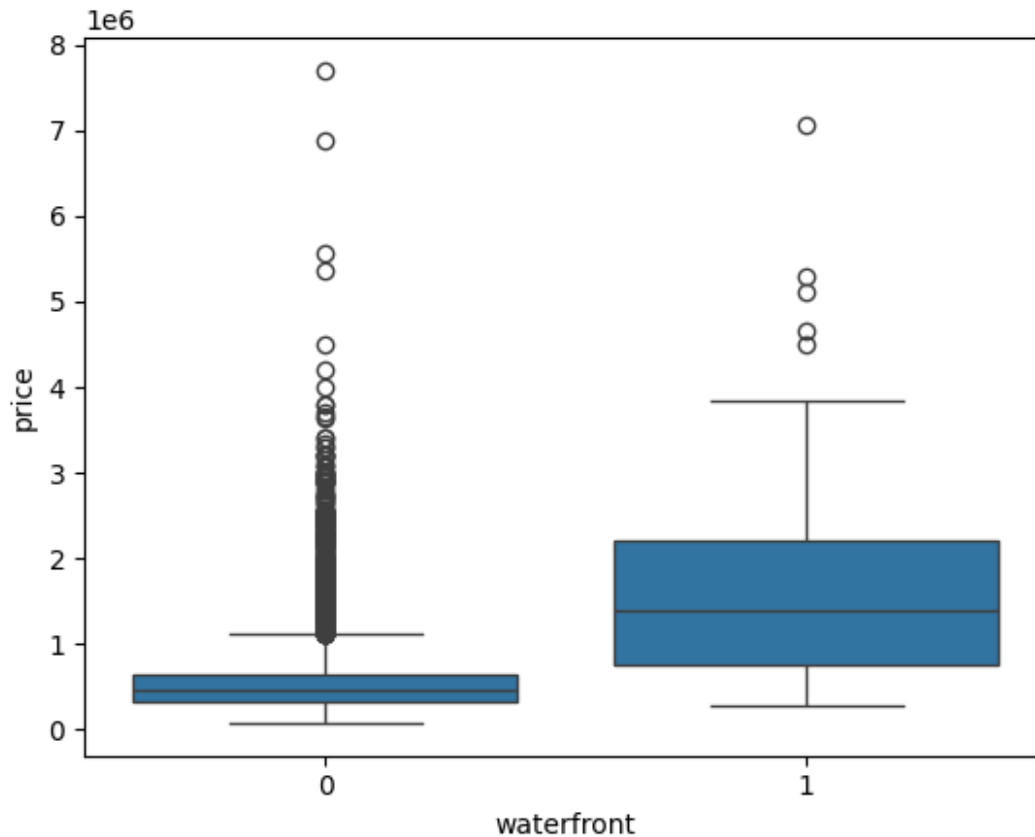
```
df["floors"].value_counts().to_frame()
```

	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. Take a screenshot of your code and boxplot. You will need to submit the screenshot for the final project.

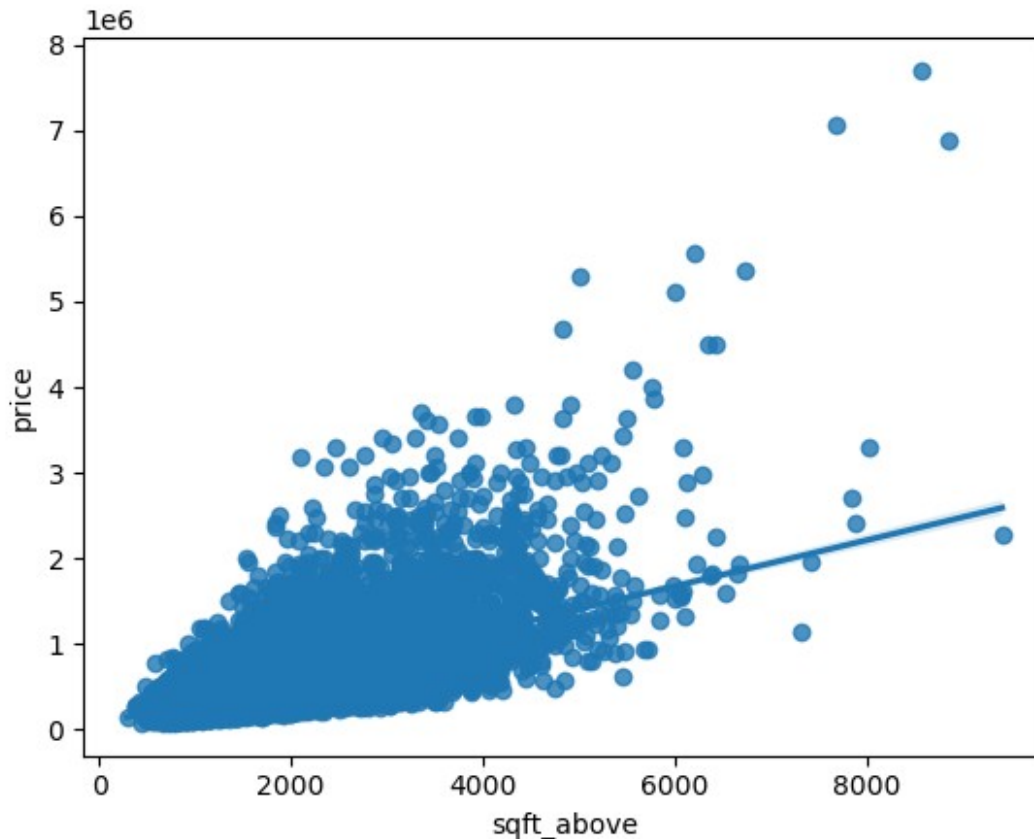
```
sns.boxplot(x="waterfront",y="price",data=df)
<AxesSubplot:xlabel='waterfront', ylabel='price'>
```



Question 5

Use the function `regplot` in the seaborn library to determine if the feature `sqft_above` is negatively or positively correlated with `price`. Take a screenshot of your code and scatterplot. You will need to submit the screenshot for the final project.

```
sns.regplot(x="sqft_above", y="price", data=df)
<AxesSubplot:xlabel='sqft_above', ylabel='price'>
```



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

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

Module 4: Model Development

We can Fit a linear regression model using the longitude feature 'long' and calculate the R^2 .

```
X = df[['long']]
Y = df['price']
lm = LinearRegression()
lm.fit(X,Y)
lm.score(X, Y)
```

```
0.00046769430149007363
```

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 . You will need to submit it for the final project.

```
X = df[['sqft_living']]
Y = df['price']
lr = LinearRegression()
lr.fit(X,Y)
lr.score(X,Y)
```

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_above","grade","sqft_living"]
```

Then calculate the R^2 . Take a screenshot of your code and the value of the R^2 . You will need to submit it for the final project.

```
lr.fit(df[features],df['price'])
lr.score(df[features],df['price'])
```

0.6576890354915759

This will help with Question 8

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

'scale'

'polynomial'

'model'

The second element in the tuple contains the model constructor

StandardScaler()

PolynomialFeatures(include_bias=False)

LinearRegression()

```
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 . Take a screenshot of your code and the value of the R^2 . You will need to submit it for the final project.

```
pipe = Pipeline(Input)
pipe
pipe.fit(df[features],df['price'])
pipe.score(df[features],df['price'])

0.7512051345272872
```

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")

done
```

We will split the data into training and testing sets:

```
features = ["floors",
"waterfront","lat" ,"bedrooms" ,"sqft_basement" ,"view" ,"bathrooms","
sqft_living15","sqft_above","grade","sqft_living"]
X = df[features]
Y = df['price']

x_train, x_test, y_train, y_test = train_test_split(X, Y,
test_size=0.15, random_state=1)

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. Take a screenshot of your code and the value of the R^2 . You will need to submit it for the final project.

```
from sklearn.linear_model import Ridge

Ridgemodel = Ridge(alpha=1)
Ridgemodel.fit(x_train,y_train)
Ridgemodel.score(x_train,y_train)

0.6594362021081352
```

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 . You will need to submit it for the final project.

```
pr = PolynomialFeatures(degree = 2)
x_train_pr = pr.fit_transform(x_train)
x_test_pr = pr.fit_transform(x_test)
RidgeModel_2 = Ridge(alpha=0.1)
RidgeModel_2.fit(x_train_pr,y_train)
RidgeModel_2.score(x_test_pr, y_test)

0.7002744263583341
```

Once you complete your notebook you will have to share it. You can download the notebook by navigating to "File" and clicking on "Download" button. This will save the (.ipynb) file on your computer. Once saved, you can upload this file in the "My Submission" tab, of the "Peer-graded Assignment" section.

Joseph Santarcangelo has a PhD in Electrical Engineering, his research focused on using machine learning, signal processing, and computer vision to determine how videos impact human cognition. Joseph has been working for IBM since he completed his PhD.

Other contributors: Michelle Carey, Mavis Zhou

Change Log

Date (YYYY-MM-DD)	Version	Changed By	Change Description
2020-12-01	2.2	Aije Egwaikhide	Coverted Data description from text to table
2020-10-06	2.1	Lakshmi Holla	Changed markdown instruction of Question1
2020-08-27	2.0	Malika Singla	Added lab to GitLab
2022-06-13	2.3	Svitlana Kramar	Updated Notebook sharing instructions

© IBM Corporation 2020. All rights reserved.

