Skills Network

Final Project: House Sales in King County, USA

Table of Contents

Estimated Time Needed: **75 min**

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
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

In [1]:
```
# All Libraries required for this lab are listed below. The libraries pre-instal
# !mamba install -qy pandas==1.3.4 numpy==1.21.4 seaborn==0.9.0 matplotlib==3.5.
# Note: If your environment doesn't support "!mamba install", use "!pip install"
```

In [2]:
```
# Surpress warnings:
def warn(*args, **kwargs):
    pass
import warnings
warnings.warn = warn
```

In [3]:
```
#!pip install -U scikit-learn
```

In [4]:
```
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.

In [7]:
```
!pip install seaborn
```

```
Requirement already satisfied: seaborn in c:\users\91939\anaconda3\lib\site-packa
ges (0.12.2)
Requirement already satisfied: numpy!=1.24.0,>=1.17 in c:\users\91939\anaconda3\l
ib\site-packages (from seaborn) (1.26.4)
Requirement already satisfied: pandas>=0.25 in c:\users\91939\anaconda3\lib\site-
packages (from seaborn) (2.1.4)
Requirement already satisfied: matplotlib!=3.6.1,>=3.1 in c:\users\91939\anaconda
3\lib\site-packages (from seaborn) (3.8.0)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\91939\anaconda3\lib\s
ite-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (1.2.0)
Requirement already satisfied: cycler>=0.10 in c:\users\91939\anaconda3\lib\site-
packages (from matplotlib!=3.6.1,>=3.1->seaborn) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\91939\anaconda3\lib
\site-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (4.25.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\91939\anaconda3\lib
\site-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (1.4.4)
Requirement already satisfied: packaging>=20.0 in c:\users\91939\anaconda3\lib\si
te-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (23.1)
Requirement already satisfied: pillow>=6.2.0 in c:\users\91939\anaconda3\lib\site
-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (10.2.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\91939\anaconda3\lib\s
ite-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (3.0.9)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\91939\anaconda3\l
ib\site-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\91939\anaconda3\lib\site-
packages (from pandas>=0.25->seaborn) (2023.3.post1)
Requirement already satisfied: tzdata>=2022.1 in c:\users\91939\anaconda3\lib\sit
e-packages (from pandas>=0.25->seaborn) (2023.3)
Requirement already satisfied: six>=1.5 in c:\users\91939\anaconda3\lib\site-pack
ages (from python-dateutil>=2.7->matplotlib!=3.6.1,>=3.1->seaborn) (1.16.0)
```

In [ ]:
```
import piplite
await piplite.install('seaborn')
```

In [ ]:
```
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())
```

In [9]:
```
filepath='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM
```

In [ ]:
```
await download(filepath, "housing.csv")
file_name="housing.csv"
```

Load the csv:

In [ ]:
```
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.

In [12]:
```
filepath='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM
df = pd.read_csv(filepath, header=None)
```

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

In [13]:
```
df.head()
```

Out[13]:

	0	1	2	3	4	5	6	
0	NaN	id	date	price	bedrooms	bathrooms	sqft_living	sqft_
1	0.0	7129300520	20141013T000000	221900.0	3.0	1.0	1180	56
2	1.0	6414100192	20141209T000000	538000.0	3.0	2.25	2570	72
3	2.0	5631500400	20150225T000000	180000.0	2.0	1.0	770	100
4	3.0	2487200875	20141209T000000	604000.0	4.0	3.0	1960	50

5 rows × 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.

In [14]:
```
#Enter Your Code, Execute and take the Screenshot
print(df.dtypes)
```

```
0      float64
1       object
2       object
3       object
4       object
5       object
6       object
7       object
8       object
9       object
10      object
11      object
12      object
13      object
14      object
15      object
16      object
17      object
18      object
19      object
20      object
21      object
dtype: object
```

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

In [15]:
```
df.describe()
```

Out[15]:

	0
count	21613.00000
mean	10806.00000
std	6239.28002
min	0.00000
25%	5403.00000
50%	10806.00000
75%	16209.00000
max	21612.00000

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.

In [29]:
```python
# Load the dataset with the first row as the header
filepath = 'https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/I
df = pd.read_csv(filepath)

# Display the first few rows of the dataframe
df.head()

# Display the columns of the dataframe
df.columns
```

Out[29]:
```
Index(['Unnamed: 0', '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'],
      dtype='object')
```

In [30]:
```python
# Dropping columns "id" and "Unnamed: 0" from the dataframe
df.drop(['id', 'Unnamed: 0'], axis=1, inplace=True)

# Obtaining statistical summary of the data
summary = df.describe()

# Displaying the statistical summary
print(summary)
```

```
              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  21613.000000
mean    1788.390691     291.509045   1971.005136     84.402258  98077.939805
std      828.090978     442.575043     29.373411    401.679240     53.505026
min      290.000000       0.000000   1900.000000      0.000000  98001.000000
25%     1190.000000       0.000000   1951.000000      0.000000  98033.000000
50%     1560.000000       0.000000   1975.000000      0.000000  98065.000000
75%     2210.000000     560.000000   1997.000000      0.000000  98118.000000
max     9410.000000    4820.000000   2015.000000   2015.000000  98199.000000

                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`

In [31]:
```
print("number of NaN values for the column bedrooms :", df['bedrooms'].isnull().
print("number of NaN values for the column bathrooms :", df['bathrooms'].isnull(
```

```
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`

In [32]:
```
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 top `True`

```
In [33]: mean=df['bathrooms'].mean()
         df['bathrooms'].replace(np.nan,mean, inplace=True)
```

```
In [34]: print("number of NaN values for the column bedrooms :", df['bedrooms'].isnull().
         print("number of NaN values for the column bathrooms :", df['bathrooms'].isnull(
```

```
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.

```
In [35]: #Enter Your Code, Execute and take the Screenshot
         # Counting number of houses with unique floor values
         floor_counts = df['floors'].value_counts().to_frame()

         # Displaying the dataframe
         print(floor_counts)
```

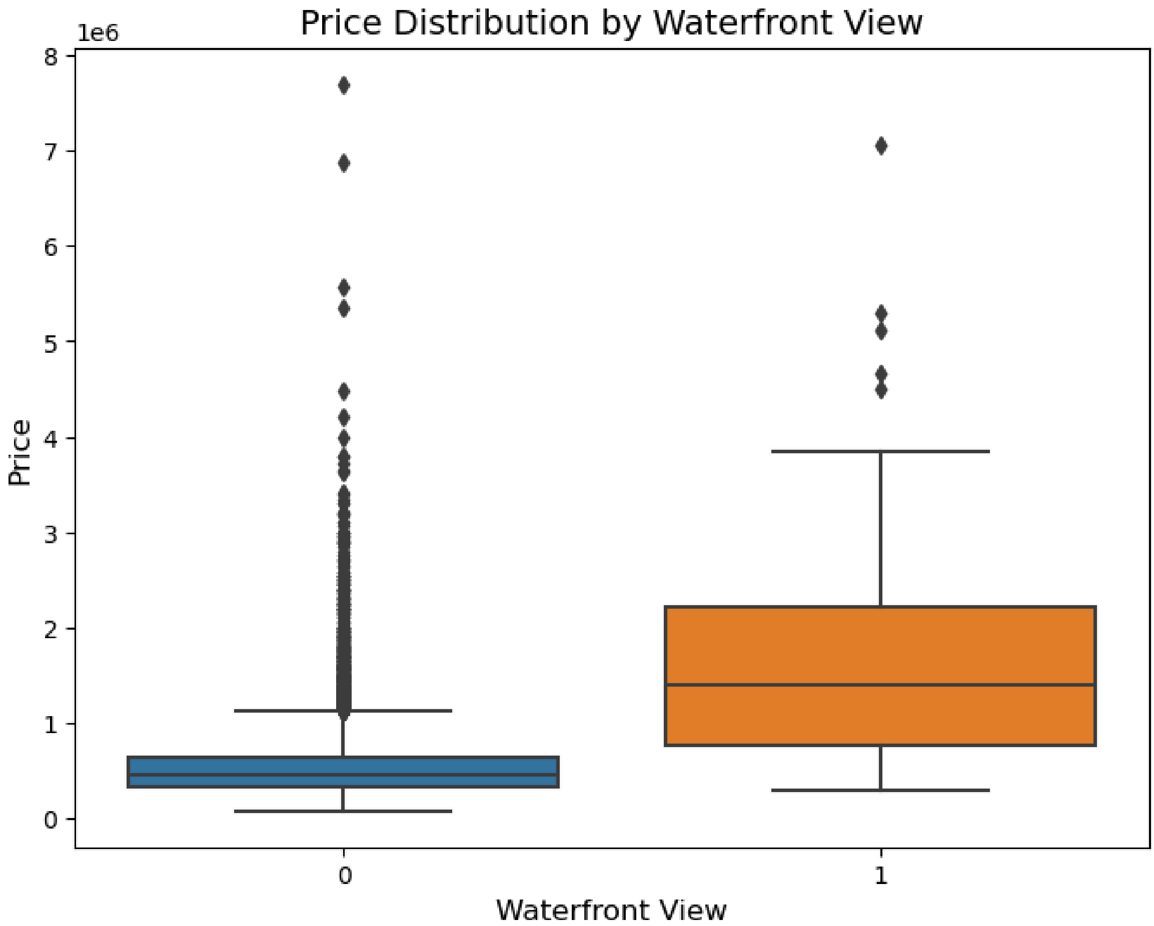
```
        count
floors
1.0     10680
2.0      8241
1.5      1910
3.0       613
2.5       161
3.5         8
```

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.

```
In [36]: import seaborn as sns
         import matplotlib.pyplot as plt

         # Set up the figure size
         plt.figure(figsize=(8, 6))

         # Create the boxplot
         sns.boxplot(x='waterfront', y='price', data=df)

         # Set the labels and title
         plt.xlabel('Waterfront View', fontsize=12)
         plt.ylabel('Price', fontsize=12)
         plt.title('Price Distribution by Waterfront View', fontsize=14)
```

```
# Show the plot
plt.show()
```



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.

In [37]:

```
#Enter Your Code, Execute and take the Screenshot
import seaborn as sns
import matplotlib.pyplot as plt

# Set up the figure size
plt.figure(figsize=(8, 6))

# Create the scatter plot with regression line
sns.regplot(x='sqft_above', y='price', data=df, scatter_kws={'alpha':0.5})

# Set the labels and title
plt.xlabel('Sqft Above', fontsize=12)
plt.ylabel('Price', fontsize=12)
plt.title('Price vs Sqft Above', fontsize=14)

# Show the plot
plt.show()
```

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

In [42]:
```
import pandas as pd

# Exclude non-numeric columns
numeric_cols = df.select_dtypes(include=['float64', 'int64']).columns

# Calculate correlation matrix for numeric columns only
correlation_matrix = df[numeric_cols].corr()

# Display correlation with 'price' column, sorted by absolute values in descendi
price_corr = correlation_matrix['price'].abs().sort_values(ascending=False)

# Print the most correlated feature (excluding 'price' itself)
most_correlated_feature = price_corr.index[1]  # Index 0 is 'price' itself, so w
print(f"The feature most correlated with price (excluding 'price' itself) is: {m
```

```
The feature most correlated with price (excluding 'price' itself) is: sqft_living
```

Module 4: Model Development

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

In [43]:
```
X = df[['long']]
Y = df['price']
lm = LinearRegression()
```

```
lm.fit(X,Y)
lm.score(X, Y)
```

Out[43]: 0.00046769430149029567

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.

In [44]:
```
#Enter Your Code, Execute and take the Screenshot
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score

# Define the feature (X) and target (y)
X = df[['sqft_living']]  # Feature
y = df['price']           # Target

# Create a linear regression model
model = LinearRegression()

# Fit the model
model.fit(X, y)

# Make predictions
y_pred = model.predict(X)

# Calculate R^2 score
r2 = r2_score(y, y_pred)

print(f"R^2 score: {r2}")
```

```
R^2 score: 0.4928532179037931
```

Question 7

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

In [45]:
```
features =["floors", "waterfront","lat" ,"bedrooms" ,"sqft_basement" ,"view" ,"b
```

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.

In [47]:
```
#Enter Your Code, Execute and take the Screenshot
X = df[features]  # Features
y = df['price']   # Target

# Create a linear regression model
model = LinearRegression()

# Fit the model
model.fit(X, y)

# Make predictions
```

```
y_pred = model.predict(X)

# Calculate R^2 score
r2 = r2_score(y, y_pred)

print(f"R^2 score: {r2}")
```

```
R^2 score: 0.6576951666037496
```

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()`

In [ ]:
```
Input=[('scale',StandardScaler()),('polynomial', PolynomialFeatures(include_bias
```

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.

In [48]:
```
#Enter Your Code, Execute and take the Screenshot
import pandas as pd
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler, PolynomialFeatures
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score

# Define the list of features
features = ["floors", "waterfront", "lat", "bedrooms", "sqft_basement",
            "view", "bathrooms", "sqft_living15", "sqft_above", "grade", "sqft_l

# Create pipeline
pipeline = Pipeline([
    ('scale', StandardScaler()),       # Step 1: Scale the features
    ('polynomial', PolynomialFeatures(include_bias=False)),  # Step 2: Create po
    ('model', LinearRegression())      # Step 3: Linear regression model
])

# Define feature matrix (X) and target vector (y)
X = df[features]  # Features
y = df['price']   # Target
```

```
# Fit the pipeline
pipeline.fit(X, y)

# Make predictions
y_pred = pipeline.predict(X)

# Calculate R^2 score
r2 = r2_score(y, y_pred)

print(f"R^2 score using pipeline: {r2}")
```

```
R^2 score using pipeline: 0.7513408009657256
```

Module 5: Model Evaluation and Refinement

Import the necessary modules:

In [49]:
```
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:

In [50]:
```
features =["floors", "waterfront","lat" ,"bedrooms" ,"sqft_basement" ,"view" ,"b
X = df[features]
Y = df['price']

x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.15, random


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.

In [ ]:
```
from sklearn.linear_model import Ridge
```

In [51]:
```
#Enter Your Code, Execute and take the Screenshot
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import Ridge
from sklearn.metrics import r2_score

# Define the list of features
features = ["floors", "waterfront", "lat", "bedrooms", "sqft_basement",
```

```
            "view", "bathrooms", "sqft_living15", "sqft_above", "grade", "sqft_l

# Define feature matrix (X) and target vector (y)
X = df[features]  # Features
y = df['price']   # Target

# Split the data into training and test sets (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_

# Create Ridge regression object with alpha=0.1
ridge = Ridge(alpha=0.1)

# Fit the Ridge model on training data
ridge.fit(X_train, y_train)

# Make predictions on test data
y_pred = ridge.predict(X_test)

# Calculate R^2 score on test data
r2 = r2_score(y_test, y_pred)

print(f"R^2 score on test data using Ridge regression: {r2}")
```

```
R^2 score on test data using Ridge regression: 0.6613982983090945
```

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.

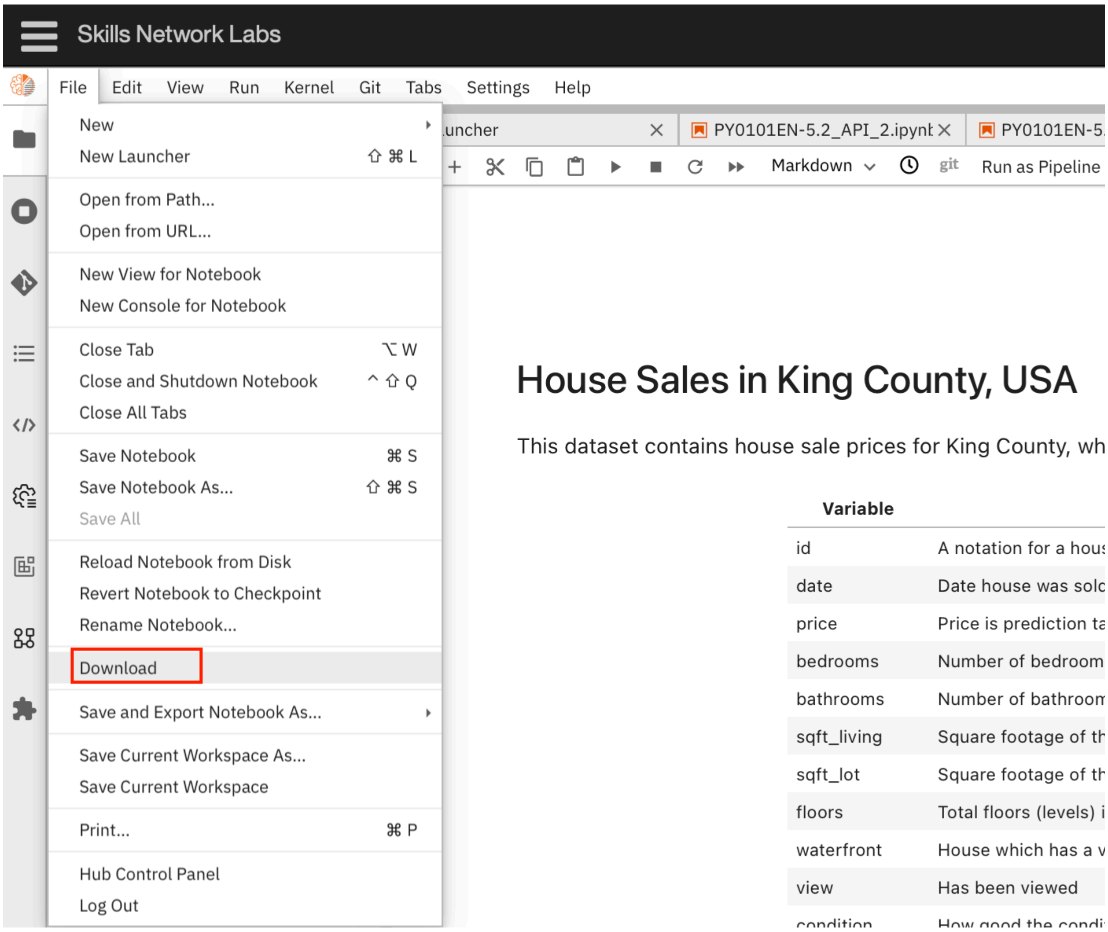
In [52]:

```
#Enter Your Code, Execute and take the Screenshot
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import Ridge
from sklearn.metrics import r2_score


# Define the list of features
features = ["floors", "waterfront", "lat", "bedrooms", "sqft_basement",
            "view", "bathrooms", "sqft_living15", "sqft_above", "grade", "sqft_l

# Define feature matrix (X) and target vector (y)
X = df[features]  # Features
y = df['price']   # Target

# Split the data into training and test sets (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_

# Create polynomial features (second order)
poly = PolynomialFeatures(degree=2)
X_train_poly = poly.fit_transform(X_train)
X_test_poly = poly.transform(X_test)

# Create Ridge regression object with alpha=0.1
ridge = Ridge(alpha=0.1)
```

```python
# Fit the Ridge model on polynomial training data
ridge.fit(X_train_poly, y_train)

# Make predictions on polynomial test data
y_pred = ridge.predict(X_test_poly)

# Calculate R^2 score on test data
r2 = r2_score(y_test, y_pred)

print(f"R^2 score on test data using Ridge regression with polynomial features:
```

```
R^2 score on test data using Ridge regression with polynomial features: 0.7000720
068163451
```

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.

About the Authors:

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 describtion 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

---

In [ ]: