Skills Network

Data Analysis with Python

House Sales in King County, USA

This dataset contains house sale prices for King County, which includes Seattle. It includes homes sold between May 2014 and May 2015.

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

Variable	Description
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)

In [5]:

```
#After executing the below command restart the kernel and run all cells.
!pip3 install scikit-learn --upgrade --user
```

```
Requirement already satisfied: scikit-learn in c:\users\akkur\appdata\roaming\python\p
ython312\site-packages (1.6.1)
Requirement already satisfied: numpy>=1.19.5 in c:\users\akkur\anaconda3\lib\site-pack
ages (from scikit-learn) (1.26.4)
Requirement already satisfied: scipy>=1.6.0 in c:\users\akkur\anaconda3\lib\site-packa
ges (from scikit-learn) (1.13.1)
Requirement already satisfied: joblib>=1.2.0 in c:\users\akkur\anaconda3\lib\site-pack
ages (from scikit-learn) (1.4.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in c:\users\akkur\anaconda3\lib\si
te-packages (from scikit-learn) (3.5.0)
```

You will require the following libraries:

In [7]:

```
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

Load the csv:

In [10]:

```
file_name='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeve
df=pd.read_csv(file_name)
```

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

In [12]:

```
df.head()
```

Out[12]:

	Unnamed: 0	id	date	price	bedrooms	bathrooms	sqft_living
0	0	7129300520	20141013T000000	221900.0	3.0	1.00	1180
1	1	6414100192	20141209T000000	538000.0	3.0	2.25	2570
2	2	5631500400	20150225T000000	180000.0	2.0	1.00	770
3	3	2487200875	20141209T000000	604000.0	4.0	3.00	1960
4	4	1954400510	20150218T000000	510000.0	3.0	2.00	1680

5 rows × 22 columns

Question 1

Display the data types of each column using the function dtypes, then take a screenshot and submit it, include your code in the image.

In [34]:

```
print(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.

In [15]:

```
df.describe()
```

Out[15]:

	Unnamed: 0	id	price	bedrooms	bathrooms	sqft_living
count	21613.00000	2.161300e+04	2.161300e+04	21600.000000	21603.000000	21613.000000
mean	10806.00000	4.580302e+09	5.400881e+05	3.372870	2.115736	2079.899736
std	6239.28002	2.876566e+09	3.671272e+05	0.926657	0.768996	918.440897
min	0.00000	1.000102e+06	7.500000e+04	1.000000	0.500000	290.000000
25%	5403.00000	2.123049e+09	3.219500e+05	3.000000	1.750000	1427.000000
50%	10806.00000	3.904930e+09	4.500000e+05	3.000000	2.250000	1910.000000
75%	16209.00000	7.308900e+09	6.450000e+05	4.000000	2.500000	2550.000000
max	21612.00000	9.900000e+09	7.700000e+06	33.000000	8.000000	13540.000000

8 rows × 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. Take a screenshot and submit it, make sure the `inplace` parameter is set to `True`

In [42]:

```
import numpy as np
import pandas as pd

df = pd.read_csv("https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/
df.drop(["id", "Unnamed: 0"], axis=1, inplace=True)

# Hatalı (uyarı veren) yöntem:
# df["bedrooms"].replace(np.nan, df["bedrooms"].mean(), inplace=True)

# Doğru yöntem:
df.replace({"bedrooms": {np.nan: df["bedrooms"].mean()},
            "bathrooms": {np.nan: df["bathrooms"].mean()}},
           inplace=True)

# Kontrol için:
print(df.isnull().sum())  # Eksik değer kaldı mı?
```

```
date             0
price            0
bedrooms         0
bathrooms        0
sqft_living      0
sqft_lot         0
floors           0
waterfront       0
view             0
condition        0
grade            0
sqft_above       0
sqft_basement    0
yr_built         0
yr_renovated     0
zipcode          0
lat              0
long             0
sqft_living15    0
sqft_lot15       0
dtype: int64
```

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

In [19]:
```
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`

In [44]:
```
mean=df['bedrooms'].mean()
df['bedrooms'].replace(np.nan,mean, inplace=True)
```

```
C:\Users\akkur\AppData\Local\Temp\ipykernel_28948\4091211281.py:2: FutureWarning: A va
lue is trying to be set on a copy of a DataFrame or Series through chained assignment
using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because th
e intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({c
ol: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform the
operation inplace on the original object.


  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 [23]:
```
mean=df['bathrooms'].mean()
df['bathrooms'].replace(np.nan,mean, inplace=True)
```

```
C:\Users\akkur\AppData\Local\Temp\ipykernel_28948\1207139423.py:2: FutureWarning: A va
lue is trying to be set on a copy of a DataFrame or Series through chained assignment
using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because th
e intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({c
ol: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform the
operation inplace on the original object.


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

In [24]:
```
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 dataframe.

In [46]:
```
df["floors"].value_counts().to_frame()
```

Out[46]:

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

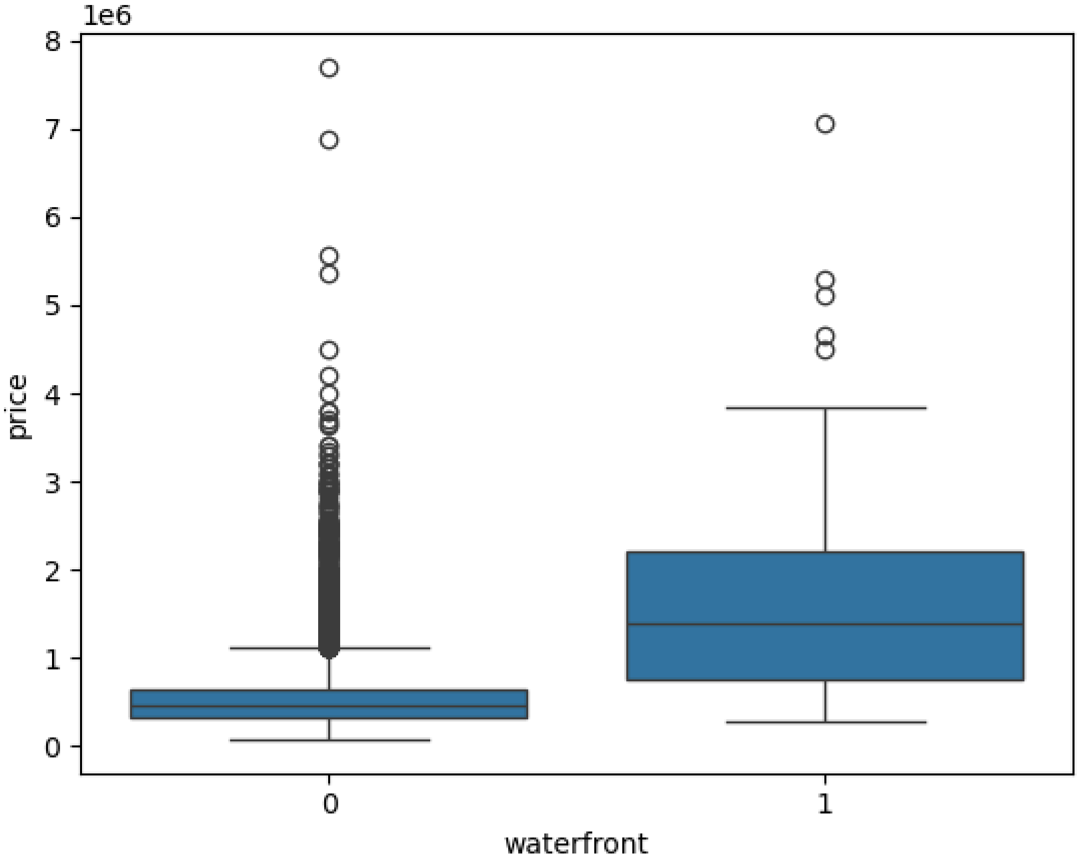
Question 4

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.

In [48]:

```
import seaborn as sns
import matplotlib.pyplot as plt

sns.boxplot(x="waterfront", y="price", data=df)
plt.show()
```



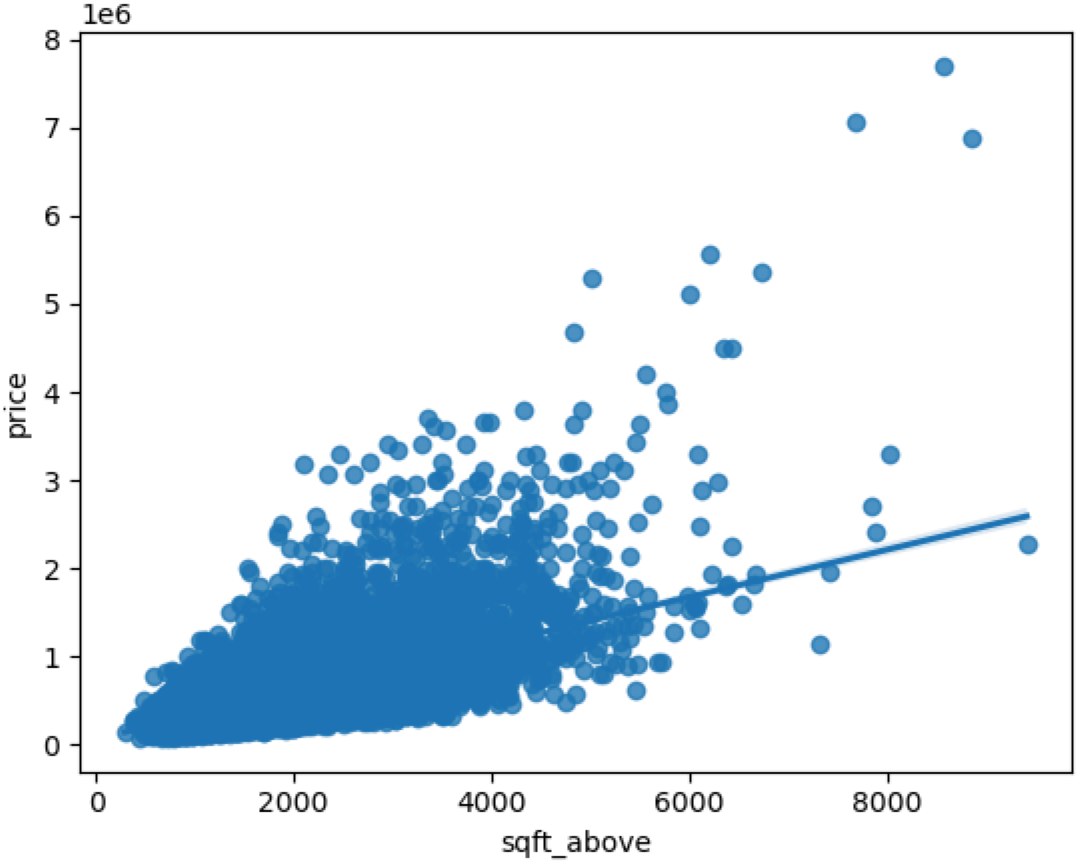
Question 5

Use the function `regplot` in the seaborn library to determine if the feature `sqft_above` is negatively or positively correlated with price.

In [50]:

```
sns.regplot(x="sqft_above", y="price", data=df)
plt.show()

# Korelasyon
df.corr()["price"].sort_values(ascending=False)
```


1e6
8
7
6
5
4
3
2
1
0
price
0
2000
4000
6000
8000
sqft_above

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[50], line 5
      2 plt.show()
      4 # Korelasyon
----> 5 df.corr()["price"].sort_values(ascending=False)

File ~\anaconda3\Lib\site-packages\pandas\core\frame.py:11049, in DataFrame.corr(self,
method, min_periods, numeric_only)
  11047 cols = data.columns
  11048 idx = cols.copy()
> 11049 mat = data.to_numpy(dtype=float, na_value=np.nan, copy=False)
  11051 if method == "pearson":
  11052     correl = libalgos.nancorr(mat, minp=min_periods)

File ~\anaconda3\Lib\site-packages\pandas\core\frame.py:1993, in DataFrame.to_numpy(se
lf, dtype, copy, na_value)
   1991 if dtype is not None:
   1992     dtype = np.dtype(dtype)
-> 1993 result = self._mgr.as_array(dtype=dtype, copy=copy, na_value=na_value)
   1994 if result.dtype is not dtype:
   1995     result = np.asarray(result, dtype=dtype)

File ~\anaconda3\Lib\site-packages\pandas\core\internals\managers.py:1694, in BlockMan
ager.as_array(self, dtype, copy, na_value)
   1692         arr.flags.writeable = False
   1693 else:
-> 1694     arr = self._interleave(dtype=dtype, na_value=na_value)
   1695     # The underlying data was copied within _interleave, so no need
   1696     # to further copy if copy=True or setting na_value
   1698 if na_value is lib.no_default:

File ~\anaconda3\Lib\site-packages\pandas\core\internals\managers.py:1753, in BlockMan
ager._interleave(self, dtype, na_value)
   1751     else:
   1752         arr = blk.get_values(dtype)
-> 1753     result[rl.indexer] = arr
   1754     itemmask[rl.indexer] = 1
   1756 if not itemmask.all():

ValueError: could not convert string to float: '20141013T000000'
```

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

In [30]:

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

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[30], line 1
----> 1 df.corr()['price'].sort_values()

File ~\anaconda3\Lib\site-packages\pandas\core\frame.py:11049, in DataFrame.corr(self, method, min_periods, numeric_only)
  11047 cols = data.columns
  11048 idx = cols.copy()
> 11049 mat = data.to_numpy(dtype=float, na_value=np.nan, copy=False)
  11051 if method == "pearson":
  11052     correl = libalgos.nancorr(mat, minp=min_periods)

File ~\anaconda3\Lib\site-packages\pandas\core\frame.py:1993, in DataFrame.to_numpy(self, dtype, copy, na_value)
   1991 if dtype is not None:
   1992     dtype = np.dtype(dtype)
-> 1993 result = self._mgr.as_array(dtype=dtype, copy=copy, na_value=na_value)
   1994 if result.dtype is not dtype:
   1995     result = np.asarray(result, dtype=dtype)

File ~\anaconda3\Lib\site-packages\pandas\core\internals\managers.py:1694, in BlockManager.as_array(self, dtype, copy, na_value)
   1692         arr.flags.writeable = False
   1693 else:
-> 1694     arr = self._interleave(dtype=dtype, na_value=na_value)
   1695     # The underlying data was copied within _interleave, so no need
   1696     # to further copy if copy=True or setting na_value
   1698 if na_value is lib.no_default:

File ~\anaconda3\Lib\site-packages\pandas\core\internals\managers.py:1753, in BlockManager._interleave(self, dtype, na_value)
   1751     else:
   1752         arr = blk.get_values(dtype)
-> 1753     result[rl.indexer] = arr
   1754     itemmask[rl.indexer] = 1
   1756 if not itemmask.all():

ValueError: could not convert string to float: '20141013T000000'
```

Module 4: Model Development

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

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

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.

In [52]:
```
from sklearn.linear_model import LinearRegression

X = df[["sqft_living"]]
y = df["price"]

lm = LinearRegression()
lm.fit(X, y)

r2 = lm.score(X, y)
print("R^2:", r2)
```

```
R^2: 0.4928532179037931
```

Question 7

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

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

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

In [54]:
```
features = ["floors", "waterfront", "lat", "bedrooms", "sqft_basement", "view", "bath
X = df[features]
y = df["price"]

lm = LinearRegression()
lm.fit(X, y)

r2 = lm.score(X, y)
print("R^2:", r2)
```

```
R^2: 0.6576951666037504
```

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=Fals
```

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.

In [56]:

```
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler

pipeline = Pipeline([
    ("scale", StandardScaler()),
    ("model", LinearRegression())
])

pipeline.fit(X, y)
print("R^2:", pipeline.score(X, y))
```

```
R^2: 0.65769516660375
```

Module 5: Model Evaluation and Refinement

Import the necessary modules:

In [ ]:

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

We will split the data into training and testing sets:

In [ ]:

```
features =["floors", "waterfront","lat" ,"bedrooms" ,"sqft_basement" ,"view" ,"bathro
X = df[features]
Y = df['price']

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


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

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.

In [ ]:

```
from sklearn.linear_model import Ridge
```

In [58]:

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import Ridge

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state
ridge = Ridge(alpha=0.1)
ridge.fit(X_train, y_train)
print("R^2:", ridge.score(X_test, y_test))
```

```
R^2: 0.6459152254891412
```

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.

In [60]:

```
from sklearn.preprocessing import PolynomialFeatures

poly = PolynomialFeatures(degree=2)
X_train_poly = poly.fit_transform(X_train)
X_test_poly = poly.transform(X_test)

ridge_poly = Ridge(alpha=0.1)
ridge_poly.fit(X_train_poly, y_train)
print("R^2:", ridge_poly.score(X_test_poly, y_test))
```

```
R^2: 0.7543633738047011
```

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
2022-07-29	2.3	Lakshmi Holla	Added library import
2020-12-01	2.2	Aije Egwaikhide	Coverted Data describtion from text to table

Date (YYYY-MM-DD)	Version	Changed By	Change Description
2020-10-06	2.1	Lakshmi Holla	Changed markdown instruction of Question1
2020-08-27	2.0	Malika Singla	Added lab to GitLab