Data Analytics for House Pricing Data Set

November 21, 2024

1 Question 1-Display the data types of each column using the attribute dtypes, then take a screenshot and submit it. Include your code in the image.

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

[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
```

[6]:
```
import sys
print(sys.executable)
```

```
C:\Users\Neha\anaconda3\python.exe
```

[22]:
```
import pandas as pd

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

# Load the dataset from the URL
df = pd.read_csv(filepath)

# Display the data types of each column
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
```

[24]:
```
import pandas as pd

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

# Load the dataset from the URL
df = pd.read_csv(filepath)

# Drop the specified columns from axis 1 (columns) and set inplace=True
df.drop(['id', 'Unnamed: 0'], axis=1, inplace=True)

# Obtain a statistical summary of the data
print(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  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
```

[26]:

```python
import pandas as pd

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

# Load the dataset from the URL
df = pd.read_csv(filepath)

# Count the number of houses with unique floor values
floor_counts = df['floors'].value_counts()

# Convert the result to a DataFrame
floor_counts_df = floor_counts.to_frame()

# Rename the column to make it descriptive
```

```
floor_counts_df.columns = ['Number of Houses']

# Display the DataFrame
print(floor_counts_df)
```

```
        Number of Houses
floors
1.0                10680
2.0                 8241
1.5                 1910
3.0                  613
2.5                  161
3.5                    8
```

[28]:

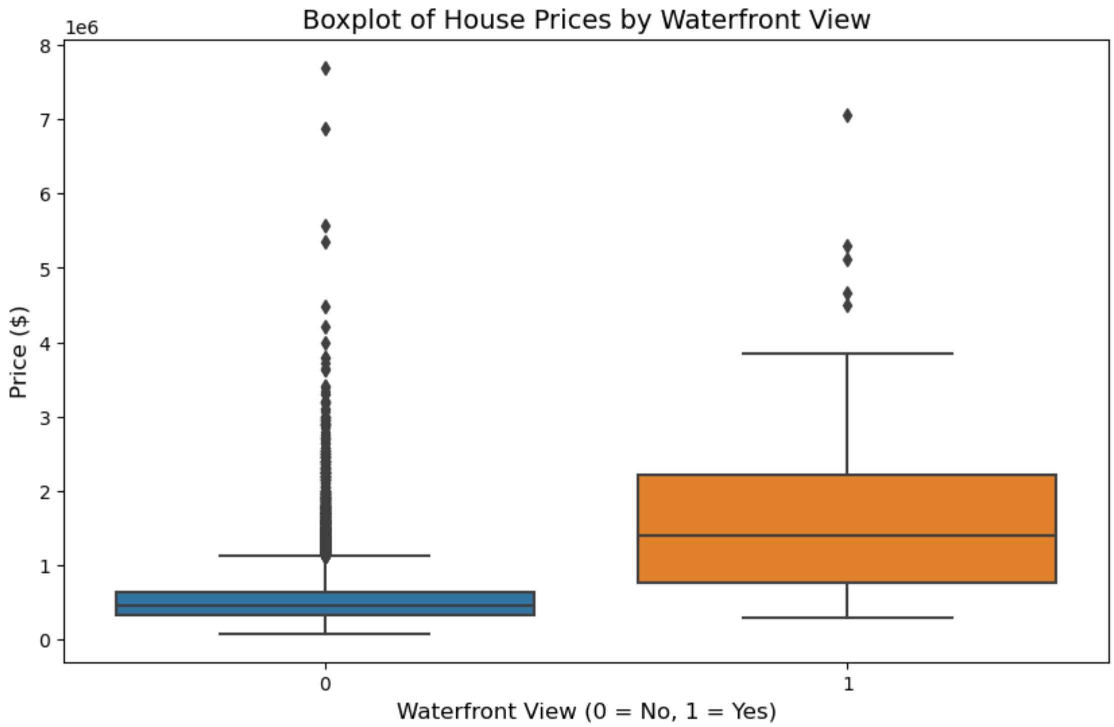
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Load the dataset from the provided URL
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)

# Create a boxplot for house prices based on the waterfront view
plt.figure(figsize=(10, 6))  # Adjust the figure size for better visibility
sns.boxplot(x='waterfront', y='price', data=df)

# Add labels and title
plt.xlabel('Waterfront View (0 = No, 1 = Yes)', fontsize=12)
plt.ylabel('Price ($)', fontsize=12)
plt.title('Boxplot of House Prices by Waterfront View', fontsize=14)

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



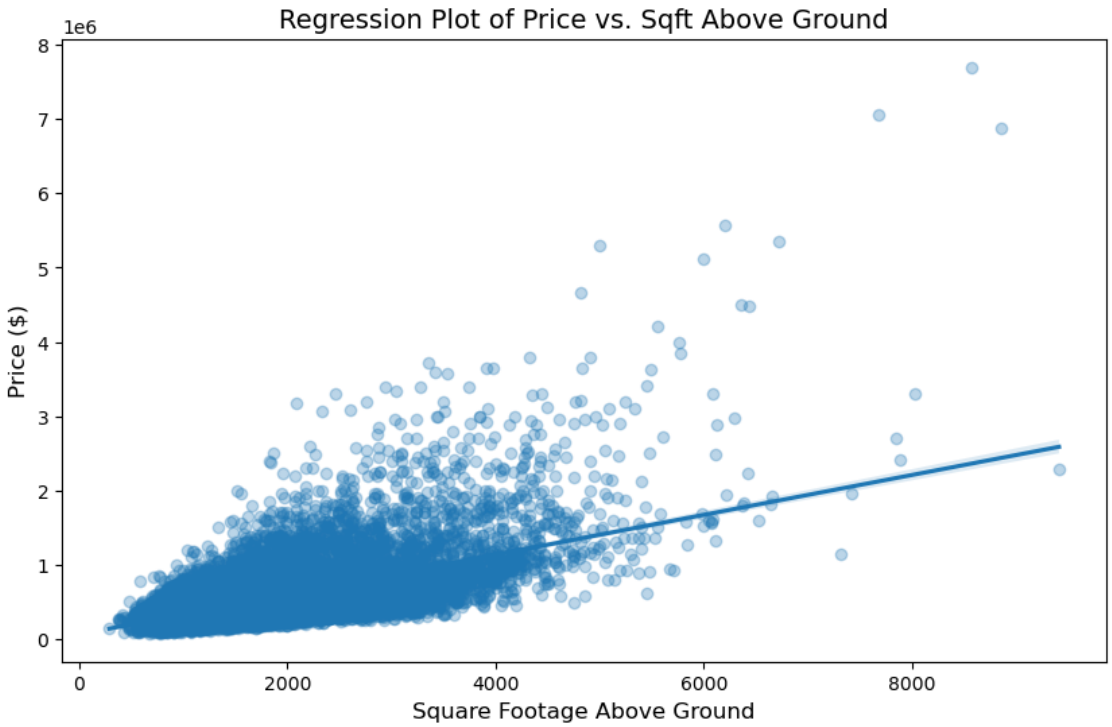
[30]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Load the dataset from the provided URL
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)

# Create a regression plot
plt.figure(figsize=(10, 6))  # Adjust the figure size for better visibility
sns.regplot(x='sqft_above', y='price', data=df, scatter_kws={'alpha': 0.3})

# Add labels and title
plt.xlabel('Square Footage Above Ground', fontsize=12)
plt.ylabel('Price ($)', fontsize=12)
plt.title('Regression Plot of Price vs. Sqft Above Ground', fontsize=14)

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



[32]:
```
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score

# Load the dataset from the provided URL
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)

# Select the feature and target variable
X = df[['sqft_living']]  # Independent variable
y = df['price']          # Dependent variable

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
 ↪random_state=42)

# Fit the linear regression model
model = LinearRegression()
model.fit(X_train, y_train)
```

```
# Predict on the test set
y_pred = model.predict(X_test)

# Calculate the R² score
r2 = r2_score(y_test, y_pred)

# Print the R² value
print(f'R² Score: {r2:.4f}')
```

```
R² Score: 0.4839
```

[36]:
```
from sklearn.impute import SimpleImputer

# Replace NaN values with the mean of each column
imputer = SimpleImputer(strategy='mean')
df[features] = imputer.fit_transform(df[features])

# Proceed with the rest of the steps
X = df[features]
y = df['price']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
 ↪random_state=42)

# Fit the linear regression model
model = LinearRegression()
model.fit(X_train, y_train)

# Predict on the test set
y_pred = model.predict(X_test)

# Calculate the R² value
r2 = r2_score(y_test, y_pred)

# Print the R² value
print(f'R² Score: {r2:.4f}')
```

```
R² Score: 0.6545
```

[38]:
```
import pandas as pd
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler, PolynomialFeatures
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score
from sklearn.impute import SimpleImputer
```

```
# Load the dataset
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)

# Define features and target
features = ["floors", "waterfront", "lat", "bedrooms", "sqft_basement",
            "view", "bathrooms", "sqft_living15", "sqft_above", "grade",␣
↪"sqft_living"]
X = df[features]
y = df['price']

# Handle missing values by filling them with the mean
imputer = SimpleImputer(strategy='mean')
X = imputer.fit_transform(X)

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
↪random_state=42)

# Create the pipeline
pipeline = Pipeline([
    ('scaler', StandardScaler()),           # Scale the data
    ('polynomial', PolynomialFeatures(degree=2)),  # Polynomial transformation
    ('model', LinearRegression())          # Linear regression model
])

# Fit the pipeline on the training data
pipeline.fit(X_train, y_train)

# Predict on the test set
y_pred = pipeline.predict(X_test)

# Calculate the R² value
r2 = r2_score(y_test, y_pred)

# Print the R² value
print(f'R² Score: {r2:.4f}')
```

```
R² Score: 0.7127
```

[40]:
```
import pandas as pd
from sklearn.linear_model import Ridge
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score
from sklearn.impute import SimpleImputer
```

```
# Load the dataset
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)

# Define features and target
features = ["floors", "waterfront", "lat", "bedrooms", "sqft_basement",
            "view", "bathrooms", "sqft_living15", "sqft_above", "grade",␣
↪"sqft_living"]
X = df[features]
y = df['price']

# Handle missing values by filling them with the mean
imputer = SimpleImputer(strategy='mean')
X = imputer.fit_transform(X)

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
↪random_state=42)

# Create and fit a Ridge regression model
ridge_model = Ridge(alpha=0.1)  # Regularization parameter set to 0.1
ridge_model.fit(X_train, y_train)

# Predict on the test set
y_pred = ridge_model.predict(X_test)

# Calculate the R² value
r2 = r2_score(y_test, y_pred)

# Print the R² value
print(f'R² Score: {r2:.4f}')
```

```
R² Score: 0.6545
```

[42]:
```
import pandas as pd
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import Ridge
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score
from sklearn.impute import SimpleImputer

# Load the dataset
```

```
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)

# Define features and target
features = ["floors", "waterfront", "lat", "bedrooms", "sqft_basement",
            "view", "bathrooms", "sqft_living15", "sqft_above", "grade",␣
↪"sqft_living"]
X = df[features]
y = df['price']

# Handle missing values by filling them with the mean
imputer = SimpleImputer(strategy='mean')
X = imputer.fit_transform(X)

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
↪random_state=42)

# Perform a second-order polynomial transform
poly = PolynomialFeatures(degree=2)
X_train_poly = poly.fit_transform(X_train)
X_test_poly = poly.transform(X_test)

# Create and fit a Ridge regression model
ridge_model = Ridge(alpha=0.1)  # Regularization parameter set to 0.1
ridge_model.fit(X_train_poly, y_train)

# Predict on the test set
y_pred = ridge_model.predict(X_test_poly)

# Calculate the R² value
r2 = r2_score(y_test, y_pred)

# Print the R² value
print(f'R² Score: {r2:.4f}')
```

```
R² Score: 0.7000
```

[ ]: