Dataset Link: https://www.kaggle.com/datasets/vedavyasv/usa-housing

Importing Necessary Libraries

In [4]:
```
#Write Your Code Here

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
import warnings

%matplotlib inline
```

Import and read dataset

In [5]:
```
#Write Your Code Here
```

```
df = pd.read_csv('USA_Housing.csv')
df.head(10)
```

Out[5]:

	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price	Address
0	79545.458574	5.682861	7.009188	4.09	23086.800503	1.059034e+06	208 Michael Ferry Apt. 674\nLaurabury, NE 3701...
1	79248.642455	6.002900	6.730821	3.09	40173.072174	1.505891e+06	188 Johnson Views Suite 079\nLake Kathleen, CA...
2	61287.067179	5.865890	8.512727	5.13	36882.159400	1.058988e+06	9127 Elizabeth Stravenue\nDanieltown, WI 06482...
3	63345.240046	7.188236	5.586729	3.26	34310.242831	1.260617e+06	USS Barnett\nFPO AP 44820
4	59982.197226	5.040555	7.839388	4.23	26354.109472	6.309435e+05	USNS Raymond\nFPO AE 09386
5	80175.754159	4.988408	6.104512	4.04	26748.428425	1.068138e+06	06039 Jennifer Islands Apt. 443\nTracyport, KS...
6	64698.463428	6.025336	8.147760	3.41	60828.249085	1.502056e+06	4759 Daniel Shoals Suite 442\nNguyenburgh, CO ...
7	78394.339278	6.989780	6.620478	2.42	36516.358972	1.573937e+06	972 Joyce Viaduct\nLake William, TN 17778-6483
8	59927.660813	5.362126	6.393121	2.30	29387.396003	7.988695e+05	USS Gilbert\nFPO AA 20957
9	81885.927184	4.423672	8.167688	6.10	40149.965749	1.545155e+06	Unit 9446 Box 0958\nDPO AE 97025

In [6]:
```
df.dtypes
```

Out[6]:
```
Avg. Area Income              float64
Avg. Area House Age           float64
Avg. Area Number of Rooms     float64
Avg. Area Number of Bedrooms  float64
Area Population               float64
Price                         float64
Address                        object
dtype: object
```

In [7]:
```
df.describe(include="all")
```

Out[7]:

	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price	Ad
count	5000.000000	5000.000000	5000.000000	5000.000000	5000.000000	5.000000e+03	
unique	NaN	NaN	NaN	NaN	NaN	NaN	
top	NaN	NaN	NaN	NaN	NaN	NaN	208 Mi Ferry 674\nLaura NE 3
freq	NaN	NaN	NaN	NaN	NaN	NaN	
mean	68583.108984	5.977222	6.987792	3.981330	36163.516039	1.232073e+06	
std	10657.991214	0.991456	1.005833	1.234137	9925.650114	3.531176e+05	
min	17796.631190	2.644304	3.236194	2.000000	172.610686	1.593866e+04	
25%	61480.562388	5.322283	6.299250	3.140000	29403.928702	9.975771e+05	
50%	68804.286404	5.970429	7.002902	4.050000	36199.406689	1.232669e+06	
75%	75783.338666	6.650808	7.665871	4.490000	42861.290769	1.471210e+06	
max	107701.748378	9.519088	10.759588	6.500000	69621.713378	2.469066e+06	

In [8]:
```
df.columns
```

Out[8]:
```
Index(['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of Rooms',
       'Avg. Area Number of Bedrooms', 'Area Population', 'Price', 'Address'],
      dtype='object')
```

In [9]:
```
columns = ['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of Rooms',
       'Avg. Area Number of Bedrooms', 'Area Population', 'Price', 'Address']
for column in columns:
    df[column] = df[column].replace('?',np.NaN)
```

Checking NaN values

In [10]:
```
df.isna().sum()
```

Out[10]:
```
Avg. Area Income              0
Avg. Area House Age           0
Avg. Area Number of Rooms     0
Avg. Area Number of Bedrooms  0
Area Population               0
Price                         0
Address                       0
dtype: int64
```

▶ **Click Here for the Hint**

Data Cleaning

a. Missing Value

In [11]:
```
#Write Your Code Here
df.isnull().sum()
```

Out[11]:
```
Avg. Area Income                0
Avg. Area House Age             0
Avg. Area Number of Rooms       0
Avg. Area Number of Bedrooms    0
Area Population                 0
Price                           0
Address                         0
dtype: int64
```

Dropping the rows where Price column has NAN values.

In [12]:
```
df.dropna(subset=['Price'],axis=0,inplace=True)
```

▶ **Click Here for the Hint**

b. Duplicate data

In [13]:
```
# Use the duplicated() method to create a Boolean DataFrame
duplicates_bool = df.duplicated()

# Count the number of True values (i.e., the number of duplicate rows)
num_duplicates = duplicates_bool.sum()

# Display the Boolean DataFrame and the number of duplicate rows
print("Boolean DataFrame indicating duplicates:")
print(duplicates_bool)

print("\nNumber of duplicate rows:", num_duplicates)
```

```
Boolean DataFrame indicating duplicates:
0       False
1       False
2       False
3       False
4       False
        ...
4995    False
4996    False
4997    False
4998    False
4999    False
Length: 5000, dtype: bool

Number of duplicate rows: 0
```

▶ **Click Here for the Hint**

Data Analysis

1. What is the correlation between the average area house age and the number of bedrooms in the houses in the area?

In [14]:
```
# Calculate the correlation between 'Avg. Area House Age' and 'Number of Bedrooms'
correlation = df['Avg. Area House Age'].corr(df['Avg. Area Number of Bedrooms'])

# Display the correlation
print("Correlation between Avg. Area House Age and Number of Bedrooms:", correlation)
```

```
Correlation between Avg. Area House Age and Number of Bedrooms: 0.0061489233483431015
```

▶ **Click Here for the Hint**

1. What is the correlation coefficient between the average area income and the price of the houses in the dataset?

In [15]:
```
# Calculate the correlation coefficient between 'Avg. Area Income' and 'Price'


correlation_coefficient = df['Avg. Area Income'].corr(df['Price'])

# Print the result
print("Correlation between Avg. Area Income and Price:", correlation_coefficient)
```

```
Correlation between Avg. Area Income and Price: 0.639733778249894
```

Data Visualization

1. Use a scatter plot to visualize the relationship between two variables (e.g. Avg. Area Income and Price):

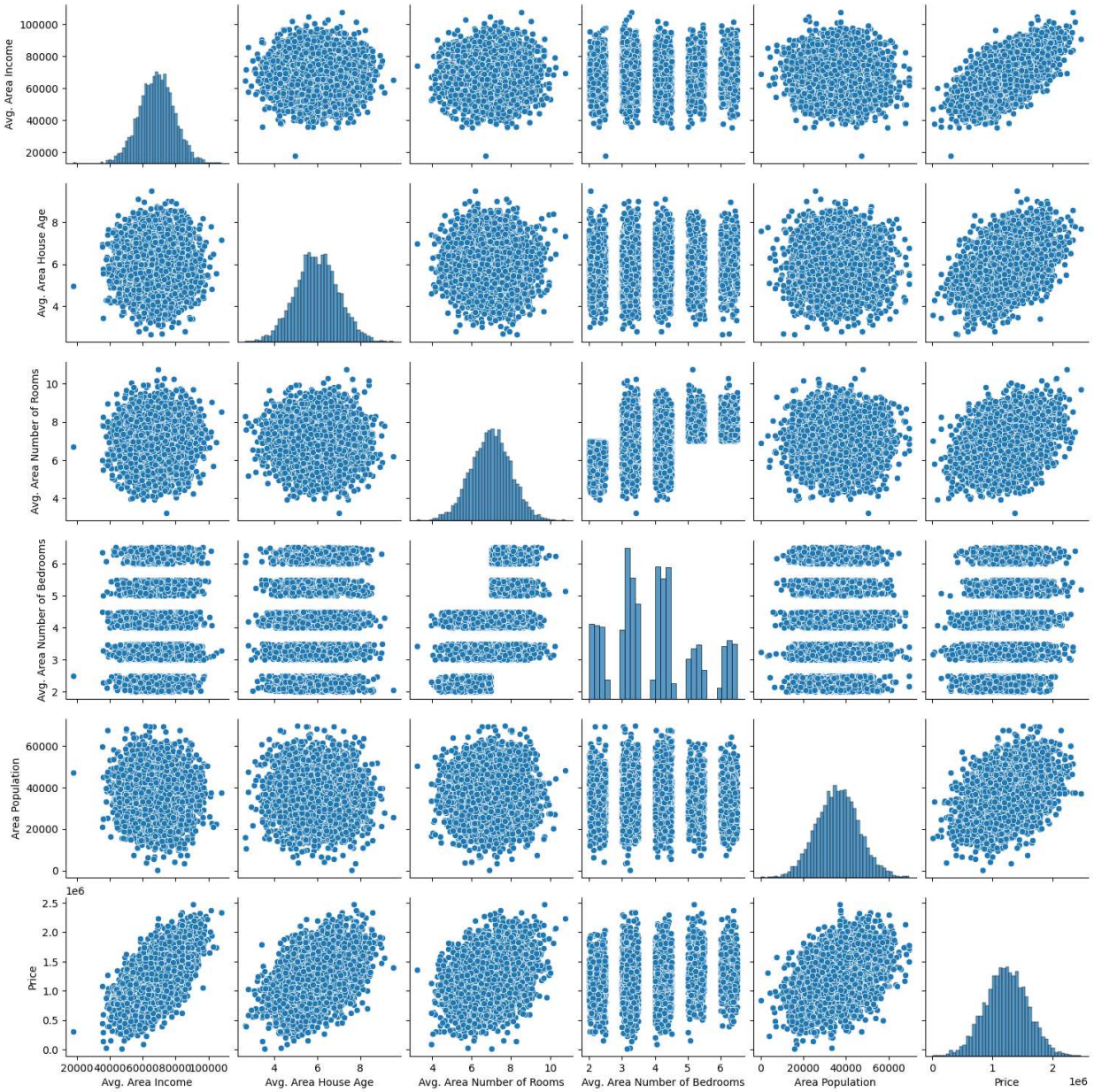
In [16]:
```
# Create a scatter plot of Avg. Area Income vs Price
plt.figure(figsize=(10, 6))
plt.scatter(df['Avg. Area Income'], df['Price'], alpha=0.6)
plt.title('Scatter Plot of Avg. Area Income vs Price')
plt.xlabel('Avg. Area Income')
plt.ylabel('Price')
plt.grid(True)
plt.show()
```

▶ **Click Here for the Hint**

1. Data visualization of dataset using sns.pairplot

In [18]:
```
# Write your code here

warnings.filterwarnings("ignore")
sns.pairplot(df)
plt.show()
```



▶ **Click Here for the Hint**

Model Development & Evaluation

In [19]:
```
df.columns
```

Out[19]:
```
Index(['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of Rooms',
       'Avg. Area Number of Bedrooms', 'Area Population', 'Price', 'Address'],
      dtype='object')
```

In [20]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report


# Create a binary target variable indicating whether a house is expensive or not
median_price = df['Price'].median()
df['Expensive'] = df['Price'] > median_price
```

```
df.head(5)
```

Out[20]:

	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price	Address	E
0	79545.458574	5.682861	7.009188	4.09	23086.800503	1.059034e+06	208 Michael Ferry Apt. 674\nLaurabury, NE 3701...	
1	79248.642455	6.002900	6.730821	3.09	40173.072174	1.505891e+06	188 Johnson Views Suite 079\nLake Kathleen, CA...	
2	61287.067179	5.865890	8.512727	5.13	36882.159400	1.058988e+06	9127 Elizabeth Stravenue\nDanieltown, WI 06482...	
3	63345.240046	7.188236	5.586729	3.26	34310.242831	1.260617e+06	USS Barnett\nFPO AP 44820	
4	59982.197226	5.040555	7.839388	4.23	26354.109472	6.309435e+05	USNS Raymond\nFPO AE 09386	

In [22]:
```
df.columns
```

Out[22]:
```
Index(['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of Rooms',
       'Avg. Area Number of Bedrooms', 'Area Population', 'Price', 'Address',
       'Expensive'],
      dtype='object')
```

▶ **Click Here for the Hint**

In [25]:
```
features = ['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of Rooms',
            'Avg. Area Number of Bedrooms', 'Area Population']
X = df[features]
y = df['Expensive']  # Assuming 'Expensive' is your binary target variable

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=

# Fit a decision tree model
model = DecisionTreeClassifier(random_state=42)
model.fit(X_train, y_train)

# Predict the classes of the testing set
predictions = model.predict(X_test)

# Calculate the accuracy of the model
accuracy = accuracy_score(y_test, predictions)
print(f"Accuracy: {accuracy:.2f}")

# Evaluate the performance of the model
# Confusion Matrix and Classification Report
conf_matrix = confusion_matrix(y_test, predictions)
class_report = classification_report(y_test, predictions)
```

```
print("\nConfusion Matrix:")
print(conf_matrix)

print("\nClassification Report:")
print(class_report)

plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', cbar=False,
            xticklabels=['Not Expensive', 'Expensive'], yticklabels=['Not Expensive',
plt.title('Confusion Matrix')
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.show()
```

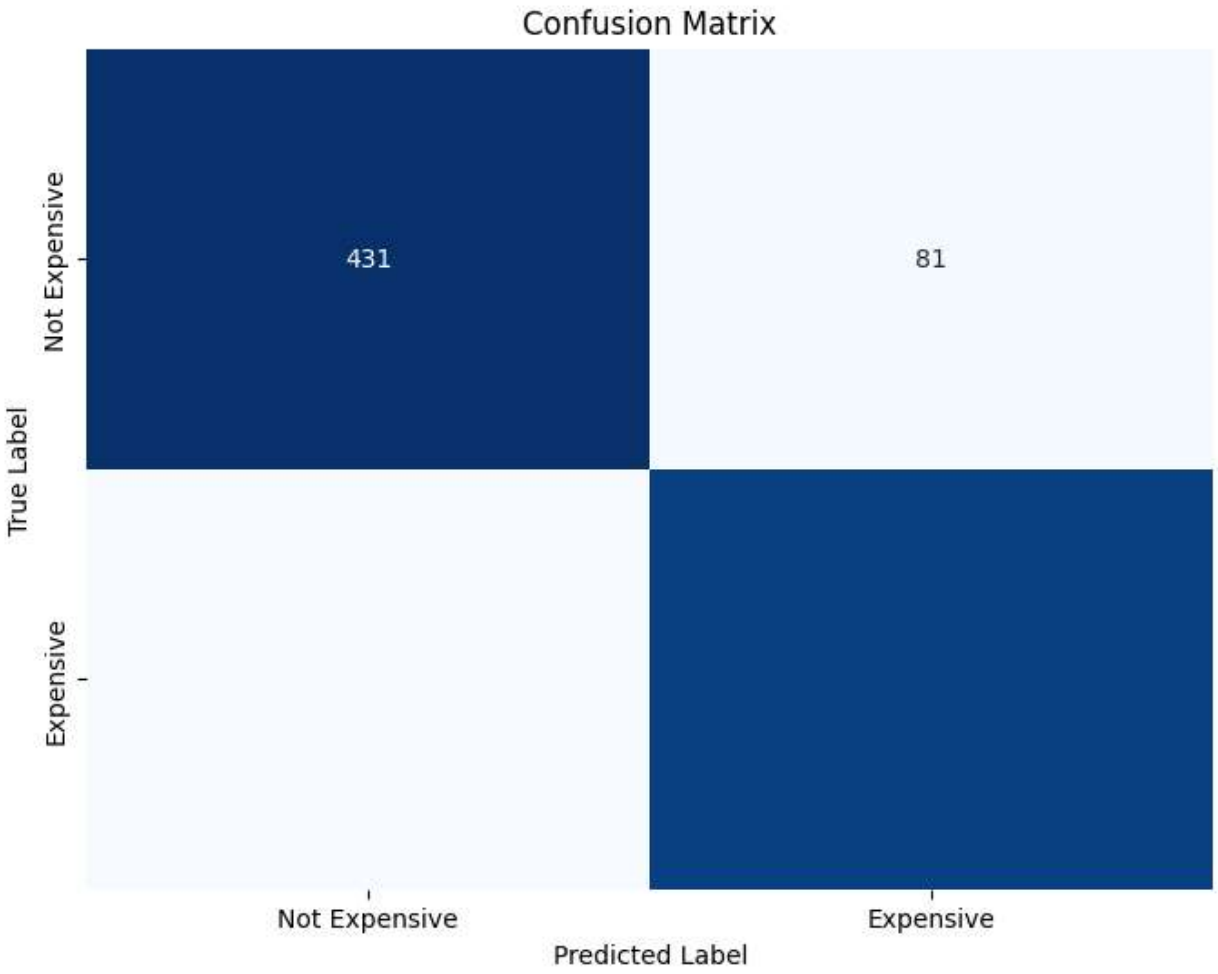
```
Accuracy: 0.84

Confusion Matrix:
[[431  81]
 [ 78 410]]

Classification Report:
              precision    recall  f1-score   support

       False       0.85      0.84      0.84       512
        True       0.84      0.84      0.84       488

    accuracy                           0.84      1000
   macro avg       0.84      0.84      0.84      1000
weighted avg       0.84      0.84      0.84      1000
```


Confusion Matrix
Not Expensive
Expensive
True Label
431
81
Not Expensive
Expensive
Predicted Label

In [ ]: