

In [1]:

```
1 import pandas as pd
2 import numpy as np
3 import matplotlib.pyplot as plt
4 %matplotlib inline
5
6 plt.style.use("seaborn")
```

In [2]:

```
1 csvpath='/home/ubuntu/桌面/Git/Deeplearning/Python课件/5-机器学习/J老师/others/USA_Housing.csv'
2 USAhousing = pd.read_csv(csvpath)
3 USAhousing.head()
```

Out[2]:

	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

In [3]:

```
1 USAhousing.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 7 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   Avg. Area Income                      5000 non-null  float64
1   Avg. Area House Age                   5000 non-null  float64
2   Avg. Area Number of Rooms             5000 non-null  float64
3   Avg. Area Number of Bedrooms          5000 non-null  float64
4   Area Population                       5000 non-null  float64
5   Price                                 5000 non-null  float64
6   Address                              5000 non-null  object
dtypes: float64(6), object(1)
memory usage: 273.6+ KB
```

▼

数据处理

In [4]:

```
1 # 判断哪些列存在数据缺失
2 USAhousing.isna().sum()
```

Out[4]:

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

In [5]:

```
1 # 判断DataFrame中是否有重复的行
2 USAhousing.duplicated().value_counts() # 注意value_counts是函数Series的函数
```

Out[5]:

False	5000
dtype:	int64

In [6]:

```
1 # 仅针对数值型特征做分析
2 USAhousing.describe() # 平均值与中位数相同或相似说明符合正态分布
```

Out[6]:

	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price
count	5000.000000	5000.000000	5000.000000	5000.000000	5000.000000	5.000000e+03
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 [7]:

```
1 # 分析object类型的Address是否可以作为特征
2 USAhousing.describe(include=[object]) # 显然不行，可以提取有用信息后去掉此列
```

Out[7]:

	Address
count	5000
unique	5000
top	208 Michael Ferry Apt. 674\nLaurabury, NE 3701...
freq	1

In [8]:

```
1 def address2district(address):
2     temp = address.split('\n')[1] #地址是以\n来进行分割的，所以[1]部分是我们所需要的
3     if ',' in temp: #其中一部分的地址是以','来进行分割的
4         district = temp.split(',')[1].split()[0]
5     else:
6         district=temp.split(' ')[1]
7     return district
8
9 USAhousing.loc[:, 'district'] = USAhousing['Address'].apply(address2district)
```

▼

可视化分析

In [9]:

```
1 # income与price的关系图 散点图 判断是否存在线性关系
2 plt.scatter(USAhousing['Avg. Area Income'], USAhousing['Price'], edgecolors='black', linewidths=0.45, alpha=0.85)
3 plt.title("Income --- Price")
4 plt.xlabel('Avg. Area Income')
5 plt.ylabel('Price')
6 plt.tight_layout()
7 # 存在线性关系
```



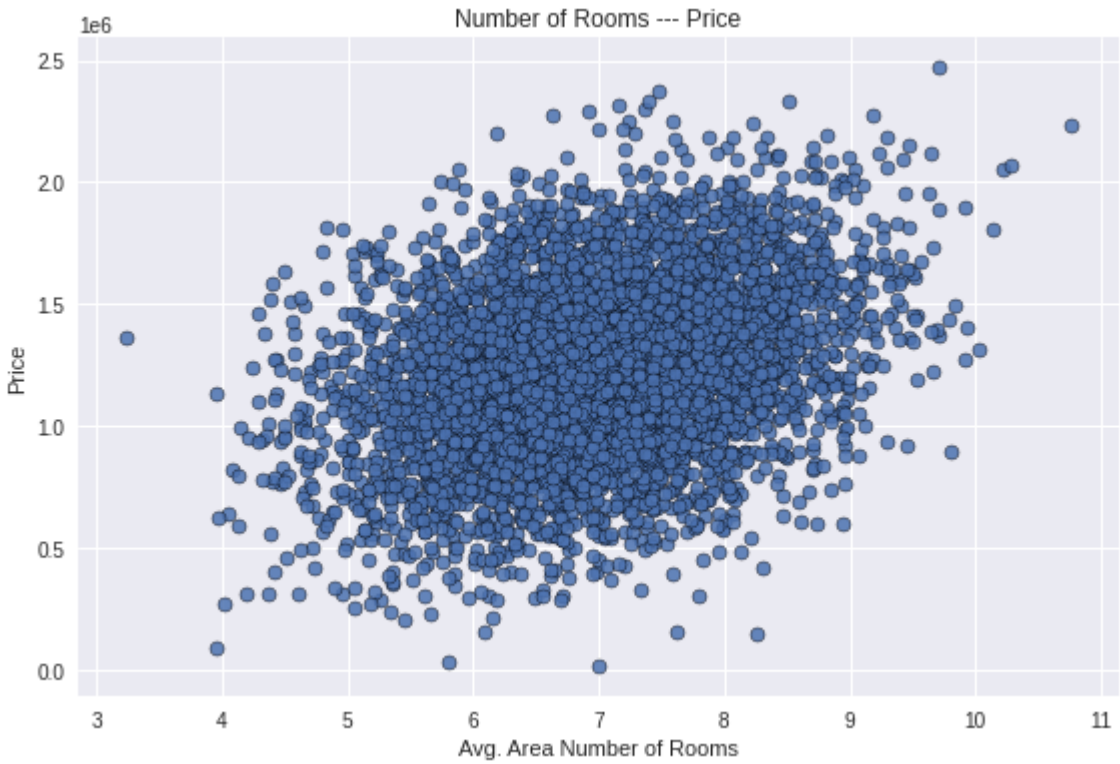
In [10]:

```
1 # house age与price的关系图 散点图 判断是否存在线性关系
2 plt.scatter(USAhousing['Avg. Area House Age'], USAhousing['Price'], edgecolors='black', linewidths=0.45, alpha=0.85)
3 plt.title("House Age --- Price")
4 plt.xlabel('Avg. Area House Age')
5 plt.ylabel('Price')
6 plt.tight_layout()
7 # 存在线性关系
```



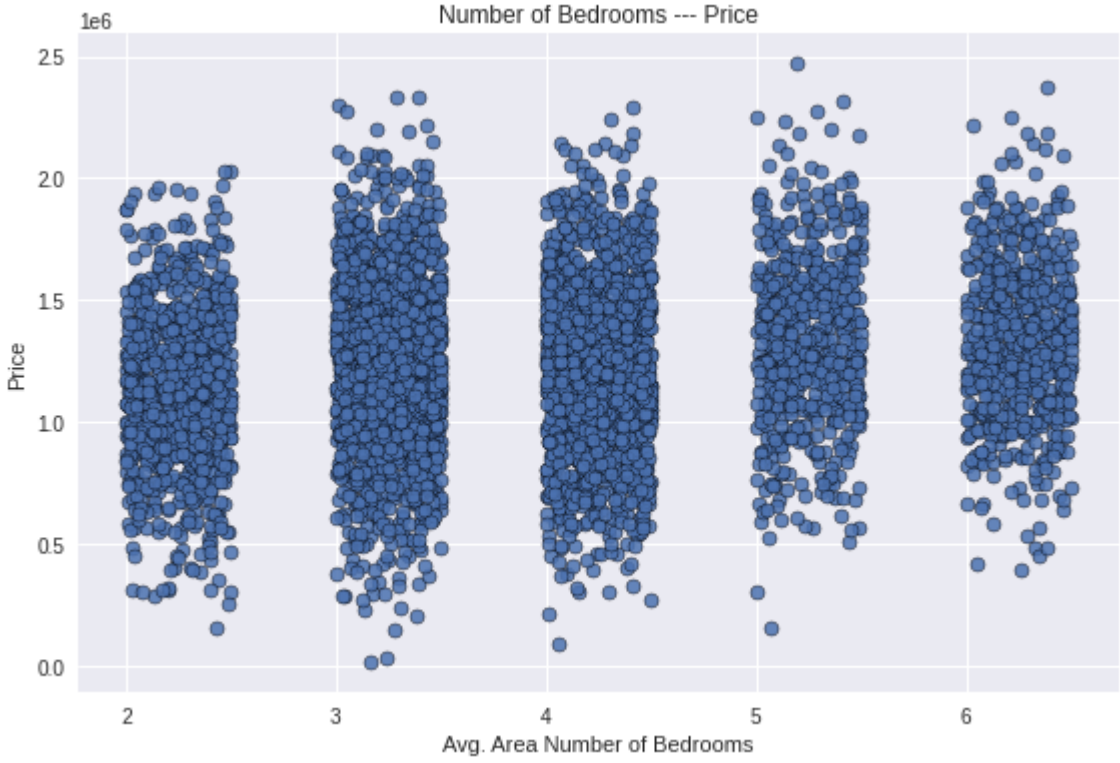
In [11]:

```
1 # number of rooms与price的关系图 散点图 判断是否存在线性关系
2 plt.scatter(USAhousing['Avg. Area Number of Rooms'], USAhousing['Price'], edgecolors='black', linewidths=0.45, alpha=0.85)
3 plt.title("Number of Rooms --- Price")
4 plt.xlabel('Avg. Area Number of Rooms')
5 plt.ylabel('Price')
6 plt.tight_layout()
7 # 存在线性关系
```



In [12]:

```
1 # number of bedrooms与price的关系图 散点图 判断是否存在线性关系
2 plt.scatter(USAhousing['Avg. Area Number of Bedrooms'], USAhousing['Price'], edgecolors='black', linewidths=0.45, alpha=0.85)
3 plt.title("Number of Bedrooms --- Price")
4 plt.xlabel('Avg. Area Number of Bedrooms')
5 plt.ylabel('Price')
6 plt.tight_layout()
7 # 可能存在一定的线性关系
```



In [13]:

```
1 # number of rooms与price的关系图 散点图 判断是否存在线性关系
2 plt.scatter(USAhousing['Area Population'], USAhousing['Price'], edgecolors='black', linewidths=0.45, alpha=0.85)
3 plt.title("Area Population --- Price")
4 plt.xlabel('Area Population')
5 plt.ylabel('Price')
6 plt.tight_layout()
7 # 存在线性关系
```



▼ 建立模型

In [14]:

```
1 # 特征编码
2 from sklearn.preprocessing import LabelEncoder
3
4 le = LabelEncoder()
5
6 le.fit(USAhousing['district'].values)
7 USAhousing['district_code'] = le.transform(USAhousing['district'].values)
8
9 # 特征筛选, 去掉Address特征
10 X = USAhousing[['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of Rooms',
11                 'Avg. Area Number of Bedrooms', 'Area Population']]
12 y = USAhousing['Price']
```

In [15]:

```
1 # 分配训练集和测试集
2 from sklearn.model_selection import train_test_split
3
4 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

In [16]:

▼

```
1 # 特征缩放: 数据集归一化
2 from sklearn.preprocessing import StandardScaler # 数据预处理类
3
4 # 归一化方法: 标准差标准化(zero-mean) 转化函数: x = (x-mean) / std
5 # 经过处理后得到的数据集符合标准正态分布, 即均值为0, 标准差为1
6 # 适用于本身服从正态分布的数据
7
8 scaler = StandardScaler()
9
10 # 下面两行代码是固定用法, 不能颠倒顺序
11 X_train = scaler.fit_transform(X_train) # 求得 训练集 的平均值和方差并应用在 训练集 上, 同时也会保存
12 X_test = scaler.transform(X_test)      # 用保存的 训练集 的平均值和方差来应用在 测试集 上
13
14 # 数据预处理中的方法:
15
16 # - fit():
17 # 解释: 简单来说, 就是求得训练集x的均值啊, 方差啊, 最大值啊, 最小值, 这些训练集x固有的属性。可以理解为一个训练过程
18
19 # - transform():
20 # 解释: 在Fit的基础上, 进行标准化, 降维, 归一化等操作 (看具体用的是哪个工具, 如PCA, StandardScaler等)
21
22 # - fit_transform():
23 # 解释: fit_transform是fit和transform的组合, 既包括了训练又包含了转换
```

In [17]:

▼

```
1 # 模型训练
2 from sklearn.linear_model import LinearRegression
3
4 lin_reg_model = LinearRegression()
5 lin_reg_model.fit(X_train, y_train) #fit生成权重系数coefficient
```

Out[17]:

▼

LinearRegression

LinearRegression()

In [18]:

▼

```
1 # 输出截距
2 print(lin_reg_model.intercept_)
```

1228219.1492415662

In [19]:

▼

```
1 # 特征权重
2 coeff_df = pd.DataFrame(lin_reg_model.coef_, X.columns, columns=['Coefficient'])
3 coeff_df.sort_values(by='Coefficient', ascending=False)
```

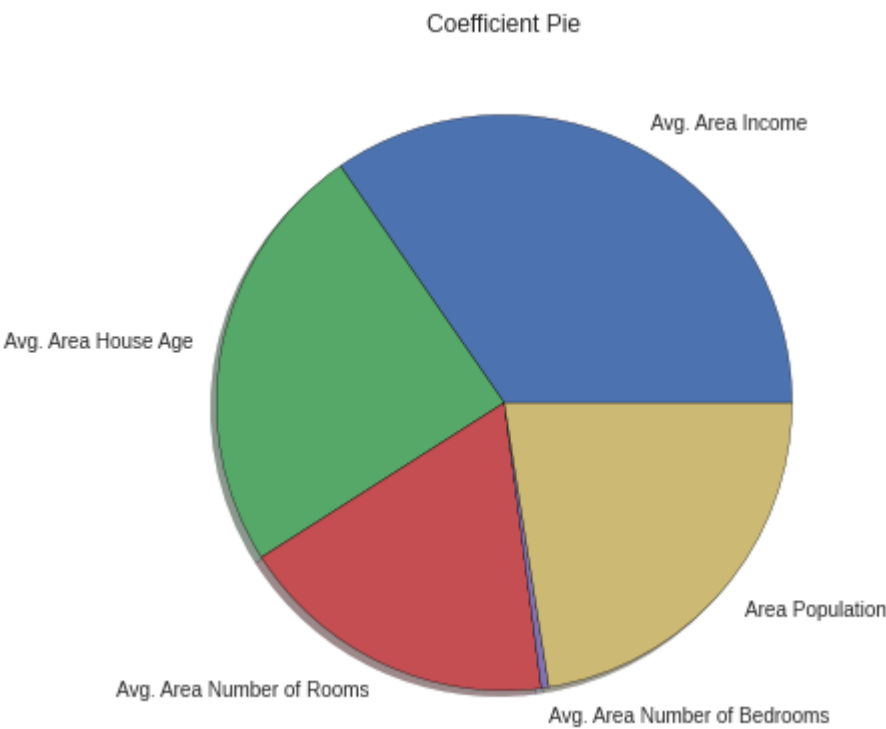
Out[19]:

	Coefficient
Avg. Area Income	232679.724643
Avg. Area House Age	163841.046593
Area Population	151252.342377
Avg. Area Number of Rooms	121110.555478
Avg. Area Number of Bedrooms	2892.815119

In [20]:

▼

```
1 # 特征权重可视化
2 plt.pie(lin_reg_model.coef_, labels=X.columns, shadow=True, wedgeprops={'edgecolor':'black'})
3 plt.title("Coefficient Pie")
4 plt.tight_layout()
```



模型评估

In [21]:

▼

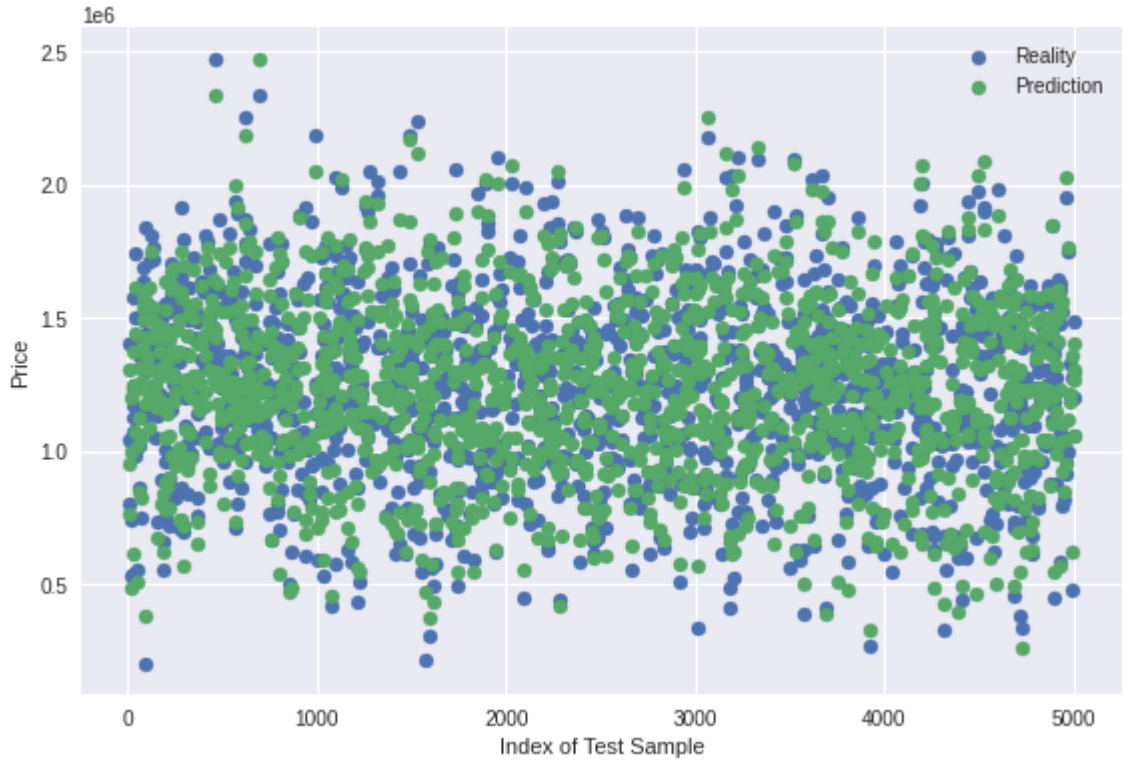
```
1 from sklearn import metrics
2
3 # 模型评估函数
4 def print_evaluate(y_test, y_predict):
5     mse = metrics.mean_squared_error(y_test, y_predict) #MSE
6     mae = metrics.mean_absolute_error(y_test, y_predict) #MAE
7     rmse = np.sqrt(mse) #RMSE
8     r2 = metrics.r2_score(y_test, y_predict) #R2 Square
9     print(f'MSE: {mse}\nRMSE: {rmse}\nMAE: {mae}\nR2: {r2}\n_____ \n')
10
11 # 模型预测 输出评估结果
12 test_pred = lin_reg_model.predict(X_test)
13
14 print("测试集计算结果: \n_____")
15 print_evaluate(y_test, test_pred)
```

测试集计算结果:

MSE:	10068422551.40088
RMSE:	100341.52954485436
MAE:	81135.56609336878
R2:	0.9146818498754016

In [22]:

```
1 # 预测与真实结果可视化
2 plt.scatter(y_test.index, y_test, label='Reality')
3 plt.scatter(y_test.index, test_pred, label='Prediction')
4 plt.legend()
5 plt.xlabel('Index of Test Sample')
6 plt.ylabel('Price')
7 plt.tight_layout()
```



In [23]:

```
1 # # 保存模型
2 # import pickle
3
4 # scalerfile = './scaler.sav'
5 # pickle.dump(scaler, open(scalerfile, 'wb')) # 参数
6 # pickle.dump(lin_reg_model, open('./HousingModel.pkl', 'wb')) # 模型
```

In [24]:

```
1 # 导入模型
2 # import joblib
3
4 # model = joblib.load('path')
5 # prediction = model.predict(X_test)
```