In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

plt.style.use("seaborn")
```

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

In [3]:
```
def address2district(address):
    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
    if ',' in temp: #其中一部分的地址是以‘,’来进行分割的
        district = temp.split(',')[1].split()[0]
    else:
        district=temp.split(' ')[1]
    return district

USAhousing.loc[:,'district'] = USAhousing['Address'].apply(address2district)
```

建立模型

In [4]:
```
# 特征编码
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

le.fit(USAhousing['district'].values)
USAhousing['district_code'] = le.transform(USAhousing['district'].values)

# 特征筛选
X = USAhousing[['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of Rooms',
                'Avg. Area Number of Bedrooms', 'Area Population', 'district_code']]

y = USAhousing['Price']
```

In [5]:
```
# 分配训练集和测试集
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=3)
```

In [6]:
```
# 特征缩放：数据集归一化
from sklearn.preprocessing import StandardScaler # 数据预处理类

# 归一化方法：标准差标准化(zero-mean) 转化函数：x = (x-mean) / std
# 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
# 适用于本身服从正态分布的数据

scaler = StandardScaler()

# 下面两行代码是固定用法，不能颠倒顺序
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
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

# 数据预处理中的方法：

# - fit():
# 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

# - transform():
# 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）

# - fit_transform():
# 解释：fit_transform是fit和transform的组合，既包括了训练又包含了转换
```

In [7]:
```
# 模型训练
from sklearn.linear_model import LassoCV,ElasticNetCV

lscv = LassoCV(alphas=[1.0, 0.1, 0.01, 0.005, 0.0025, 0.001, 0.00025])

lscv.fit(X_train, y_train)

print(lscv.alpha_)

encv = ElasticNetCV(alphas=[0.1, 0.01, 0.005, 0.0025, 0.001],l1_ratio=[0.1, 0.25, 0.5, 0.75, 0.8])

encv.fit(X_train, y_train)

print((encv.alpha_, encv.l1_ratio_))
```

```
1.0
(0.001, 0.8)
```

模型评估

In [8]:
```
from sklearn import metrics

# 模型评估函数
def print_evaluate(y_test, y_predict):
    mse = metrics.mean_squared_error(y_test, y_predict)  #MSE
    mae = metrics.mean_absolute_error(y_test, y_predict) #MAE
    rmse = np.sqrt(mse) #RMSE
    r2 = metrics.r2_score(y_test, y_predict) #R2 Square
    print(f'MSE:  {mse}\nRMSE: {rmse}\nMAE:  {mae}\nR2:   {r2}\n_____________________________\n')

# 模型预测 输出评估结果
test_pred1 = lscv.predict(X_test)
test_pred2 = encv.predict(X_test)
print("Lasso测试集计算结果：\n_____________________________")
print_evaluate(y_test, test_pred1)
print("ElasticNet测试集计算结果：\n_____________________________")
print_evaluate(y_test, test_pred2)
```

```
Lasso测试集计算结果：
_____________________________
MSE:  10776609405.682356
RMSE: 103810.44940506884
MAE:  83684.59354924301
R2:   0.9151206757696965
_____________________________

ElasticNet测试集计算结果：
_____________________________
MSE:  10777410385.78996
RMSE: 103814.30723069899
MAE:  83688.86956334059
R2:   0.9151143670460807
_____________________________
```