**使用简单线性回归根据年龄预测医疗费用**

**1. 实验目的**

* 熟练掌握使用简单线性回归构建回归模型
* 熟练掌握简单线性回归模型的调参过程

**2. 实验内容**

预测个人未来医疗费用支出，用于辅助医疗保险作出收取保费的决策，对保险公司有重要意义。

本实验使用简单线性回归模型，通过年龄字段预测个人未来的医疗费用支出。

**3. 实验数据**

**数据集位置**

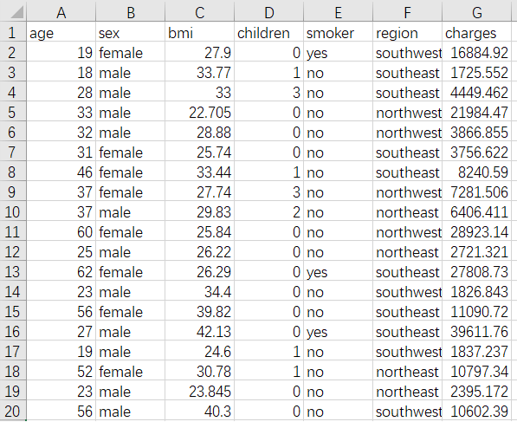
数据集存放在/home/dataset目录下。数据集名字是insurance.csv

**数据集格式**

数据集关于医疗费用，给保险公司用。

数据集包含 1338 个样本数据（行）和 7 个字段（列）。

7个列分别是年龄、性别、bmi、孩子数量、是否抽烟、所在区域、过去医疗费用支出。



**数据字段解释**

* age：数值型；年龄
* sex：字符串类型；性别；有2个值，分别是female（女性）、male（男性）
* bmi：数值型；身体bmi指数
* children：数值型；有几个孩子
* smoker：字符串类型；是否是吸烟者；有2个值，分别是yes（是吸烟者）、no（不是吸烟者）
* region：字符串类型；所在区域；有4个值，分别是southwest（西南部）、southeast（东南部）、northwest（西北部）、northeast（东北部）
* charges：数值型；客户的医疗花费

**4. 实验知识点**

* 简单线性回归模型
* MSE

**5. 实验时长**

1学时

**6. 实验环境**

* Linux Ubuntu 操作系统
* Jupyter 代码编辑器
* Python 3.6.9
* numpy 1.18.5
* pandas 1.1.5
* matplotlib 3.3.3
* scikit-learn 0.24.2

**7. 实验分析**

1. 导入包、导入数据、做数据预处理
2. 构建简单线性回归模型
3. 使用不同的参数，比较简单线性回归模型性能

**8. 实验过程**

**8.1 导入包**

In [1]:

*# -\*- coding: utf-8 -\*-*

*# 导入包*

**import** **numpy** **as** **np**

**import** **pandas** **as** **pd**

**import** **matplotlib.pyplot** **as** **plt**

**8.2 导入数据集**

In [2]:

*# 导入数据集*

data = pd.read\_csv('/home/dataset/insurance.csv')

data.head(5)

Out[2]:

|  | **age** | **sex** | **bmi** | **children** | **smoker** | **region** | **charges** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 19 | female | 27.900 | 0 | yes | southwest | 16884.92400 |
| **1** | 18 | male | 33.770 | 1 | no | southeast | 1725.55230 |
| **2** | 28 | male | 33.000 | 3 | no | southeast | 4449.46200 |
| **3** | 33 | male | 22.705 | 0 | no | northwest | 21984.47061 |
| **4** | 32 | male | 28.880 | 0 | no | northwest | 3866.85520 |

**8.3 数据预处理**

**8.3.1 检测缺失值**

In [3]:

*# 检测缺失值*

null\_df = data.isnull().sum()

null\_df

Out[3]:

age 0

sex 0

bmi 0

children 0

smoker 0

region 0

charges 0

dtype: int64

**8.3.2 筛选数据**

年龄（age）和医疗费用支出（charges）不一定有线性关系。如果他们不具有线性关系，那么就筛选一部分数据（在这些数据中，age和charges有明显的线性关系）。

In [4]:

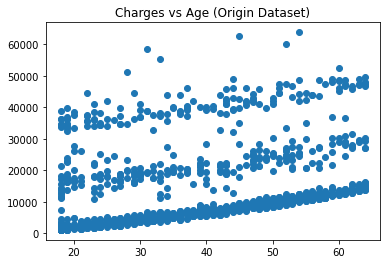
*# 画出age和charges的散点图*

plt.figure()

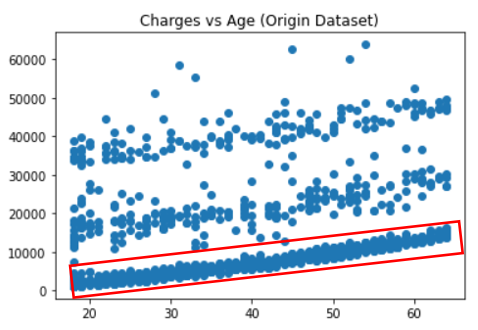
plt.scatter(data['age'], data['charges'])

plt.title('Charges vs Age (Origin Dataset)')

plt.show()



由上图（age和charges的散点图）可知，他们不具有线性关系，但是部分数据有线性关系。于是筛选具有线性关系的数据，如下图所示。



In [5]:

*# 筛选数据*

new\_data\_1 = data.query('age<=40 & charges<=10000') *# 40岁以下 且 10000元以下*

new\_data\_2 = data.query('age>40 & age<=50 & charges<=12500') *# 40岁至50岁之间 且 12500元以下*

new\_data\_3 = data.query('age>50 & charges<=17000') *# 50岁以上 且 17000元以下*

new\_data = pd.concat([new\_data\_1, new\_data\_2, new\_data\_3], axis=0)

In [6]:

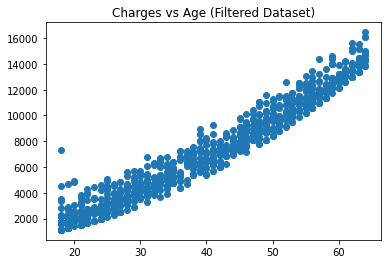
*# 画出age和charges的散点图*

plt.figure()

plt.scatter(new\_data['age'], new\_data['charges'])

plt.title('Charges vs Age (Filtered Dataset)')

plt.show()



由上图可见，现在age和charges具有相关性了。

In [7]:

*# 检查age和charges的相关性*

print('age和charges的相关性是：**\n**', np.corrcoef(new\_data['age'], new\_data['charges']))

age和charges的相关性是：

[[1. 0.97552029]

[0.97552029 1. ]]

age和charges相关系数是0.9755，接近于1，说明相关性非常高。

**8.3.3 得到自变量和因变量**

In [8]:

*# 得到自变量和因变量*

x = new\_data.iloc[:, 0:1].values

y = new\_data['charges'].values

**8.3.4 拆分训练集和测试集**

In [9]:

*# 拆分训练集和测试集*

**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 = 1)

print(x\_train.shape)

print(x\_test.shape)

print(y\_train.shape)

print(y\_test.shape)

(772, 1)

(194, 1)

(772,)

(194,)

**8.4 构建不同参数的简单线性回归模型**

由于不同的参数影响着简单线性回归模型的性能，所以下面使用不同的参数构建3个简单线性回归模型，理解参数对模型性能的影响。

**8.4.1 模型1**

参数 normalize = True, fit\_intercept = True

**8.4.1.1构建简单线性回归模型**

In [10]:

*# 构建不同参数的简单线性回归模型*

*# 模型1：构建线性回归模型（normalize = True, fit\_intercept = True）*

**from** **sklearn.linear\_model** **import** LinearRegression

regressor = LinearRegression(normalize = **True**, fit\_intercept = **True**)

regressor.fit(x\_train, y\_train)

Out[10]:

LinearRegression(normalize=True)

**8.4.1.2 得到模型线性表达式**

In [11]:

*# 得到线性表达式： Charges = a \* Age + b*

print('Charges = **%.2f** \* Age + (**%.2f**)' %(regressor.coef\_, regressor.intercept\_))

Charges = 266.97 \* Age + (-3461.37)

**8.4.1.3 预测测试集**

In [12]:

*# 预测测试集*

y\_pred = regressor.predict(x\_test)

**8.4.1.4 得到模型 MSE**

In [13]:

*# 得到模型的MSE*

**from** **sklearn.metrics** **import** mean\_squared\_error

mse\_score = mean\_squared\_error(y\_test, y\_pred)

print('简单线性回归模型的MSE是：**%.2f**' %(mse\_score)) *# 738,002.02*

简单线性回归模型的MSE是：738002.02

**8.4.1.5 可视化测试集**

In [14]:

*# 可视化测试集*

plt.scatter(x\_test, y\_test, color = 'red')

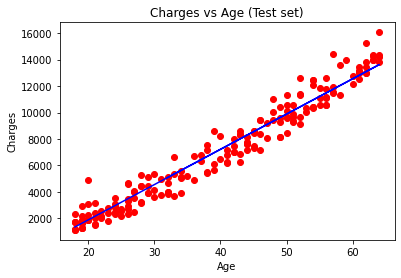
plt.plot(x\_test, y\_pred, color = 'blue')

plt.title('Charges vs Age (Test set)')

plt.xlabel('Age')

plt.ylabel('Charges')

plt.show()



由上图可见，预测的直线拟合样本点拟合的较好

**8.4.2 模型2**

参数 normalize = False, fit\_intercept = True

In [15]:

*# 模型2：构建线性回归模型（normalize = False, fit\_intercept = True）*

regressor = LinearRegression(normalize = **False**, fit\_intercept = **True**)

regressor.fit(x\_train, y\_train)

Out[15]:

LinearRegression()

In [16]:

*# 得到线性表达式： Charges = a \* Age + b*

print('Charges = **%.2f** \* Age + (**%.2f**)' %(regressor.coef\_, regressor.intercept\_))

Charges = 266.97 \* Age + (-3461.37)

In [17]:

*# 预测测试集*

y\_pred = regressor.predict(x\_test)

In [18]:

*# 得到模型的MSE*

mse\_score = mean\_squared\_error(y\_test, y\_pred)

print('简单线性回归模型的MSE是：**%.2f**' %(mse\_score)) *# 738,002.02*

简单线性回归模型的MSE是：738002.02

In [19]:

*# 可视化测试集*

plt.scatter(x\_test, y\_test, color = 'red')

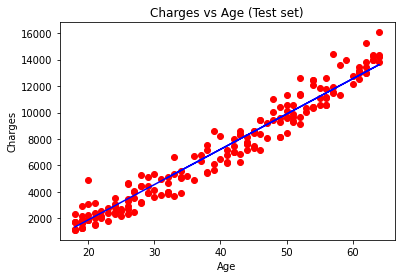
plt.plot(x\_test, y\_pred, color = 'blue')

plt.title('Charges vs Age (Test set)')

plt.xlabel('Age')

plt.ylabel('Charges')

plt.show()



由上图可见，预测的直线拟合样本点拟合的较好

**8.4.3 模型3**

参数 normalize = False, fit\_intercept = False

In [20]:

*# 模型3：构建线性回归模型（normalize = False, fit\_intercept = False）*

regressor = LinearRegression(normalize = **False**, fit\_intercept = **False**)

regressor.fit(x\_train, y\_train)

Out[20]:

LinearRegression(fit\_intercept=False)

In [21]:

*# 得到线性表达式： Charges = a \* Age + b*

print('Charges = **%.2f** \* Age' %(regressor.coef\_))

Charges = 188.91 \* Age

In [22]:

regressor.intercept\_

Out[22]:

0.0

In [23]:

*# 预测测试集*

y\_pred = regressor.predict(x\_test)

In [24]:

*# 得到模型的MSE*

mse\_score = mean\_squared\_error(y\_test, y\_pred)

print('简单线性回归模型的MSE是：**%.2f**' %(mse\_score)) *# 2,404,889.50*

简单线性回归模型的MSE是：2404889.50

In [25]:

*# 可视化测试集预测结果*

plt.scatter(x\_test, y\_test, color = 'red')

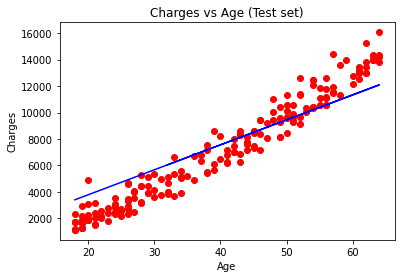
plt.plot(x\_test, y\_pred, color = 'blue')

plt.title('Charges vs Age (Test set)')

plt.xlabel('Age')

plt.ylabel('Charges')

plt.show()



由上图可见，预测的直线拟合样本点拟合的不太好。

对比模型3与模型2的MSE值，模型3的MSE是百万级别，模型2的MSE是十万级别，模型3的MSE较大，也印证了模型3预测的直线不如模型2预测的直线准确。

**9. 实验结果（结论）**[**¶**](javascript:;)

1. 如果自变量和因变量具有线性关系，那么简单线性回归模型可以通过一个自变量来预测因变量；
2. 不同的参数影响着简单线性回归模型的性能。