

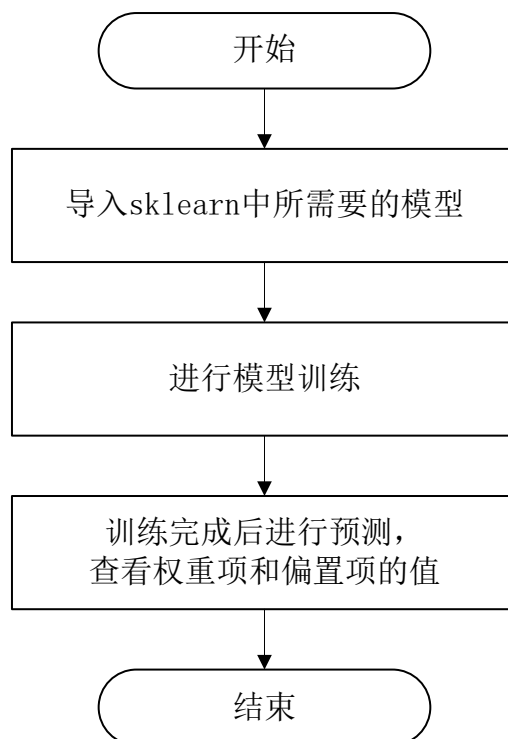
一、实验流程

1. 简单的线性回归模型

基本步骤为：

- ①导入 sklearn 中的线性回归模型
- ②给出一组简单的数据，进行模型训练
- ③训练完成后进行预测，查看权重项和偏置项的值

流程图设计如下：



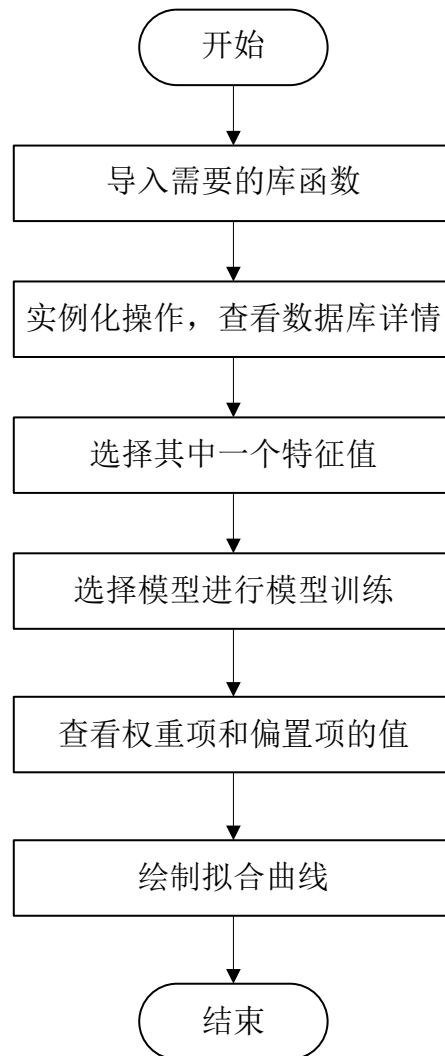
2. 单特征波士顿房价预测

基本步骤为：

- ①导入所需库
- ②提取数据库，并查看详情
- ③只取每栋住宅的房间数作为模型唯一的特征
- ④训练模型

⑤绘制拟合曲线

流程图展示具体的操作步骤为：



3. 多特征波士顿房价预测

基本步骤为

①导入库

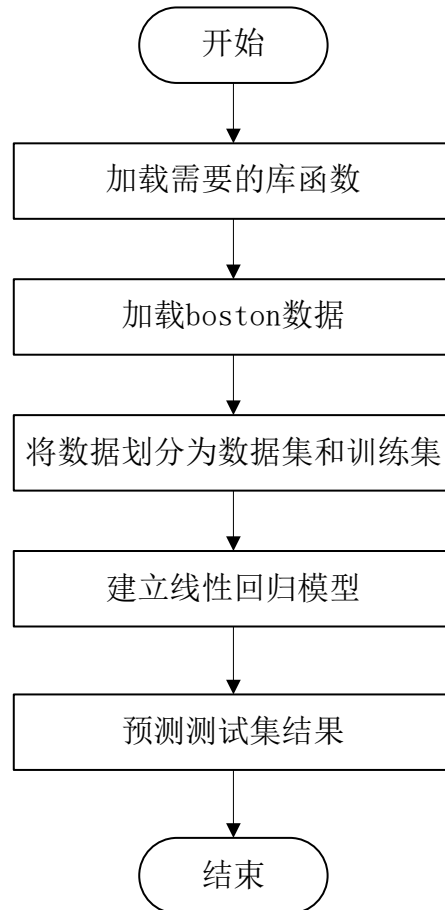
②加载波士顿数据

③将数据划分为训练集和测试集

④建立线性回归模型

⑤在测试集上进行预测，并进行性能度量

流程图设计展示为：



二、实验代码

1. 简单的线性回归模型

#从 sklearn 库中导入所需模型

```
from sklearn.linear_model import LinearRegression
```

```
clf.fit([[3, 3], [4, 4], [5, 5]], [7, 8, 9])
```

模型训练，可以看出特征有两个：x1和x2，训练集的数据有三个

```
pre = clf.predict([[3, 3]]) # 模型预测
```

```
clf.coef_#权重项
```

```
clf.intercept_#偏置项
```

```
print(pre)#打印预测值
```

特征值为 0，1，2 时的情况：



Jupyter 实验一_线性回归_波士顿房价预测 Last Checkpoint: 3 小时前 (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

- 导入sklearn中的线性回归模型
- 给出一组简单的数据，进行模型训练
- 训练完成后进行预测，查看权重项和偏置项的值

```
In [7]: #从sklearn库中导入所需模型
from sklearn.linear_model import LinearRegression

In [8]: clf = LinearRegression()
clf
Out[8]: LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)

In [10]: clf.fit([[0, 0], [1, 1], [2, 2]], [0, 1, 2]) # 模型训练，可以看出，特征有两个：x1和x2，训练集的数据有三个
clf
Out[10]: LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)

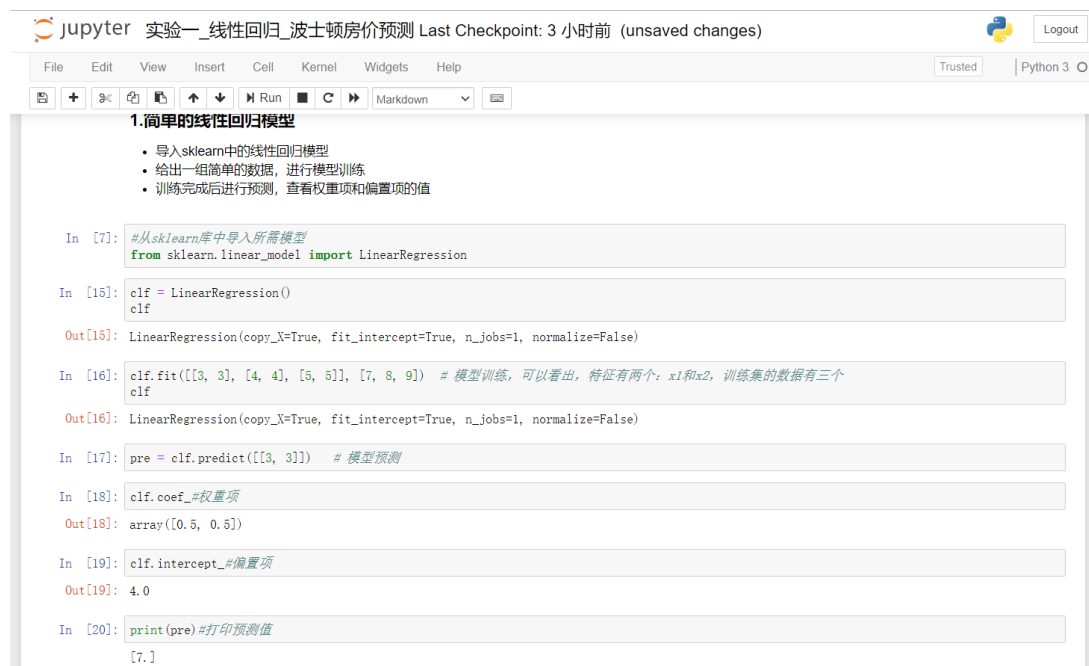
In [11]: pre = clf.predict([[3, 3]]) # 模型预测

In [12]: clf.coef_#权重项
Out[12]: array([0.5, 0.5])

In [13]: clf.intercept_#偏置项
Out[13]: 1.1102230246251565e-16

In [14]: print(pre)#打印预测值
[3.]
```

特征值为 3, 4, 5 的情况:



The image shows a Jupyter Notebook interface with the title "实验一_线性回归_波士顿房价预测 Last Checkpoint: 3 小时前 (unsaved changes)". The notebook contains a single cell with the following code and output:

```
In [7]: #从sklearn库中导入所需模型
from sklearn.linear_model import LinearRegression

In [15]: clf = LinearRegression()
clf

Out[15]: LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)

In [16]: clf.fit([[3, 3], [4, 4], [5, 5]], [7, 8, 9]) # 模型训练, 可以看出, 特征有两个: x1和x2, 训练集的数据有三个
clf

Out[16]: LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)

In [17]: pre = clf.predict([[3, 3]]) # 模型预测

In [18]: clf.coef_ #权重项
Out[18]: array([0.5, 0.5])

In [19]: clf.intercept_ #偏置项
Out[19]: 4.0

In [20]: print(pre) #打印预测值
[7.]
```

2. 单特征波士顿房价预测

`import matplotlib.pyplot as plt` #绘制图像相关

`boston = load_boston()` # 实例化

`clf = LinearRegression()` #选择模型

`print('建立的 LinearRegression 模型为: ', '\n', clf)`

`clf.fit(x, boston.target)` # 模型训练

`y_pre = clf.predict(x)` # 模型输出值

`y_pre`

`y_pre.shape`

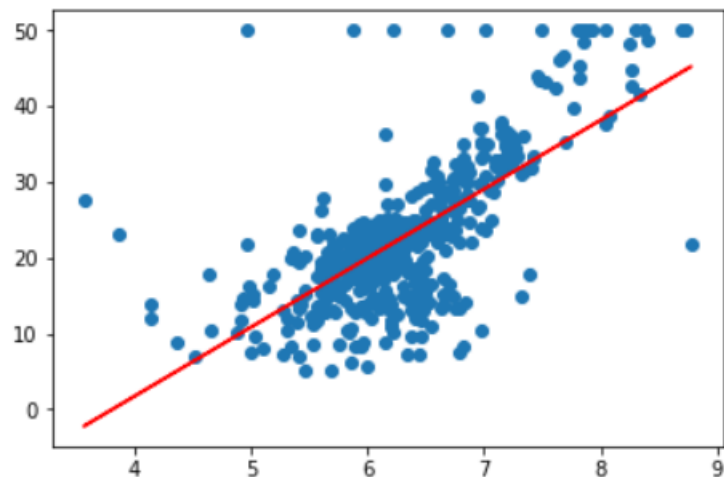
`plt.scatter(x, boston.target)` # 样本实际分布

`plt.plot(x, y_pre, color='red')` # 绘制拟合曲线

`plt.show()`

绘制的图像为

```
In [47]: plt.scatter(x, boston.target) # 样本实际分布
plt.plot(x, y_pre, color='red') # 绘制拟合曲线
plt.show()
```



3. 多特征波士顿房价预测

加载 boston 数据

```
boston = load_boston()
```

```
x = boston['data']
```

```
y = boston['target']
```

```
names = boston['feature_names']
```

将数据划分为训练集测试集

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2,
random_state=125)
```

建立线性回归模型

```
clf = LinearRegression().fit(x_train, y_train)
```

```
print('建立的 LinearRegression 模型为: ', '\n', clf)
```

预测测试集结果

```
y_pred = clf.predict(x_test)
```

```
print('预测前20个结果为: ', '\n', y_pred[:20])
```

输出的重要值如下：

预测前20个结果为:

```
[21.12953164 19.67578799 22.01735047 24.62046819 14.45164813 23.32325459
16.6468677 14.9175848 33.58466804 17.48328609 25.50385719 36.60215179
25.95309333 28.48503161 19.34928078 20.16966217 25.9788081 18.25959831
16.52754056 17.08448854]
```

Boston数据线性回归模型的平均绝对误差为: 3.377642697362791

Boston数据线性回归模型的均方误差为: 31.15059667690467

Boston数据线性回归模型的中值绝对误差为: 1.777421315736218

Boston数据线性回归模型的可解释方差值为: 0.7105949626282937

Boston数据线性回归模型的R方值为: 0.7068954225782444

三、实验问题

Q1.1: 此模型权重项和偏置项的值各为多少?

权重项 `array([0.5, 0.5])`

偏置项 `1.1102230246251565e-16`

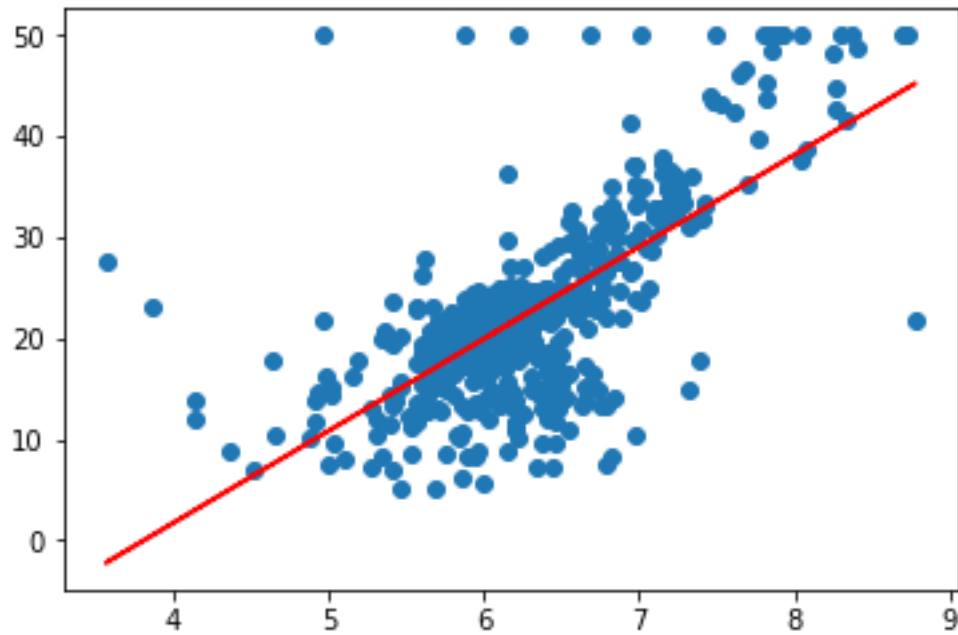
Q1.2: 将训练集数据改为`[[3, 3], [4, 4], [5, 5]]`, `[7, 8, 9]`后, 再进行一次训练, 此时权重项和偏置项的值各为多少? 预测值为多少?

权重项 `array([0.5, 0.5])`

偏置项 `4.0`

预测值 `7`

Q2.1: 附上你绘制的拟合曲线, 在单个特征下模型的拟合效果是好还是不好? 讨论原因。



单特征模型拟合效果不好，因为数据集有 14 个特征值，每一个特征值对房价的影响不同，只考虑一个特征对结果存在极大的偶然性

Q3.1: 预测前 20 个结果是什么？

预测前 20 个结果为：

```
[21.12953164    19.67578799    22.01735047    24.62046819
14.45164813    23.32325459    16.6468677     14.9175848
33.58466804  17.48328609  25.50385719  36.60215179  25.95309333
28.48503161  19.34928078  20.16966217  25.9788081   18.25959831
16.52754056  17.08448854]
```

Q3.2: 结合模型的平均绝对误差、均方误差、中值绝对误差、可解释方差值、R 方值等性能度量参数，讨论多特征下模型的性能。

Boston 数据线性回归模型的平均绝对误差为： 3.377642697362791

Boston 数据线性回归模型的均方误差为： 31.15059667690467

Boston 数据线性回归模型的中值绝对误差为： 1.777421315736218

Boston 数据线性回归模型的可解释方差值为： 0.7105949626282937

Boston 数据线性回归模型的 R 方值为： 0.7068954225782444

分析可得:平均绝对误差, 均方误差和中值绝对误差的值越靠近零, 模型性能越好, 可解释方差值和 R 平方值越靠近 1, 性能越好。由此可知, 该线性回归模型的性能较好, 但还可以进行更大的优化。

四、总结与体会

通过学习波士顿房价预测相关的知识, 很好的为我打开了人工智能和机器学习的大门, 虽然它只是机器学习的第一节入门课, 也是第一个实践课, 但也能让我在通信专业的基础上拓宽思路, 学习更多新颖的知识。同时我也意识到 anaconda 是学习 python 语言的有利工具, 我会慢慢琢磨、利用好这个工具。本次实验也让我意识到, 数学知识的重要性, 一切分析的来源都是基本的数学知识进行拓展, 可以说数学就是一切。在考研路上, 数学是一头拦路虎, 我有信心通过自己的努力, 一步一步克服困难, 坚持走下去, 打破前进路上所有的障碍困难, 实现自己的目标。

立于皓月之边, 不弱星光之势。