Linear regression

1. 算法原理

1）线性回归的模型函数和损失函数

线性回归遇到的问题一般是这样的。我们有 m 个样本，每个样本对应于 n 维特征和一个结果输出，如下：

$$(x(0)_1, x(0)_2, \ldots x(0)_n, y_0), (x(1)_1, x(1)_2, \ldots x(1)_n, y_1), \ldots (x(m)_1, x(m)_2, \ldots x_n^{(m)}, y_n)$$

我们的问题是，对于一个新的 x=y，他所对应的 yx 是多少呢？ 如果这个问题里面的 y 是连续的，则是一个回归问题，否则是一个分类问题。

对于 n 维特征的样本数据，如果我们决定使用线性回归，那么对应的模型是这样的：$h_\theta(x_1,x_2,\ldots x_n)=\theta_0+\theta_1 x_1+\ldots+\theta_n x_n$，其中 $\theta_i(i=0,1,2\ldots.n)$ 为模型参数，$x_i(i=0,1,2\ldots.n)$为每个样本的 n 个特征值。这个表示可以简化，我们增加一个特征 $x_0=1$，这样 $h_\theta(x_0,x_1,\ldots x_n)=\sum_{i=0}^{n}\theta_i x_i$。

进一步用矩阵形式表达更加简洁如下：$h_\theta(X)=X\theta$ 其中， 假设函数 $h_\theta(X)$为 mx1 的向量, θ 为 nx1 的向量，里面有 n 个代数法的模型参数。X 为 mxn 维的矩阵。m 代表样本的个数，n 代表样本的特征数。

得到了模型，我们需要求出需要的损失函数，一般线性回归我们用均方误差作为损失函数。损失函数的代数法表示如下：

$$J(\theta_0, \theta_1 \ldots., \theta_n) = \sum_{i=0} m(h_\theta(x_0, x_1, \ldots x_n) - y_i)2$$

进一步用矩阵形式表达损失函数：$J(\theta)=12(X\theta-Y)^T(X\theta-Y)$由于矩阵法表达比较的简洁，后面我们将统一采用矩阵方式表达模型函数和损失函数。

2）线性回归算法

对于线性回归的损失函数$J(\theta) = \frac{1}{2}(X\theta - Y)^T(X\theta - Y)$，我们常用的有两种方法来求损失函数最小化时候的 θ 参数：一种是梯度下降法，一种是最小二乘法。

如果采用梯度下降法，则 θ 的迭代公式是这样的：

$$\theta = \theta - \alpha X^T(X\theta - Y)$$

通过若干次迭代后，我们可以得到最终的 θ 的结果

如果采用最小二乘法，则 θ 的结果公式如下：

$$\theta = (XTX)-1X^TY$$

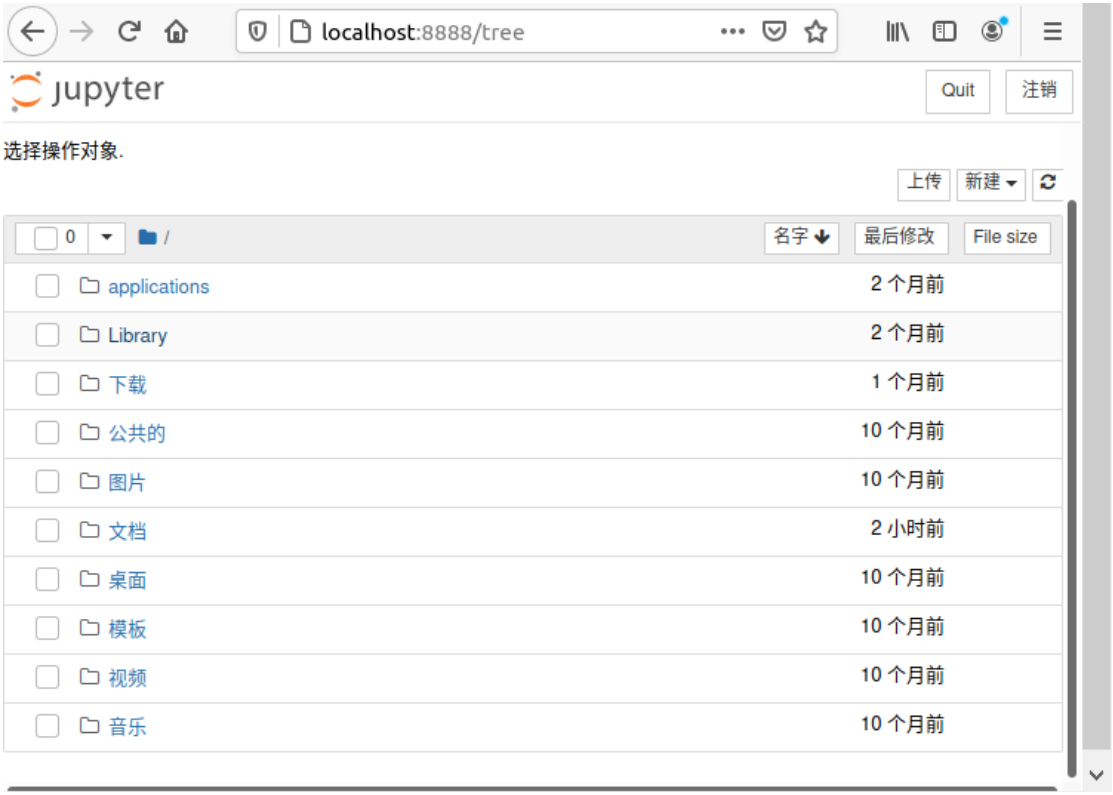
当然线性回归，还有其他的常用算法，比如牛顿法和拟牛顿法，这里不详细描述。

2. 实验环境

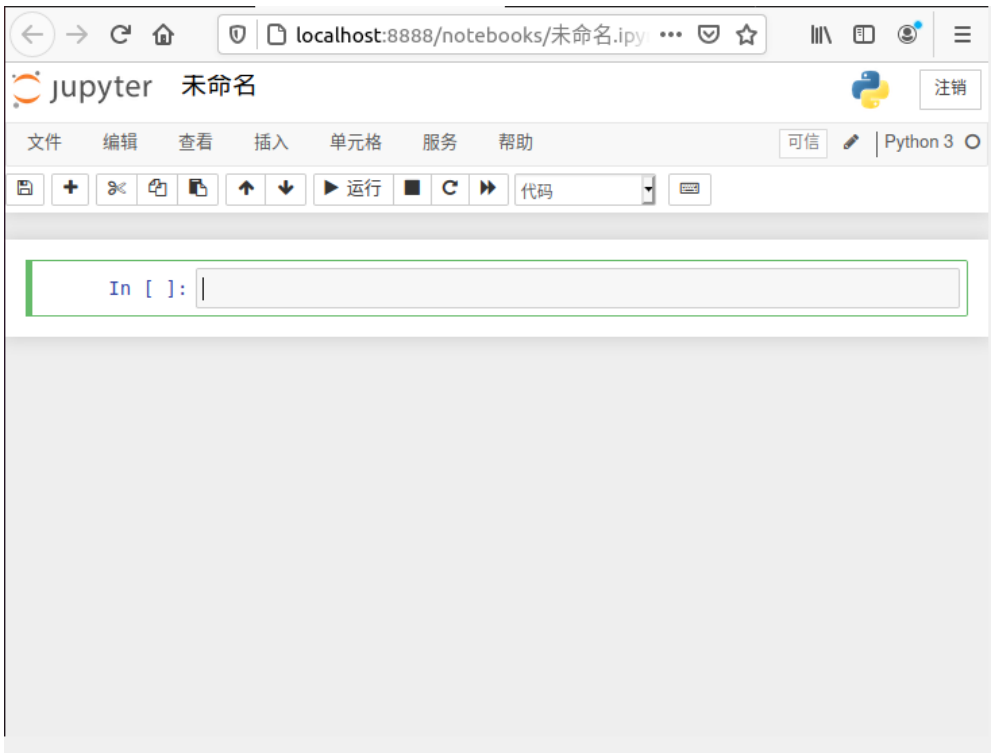
Ubuntu 20.04
Python 3.6
Jupyter notebook

3. 实验步骤

1)打开终端，然后输入 jupyter notebook，出现如下界面



2)选定特定文件夹，新建 ipynb 文件，在未命名出可重命名文件



4. 实操

Step 1：预处理数据

1. 导入库
2. 导入数据集
3. 检查缺失数据
4. 分割数据集

```
#导入库
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

#导入数据集
dataset = pd.read_csv('studentscores.csv')

#查看数据，检查有没有缺失数据
X = dataset.iloc[ : ,   : 1 ].values
Y = dataset.iloc[ : , 1 ].values


#分割数据集
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split( X, Y, test_size =
1/4, random_state = 0)
```

Step 2:把数据集拟合到简单线性回归模型中

```
from sklearn.linear_model import LinearRegression
regressor = LinearRegression() ## 创建一个 regressor 对象
regressor = regressor.fit(X_train, Y_train) ### 对象拟合到数据集里面
```

Step 3:预测结果

- 在训练好的 regressor 中使用预测模型
- 结果输出到向量 Y_pred 中

```
Y_pred = regressor.predict(X_test)
```

Step 4：可视化

```
    plt.scatter(X_train , Y_train, color = 'red')
    plt.plot(X_train , regressor.predict(X_train), color ='blue')   ## 训练结果
    plt.scatter(X_test , Y_test, color = 'green')
    plt.plot(X_test , regressor.predict(X_test), color ='blue')   ## 测试结果
```