python 简介

python 是非常好用的一门编程语言，简单易用轻量级，扩展性强，有着强大的第三方生态。Python环境的配置建议：

python 3.x —--> Anaconda ——-> pycharm/VS code

1. 数据处理

1.1 数据分析三连

在使用Python进行数据分析时，通常使用的第三方库包括numpy、pandas、matplotlib。

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

其中，numpy主要被用来做数组（向量、矩阵等）运算，与之对应的数据类型为array、ndarray；

pandas主要被用做表格数据读入与后续处理，与之对应的数据类型为dataframe；

matplotlib主要被用来做初级画图。

由于每个库都比较强大，可用功能较多，下面仅做简要讲解，实战中再做展开。

pandas

- 读入数据：

  ```
  df = pd.read_csv('123.csv')
  df = pd.read_excel('123.xlsx')
  ```

- 去重：

  ```
  df = df.drop_duplicates()
  ```

  去除空值：

  ```
  df = df.dropna()
  ```

numpy

- 将上述表格转化为数组：

  ```
  data = np.array(df)
  ```

- 构建1矩阵，0矩阵：

```
ones = np.ones(4, 4)
zeros = np.zeros(5)
```

- 计算向量的范数：

```
vec = np.array([1, 2, 3])

l1 = np.linalg.norm(vec, ord=1) # 1范数
l2 = np.linalg.norm(vec, odf=2) # 2范数
```

- 做简要的矩阵运算：

```
from numpy.linalg import inv

B = inv(X.transpose().dot(X)).dot(X.transpose()).dot(Y)
```

在numpy里，将两个数组用 * 来进行乘法，得到的结果是对应相乘。

matplotlib

- 画连线图：

```
x = [1, 2, 3, 4, 5]
y = [(i + 1)**2 for i in range(5)]

plt.plot(x, y)
plt.show()
```

- 画散点图：

```
x = [1, 2, 3, 4, 5]
y = [2, 3, 4, 5, 6]

plt.scatter(x, y)
plt.show()
```

- 对于一些更好看的图表，可以使用seaborn库。

其余数据预处理

剩下的数据预处理，包括数据归一化，划分训练集与测试集，划分交叉验证等均采用scikit-learn库实现。sklearn是非常强大的机器学习库，其中包含有非常多的回归、分类等算法实现。

- 数据标准化：

```
from sklearn.preprocessing import scale, StandardScaler

# 减均值除方差
x = scale(x)
# 或者
scaler = StandardScaler()
x = scaler.fit_transform(x)
```

- 将特征缩放至0-1：

```
from sklearn.preprocessing import MinMaxScaler

x = MinMaxScaler.fit_transform(x)
```

- 划分测试集与训练集：

```
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.3, random_state=13)
```

- 交叉验证集的划分略微复杂，这里不讲。

模型建立

模型的建立同样是采用sklearn。下面看一段完整的代码。

```
#-*- coding = UTF-8 -*-

import numpy as np
from numpy.linalg import inv
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression

x = [0.03, 0.04, 0.05, 0.07, 0.09, 0.1, 0.12, 0.15, 0.17,   0.2]
y = [40.5, 39.5, 41, 41.5, 43, 42, 45, 47.5, 53, 56]

'''首先采用推导式求系数做线性回归：'''

X = np.ones((10, 2))
Y = np.ones((10, 1))
for i in range(10):
    X[i, 1] = x[i]
    Y[i, 0] = y[i]
B = inv(X.transpose().dot(X)).dot(X.transpose()).dot(Y)
print('B = {}\n'.format(B)) # 回归参数组成的向量


'''下面采用sklearn方法做线性回归：'''

f = lambda x: round(x, 8)

X = np.ones((10, 1))
for i in range(10):
    X[i, 0] = x[i]

model = LinearRegression()
model.fit(X, Y)
theta_0 = float(model.intercept_)
theta_1 = float(model.coef_)
print(theta_0, '\n', theta_1)
# 由于浮点数精度的原因，两个小数不能直接做比较，
# 将不同方法计算得到的参数取8位小数后再做比较
```

```
print(f(theta_0) == f(B[0, 0]))
print(f(theta_1) == f(B[1, 0]))


plt.plot([0.02, 0.22], B[0, 0] + B[1, 0] * np.array([0.02, 0.22]),
color='red')
plt.scatter(x, y)
plt.show()
```

对于LASSO回归与岭回归，我们有：

```
from sklearn.linear_model import Lasso, Ridge
```

更详细的介绍可以看官网。

Lasso: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Lasso.html#sklearn.linear_model.Lasso

Ridge: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Ridge.html#sklearn.linear_model.Ridge

其具体的使用方法同样较为简单。另外，sklearn还做为这两个模型分别做了交叉验证类，即有：

```
from sklearn.linear_model import LassoCV, RidgeCV
```

如果想要使用交叉验证，仅需指定对应参数即可。并且，在模型评估中同样有着cross_val_score，其具体的网站为：https://scikit-learn.org/stable/modules/model_evaluation.html#scoring-parameter

```
from sklearn import datasets    #自带数据集
from sklearn.model_selection import train_test_split,cross_val_score    #划分数据 交叉验证
from sklearn.neighbors import KNeighborsClassifier  #一个简单的模型，只有K一个参数，类似K-means
import matplotlib.pyplot as plt
iris = datasets.load_iris()     #加载sklearn自带的数据集
X = iris.data           #这是数据
y = iris.target         #这是每个数据所对应的标签
train_X,test_X,train_y,test_y = train_test_split(X,y,test_size=1/3,random_state=3)  #这里划分数据以1/3的来划分训练集训练结果 测试集测试结果
k_range = range(1,31)
cv_scores = []      #用来放每个模型的结果值
for n in k_range:
    knn = KNeighborsClassifier(n)   #knn模型，这里一个超参数可以做预测，当多个超参数时需要使用另一种方法GridSearchCV
    scores = cross_val_score(knn,train_X,train_y,cv=10,scoring='accuracy')
 #cv：选择每次测试折数  accuracy：评价指标是准确度,可以省略使用默认值，具体使用参考下面。
    cv_scores.append(scores.mean())
plt.plot(k_range,cv_scores)
plt.xlabel('K')
plt.ylabel('Accuracy')      #通过图像选择最好的参数
plt.show()
best_knn = KNeighborsClassifier(n_neighbors=3)  # 选择最优的K=3传入模型
best_knn.fit(train_X,train_y)           #训练模型
```

```
print(best_knn.score(test_X,test_y))    #看看评分
```