1. **如何使用scikit包训练一个简单的线性回归模型？**

1. 读取数据和预处理函数  
**import** pandas **as** pd  
  
*# 从excel文件中读取数据***def** loaddata(datafile):  
 *# 读取自变量和因变量数据 index:温度数据0-12 DYB数据13-38 挠度数据39-40* df = pd.read\_excel(datafile, header=**None**)  
 data = df.values  
 data = np.delete(data, 0, 0)  
 data = np.delete(data, [0, 1], 1)  
 data = data.astype(np.float)  
 *# 温度数据* X1 = data[:, 0:13]  
 *# 动应变数据* t1 = []  
 t2 = []  
 **for** i **in** range(26):  
 **if** i % 2 == 0:  
 t1.append(13 + i)  
 **else**:  
 t2.append(13 + i)  
 X21 = data[:, t1]  
 X22 = data[:, t2]  
 *# 挠度数据* Y11 = data[:, 39:40]  
 Y12 = data[:, 40:41]  
 **return** X1, X21, X22, Y11, Y12  
  
*# 根据数据集Data.xlsx***def** main():  
 datafile = **"./data/Data.xlsx"** X1, X21, X22, Y11, Y12 = loaddata(datafile)  
 *# 完整正值数据* X11 = np.concatenate([X1, X21], axis=1)  
 *# 完整负值数据* X12 = np.concatenate([X1, X22], axis=1)  
 **return** X11, X12, Y11, Y12  
  
2. 准备算法的输入数据，划分训练集和测试集  
**if** \_\_name\_\_ == **"\_\_main\_\_"**:  
 X11, X12, Y11, Y12 = main()  
 X\_data\_1 = X11 *# (677,26)* Y\_data\_1 = Y11 *# (677,1)* X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_data\_1, Y\_data\_1, test\_size=0.3, random\_state=2018)  
 print(np.array(X\_train).shape) *# (473,26)* print(np.array(X\_test).shape) *# (204,26)*3. 训练scikit-learn的线性模型  
**from** sklearn.linear\_model **import** LinearRegression  
  
*# 下面为sklearn线性回归过程* linreg = LinearRegression()  
 linreg.fit(X\_train, y\_train)  
*# 查看模型的系数结果* print(linreg.intercept\_)  
 print(linreg.coef\_)  
  
4. 模型预测和评价  
 *# 预测* y\_pred = linreg.predict(X\_test)  
  
 *# 用scikit-learn计算MSE* print(**"一次交叉验证之后结果---------------------"**)  
 print(**"MSE:"**,metrics.mean\_squared\_error(y\_test, y\_pred))  
 *# 用scikit-learn计算RMSE* print(**"RMSE:"**, np.sqrt(metrics.mean\_squared\_error(y\_test, y\_pred)))  
 print(**"R2\_score:"**, metrics.r2\_score(y\_test, y\_pred))  
  
 *# 我们可以通过交叉验证来持续优化模型，代码如下，我们采用10折交叉验证，即cross\_val\_predict中的cv参数为10：* predicted = cross\_val\_predict(linreg, X\_data\_1, Y\_data\_1, cv=10)  
 *# 用scikit-learn计算MSE* print(**"十折交叉验证之后结果---------------------"**)  
 print(**"MSE:"**,metrics.mean\_squared\_error(Y\_data\_1, predicted))  
 *# 用scikit-learn计算RMSE* print(**"RMSE:"**, np.sqrt(metrics.mean\_squared\_error(Y\_data\_1, predicted)))  
 print(**"explained\_variance\_score:"**, metrics.explained\_variance\_score(Y\_data\_1, predicted))  
 print(**"mean\_absolute\_error:"**, metrics.mean\_absolute\_error(Y\_data\_1, predicted))  
 print(**"median\_absolute\_error:"**, metrics.median\_absolute\_error(Y\_data\_1, predicted))  
 print(**"R2\_score:"**, metrics.r2\_score(Y\_data\_1, predicted))  
  
自己实现的验证：  
  
**''' 均方误差根 '''  
def** rmse(y\_test, y\_true):  
 **return** sp.sqrt(sp.mean((y\_test - y\_true) \*\* 2))  
  
  
**''' 与均值相比的优秀程度，介于[0~1]。0表示不如均值。1表示完美预测.这个版本的实现是参考scikit-learn官网文档 '''  
def** R2(y\_test, y\_true):  
 **return** 1 - ((y\_test - y\_true) \*\* 2).sum() / ((y\_true - np.mean(y\_true)) \*\* 2).sum()  
  
  
**''' 这是Conway&White《机器学习使用案例解析》里的版本 '''  
def** R22(y\_test, y\_true):  
 y\_mean = np.array(y\_true)  
 y\_mean[:] = y\_mean.mean()  
 **return** 1 - rmse(y\_test, y\_true) / rmse(y\_mean, y\_true)  
  
 *# print("RMSE\_1:",rmse(y\_pred , y\_test))  
 # print("R2\_score\_1:",R2(y\_pred , y\_test))  
 # print("R22\_score\_1:",R22(y\_pred , y\_test))*

**2.列举几个常用的Python数据分析包及其作用？**

NumPy系统是Python的一种开源的数值计算扩展。这种工具可用来存储和处理大型矩阵，比Python自身的嵌套列表（nested list structure)结构要高效的多（该结构也可以用来表示矩阵（matrix））。NumPy（Numeric Python）提供了许多高级的数值编程工具，如：矩阵数据类型、矢量处理，以及精密的运算库。专为进行严格的数字处理而产生。多为很多大型金融公司使用，以及核心的科学计算组织如：Lawrence Livermore，NASA用其处理一些本来使用C++，Fortran或[Matlab](https://baike.baidu.com/item/Matlab" \t "_blank)等所做的任务。

SciPy是一款方便、易于使用、专为科学和工程设计的Python工具包.它包括统计,优化,整合,线性代数模块,傅里叶变换,信号和图像处理,常微分方程求解器等等.

Pandas [1]  是python的一个数据分析包，最初由AQR Capital Management于2008年4月开发，并于2009年底开源出来，目前由专注于Python数据包开发的PyData开发team继续开发和维护，属于PyData项目的一部分。Pandas最初被作为金融数据分析工具而开发出来，因此，pandas为时间序列分析提供了很好的支持。 Pandas的名称来自于面板数据（panel data）和python数据分析（data analysis）。panel data是经济学中关于多维数据集的一个术语，在Pandas中也提供了panel的数据类型。

**3.如何使用numpy对数列的前n项进行排序？**

用ndarray.sort内建函数排序

**4.如何检验一个数据集或者时间序列是随机分布的？**

画lag plot（Correlogram：相关图），如果图上的点呈散乱分布，则为随机

**5.在Python中如何创建包含不同类型数据的dataframe**

利用pandas包的DataFrame函数的serias创建列然后用dtype定义类型

**6.简单描述numpy array对比Python list的优势**

a. numpy array比python list更紧凑，存储数据占的空间小，读写速度快。(这是由于python list储存的是指向对象（至少需要16个字节）的指针（至少4个字节）；而array中储存的是单一变量（比如单精度浮点数为4个字节，双精度为8）)

b. array可以直接使用vector和matrix类型的处理函数，非常方便。

**7.创建一个长度为10的一维全为0的ndarray对象，然后让第5个元素为1**

Import numpy as np

S = Np.zeros(shape = 10)

S[4] = 1

**8.使用np.random.random创建一个10\*10的ndarray对象，并打印出最大最小元素**

Import numpy as np

S = Np.random.random(size = (10,10))

Smax = s.max()

**9.创建一个10\*10的ndarray对象，且矩阵边界全为1，里面全为0**

Import numpy as np

S = Np.zeros(shape=(10,10),dtype = np.int8)

S[[0,9]] = 1

S[:,[0,9]] = 1

**10.创建一个范围在(0,1)之间的长度为12的等差数列**

Import numpy as np

Np.linspace(0,1,12)

**11.给定一个二维矩阵，如何交换任意两行的元素？**

Import numpy as np

S = np.random.randint(0,100,size = (3,3))

S[[1,0,2]]

**12.正则化一个5\*5的随机矩阵**

正则的概念：假设a是矩阵中的一个元素，max/min分别是矩阵元素的最大最小值，则正则化后a = (a - min)/(max - min)