北京郵電大學

Beijing University of Posts and Telecommunications

数据科学

题目：第一次实验

学院：信息与通信工程学院

班级：2019211127

姓名：王兴睿

学号：2018211756

目录

1. 每个任务附上对应关键代码段并将运行结果截图，导出为 pdf 后发送到助教邮箱（必做）

① numpy 创建数组，数组形状修改结果截图

创建数组

```
In [1]:
import numpy as np
```

```
In [3]:
x = np.arange(15)
print(x)
```

```
[ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14]
```

数组形状修改

```
In [5]:
print("原始数组：",x)
print("修改后的数组：",x.reshape(3,5))
```

```
原始数组： [ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14]
修改后的数组： [[ 0  1  2  3  4]
 [ 5  6  7  8  9]
 [10 11 12 13 14]]
```

② 输出糖尿病数据集所有变量值及其数组形状

输出糖尿病数据集所有变量值

```
In [7]:
import matplotlib as plt
from sklearn.datasets import load_diabetes
```

```
In [11]:
diabetes = load_diabetes()
print(diabetes.data)
```

```
[[ 0.03807591  0.05068012  0.06169621 ... -0.00259226  0.01990842
  -0.01764613]
 [-0.00188202 -0.04464164 -0.05147406 ... -0.03949338 -0.06832974
  -0.09220405]
 [ 0.08529891  0.05068012  0.04445121 ... -0.00259226  0.00286377
  -0.02593034]
 ...
 [ 0.04170844  0.05068012 -0.01590626 ... -0.01107952 -0.04687948
   0.01549073]
 [-0.04547248 -0.04464164  0.03906215 ...  0.02655962  0.04452837
  -0.02593034]
 [-0.04547248 -0.04464164 -0.0730303  ... -0.03949338 -0.00421986
   0.00306441]]
```

输出数组形状

```
In [14]:
print(diabetes.data.shape)
print(diabetes.target.shape)
```

```
(442, 10)
(442,)
```

③ 输出糖尿病数据所有样本真实标签及其数组形状

```
In [10]:
print(diabetes.target)
```

```
[151.  75. 141. 206. 135.  97. 138.  63. 110. 310. 101.  69. 179. 185.
 118. 171. 166. 144.  97. 168.  68.  49.  68. 245. 184. 202. 137.  85.
 131. 283. 129.  59. 341.  87.  65. 102. 265. 276. 252.  90. 100.  55.
  61.  92. 259.  53. 190. 142.  75. 142. 155. 225.  59. 104. 182. 128.
  52.  37. 170. 170.  61. 144.  52. 128.  71. 163. 150.  97. 160. 178.
  48. 270. 202. 111.  85.  42. 170. 200. 252. 113. 143.  51.  52. 210.
  65. 141.  55. 134.  42. 111.  98. 164.  48.  96.  90. 162. 150. 279.
  92.  83. 128. 102. 302. 198.  95.  53. 134. 144. 232.  81. 104.  59.
 246. 297. 258. 229. 275. 281. 179. 200. 200. 173. 180.  84. 121. 161.
  99. 109. 115. 268. 274. 158. 107.  83. 103. 272.  85. 280. 336. 281.
 118. 317. 235.  60. 174. 259. 178. 128.  96. 126. 288.  88. 292.  71.
 197. 186.  25.  84.  96. 195.  53. 217. 172. 131. 214.  59.  70. 220.
 268. 152.  47.  74. 295. 101. 151. 127. 237. 225.  81. 151. 107.  64.
 138. 185. 265. 101. 137. 143. 141.  79. 292. 178.  91. 116.  86. 122.
  72. 129. 142.  90. 158.  39. 196. 222. 277.  99. 196. 202. 155.  77.
 191.  70.  73.  49.  65. 263. 248. 296. 214. 185.  78.  93. 252. 150.
  77. 208.  77. 108. 160.  53. 220. 154. 259.  90. 246. 124.  67.  72.
 257. 262. 275. 177.  71.  47. 187. 125.  78.  51. 258. 215. 303. 243.
  91. 150. 310. 153. 346.  63.  89.  50.  39. 103. 308. 116. 145.  74.
  45. 115. 264.  87. 202. 127. 182. 241.  66.  94. 283.  64. 102. 200.
 265.  94. 230. 181. 156. 233.  60. 219.  80.  68. 332. 248.  84. 200.
  55.  85.  89.  31. 129.  83. 275.  65. 198. 236. 253. 124.  44. 172.
 114. 142. 109. 180. 144. 163. 147.  97. 220. 190. 109. 191. 122. 230.
```

输出数组形状

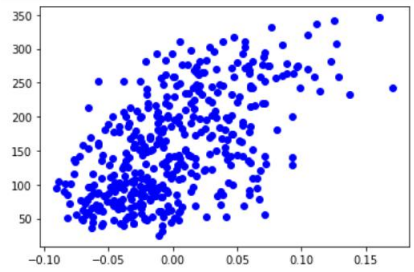
```
In [14]:
print(diabetes.data.shape)
print(diabetes.target.shape)
```

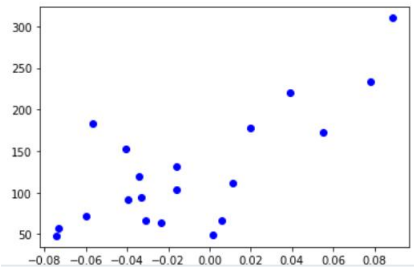
```
(442, 10)
(442,)
```

④ 输出测试数据散点图（学号尾号为基数散点图为红色方形，学号尾号为偶数散点图为蓝色圆形）

输出测试数据散点图

n [8]:
```
diabetes_X = diabetes.data[:,2]
diabetes_X_train = diabetes_X[:-20]
diabetes_X_test = diabetes_X[-20:]
diabetes_Y_train = diabetes.target[:-20]
diabetes_Y_test = diabetes.target[-20:]
plt.scatter(diabetes_X_train,diabetes_Y_train,color='blue',marker='o')
plt.show()
plt.scatter(diabetes_X_test,diabetes_Y_test,color='blue',marker='o')
plt.show()
```





⑤ diabetes_X_train=np.array(diabetes_X_train).reshape(-1,1)句的意义？

在新版 sklearn 中，所有数据都应该是二维矩阵，np.array(diabetes_X_train)只是单独一行或一列，需要使用.reshape(1,-1)进行转换。

⑥ 线性回归回归系数计算

回归系数计算

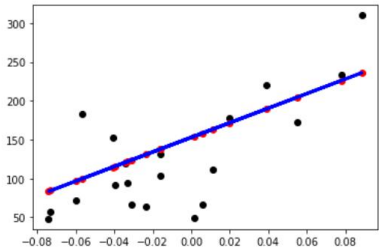
In [23]:
```
print("coefficients:",regr.coef_)
```

```
coefficients: [938.23786125]
```

⑦ 线性回归的回归结果折线图及散点图展示

回归结果折线图及散点图

In [15]:
```
plt.scatter(diabetes_X_test,diabetes_Y_test,color = 'black')
plt.scatter(diabetes_X_test,diabetes_Y_pred,color = 'red')
plt.plot(diabetes_X_test,diabetes_Y_pred,color='blue',linewidth=3)
plt.show()
```



⑧ 逻辑回归回归系数计算

逻辑回归回归系数计算

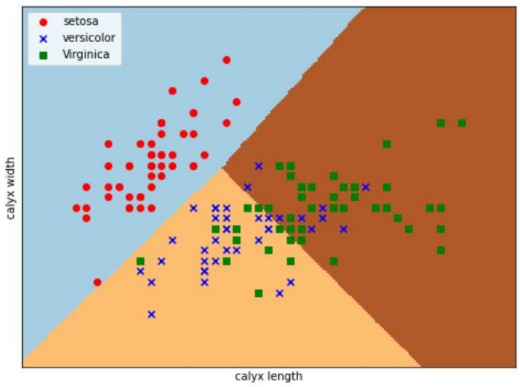
In [44]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.datasets import load_iris
iris = load_iris()
#print(iris.data)
#print(iris.target)
X = iris.data[:, :2] #获取鸢尾花前两列数据
Y = iris.target
clf = LogisticRegression(C=1e5)
clf.fit(X,Y)
print("coefficients:",clf.coef_)
```

```
coefficients: [[-32.00790225  29.63282048]
 [  0.1294651   -3.21259106]
 [  2.6022093   -0.74600204]]
```

⑨ 逻辑回归回归散点图展示

散点图展示

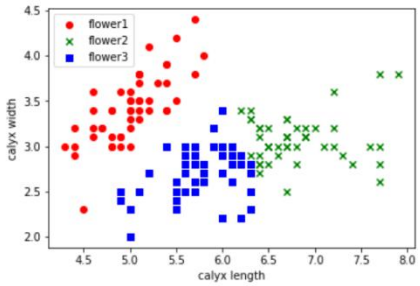
In [15]:
```
h = .02
x_min, x_max = X[:, 0].min() - .5, X[:, 0].max() + .5
y_min, y_max = X[:, 1].min() - .5, X[:, 1].max() + .5
xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min, y_max, h))
Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)
plt.figure(1, figsize=(8, 6))
plt.pcolormesh(xx, yy, Z, cmap=plt.cm.Paired)
plt.scatter(X[:50, 0], X[:50, 1], color = 'red', marker = 'o', label = 'setosa')
plt.scatter(X[50:100, 0], X[50:100, 1], color = 'blue', marker = 'x', label = 'versicolor')
plt.scatter(X[100:, 0], X[100:, 1], color = 'green', marker = 's', label = 'Virginica')
plt.xlabel('calyx length')
plt.ylabel('calyx width')
plt.xlim(xx.min(),xx.max())
plt.ylim(yy.min(),yy.max())
plt.xticks(())
plt.yticks(())
plt.legend(loc=2)
plt.show()
```



⑩ 对鸢尾花数据进行 K-means 聚类，绘制聚类中心为 3 的聚类结果图

K-means聚类

In [17]:
```
from sklearn.cluster import KMeans
estimator = KMeans(n_clusters=3) #构造聚类器
estimator.fit(X) #聚类
label_pred = estimator.labels_ #获取聚类标签
#绘制k-means结果
x0 = X[label_pred ==0]
x1 = X[label_pred ==1]
x2 = X[label_pred ==2]
plt.scatter(x0[:,0],x0[:,1],c='red',marker='o',label='flower1')
plt.scatter(x1[:,0],x1[:,1],c='green',marker='x',label='flower2')
plt.scatter(x2[:,0],x2[:,1],c='blue',marker='s',label='flower3')
plt.xlabel('calyx length')
plt.ylabel('calyx width')
plt.legend(loc=2)
plt.show()
```



2. 有余力的同学在进行以下实验并进行创新，给出最终实验结果（选做）

①通过均方差评估下不同参数下回归模型的能力

本次实验主要研究了 LogisticRegression()中参数 c 不同取值时，均方差的大小。c 是正则化系数λ的倒数，float 类型，默认为 1.0。必须是正浮点型数。像 SVM 一样，越小的数值表示越强的正则化，下面列表展示了不同的结果。

API	参数	意义	备注
LogisticRegression 的 parameters	Penalty	Str类型，可选项为'l1'和'l2'，默认为'l2'，用来确定惩罚项的规范。'newton-cg'，'sag'和'lbfgs'仅支持'l2'惩罚项。	惩罚项是用来添加参数避免过拟合，可以理解为对当前训练样本的惩罚，用以提高函数的泛化能力。
	dual	布尔类型，默认为‘False’，dual仅仅用在l2惩罚项的liblinear解法上。	Liblinear好像是线性多核，但不太清楚详细意思。
	tol	float类型，默认为‘1e-4’，表示满足该精度时训练停止。	停止求解的标准，个人感觉应该是精度的意思。
	C	float类型，默认为‘1.0’，正则化系数的倒数，取值必须为正浮点数。	例如在SVM中，值越小表示越强的正则化。
	fit_intercept	布尔类型，默认为‘Ture’，确定是否有一个偏差或者截距应该被添加进决策函数。	
	intercept_scaling	float类型，默认为‘1.’，仅在解法‘liblinear’被使用且fit_intercept被置为True。	
	class_weight	字典类型或‘balanced‘，默认为‘None’，表示为各个类型样本的权重。如果没有给定，所有类型的权重为1。参数为‘balanced’则依照输入样本的频率自动调整权重。	‘balanced’参数的情况下，会按照比例调整比较多的类别的样本的权重降低，比较少样本的类别的样本权重依比提高。

通过均方差评估下不同参数下回归模型的能力

In [38]:

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import mean_squared_error
clf = LogisticRegression(C=1e5)
clf.fit(diabetes_X_train.reshape(-1,1),diabetes_Y_train)
diabetes_Y_pred = clf.predict(diabetes_X_test.reshape(-1,1))
print("Mean squred error:%.2f"
      % mean_squared_error(diabetes_Y_test,diabetes_Y_pred))
clf = LogisticRegression(C=1e3)
clf.fit(diabetes_X_train.reshape(-1,1),diabetes_Y_train)
diabetes_Y_pred = clf.predict(diabetes_X_test.reshape(-1,1))
print("Mean squred error:%.2f"
      % mean_squared_error(diabetes_Y_test,diabetes_Y_pred))
clf = LogisticRegression(C=1e2)
clf.fit(diabetes_X_train.reshape(-1,1),diabetes_Y_train)
diabetes_Y_pred = clf.predict(diabetes_X_test.reshape(-1,1))
print("Mean squred error:%.2f"
      % mean_squared_error(diabetes_Y_test,diabetes_Y_pred))
clf = LogisticRegression()
clf.fit(diabetes_X_train.reshape(-1,1),diabetes_Y_train)
diabetes_Y_pred = clf.predict(diabetes_X_test.reshape(-1,1))
print("Mean squred error:%.2f"
      % mean_squared_error(diabetes_Y_test,diabetes_Y_pred))
```

C	均方差
1.0	10277.60
1e2	5886.55
1e3	5893.10
1e5	5966.10

当 c 的值为默认值 1.0 时，这时均方差达到了 10277.60，说明模型拟合的效果并不好，而在 c 的值分别取 1e2、1e3 和 1e5 时，均方差在 5900 左右，模型的拟合效果大大提升。

②对于 matplotlib 的使用

matplotlib 是受 MATLAB 的启发构建的，在 matplotlib.pyplot 模块中有一套完全仿照 MATLAB 的函数形式的绘图接口。在绘图结构中，figure 创建窗口，subplot 创建子图。所有的绘画只能在子图上进行。plt 表示当前子图，若没有就创建一个子图。通过下面的命令来配置参数：

axex: 设置坐标轴边界和表面的颜色、坐标刻度值大小和网格的显示

figure: 控制 dpi、边界颜色、图形大小、和子区(subplot)设置

font: 字体集（font family）、字体大小和样式设置

grid: 设置网格颜色和线性

legend: 设置图例和其中的文本的显示

line: 设置线条（颜色、线型、宽度等）和标记

patch: 是填充 2D 空间的图形对象，如多边形和圆。控制线宽、颜色和抗锯齿设置等。

savefig: 可以对保存的图形进行单独设置。例如，设置渲染的文件的背景为白色。

verbose: 设置 matplotlib 在执行期间信息输出，如 silent、helpful、debug 和 debug-annoying。

xticks 和 yticks: 为 x,y 轴的主刻度和次刻度设置颜色、大小、方向，以及标签大小。

参考链接：

http://matplotlib.org/

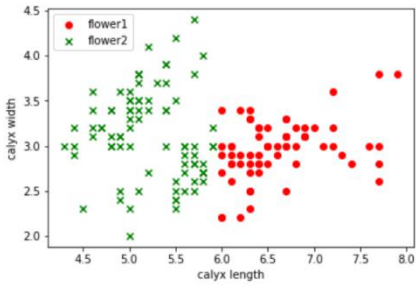
https://www.jianshu.com/p/da385a35f68d

③不同聚类数量对于结果的区别

2类

In [18]:

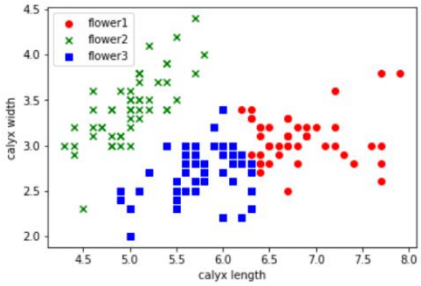
```
from sklearn.cluster import KMeans
estimator = KMeans(n_clusters=2) #构造聚类器
estimator.fit(X) #聚类
label_pred = estimator.labels_ #获取聚类标签
#绘制k-means结果
x0 = X[label_pred ==0]
x1 = X[label_pred ==1]
plt.scatter(x0[:,0],x0[:,1],c='red',marker='o',label='flower1')
plt.scatter(x1[:,0],x1[:,1],c='green',marker='x',label='flower2')
plt.xlabel('calyx length')
plt.ylabel('calyx width')
plt.legend(loc=2)
plt.show()
```



3类

In [19]:

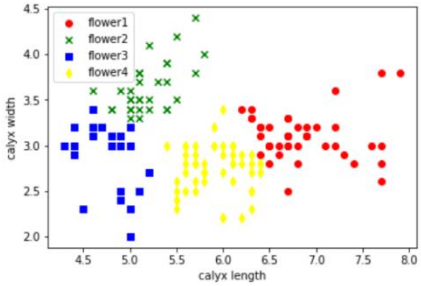
```
from sklearn.cluster import KMeans
estimator = KMeans(n_clusters=3) #构造聚类器
estimator.fit(X) #聚类
label_pred = estimator.labels_ #获取聚类标签
#绘制k-means结果
x0 = X[label_pred ==0]
x1 = X[label_pred ==1]
x2 = X[label_pred ==2]
plt.scatter(x0[:,0],x0[:,1],c='red',marker='o',label='flower1')
plt.scatter(x1[:,0],x1[:,1],c='green',marker='x',label='flower2')
plt.scatter(x2[:,0],x2[:,1],c='blue',marker='s',label='flower3')
plt.xlabel('calyx length')
plt.ylabel('calyx width')
plt.legend(loc=2)
plt.show()
```



4类

In [20]:

```
from sklearn.cluster import KMeans
estimator = KMeans(n_clusters=4) #构造聚类器
estimator.fit(X) #聚类
label_pred = estimator.labels_ #获取聚类标签
#绘制k-means结果
x0 = X[label_pred ==0]
x1 = X[label_pred ==1]
x2 = X[label_pred ==2]
x3 = X[label_pred ==3]
plt.scatter(x0[:,0],x0[:,1],c='red',marker='o',label='flower1')
plt.scatter(x1[:,0],x1[:,1],c='green',marker='x',label='flower2')
plt.scatter(x2[:,0],x2[:,1],c='blue',marker='s',label='flower3')
plt.scatter(x3[:,0],x3[:,1],c='yellow',marker='d',label='flower4')
plt.xlabel('calyx length')
plt.ylabel('calyx width')
plt.legend(loc=2)
plt.show()
```



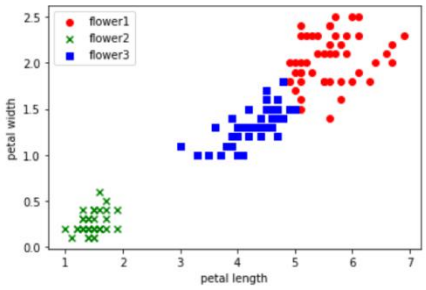
④使用不同的特征进行特征聚类会有什么结果

使用鸢尾花的后两列数据，即花瓣长度和花瓣宽度来进行聚类，聚类数量为 3：

使用不同的特征进行特征聚类会有什么结果

In [26]:

```
from sklearn.cluster import KMeans
from sklearn.linear_model import LogisticRegression
from sklearn.datasets import load_iris
iris = load_iris()
X = iris.data[:,2:4] #获取鸢尾花后两列数据
Y = iris.target
clf = LogisticRegression(C=1e5)
clf.fit(X,Y)
estimator = KMeans(n_clusters=3) #构造聚类器
estimator.fit(X) #聚类
label_pred = estimator.labels_ #获取聚类标签
#绘制k-means结果
x0 = X[label_pred ==0]
x1 = X[label_pred ==1]
x2 = X[label_pred ==2]
plt.scatter(x0[:,0],x0[:,1],c='red',marker='o',label='flower1')
plt.scatter(x1[:,0],x1[:,1],c='green',marker='x',label='flower2')
plt.scatter(x2[:,0],x2[:,1],c='blue',marker='s',label='flower3')
plt.xlabel('petal length')
plt.ylabel('petal width')
plt.legend(loc=2)
plt.show()
```



⑤⑥尝试更多的数据集，进行 2007 年前中国男足在亚洲水平的聚类实验

In [3]:
```
X_train=[
    [1,0.5],
    [0.3,0.19],
    [0,0.13],
    [0.24,0.25],
    [0.3,0.06],
    [1,0],
    [1,0.5],
    [1,0.5],
    [0.7,0.25],
    [1,0.5],
    [1,0.25],
    [1,0.5],
    [0.7,0.5],
    [0.7,1],
    [1,0.5]
]
kmeans = KMeans(n_clusters=3)
kmeans.fit(X_train)
y_pre = kmeans.predict(X_train)
y_pre
```

Out[3]: array([0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 2, 0])

将聚类结果列表如下：

国家	水平
日本、韩国、伊朗、沙特	亚洲一流
朝鲜	亚洲二流
其他国家	亚洲三流

题目要求对 2007 年之前的数据进行聚类，故选取了 2006 年和 2007 年的数据，聚类结果如上，中国属于亚洲三流