Python与金融数据挖掘(15)

文欣秀

wenxinxiu@ecust.edu.cn

机器学习分类

- 有监督学习（分类、回归）
- 无监督学习（聚类、降维）
- 强化学习
- 半监督学习

分类算法

- K近邻算法(KNN)
- 朴素贝叶斯算法(NB)
- 支持向量机(SVM)
- 决策树(DT)
- 逻辑回归(LR)

案例分析

	A	B	C	D
1	Sex	Height	Weight	Type
2	1	180	75	normal
3	1	180	85	normal
4	1	180	90	overweight
5	1	180	100	overweight
6	1	175	90	overweight
7	1	175	80	overweight
8	1	175	65	normal
9	1	175	55	underweight
10	1	170	60	normal
11	1	170	70	normal
12	1	170	80	overweight
13	1	185	90	overweight
14	1	185	75	normal
15	1	175	60	underweight
16	1	180	65	underweight
17	1	160	75	overweight
18	1	160	60	normal
19	1	170	68	normal
20	1	165	62	normal
21	1	190	75	underweight

胖瘦预测KNN算法

```
import pandas as pd
from sklearn.neighbors import KNeighborsClassifier
from warnings import simplefilter
simplefilter(action='ignore', category=FutureWarning)
data=pd.read_csv('info.csv',encoding='gb2312')
model= KNeighborsClassifier()
model.fit(data[['Sex','Height','Weight']].values, data['Type'].values)
height=int(input("Your Height:"))
weight=int(input("Your Weight:"))
print('Type：',model.predict([[1,height,weight]]))
```

胖瘦预测NB算法

```
import pandas as pd
from sklearn.naive_bayes import GaussianNB
data=pd.read_csv('info.csv',encoding='gb2312')
model= GaussianNB()
model.fit(data[['Sex','Height','Weight']].values, data['Type'].values)
height=int(input("Your Height:"))
weight=int(input("Your Weight:"))
print('Type：',model.predict([[1,height,weight]]))
```

胖瘦预测SVC算法

```
import pandas as pd
from sklearn.svm import SVC
data=pd.read_csv('info.csv',encoding='gb2312')
model= SVC( )
model.fit(data[['Sex','Height','Weight']].values, data['Type'].values)
height=int(input("Your Height:"))
weight=int(input("Your Weight:"))
print('Type：',model.predict([[1,height,weight]]))
```

胖瘦预测DT算法

```
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
data=pd.read_csv('info.csv',encoding='gb2312')
model= DecisionTreeClassifier()
model.fit(data[['Sex','Height','Weight']].values, data['Type'].values)
height=int(input("Your Height:"))
weight=int(input("Your Weight:"))
print('Type：  ',model.predict([[1,height,weight]]))
```

胖瘦预测LR算法

```
import pandas as pd
from sklearn.linear_model import LogisticRegression
data=pd.read_csv('info.csv',encoding='gb2312')
model= LogisticRegression(max_iter=10000)
model.fit(data[['Sex','Height','Weight']].values, data['Type'].values)
height=int(input("Your Height:"))
weight=int(input("Your Weight:"))
print('Type：  ',model.predict([[1,height,weight]]))
```

思考题一

问题： 如何比较以上算法的准确率从而选择有效的算法？

方案： 使用相同的数据，相同的方法来评估不同的算法，以便得到一个准确的结果。

	A	B	C	D	E
1	Sepal_Length	Sepal_Width	Petal_Length	Petal_Width	Species
2	5.1	3.5	1.4	0.2	setosa
3	4.9	3	1.4	0.2	setosa
4	4.7	3.2	1.3	0.2	setosa
5	4.6	3.1	1.5	0.2	setosa
6	5	3.6	1.4	0.2	setosa
7	5.4	3.9	1.7	0.4	setosa
8	4.6	3.4	1.4	0.3	setosa
9	5	3.4	1.5	0.2	setosa
10	4.4	2.9	1.4	0.2	setosa
11	4.9	3.1	1.5	0.1	setosa

鸢尾花(iris)数据集分析

Iris 数据集： 是一个经典数据集，在统计学习和机器学习领域都经常被用作示例。数据集内包含 3 类共 150 条记录，每类各 50 个数据，每条记录都有 4 项特征：花萼长度(Sepal Length)、花萼宽度(Sepal Width)、花瓣长度(Petal Length)、花瓣宽度(Petal Width)，可以通过这4个特征预测鸢尾花卉属于iris-setosa(山鸢尾)，iris-versicolour(变色鸢尾), iris-virginica(维吉尼鸢尾)中的哪一品种。

十折交叉验证

十折交叉验证（10-fold cross-validation）：用来测试算法准确性。是常用的测试方法。**将数据集分成10份，轮流将其中9份作为训练数据，1份作为测试数据进行试验**。每次试验都会得出相应的正确率（或差错率）。10次的结果的正确率（或差错率）的平均值作为对算法精度的估计，一般还需要进行多次10折交叉验证（例如8次10折交叉验证），再求其均值，作为对算法准确性的估计。

算法比较（1）

```
import pandas as pd
from sklearn.model_selection import KFold
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.model_selection import cross_val_score
```

算法比较（2）

```
from warnings import simplefilter
simplefilter(action='ignore', category=FutureWarning)
# 导入数据
dataset = pd.read_csv('iris.csv')
#显示数据维度
print('数据维度: 行 %s，列 %s' % dataset.shape)
# 查看数据的前10行
print(dataset.head(10))
```

```
数据维度: 行 150，列 5
```

```
数据维度: 行 150，列 5
   Sepal_Length  Sepal_Width  Petal_Length  Petal_Width Species
0           5.1          3.5           1.4          0.2  setosa
1           4.9          3.0           1.4          0.2  setosa
2           4.7          3.2           1.3          0.2  setosa
3           4.6          3.1           1.5          0.2  setosa
4           5.0          3.6           1.4          0.2  setosa
5           5.4          3.9           1.7          0.4  setosa
6           4.6          3.4           1.4          0.3  setosa
7           5.0          3.4           1.5          0.2  setosa
8           4.4          2.9           1.4          0.2  setosa
9           4.9          3.1           1.5          0.1  setosa
>>>
```

算法比较（3）

```
print(dataset.describe()) # 统计描述数据信息
print(dataset.groupby('Species').size()) # 分类分布情况
# 分离数据集
array = dataset.values
X = array[:, 0:4]
Y = array[:, 4]
seed = 7
kfold = KFold(n_splits=10, shuffle=True,random_state=seed)
```

	Sepal_Length	Sepal_Width	Petal_Length	Petal_Width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.057333	3.758000	1.199333
std	0.828066	0.435866	1.765298	0.762238
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

Species	
setosa	50
versicolor	50
virginica	50

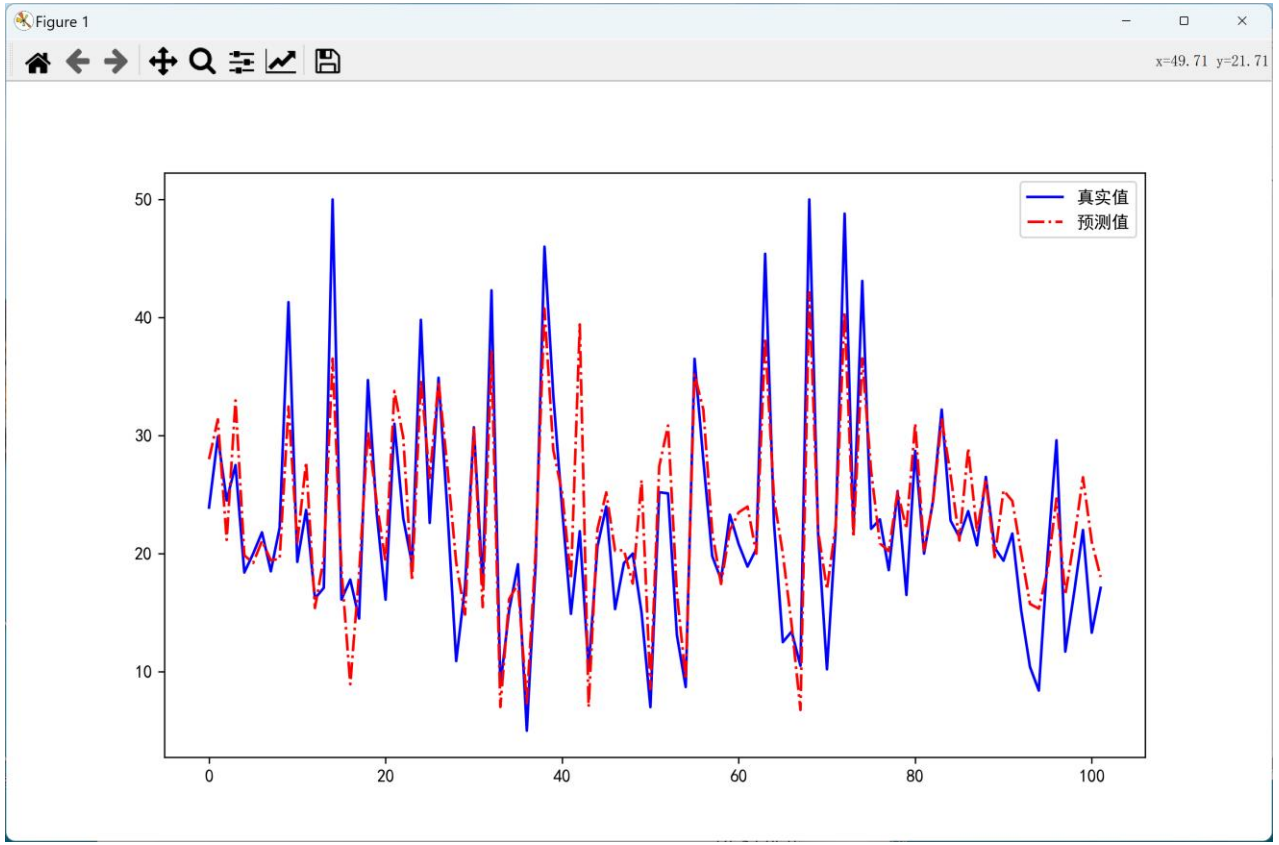
算法比较（4）

```
# 算法审查
models = {}
models['LR'] = LogisticRegression(max_iter=10000)
models['DT'] = DecisionTreeClassifier()
models['KNN'] = KNeighborsClassifier()
models['NB'] = GaussianNB()
models['SVM'] = SVC()
```

算法比较（5）

```
# 评估算法
results = []
for key in models:
    #cross_val_score:得到K折验证中每一折的得分
    cv_results = cross_val_score(models[key], X, Y, cv=kfold)
    results.append(cv_results)
    print('%s: %f (%f)' %(key, cv_results.mean(), cv_results.std()))
```

波士顿房价问题



Python实现线性回归步骤

- 导入对应库
- 加载数据集并划分数据集
- 在训练集上训练线性回归模型
- 使用测试集实现预测
- 绘图输出，结果可视化对比

波士顿房价

sklearn提供的波士顿房价数据集统计20世纪70年代中期**波士顿郊区房价**。该数据集包含**506**条记录，**13**个特征指标，第**14列**通常为目标列房价。试图能找到特征指标与房价的关系。

波士顿房价

本例首先将506组数据的数据集划分为训练集和测试集，其中**404组数据是训练样本**，剩下的**102组数据作为验证样本**。然后构建回归模型并训练模型，查看模型的13个特征的系数以及截距，获取模型的预测结果，最后绘制折线图对比预测值和真实。

波士顿房价回归模型

```
# （1）导入库
from sklearn.datasets import load_boston
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt
from matplotlib import rcParams
from warnings import simplefilter
simplefilter(action='ignore', category=FutureWarning)
```

波士顿房价回归模型

```
# （2）加载数据集
boston=load_boston()
x=boston['data']
y=boston['target']
names=boston['feature_names']
# 分割数据为训练集和测试集
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=22)
print('x_train前3行数据为：', x_train[0:3])
print('y_train前3行数据为：',y_train[0:3])
```

```
x_train前3行数据为： [[2.24236e+00 0.00000e+00 1.95800e+01 0.00000e+00 6.05000e-
01 5.85400e+00
  9.18000e+01 2.42200e+00 5.00000e+00 4.03000e+02 1.47000e+01 3.95110e+02
  1.16400e+01]
 [2.61690e-01 0.00000e+00 9.90000e+00 0.00000e+00 5.44000e-01 6.02300e+00
  9.04000e+01 2.83400e+00 4.00000e+00 3.04000e+02 1.84000e+01 3.96300e+02
  1.17200e+01]
 [6.89900e-02 0.00000e+00 2.56500e+01 0.00000e+00 5.81000e-01 5.87000e+00
  6.97000e+01 2.25770e+00 2.00000e+00 1.88000e+02 1.91000e+01 3.89150e+02
  1.43700e+01]]
y_train前3行数据为： [22.7 19.4 22. ]
```

波士顿房价回归模型

```
# （3）创建线性回归模型对象
lr=LinearRegression()
#使用训练集训练模型
lr.fit(x_train,y_train)
#显示模型
print(lr)
print("13个系数:",lr.coef_)
print("模型截距:",lr.intercept_ )
# （4）使用测试集获取预测结果
print("预测结果:",lr.predict(x_test[:5]))
```

```
LinearRegression()
```

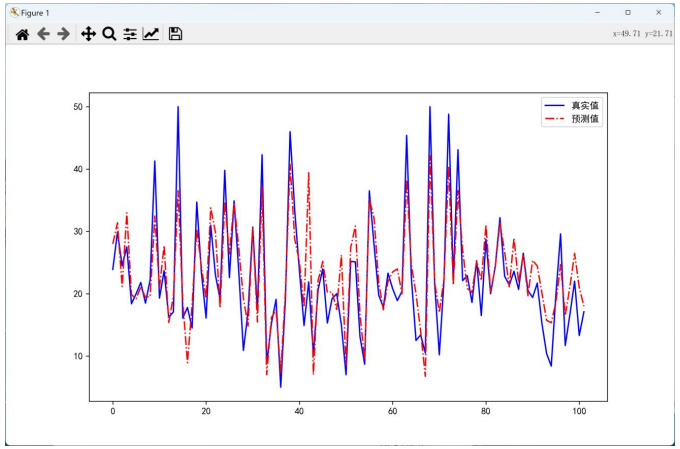
```
13个系数: [-1.01199845e-01  4.67962110e-02 -2.06902678e-02  3.58072311e+00
 -1.71288922e+01  3.92207267e+00 -5.67997339e-03 -1.54862273e+00
  2.97156958e-01 -1.00709587e-02 -7.78761318e-01  9.87125185e-03
 -5.25319199e-01]
```

```
模型截距: 32.428252866991 19
```

```
预测结果: [27.99617259 31.37458822 21.16274236 32.97684211 19.85350998]
```

波士顿房价回归模型

```
# （5）绘图对比预测值和真实值
rcParams['font.sans-serif']='SimHei'
fig=plt.figure(figsize=(10,6))
y_pred=lr.predict(x_test)
plt.plot(range(y_test.shape[0]),y_test,color="blue",linestyle="-")
plt.plot(range(y_test.shape[0]),y_pred,color="red",linestyle="-.")
plt.legend(['真实值','预测值'])
plt.show()
```



机器学习分类

- 有监督学习（分类、回归）
- 无监督学习（聚类、降维）
- 强化学习
- 半监督学习

聚 类

聚类(Clustering Approach)：是按一定的距离或相似性系数将数据分成一系列相互区分的组，常用的经典聚类方法有K-means, K-medoids，isodata等。

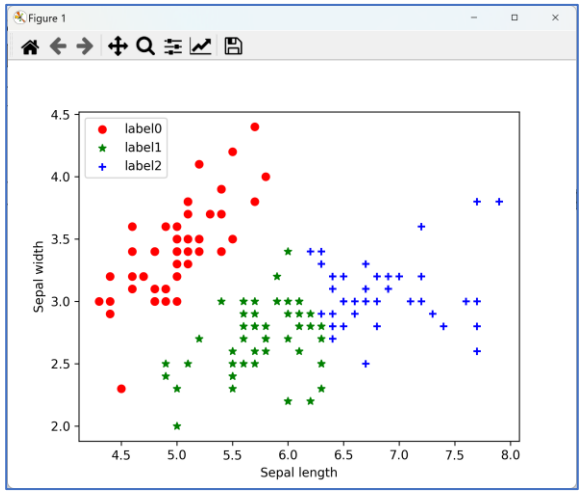
聚类算法的应用场景：市场分析、商业经营、图像处理、决策支持、模式识别。

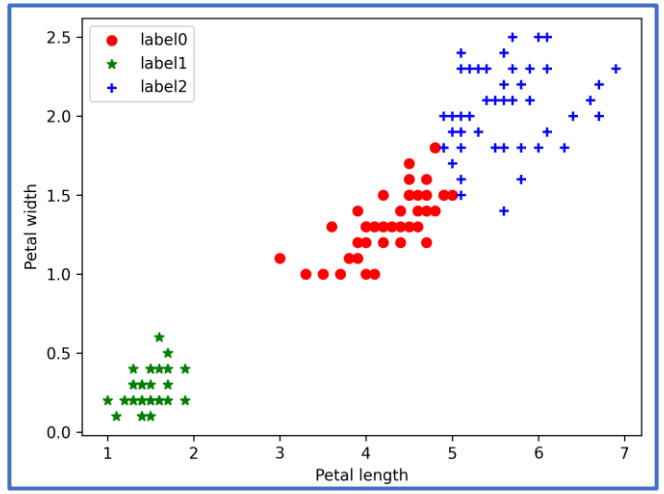
K-Means聚类算法

- K-Means算法属于聚类分析中划分方法里较为经典的一种，由于该算法的效率高，所以在对大规模数据进行聚类时被广泛应用。
- K-Means算法通过将样本划分k个簇类来实现数据聚类，该算法需要指定划分类的个数。

鸢尾花聚类问题

	A	B	C	D
1	Sepal_Length	Sepal_Width	Petal_Length	Petal_Width
2	5.1	3.5	1.4	0.2
3	4.9	3	1.4	0.2
4	4.7	3.2	1.3	0.2
5	4.6	3.1	1.5	0.2
6	5	3.6	1.4	0.2
7	5.4	3.9	1.7	0.4
8	4.6	3.4	1.4	0.3
9	5	3.4	1.5	0.2
10	4.4	2.9	1.4	0.2
11	4.9	3.1	1.5	0.1
12	5.4	3.7	1.5	0.2
13	4.8	3.4	1.6	0.2





K- Means聚类算法

Scikit-learn的Cluster类提供聚类分析的方法：

- 模型初始化

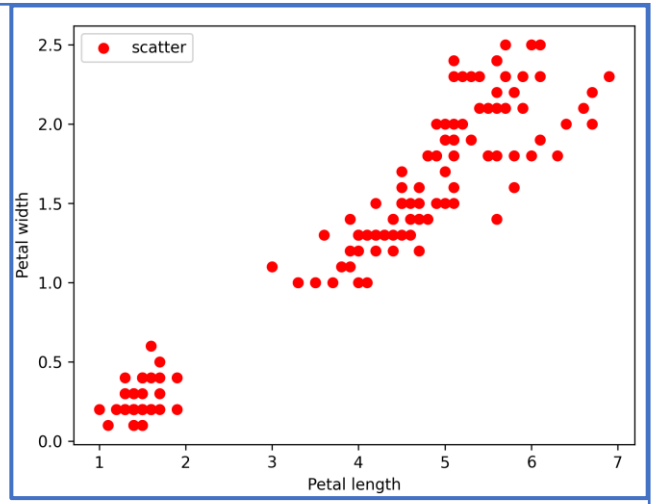
```
kmeans=Kmeans(n_clusters)    #参数为簇的个数
```

- 模型学习

```
kmeans. fit(X)                    #参数为样本二维数组
```

鸢尾花问题K- Means模型（1）

```
import matplotlib.pyplot as plt
import numpy as np
from sklearn.cluster import KMeans
import pandas as pd   #导入模块
iris = pd.read_csv("iris.csv")
X = iris.loc[:,['Petal_Length', 'Petal_Width']] #读出数据
plt.scatter(X['Petal_Length'], X['Petal_Width'], c = "red", marker='o', label='scatter')
plt.xlabel('Petal length')
plt.ylabel('Petal width')
plt.legend(loc=2)
plt.show()
```

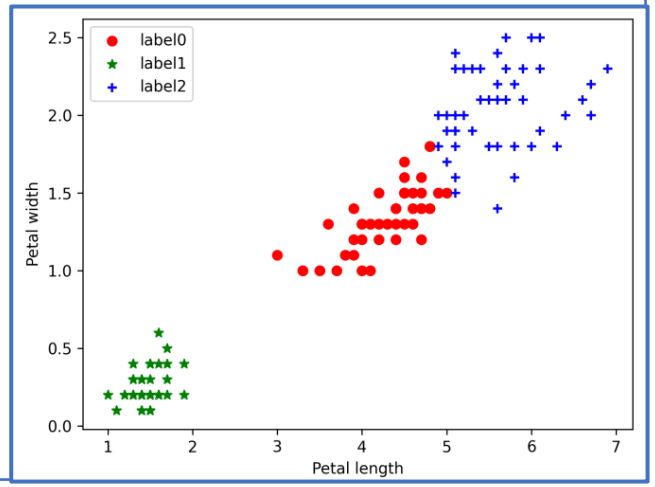


鸢尾花问题K- Means模型（2）

```
estimator = KMeans(n_clusters=3)#模型初始化
estimator.fit(X) #模型学习
label_pred = estimator.labels_ #获取聚类标签
x0 = X[label_pred == 0]
x1 = X[label_pred == 1]
x2 = X[label_pred == 2]
print(x0)
print(x1)
print(x2)
```

鸢尾花问题K- Means模型（3）

```
plt.scatter(x0['Petal_Length'], x0['Petal_Width'], c = "red", marker='o', label='label0')

plt.scatter(x1['Petal_Length'], x1['Petal_Width'], c = "green", marker='*', label='label1')

plt.scatter(x2['Petal_Length'], x2['Petal_Width'], c = "blue", marker='+', label='label2')

plt.xlabel('Petal length')

plt.ylabel('Petal width')

plt.legend(loc=2)

plt.show()
```



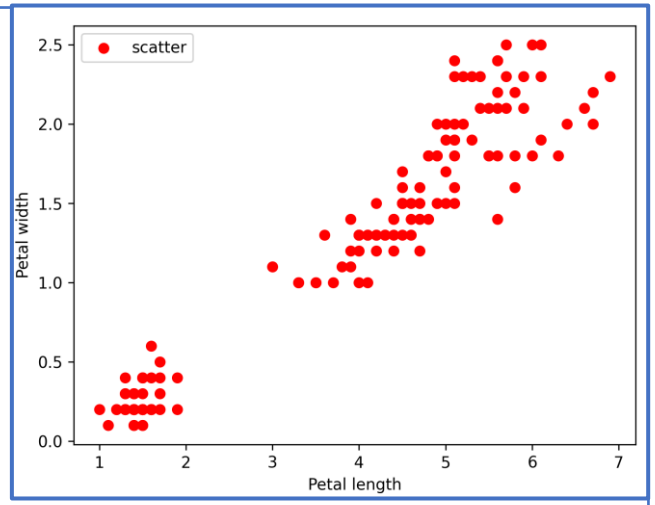
鸢尾花数据集获取

```
>>> from sklearn import datasets     # 导入数据集包
>>> dir (datasets)                   # 查看数据集
>>> iris = datasets.load_iris()
>>> X = iris.data[:,:4]
>>> print(X)
```

导入数据的函数名称	对应的数据集
load_boston()	波士顿房价数据集
load_breast_cancer()	乳腺癌数据集
load_iris()	鸢尾花数据集
load_diabetes()	糖尿病数据集
load_digits()	手写数字数据集
load_linnerud()	体能训练数据集
load_wine()	红酒品类数据集

鸢尾花问题K- Means模型（1修改）

```
import matplotlib.pyplot as plt
import numpy as np
from sklearn.cluster import KMeans
from sklearn import datasets     # 导入数据集包
import pandas as pd   #导入模块
iris = datasets.load_iris()          # 加载数据据集
X = iris['data'] #读出数据
plt.scatter(X[:,2], X[:,3], c = "red", marker='o', label='scatter')
plt.xlabel('Petal length')
plt.ylabel('Petal width')
plt.legend(loc=2)
plt.show()
```

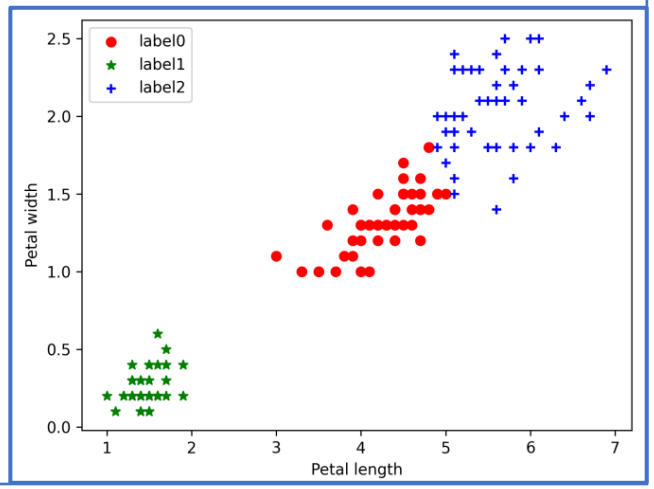


鸢尾花问题K- Means模型（2）

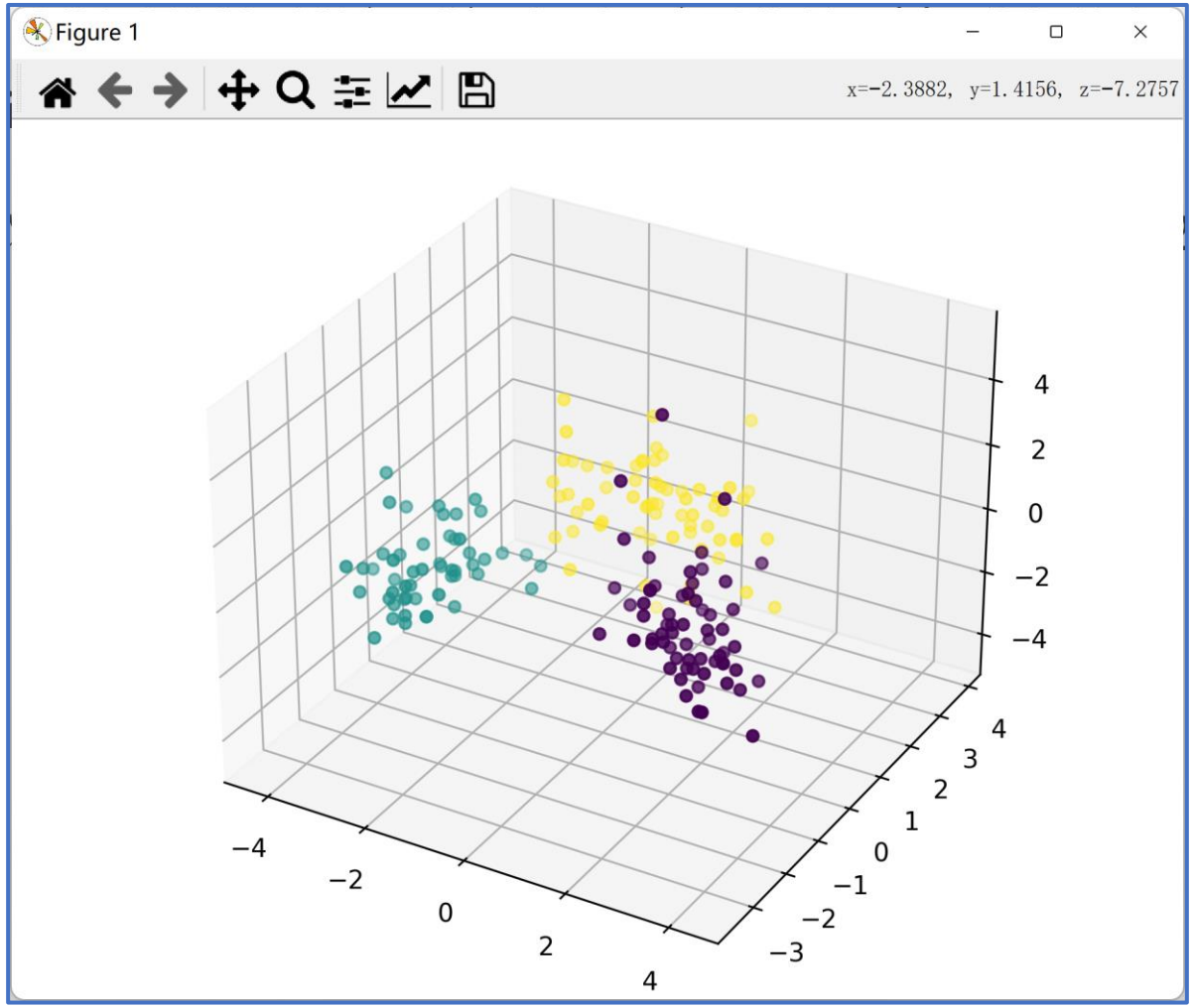
```
estimator = KMeans(n_clusters=3)#模型初始化
estimator.fit(X) #模型学习
label_pred = estimator.labels_ #获取聚类标签
x0 = X[label_pred == 0]
x1 = X[label_pred == 1]
x2 = X[label_pred == 2]
```

鸢尾花问题K- Means模型（3修改）

```
plt.scatter(x0[:,2], x0[:,3], c = "red", marker='o', label='label0')
plt.scatter(x1[:,2], x1[:,3], c = "green", marker='*', label='label1')
plt.scatter(x2[:,2], x2[:,3], c = "blue", marker='+', label='label2')
plt.xlabel('Petal length')
plt.ylabel('Petal width')
plt.legend(loc=2)
plt.show()
```

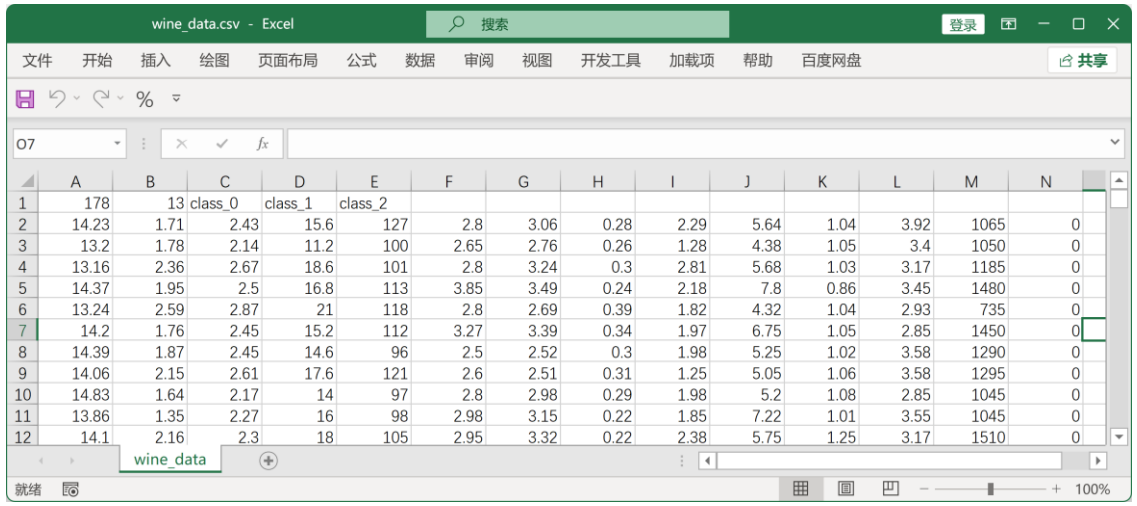


数据降维案例分析



葡萄酒(wine)数据集分析

wine数据集： 该数据集为意大利同一地区生产的三个不同种类的葡萄酒成分数据,每行代表一种酒的样本，共有178个样本，每个样本有13个特征，分为3个类别。



	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	178	13	class_0	class_1	class_2									
2	14.23	1.71	2.43	15.6	127	2.8	3.06	0.28	2.29	5.64	1.04	3.92	1065	0
3	13.2	1.78	2.14	11.2	100	2.65	2.76	0.26	1.28	4.38	1.05	3.4	1050	0
4	13.16	2.36	2.67	18.6	101	2.8	3.24	0.3	2.81	5.68	1.03	3.17	1185	0
5	14.37	1.95	2.5	16.8	113	3.85	3.49	0.24	2.18	7.8	0.86	3.45	1480	0
6	13.24	2.59	2.87	21	118	2.8	2.69	0.39	1.82	4.32	1.04	2.93	735	0
7	14.2	1.76	2.45	15.2	112	3.27	3.39	0.34	1.97	6.75	1.05	2.85	1450	0
8	14.39	1.87	2.45	14.6	96	2.5	2.52	0.3	1.98	5.25	1.02	3.58	1290	0
9	14.06	2.15	2.61	17.6	121	2.6	2.51	0.31	1.25	5.05	1.06	3.58	1295	0
10	14.83	1.64	2.17	14	97	2.8	2.98	0.29	1.98	5.2	1.08	2.85	1045	0
11	13.86	1.35	2.27	16	98	2.98	3.15	0.22	1.85	7.22	1.01	3.55	1045	0
12	14.1	2.16	2.3	18	105	2.95	3.32	0.22	2.38	5.75	1.25	3.17	1510	0

葡萄酒(wine)数据集特征

13个特征：酒精、苹果酸、灰、灰分的碱度、镁、总酚、黄酮类化合物、非黄烷类酚类、原花色素、颜色强度、色调、稀释葡萄酒的OD280/OD315、脯氨酸。其中第1类有59个样本，第2类有71个样本，第3类有48个样本。

PCA主成分分析技术

PCA主成分分析技术：旨在利用降维的思想，把多指标转化为少数几个综合指标。

思考：特征降维能够有效降低数据量，wine数据集的13个特征是否都对分类提供帮助？能否进行降维处理？

wine数据集主成分分析PCA（1）

```
from sklearn.cluster import Kmeans   #K-Means聚类模型
from sklearn.datasets import load_wine   #wine数据集
from sklearn.decomposition import PCA  #pca降维
from sklearn.preprocessing import scale    #数据标准化
from sklearn.preprocessing import StandardScaler #标准化
from sklearn import metrics
import numpy as np
import matplotlib.pyplot as plt#数据可视化
from mpl_toolkits.mplot3d import Axes3D #三D绘图
```

wine数据集主成分分析PCA（2）

```
#导入数据集
wine = load_wine()
X = wine['data']
y = wine['target']
print("data:")
print(X)
print("target")
print(y)
```

```
data:
[[1.423e+01 1.710e+00 2.430e+00 ... 1.040e+00 3.920e+00 1.065e+03]
 [1.320e+01 1.780e+00 2.140e+00 ... 1.050e+00 3.400e+00 1.050e+03]
 [1.316e+01 2.360e+00 2.670e+00 ... 1.030e+00 3.170e+00 1.185e+03]
 ...
 [1.327e+01 4.280e+00 2.260e+00 ... 5.900e-01 1.560e+00 8.350e+02]
 [1.317e+01 2.590e+00 2.370e+00 ... 6.000e-01 1.620e+00 8.400e+02]
 [1.413e+01 4.100e+00 2.740e+00 ... 6.100e-01 1.600e+00 5.600e+02]]
target
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2]
```

wine数据集主成分分析PCA（3）

```
#pca降维
pca = PCA(n_components=3)
X_scale=StandardScaler().fit_transform(X)  #标准化矩阵
X_reduce=pca.fit_transform(scale(X_scale))


#聚类模型
model = KMeans(n_clusters=3)
model.fit(X_reduce)
labels=model.labels_
print(labels)
```

```
[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 2 0 0 0 0 0 0 0 0 0 0 0 1
 0 0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 2 0 0 1 0 0 0 0 0 0 0 0 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2]
```

```
target
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2]
```

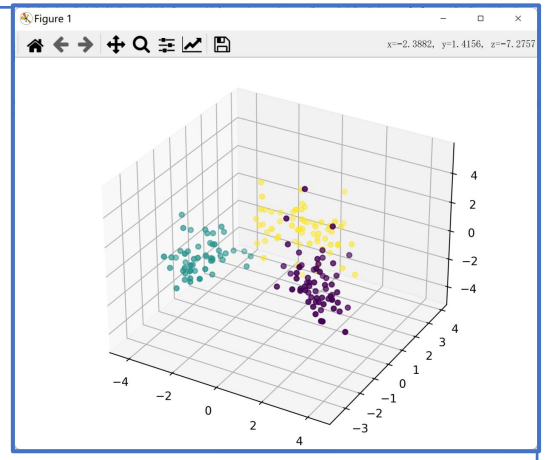
wine数据集主成分分析PCA（4）

0.879 0.873 0.876 0.897 0.875 0.454

```
#输出模型的准确度
print('%.3f  %.3f  %.3f  %.3f  %.3f  %.3f' % \
(metrics.homogeneity_score(y, labels),
 metrics.completeness_score(y, labels),
 metrics.v_measure_score(y, labels),
 metrics.adjusted_rand_score(y, labels),
 metrics.adjusted_mutual_info_score(y, labels),
 metrics.silhouette_score(X_reduce, labels)))
```

wine数据集主成分分析PCA（5）

```
#绘制模型的分布图
fig=plt.figure()
ax=Axes3D(fig, auto_add_to_figure=False)
fig.add_axes(ax)
ax.scatter(X_reduce[:, 0], X_reduce[:, 1], X_reduce[:, 2], \
                                    c=labels.astype(np.float64))
plt.show()
```



数字图像处理

数字图像处理(Digital Image Processing)：是通过计算机对图像进行去除噪声、增强、复原、分割、提取特征等处理的方法和技术。

数字图像处理目的：

- 提高图像的视感质量
- 提取图像中所包含的某些特征或特殊信息
- 图像数据的变换、编码和压缩，便于存储和传输

图像尺寸案例

```
from PIL import Image
from numpy import array
import matplotlib. pylab as plt
a=Image. open("money.jpg")  #返回一个PIL图像对象
result=array(a)  #把图像对象转换为数组
print(result. shape) #返回图像的行数,列数,色彩通道数
print(result)
plt. imshow(a)#对图像进行处理，并显示其格式
plt. show()
```


```
(720, 1080, 3)
```

```
[[[159 157 160]
  [159 157 160]
  [158 158 160]
  ...
  [166 177 173]
  [166 177 173]
  [167 178 174]]

 [[159 157 160]
  [159 157 160]
  [158 158 160]
```

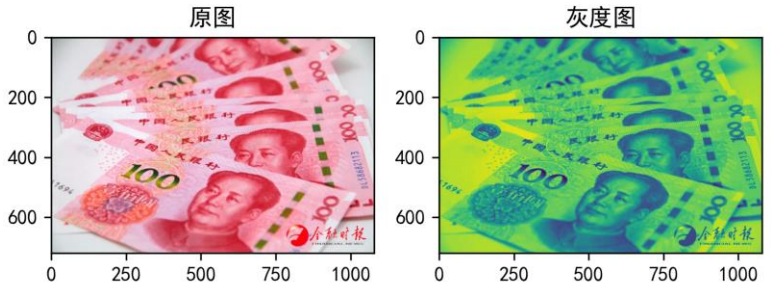
图像类型

二值图像：一幅二值图像的二维矩阵仅由0、1两个值构成，“0”代表黑色，“1”代白色，数据类型通常为1个二进制位。二值图像通常用于文字、线条图的扫描识别（OCR）。

灰度图像：灰度图像矩阵元素的取值范围通常为[0，255]，数据类型一般为8位无符号整数的(int8)，“0”表示纯黑色，“255”表示纯白色，如医学图像、遥感图像。

灰度图像案例

```
from PIL import Image
from numpy import array
import matplotlib. pylab as plt
a=Image. open("money.jpg")  #返回一个PIL图像对象
b=a. convert("L")  #转换为灰度图像对象
plt. rc('font',family="SimHei")
plt. subplot(121); plt. imshow(a); plt. title("原图")
plt. subplot(122); plt. imshow(b); plt. title("灰度图")
plt.show()
```



图像类型

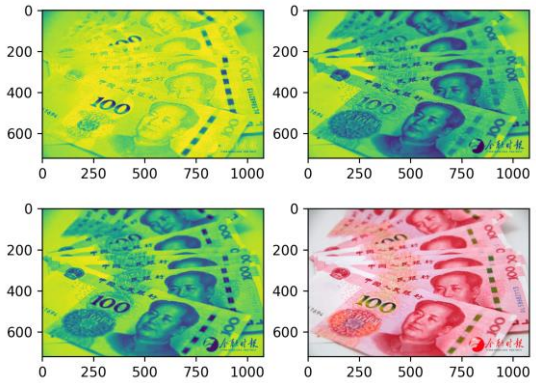
索引图像：除了存放图像的二维矩阵外，还包括一个称之为颜色索引矩阵MAP的二维数组，数据类型一般为**8位无符号整形（int8）**，MAP的大小由存放图像的矩阵元素值域决定。索引图像一般用于**存放色彩要求比较简单的图像**，如Windows中色彩构成比较简单的壁纸。

图像类型

RGB图像：与索引图像一样，它分别用红（**R**）、绿（**G**）、蓝（**B**）三原色的组合来表示每个像素的颜色。与索引图像不同的是，RGB图像每一个像素的颜色值直接存放在图像矩阵中，M、N分别表示图像的行列数。RGB图像的数据类型一般为**8位无符号整型**，通常用于表示和存放**真彩色**图像，当然也可以存放灰度图像。

Image图像分割与合并

```
from PIL import Image
from matplotlib.pyplot import *
a=Image. open('money.jpg') #读入图像
ra,ga,ba=a. split() #图像分割成R、G、B三个通道
c=Image. merge('RGB',(ra,ga,ba)) #三个通道合成一张彩色图像
subplot(221); imshow(ra); subplot(222); imshow(ga)
subplot(223); imshow(ba); subplot(224); imshow(c);
show()
```



53

谢　谢