实验六　使用 scikit-learn 构建模型

1.实验类型

设计型实验

2.实验目的和要求

（1） 掌握 sklearn 转换器的用法。
（2） 掌握训练集、测试集划分的方法。
（3） 掌握使用 sklearn 进行 PCA 降维的方法。
（4） 了解 sklearn 估计器的用法。
（5） 掌握聚类模型的构建方法。
（6） 掌握聚类模型的评价方法。
（7） 使用 sklearn 库建立 SVM 模型。
（8） 根据分类模型的评分指标评价 SVM 模型。

3.实验内容

题目一：使用 sklearn 处理 wine 和 wine_quality 数据集

(1)训练要点

- 掌握 sklearn 转换器的用法。
- 掌握训练集、测试集划分的方法。
- 掌握使用 sklearn 进行 PCA 降维的方法。

(2)需求说明

wine 数据集和 wine_quality 数据集分别是两份和酒有关的数据集。wine 数据集包含 3 种不同起源的葡萄酒的记录共 178 条。其中，每个特征对应葡萄酒的每种化学成分，并且都属于连续型数据。wine_quality 数据集共有 4898 个观察值，11 个输入特征和一个标签。其中，每个类的观察值数量不均等，所有特征为连续型数据。

希望可以通过 wine 化学分析可以来推断葡萄酒的起源，以及酒的各类化学成分，预测该红酒的评分。通过 wine 和 wine_quality 这两个数据集来完成数据的预处理工作。其中，包括导入相关函数的库；正确理解数据预处理部分的相关函数及功能。

(3)实现步骤

- 使用 pandas 库分别读取 wine 数据集和 win_quality 数据集。
- 将 wine 数据集和 wine_quality 数据集的数据和标签拆分开。
- 将 wine_quality 数据集划分为训练集和测试集。

- 标准化 wine 数据集和 wine_quality 数据集。
- 对 wine 数据集和 wine_quality 数据集进行 PCA 降维。

题目二：构建基于 wine 数据集的 K-Means 聚类模型

(1)训练要点

- 了解 sklearn 估计器的用法。
- 掌握聚类模型的构建方法。
- 掌握聚类模型的评价方法。

(2)需求说明

wine 数据集的红酒总共分为 3 种，通过将 wine 数据集的数据进行聚类，聚集为 3 个簇，能够实现红酒的类别划分。

(3)实现步骤

- 根据题目一的 wine 数据集处理的结果，构建聚类数目为 3 的 K-Means 模型。
- 对比真实标签和聚类标签求取 FMI。
- 在聚类数目为 2 至 10 类之间确定最优聚类数目。
- 求取模型的轮廓系数，绘制轮廓系数折线图，确定最优聚类数目。
- 求取 Calinski-Harabasz 指数，确定最优聚类数目。

题目三：构建基于 wine 数据集的 SVM 分类模型

(1)训练要点

- 使用 sklearn 库建立 SVM 模型。
- 根据分类模型的评分指标评价 SVM 模型。

(2)需求说明

wine 数据集中的红酒类别为 3 种，将 wine 数据集划分为训练集和测试集，使用训练集训练 SVM 分类模型，并使用训练完成的模型预测 wine 测试集的红酒类别归属。

(3)实现步骤

- 读取 wine 数据集，区分标签和数据。
- 将 wine 数据集划分为训练集和测试集。
- 使用离差标准化方法标准化 wine 数据集。
- 构建 SVM 模型，并预测测试集结果。
- 打印出分类报告，评价分类模型性能

4.实验背景知识

Scikit-learn 是专门面向机器学习的 Python 开源框架，它实现了各种成熟的算法，并且易于安装与使用。

（1）读取数据

使用 pandas 库分别读取 wine 数据集和 win_quality 数据集，并查看数据的长度。如代码 1 所示。

代码 1 读取 wine 数据集和 wine_quality 数据集

In[1]:

```
# 读取 wine 数据集
import pandas as pd
wine = pd.read_csv('./第 6 章/data/wine.csv',encoding='gb18030')
print('wine 数据集的长度为：',len(wine))  #字典
#wine 有 5 条记录    每条记录有 12 个属性
print('wine 数据集的长度为：',wine.shape)
print('wine_q 数据集的前五行：\n',wine.head())
```

Out[1]:

```
wine 数据集的长度为： 178
wine_q 数据集的前五行：
   Class  Alcohol  ...  OD280/OD315_of_diluted_wines  Proline
0      1    14.23  ...                          3.92     1065
1      1    13.20  ...                          3.40     1050
2      1    13.16  ...                          3.17     1185
3      1    14.37  ...                          3.45     1480
4      1    13.24  ...                          2.93      735

[5 rows x 14 columns]
```

In[2]:

```
# 读取 wine_quality 数据集
#读取.csv 文件  sep=';'列与列之前用分开
wine_quality = pd.read_csv('../数据/第 6 章/data/wine_quality.csv',encoding='gb18030',sep=';')
print('wine_quality 数据集的大小：',wine_quality.shape)
print('wine_quality 数据集的前 5 行：\n',wine_quality.head())
```

Out[2]:

```
wine_quality 数据集的大小： (1599, 12)
wine_quality 数据集的前 5 行：
   fixed acidity  volatile acidity  ...  alcohol  quality
0            7.4              0.70  ...      9.4        5
1            7.8              0.88  ...      9.8        5
2            7.8              0.76  ...      9.8        5
3           11.2              0.28  ...      9.8        6
4            7.4              0.70  ...      9.4        5
```

```
[5 rows x 12 columns]
```

（2）数据拆分

对 wine 数据集和 win_quality 数据集分别提取数据和标签。具体实现代码和如代码 2 所示。

代码 2 wine 数据集和 wine_quality 数据集拆分

In[3]:
```
# 拆分 wine 数据集
wine_data = wine.iloc[:,1:14]
wine_label = wine.iloc[:,0]
print('wine 数据集的标签为：\n',wine_label)
```

Out[3]:
```
wine 数据集的标签为：
0      1
1      1
2      1
……
175    3
176    3
177    3
Name: Class, Length: 178, dtype: int64
```

In[4]:
```
# 拆分 wine_quality 数据集
wine_quality_data = wine_quality.iloc[:,0:11]
wine_quality_label = wine_quality.iloc[:,11]
print('wine_quality 数据集的标签为：\n',wine_quality_label)
```

Out[4]:
```
wine_quality 数据集的标签为：
0       5
1       5
2       5
……
1596    6
1597    5
1598    6
Name: quality, Length: 1599, dtype: int64
```

（3）数据划分

在数据分析过程中，为了保证模型在实际系统中能够起到预期作用，一般需要将样本分成独立的两部分：训练集和测试集。sklearn 的 model_selection 模块提供了 train_test_split 函数，能够对数据集进行拆分。对 wine 数据集和 wine_quality 数据集划分训练集、测试集，如代码 3 所示。

代码 3 对 wine 数据集和 wine_quality 数据集划分训练集、测试集

```
In[5]:
# 对 wine 数据集进行划分
from sklearn.model_selection import train_test_split
wine_train,wine_test,wine_train_label,wine_test_label = train_test_split\
(wine_data,wine_label,test_size=0.2, random_state=42)
wine_train_label.to_csv('./第 6 章/tmp/wine_train_label.csv',index = False)
wine_test_label.to_csv('./第 6 章/tmp/wine_test_label.csv',index = False)
print('wine 原始数据集数据的形状为：',wine_data.shape)
print('wine 训练集数据的形状为：',wine_train.shape)
print('wine 训练集标签的形状为：',wine_train_label.shape)
print('wine 测试集数据的形状为：',wine_test.shape)
print('wine 测试集标签的形状为：',wine_test_label.shape)

Out[5]:
wine 原始数据集数据的形状为： (178, 13)
wine 训练集数据的形状为： (142, 13)
wine 训练集标签的形状为： (142,)
wine 测试集数据的形状为： (36, 13)
wine 测试集标签的形状为： (36,)

In[6]:
# 对 wine_quality 数据集进行划分
from sklearn.model_selection import train_test_split
wine_quality_train,wine_quality_test,wine_quality_train_label,wine_quality_test_label = \
train_test_split(wine_quality_data,wine_quality_label,test_size=0.2, random_state=42)
wine_quality_train_label.to_csv('./第 6 章/tmp/wine_quality_train_label.csv',index = False)
wine_quality_test_label.to_csv('./第 6 章/tmp/wine_quality_test_label.csv',index = False)
print('wine_quality 原始数据集数据的形状为：',wine_quality_data.shape)
print('wine_quality 训练集数据的形状为：',wine_quality_train.shape)
print('wine_quality 训练集标签的形状为：',wine_quality_train_label.shape)
print('wine_quality 测试集数据的形状为：',wine_quality_test.shape)
print('wine_quality 测试集标签的形状为：',wine_quality_test_label.shape)

Out[6]:
wine_quality 原始数据集数据的形状为： (1599, 11)
wine_quality 训练集数据的形状为： (1279, 11)
wine_quality 训练集标签的形状为： (1279,)
wine_quality 测试集数据的形状为： (320, 11)
wine_quality 测试集标签的形状为： (320,)
```

（4）数据标准化

由于不同特征的数据特征之间往往有不同的量纲，这样一来就会造成数值之间产生较大的差异，后面会影响到数据分析结果的准确性。因此，为了消除特征之间纲量和取值范围差异所造成的影响，需要对数据进行标准化处理。对 wine 数据集和 wine_quality 数据集进行标准化，如代码 4 所示。

代码 4 wine 数据集和 wine_quality 数据集标准化

```
In[7]:
from sklearn.preprocessing import MinMaxScaler
```

```
# 标准化 wine 训练集
Scaler = MinMaxScaler().fit(wine_train)  # 生成规则
# 将规则应用于 wine_quality 训练集
wine_train_Scaler = Scaler.transform(wine_train)
# 将 wine_train_Scaler 转为 DataFrame
wine_train_Scaler = pd.DataFrame(wine_train_Scaler)
# 重命名 wine_train_Scaler
wine_train_Scaler.columns = wine_train.columns
wine_train_Scaler.to_csv('./第 6 章/tmp/wine_train_Scaler.csv', index = False)
print('离差标准化后 wine 训练集前 5 行的数据为：\n',wine_train_Scaler.head())
```

Out[7]:

离差标准化后 wine 训练集前 5 行的数据为：

	Alcohol	Malic_acid	...	OD280/OD315_of_diluted_wines	Proline
0	0.871053	0.160896	...	0.252747	0.301024
1	0.394737	0.940937	...	0.153846	0.186761
2	0.352632	0.036660	...	0.549451	0.301024
3	0.644737	0.158859	...	0.186813	0.269504
4	0.536842	0.124236	...	0.520147	0.584712

[5 rows x 13 columns]

In[8]:

```
# 标准化 wine 测试集
Scaler = MinMaxScaler().fit(wine_test)  # 生成规则
# 将规则应用于 wine_quality 训练集
wine_test_Scaler = Scaler.transform(wine_test)
# 将 wine_test_Scalerr 转为 DataFrame
wine_test_Scaler = pd.DataFrame(wine_test_Scaler)
# 重命 wine_test_Scaler
wine_train_Scaler.columns = wine_test.columns
wine_test_Scaler.to_csv('./第 6 章/tmp/wine_test_Scaler.csv', index = False)
print('离差标准化后 wine 测试集前 5 行的数据为：\n',wine_test_Scaler.head())
```

Out[8]:

离差标准化后 wine 测试集前 5 行的数据为：

	0	1	2	...	10	11	12
0	0.771626	0.611399	0.741379	...	0.527027	0.817460	0.383764
1	0.968858	0.854922	0.637931	...	0.405405	0.805556	0.557196
2	0.525952	0.536269	0.862069	...	0.270270	0.400794	0.202952
3	0.802768	0.196891	0.862069	...	0.837838	0.559524	0.708487
4	0.332180	0.111399	0.189655	...	0.743243	0.865079	0.136531

[5 rows x 13 columns]

In[9]:

```
# 标准化 wine_quality 训练集
Scaler = MinMaxScaler().fit(wine_quality_train)  # 生成规则
# 将规则应用于 wine_quality 训练集
wine_quality_train_Scaler = Scaler.transform(wine_quality_train)
```

```
print('离差标准化后 wine_quality 训练集前 5 行的数据为：\n',
      wine_quality_train_Scaler[0:5,:])
```

Out[9]:

```
离差标准化后 wine_quality 训练集前 5 行的数据为：
 [[0.36283186 0.39041096 0.31         0.14383562 0.12353923 0.32835821
   0.26501767 0.74375918 0.58267717 0.22699387 0.49230769]
  [0.13274336 0.06164384 0.4          0.03424658 0.09015025 0.58955224
   0.56183746 0.08296623 0.4015748   0.13496933 0.53846154]
  [0.55752212 0.18493151 0.47         0.06164384 0.1769616  0.07462687
   0.02826855 0.5969163  0.44094488 0.23312883 0.21538462]
  [0.37168142 0.3869863  0.26         0.04794521 0.12687813 0.2238806
   0.06007067 0.50440529 0.45669291 0.06134969 0.15384615]
  [0.33628319 0.62671233 0.15         0.34931507 0.10183639 0.14925373
   0.16961131 0.65565345 0.49606299 0.07361963 0.23076923]]
```

In[10]:

```
# 标准化 wine_quality 测试集
Scaler = MinMaxScaler().fit(wine_quality_test)  # 生成规则
# 将规则应用于 wine_quality 训练集
wine_quality_test_Scaler = Scaler.transform(wine_quality_test)
print('离差标准化后 wine_quality 测试集前 5 行的数据为：\n',
      wine_quality_test_Scaler[0:5,:])
```

Out[10]:

```
离差标准化后 wine_quality 测试集前 5 行的数据为：
 [[0.25714286 0.44221106 0.10126582 0.10655738 0.19582245 0.18309859
   0.14391144 0.51012146 0.33043478 0.32038835 0.21428571]
  [0.26666667 0.38190955 0.21518987 0.03278689 0.11227154 0.28169014
   0.35055351 0.42105263 0.46086957 0.14563107 0.19642857]
  [0.54285714 0.55276382 0.27848101 0.12295082 0.17754569 0.22535211
   0.099631   0.77732794 0.36521739 0.63106796 0.26785714]
  [0.33333333 0.34170854 0.39240506 0.08606557 0.10182768 0.43661972
   0.18819188 0.58299595 0.40869565 0.2038835  0.25       ]
  [0.16190476 0.34170854 0.30379747 0.04098361 0.09921671 0.23943662
   0.099631   0.32388664 0.46086957 0.26213592 0.39285714]]
```

（5）数据降维

降维是将高维度的数据保留下最重要的一些特征，去除噪声和不重要的特征，从而实现提升数据处理速度的目的。对 wine 数据集和 wine_quality 数据集实现 PCA（主成分分析方法）降维，如代码 5 所示。

代码 5 对 wine 数据集和 wine_quality 数据集降维

In[11]:

```
# 对 wine_quality 训练集进行降维
wine_quality_pca_model = PCA(n_components=8).fit(wine_quality_train)  # 生成规则
wine_quality_trainPca = wine_quality_pca_model.transform(wine_quality_train)  # 将规则应用于训练集
# 将 wine_quality_trainPca 转为 DataFrame
```

```
wine_quality_trainPca = pd.DataFrame(wine_quality_trainPca)
wine_quality_trainPca.to_csv('./第 6 章/tmp/wine_quality_trainPca.csv', index = False)
print('PCA 降维前 wine_quality 训练集数据的形状为：',wine_quality_train.shape)
print('PCA 降维后 wine_quality 训练集数据的形状为：',wine_quality_trainPca.shape)
```

Out[11]:

```
PCA 降维前 wine_quality 训练集数据的形状为： (1279, 11)
PCA 降维后 wine_quality 训练集数据的形状为： (1279, 8)
```

In[12]:

```
# 对 wine_quality 测试集进行降维
wine_quality_testPca = wine_quality_pca_model.transform(wine_quality_test)  # 将规则应用于测试集
# 将 wine_quality_testPca 转为 DataFrame
wine_quality_testPca = pd.DataFrame(wine_quality_testPca)
wine_quality_testPca.to_csv('./第 6 章/tmp/wine_quality_testPca.csv', index = False)
print('PCA 降维前 wine_quality 测试集数据的形状为：',wine_quality_test.shape)
print('PCA 降维后 wine_quality 测试集数据的形状为：',wine_quality_testPca.shape)
```

Out[12]:

```
PCA 降维前 wine_quality 测试集数据的形状为： (320, 11)
PCA 降维后 wine_quality 测试集数据的形状为： (320, 8)
```

（6）构建 K-Means 模型

数据预处理的部分包括数据读取、数据与标签的拆分、数据标准化处理以及划分数据为训练集和测试集。最后，利用 sklearn 包中提供的 K-Means 算法对数据建模，聚类数目设定为 3，如代码 6 所示。

代码 6 数据预处理及 K-Means 模型构建

In[1]:

```
import pandas as pd
from sklearn.cluster import KMeans  #导入分类器库

# 读取训练集数据
wine_train_Scaler = pd.read_csv('./第 6 章/tmp/wine_train_Scaler.csv')
wine_train_label = pd.read_csv('./第 6 章/tmp/wine_train_label.csv',header = None)
wine_train_label.columns = ['label']  # 重命名列名
# 构建并训练模型
kmeans = KMeans(n_clusters = 3,random_state=123).fit(wine_train_Scaler)
print('构建 k-means 模型为：\n',kmeans)
```

Out[1]:

```
构建 k-means 模型为：
 KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
     n_clusters=3, n_init=10, n_jobs=1, precompute_distances='auto',
     random_state=123, tol=0.0001, verbose=0)
```

（7）模型评价

sklearn 的 metircs 模块提供一些聚类模型的评价的指标，这里使用的是 FMI

评价法，如代码 7 所示。

代码 7 模型评价

In[2]:
```
# 导入 FMI 方法的评价函数
from sklearn.metrics import fowlkes_mallows_score
kmeans = KMeans(n_clusters= 3).fit(wine_train_Scaler)
score = fowlkes_mallows_score(wine_train_label['label'].tolist(),kmeans.labels_)
print('wine 数据集的类中心为 3 时，其 FMI 的评价分值为：%f'%score)
```

Out[2]:
```
wine 数据集的类中心为 3 时，其 FMI 的评价分值为：0.901579
```

In[3]:
```
for i in range(2,11):
    ##构建并训练模型
    kmeans = KMeans(n_clusters = i,random_state=123).fit(wine_train_Scaler)
    score = fowlkes_mallows_score(wine_train_label['label'],kmeans.labels_)
    print('wine 数据聚%d 类 FMI 评价分值为：%f' %(i,score))
```

Out[3]:
```
wine 数据聚 2 类 FMI 评价分值为：0.637271
wine 数据聚 3 类 FMI 评价分值为：0.901579
wine 数据聚 4 类 FMI 评价分值为：0.809973
wine 数据聚 5 类 FMI 评价分值为：0.750412
wine 数据聚 6 类 FMI 评价分值为：0.715938
wine 数据聚 7 类 FMI 评价分值为：0.686684
wine 数据聚 8 类 FMI 评价分值为：0.597405
wine 数据聚 9 类 FMI 评价分值为：0.576837
wine 数据聚 10 类 FMI 评价分值为：0.521399
```

（8）确定最优聚类数

为了更好的确定 k 值，观察 k 取 2 到 10 时，使用轮廓系数评估 KMeans 模型，然后做出轮廓系数走势图，根据图形判断聚类效果，如代码 8 所示。

代码 8 轮廓系数评估 KMeans 模型

In[4]:
```
from sklearn.metrics import silhouette_score
import matplotlib.pyplot as plt
silhouettteScore = []
for i in range(2,11):
    kmeans = KMeans(n_clusters = i,random_state=123).fit(wine_train_Scaler)
    score = silhouette_score(wine_train_Scaler,kmeans.labels_)
    silhouettteScore.append(score)
plt.figure(figsize=(10,6))
plt.plot(range(2,11),silhouettteScore,linewidth=1.5, linestyle="-")
plt.show()
```

使用 Calinski-Harabasz 指数评估 K-Means 模型，如代码 9 所示。

代码 9 使用 Calinski-Harabasz 指数评价 K-Means 聚类

In[5]:

```
from sklearn.metrics import calinski_harabaz_score
for i in range(2,11):
    ##构建并训练模型
    kmeans = KMeans(n_clusters = i,random_state=1).fit(wine_train_Scaler)
    score = calinski_harabaz_score(wine_train_Scaler,kmeans.labels_)
    print('wine 数据聚%d 类 calinski_harabaz 指数为：%f'%(i,score))
```

(9) 构建 SVM 模型

根据实训一处理后的 wine 训练集建立 SVM 模型，如代码 10 所示。

代码 10 建立 SVM 模型

In[1]:

```
import pandas as pd
from sklearn.svm import SVC
from sklearn.metrics import classification_report  # 预测报告
train_Scaler = pd.read_csv('./第 6 章/tmp/wine_train_Scaler.csv')
wine_target_train = pd.read_csv('./第 6 章/tmp/wine_train_label.csv',header = None)
test_Scaler = pd.read_csv('./第 6 章/tmp/wine_test_Scaler.csv')
wine_target_test = pd.read_csv('./第 6 章/tmp/wine_test_label.csv',header = None)
print('训练集形状：',train_Scaler.shape,'\n',
      '训练集标签形状：',wine_target_train.shape,'\n',
      '测试集形状',test_Scaler.shape,'\n',
      '测试集标签形状',wine_target_test.shape)
# 构建 SVM 模型，并预测测试集结果。
svm = SVC().fit(train_Scaler,wine_target_train)
print('建立的 SVM 模型为：\n',svm)
```

Out[1]:

```
建立的 SVM 模型为：
 SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma='auto', kernel='rbf',
  max_iter=-1, probability=False, random_state=None, shrinking=True,
  tol=0.001, verbose=False)
```

(10) 预测 wine 测试集

使用代码 11 的模型预测 wine 测试集的红酒类别。

代码 11 建立 SVM 模型

In[3]:

```
# SVM 模型预测结果
wine_target_pred = svm.predict(test_Scaler)
print('测试集的预测结果为：\n',wine_target_pred)
```

(11) 评价评估

导入 sklearn.metrics 模块的 classification_report 函数，来评价 SVM 模型的

有效性，如代码 12 所示。

代码 12 模型评估

In[4]:	print('SVM 模型分类报告：\n',classification_report(wine_target_test,wine_target_pred))

5.实验思考

（1） 常见的模型算法使用场景有哪些？
（2） 聚类和分类的区别是什么？
（3） 回归和分类的区别又是什么？