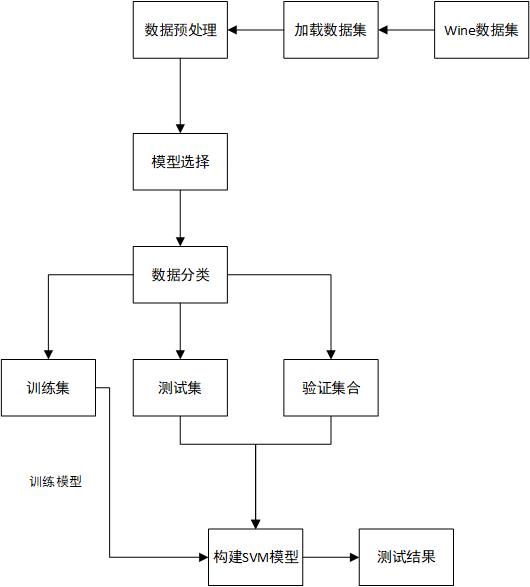
1. **wine数据集的描述**

本文Wine数据来自UCI数据库中下载的wine.data文件，并且使用核函数SVM支持向量机的方式进行机器学习分析，从而实现计算机对三种葡萄酒的成分分析并且进行自动识别类别。

分析wine.data文件我们可以看出，wine.data文件中分为178行、14列，分别对应着178行的不同葡萄酒的成分数据；第1列葡萄酒的种类，使用1/2/3来进行区分，后13列分别代表13种不同的葡萄酒的成分数据，分别为1) Alcohol，2) Malic acid，3) Ash，4) Alcalinity of ash，5) Magnesium，6) Total phenols，7) Flavanoids，8) Nonflavanoid phenols，9) Proanthocyanins，10)Color intensity，11)Hue，12)OD280/OD315 of diluted wines，13)Proline，然后进行训练与分类。

1. **分类系统流程图，并简述怎样训练怎样识别**



本文先处理wine.data数据，将各个变量进行命名，然后处理关联性较高的数据进行取舍，减小过拟合的可能；接着将数据随机混合并且按照5：3：2的比例分割为训练集、验证集和测试集，然后进行归一化处理；之后从线性核函数、多项式核函数、RBF高斯核函数和sigmoid核函数4个不同的核函数kernel选择核函数的类型，通过判断准确率来进行选择核函数类型；选择好核函数类型之后通过多个C惩罚系数和gamma进行筛选，得到最佳的C与gamma，之后再测试集与验证集中进行评价；同时使用归一化处理后的训练集构建随机森林模型，再使用测试集评价随机森林模型，并得到数据集的特征重要性以及最重要和最不重要的决策树。

1. **算法描述、评价指标**
2. SVM算法描述

支持向量机（support vector machines，SVM）是一种二分类模型，它将实例的特征向量映射为空间中的一些点，SVM 的目的就是想要画出一条线，以 “最好地” 区分这两类点，以至如果以后有了新的点，这条线也能做出很好的分类。SVM 适合中小型数据样本、非线性、高维的分类问题。由于本实验数据为非线性数据，因此我们采用核函数进行本次实验。

1. 随机森林算法描述

随机森林是一种集成学习方法，通过构建多个决策树并将它们组合起来进行预测。每棵决策树都是基于对原始数据集的随机抽样和随机特征选择建立的，使得每棵树都具有一定的差异性。在预测时，随机森林集成了所有决策树的结果，由于随机森林能够处理高维数据和大规模数据集，并且具有较好的泛化能力和抗过拟合能力。

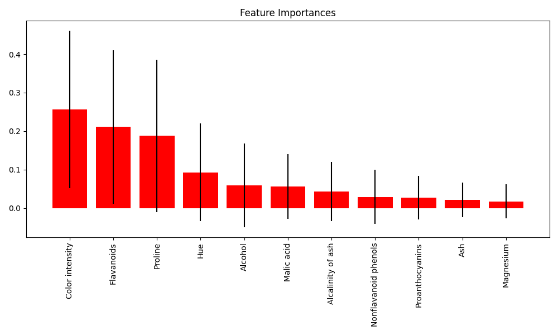
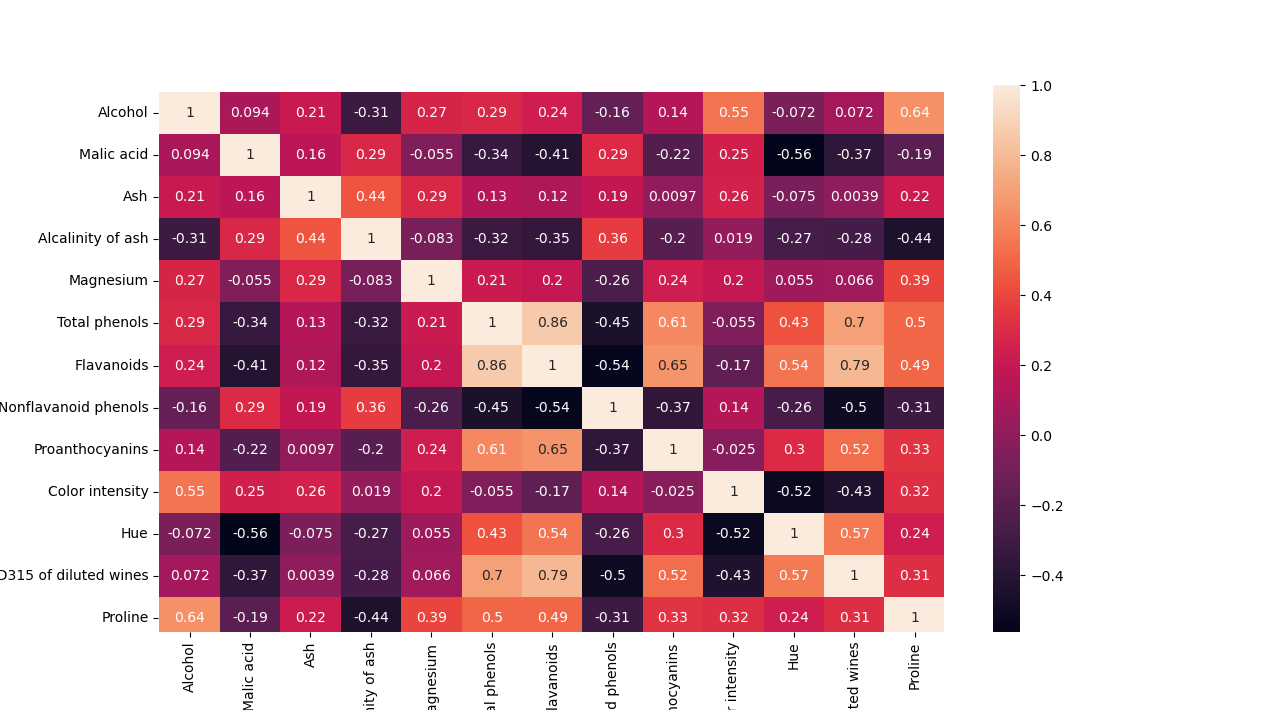
1. 评价指标

本文的评价指标为5类，分别为精确率precision、召回率recall、F1-score、准确率accuracy、宏观平均值macro recall。

1. 精确度：精确度最直观的表明了分类器不将负样本标记为正样本的能力，即
2. 召回率：召回率表示了分类学找到的所有的正样本的能力，即
3. F1-score：F1分数是精确率和召回率的调和平均数，最大值为1，最小值为0.公式如下：
4. 准确率：代表分类器对于整个样本判断正确的比重，
5. 宏观平均值：先针对每个类计算评估指标，再对他们取平均。
6. **分类结果**
7. **如何建立的训练集、验证集和测试集**

本文将特征类似的数据去除，降低了数据的过拟合的可能。

数据预处理的方式是使用corr()函数计算每两个数据之间的相似性程度，同时使用了sns.heatmap()函数显示13个数据之间相似程度的热力图分布情况，同时使用访问rf\_classifier.feature\_importances\_参数得到数据集中各特征的重要性，如下图所示：

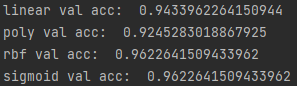


热力图 特征重要性图

从上述热力图中我们可以看出，参数6) Total phenols、7) Flavanoids和12)OD280/OD315 of diluted wines的相似度较高，存在很强的多重共线性。在进行后续处理时进行剔除三者中的两个变量，以免导致因多重共线性造成的过拟合。同时根据特征重要性图，Flavanoids的重要性比其他两个高。因此，我们在进行建立训练集、验证集和测试集之前使用feature\_remain保留1) Alcohol，2) Malic acid，3) Ash，4) Alcalinity of ash，5) Magnesium，6) Flavanoids，8) Nonflavanoid phenols，9) Proanthocyanins，10)Color intensity，11)Hue，13)Proline这11个变量。然后将数据随机混合并且按照5：3：2的比例分割为训练集、验证集和测试集，并进行归一化。

1. **SVM数据分类效果，附上分类结果图**
2. **核函数种类选择**

本文采用了四个核函数种类，分别为线性核函数、多项式核函数、RBF高斯核函数和sigmoid核函数，我们在判断同样的情况下进行分类测试准确性。



由上图我们可以看出，在四个核函数种类的准确性上明显RBF高斯核函数较高，达到了0.962，因此我们选择在后面选择RBF高斯核函数进行后续的实验和分类。

1. **分类效果**

最开始未进行调参选择时候，验证集的准确性等各项评价指标计算结果如表1、图1。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Precision | Recall | F1-score | Support |
| 1 | 0.94 | 1.00 | 0.97 | 15 |
| 2 | 0.96 | 0.96 | 0.96 | 23 |
| 3 | 1.00 | 0.93 | 0.97 | 15 |
| Accuracy |  |  | 0.96 | 53 |
| Macro avg | 0.96 | 0.96 | 0.96 | 53 |
| Weighted avg | 0.96 | 0.96 | 0.96 | 53 |

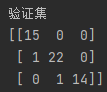


图1 未进行调参的验证集分类 表1 未进行调参的验证集分析

之后使用GridSearchCV()函数进行调参选择，惩罚因子C分别取0.1，0.2，0.3，0.4，0.5，0.6，0.7，0.8，0.9，1,5，10，100，1000和gamma参数分别取10，1，0.5，0.1，0.09，0.08，0.07，0.06，0.05，0.04，0.03，0.02，0.01，0.001，0.0001，选择发现最佳的惩罚因子C和gamma参数的分别为0.9和0.03。验证集的准确性等各项评价指标计算结果如表2、图2。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Precision | Recall | F1-score | Support |
| 1 | 1.00 | 1.00 | 1.00 | 15 |
| 2 | 0.96 | 1.00 | 0.98 | 23 |
| 3 | 1.00 | 0.93 | 0.97 | 15 |
| Accuracy |  |  | 0.98 | 53 |
| Macro avg | 0.99 | 0.98 | 0.98 | 53 |
| Weighted avg | 0.98 | 0.98 | 0.98 | 53 |

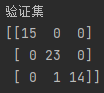


图2 使用GridSearchCV()函数调参后的验证集分类 表2 使用GridSearchCV()函数调参后的验证集分析

使用GridSearchCV()函数进行调参选择后最佳的惩罚因子C和gamma参数分类测试集，准确性等各项评价指标计算结果如表3、图3。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Precision | Recall | F1-score | Support |
| 1 | 1.00 | 0.92 | 0.96 | 12 |
| 2 | 0.93 | 1.00 | 0.97 | 14 |
| 3 | 1.00 | 1.00 | 1.00 | 10 |
| Accuracy |  |  | 0.97 | 36 |
| Macro avg | 0.98 | 0.97 | 0.97 | 36 |
| Weighted avg | 0.97 | 0.97 | 0.97 | 36 |

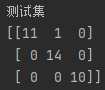


图3 使用GridSearchCV()函数调参后的测试集分类 表3 使用GridSearchCV()函数调参后的测试集分析

1. **随机森林数据分类效果，附上分类结果图**
2. **分类效果**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Precision | Recall | F1-score | Support |
| 1 | 0.92 | 0.92 | 0.92 | 12 |
| 2 | 0.93 | 0.93 | 0.93 | 14 |
| 3 | 1.00 | 1.00 | 1.00 | 10 |
| Accuracy |  |  | 0.94 | 36 |
| Macro avg | 0.95 | 0.95 | 0.95 | 36 |
| Weighted avg | 0.94 | 0.94 | 0.94 | 36 |

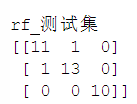


图4 使用rf的测试集分类 表4 使用rf的测试集分析

1. **最重要决策树和最差决策树分析**

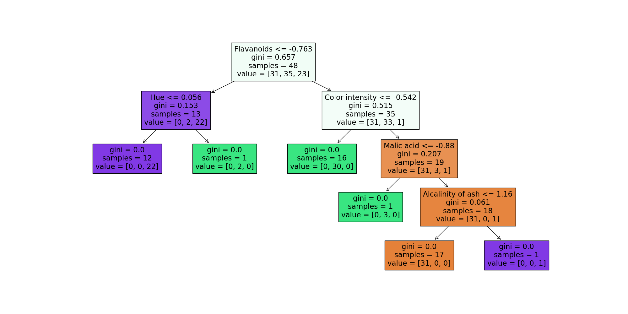
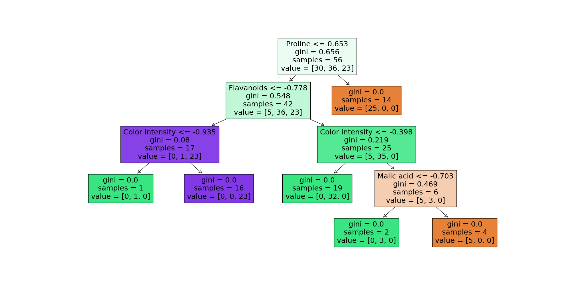


图5 worst\_tree 图6 most important tree

在两个树高度相同的情况下，我们不难发现根据重要特征图中，重要性在前三的三个特征在最重要的树中占了四个节点，而在最不重要的树中仅仅只占了两个节点。可见重要特征的使用会影响一个决策树是否重要。

1. **附录**

from sklearn import svm

import numpy as np

import matplotlib

from sklearn.model\_selection import train\_test\_split

import pandas as pd

import seaborn as sns

from sklearn.preprocessing import StandardScaler

from sklearn.metrics import confusion\_matrix

from sklearn.metrics import classification\_report

from sklearn.metrics import accuracy\_score

from sklearn.model\_selection import StratifiedKFold

from sklearn.model\_selection import GridSearchCV

import matplotlib.pyplot as plt

# 加载数据

path = 'F:\作业\人工智能导论\wine.data'

names = ['ID number', 'Alcohol','Malic acid','Ash','Alcalinity of ash','Magnesium','Total phenols','Flavanoids',

'Nonflavanoid phenols','Proanthocyanins','Color intensity','Hue','OD280/OD315 of diluted wines','Proline']

data = pd.read\_csv(path, header=None, names=names)

data.to\_csv( 'F:\作业\人工智能导论\wine.csv' )

# 显示数据

pd.set\_option('display.max\_columns', None)

print(data.columns)

print(data.head())

print(data.describe())

print(data)

# 特征字段分组，mean、se、worst

feature\_mean = list(data.columns[1:14])

# 诊断结果可视化

corr = data[feature\_mean].corr()

plt.figure(figsize=(17, 17))

sns.heatmap(corr, annot=True)

plt.show()

# 缩减属性（去除高度相关的，减小计算量）

feature\_remain = ['Alcohol' , 'Malic acid', 'Ash', 'Alcalinity of ash', 'Magnesium', 'Flavanoids','Nonflavanoid phenols', 'Proanthocyanins', 'Color intensity', 'Hue', 'Proline']

# 抽取数据，1：0.6：0.4

train, test1 = train\_test\_split(data, test\_size=0.5, random\_state=111)

val, test = train\_test\_split(test1, test\_size=0.4, random\_state=121)

print(len(train), len(val), len(test))

train\_x = train[feature\_remain]

train\_y = train['ID number']

val\_x = val[feature\_remain]

val\_y = val['ID number']

test\_x = test[feature\_remain]

test\_y = test['ID number']

# 对数据进行z-score归一化

ss = StandardScaler()

train\_x = ss.fit\_transform(train\_x)

val\_x = ss.fit\_transform(val\_x)

test\_x = ss.fit\_transform(test\_x)

# 创建SVM分类器

print("-------调参前---------")

clf = svm.SVC()

clf.fit(train\_x, train\_y)

predictions = clf.predict(val\_x)

print(confusion\_matrix(val\_y, predictions))

print(classification\_report(val\_y, predictions))

Kernel = ["linear" , "poly", "rbf", "sigmoid"]

print("-------选择核函数---------")

for kernel in Kernel:

model= svm.SVC(kernel=kernel).fit(train\_x, train\_y)

validation = model.predict(val\_x)

print(kernel, 'val acc: ', accuracy\_score(validation, val\_y))

# 选择rbf高斯通核函数

model = svm.SVC(kernel="rbf")

model.fit(train\_x, train\_y)

# 验证集

print('验证集')

predictions = model.predict(val\_x)

print(confusion\_matrix(val\_y, predictions))

print(classification\_report(val\_y, predictions))

param\_grid = {'C':[0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1,5,10,100,1000],

'gamma':[10,1,0.5,0.1,0.09,0.08,0.07,0.06,0.05,0.04,0.03,0.02,0.01,0.001,0.0001]}

kflod = StratifiedKFold(n\_splits=10, shuffle = True , random\_state=7)

grid\_model = GridSearchCV(svm.SVC(), param\_grid , cv = kflod, verbose=0)

grid\_model.fit(train\_x, train\_y)

print(grid\_model.best\_params\_)

print('验证集')

grid\_predictions\_train = grid\_model.predict(val\_x)

print(confusion\_matrix(val\_y, grid\_predictions\_train))

print(classification\_report(val\_y, grid\_predictions\_train))

# 测试集

print('测试集')

grid\_predictions\_test = grid\_model.predict(test\_x)

print(confusion\_matrix(test\_y, grid\_predictions\_test))

print(classification\_report(test\_y, grid\_predictions\_test))