人工智能课程实验一

姓名：叶宏庭

学号：71118415

邮箱：213182964@seu. edu. cn

2021 年 1 月 4 日

目录

一、学习任务

1 数据集一

数据集一为车辆信息数据集，提供特征属性包括购买价格、维修费用、容量等，要求根据提供的六个特征属性，对车辆进行分类，是一个四分类任务(unacc、acc、good、vgood)。

2 数据集二

数据集二为 sklearn 自带的数据集 iris（鸢尾花），提供特征属性包括花萼长度、花萼宽度、花瓣长度、花瓣宽度，特征值均为浮点数，这是一个三分类任务(Iris Setosa、Iris Versicolour、Iris Virginica)。

3 数据集三

数据集三为 sklearn 自带的数据集 digits（手写数字识别），提供的数字图像为 8*8 大小，因此共有 64 的特征属性，每个特征属性的取值均是 0-16 的 Integer，这是一个十分类的分类任务（[0-9]十个类别的数据）。

4 数据集四

数据集四为 sklearn 自带的数据集 Wine(红酒分类)，提供的特征属性包括红酒中的 13 种化学成分的含量，根据这些特征属性进行红酒的档次分类，这是一个三分类的任务。

二、性能指标度量

1 错误率与精度

错误率是分类错误的样本数占样本总数的比例，精度则是分类正确的样本数占样本总数的比例。对于样本集 D，分类错误率定义为

$$\mathrm{E(f;D)} = \frac{1}{m}\sum_{i=1}^{m} I(f(x_i) \neq y_i) \tag{1}$$

精度定义则为

$$\mathrm{acc(f;D)} = \frac{1}{m}\sum_{i=1}^{m} I(f(x_i) = y_i) = 1 - \mathrm{E(f;D)} \tag{2}$$

三、数据集信息

数据集名称	样本数	属性数	类别数	各类别样本数	
Car Evaluation	1728	6	4	unacc	1210
				acc	384
				good	69
				vgood	65
Iris	150	4	3	Setosa	50
				Versicolour	50
				Virginica	50
Digits	1797	64	10	[0-9]各类别均≈180个样本	
Wine	178	13	3	class_1	59
				class_2	71
				class_3	48

四、数据集预处理

1 数据集一

针对 Car Evaluation 数据集，采用 python 自行编码，实现 dataloader，根据属性标签进行划分，采用 train_test_split()进行随机划分，设置 test_size 属性为 0.1。

采用分类编码器，将文本类型的特征值转划为机器学习算法可处理的数字类型。

dataloader.py > preprocess_car

```python
from sklearn import preprocessing
import numpy as np
from sklearn.model_selection import train_test_split

def preprocess_car(data1, data2):
    tar = preprocessing.LabelEncoder().fit(['unacc', 'acc', 'good', 'vgood'])
    arr = preprocessing.OrdinalEncoder().fit([
        ['vhigh', 'vhigh', '2', '2', 'small', 'low'],
        ['high', 'high', '3', '4', 'med', 'med'],
        ['med', 'med', '4', 'more', 'big', 'high'],
        ['low', 'low', '5more', 'more', 'big', 'high']
    ])
    # print(tar.transform(['unacc', 'acc', 'good', 'vgood']))
    re_data1 = arr.transform(data1)
    re_data2 = tar.transform(data2).reshape([len(data1), 1])
    # print(re_data1[0:2])
    # print(re_data2[0:2])
    re_data = np.concatenate([re_data1, re_data2], axis=1)
    # print(re_data[1155:1165])
    # print(re_data[0:5])

    return re_data
```

```python
def dataload():

    train_x = []
    train_y = []

    test_x = []
    test_y = []

    all_data1 = []
    all_data2 = []

    with open("./dataset/car_data.txt", "r", encoding='utf-8') as f:
        reader = f.readlines()
        # print(reader)
        for i in reader:
            i = i.replace("\n", "")

            arrs = i.split(",")

            all_data1.append(arrs[:-1])
            all_data2.append(arrs[-1])

    redata = preprocess_car(all_data1, all_data2)

    x = redata[:, :-1]
    y = redata[:, -1:]

    return x, y
    train_x, test_x, train_y, test_y = train_test_split(x, y, test_size = 0.1)

    # print(len(train_x), train_y)
    return train_x, train_y, test_x, test_y

# dataload()
```

2 数据集二

该数据集为 sklearn 内置数据集，已经经过预处理操作，无缺损值等情况。下面给出数据集加载代码。

iris.py > ...

```python
from sklearn import datasets
import sklearn.tree as st
import sklearn.metrics as sm
from sklearn.model_selection import cross_validate
import numpy as np
from sklearn.neural_network import MLPClassifier
import sklearn.svm as svm

# iris：鸢尾花数据集：--> 用于分类
# 有150个数据集，共分为3类，每类50个样本。每个样本有4个特征。

# 加载数据集
iris = datasets.load_iris()  # 加载iris数据集

x, y = iris.data, iris.target

```

3 数据集三

该数据集为 sklearn 内置数据集，已经结果预处理操作，无缺损值等情况。下面给出数据集加载代码。

digit.py > ...

```python
from sklearn import datasets
import sklearn.tree as st
import sklearn.metrics as sm
from sklearn.model_selection import cross_validate
import numpy as np
from sklearn.neural_network import MLPClassifier
import sklearn.svm as svm

# Digits：Digits数据集：--> 用于分类
# 有1797个数据集，共分10类，每类约180个样本。每个样本有64个特征。

# 加载数据集
digits = datasets.load_digits()  # 加载Digits数据集

x, y = digits.data, digits.target

```

4 数据集四

该数据集为 sklearn 内置数据集，已经结果预处理操作，无缺损值等情况。下面给出数据集加载代码。

```python
from sklearn import datasets
import sklearn.tree as st
import sklearn.metrics as sm
from sklearn.model_selection import cross_validate
import numpy as np
from sklearn.neural_network import MLPClassifier
import sklearn.svm as svm

# wine：wine数据集：--> 用于分类
# 有178个数据集，共分3类，每类(59, 71, 48)个样本。每个样本有13个特征。

# 加载数据集
wine = datasets.load_wine()  # 加载Wine数据集

x, y = wine.data, wine.target
```

五、对比算法

1 对比算法

本实验采用三个算法进行实验对比，分别是决策树算法、MLP 多层感知机网络、SVM 算法。

2 参数设定

2.1 决策树算法

DecisionTreeClassifier()参数设定如下：

参数名	参数设定值	说明
criterion	gini	使用基尼系数作为评判标准
splitter	best	在所有特征中寻找最好切分点
max_features	None	采用所有特征
max_depth	7	最大深度为 7 层，经过测试 4-10 层深度，最终采取 7 层使得算法效果最好
其他参数	default	采用默认参数

2.2 MLP 多层感知机网络算法

MLPClassifier()参数设定如下：

参数名	参数设定值	说明
hidden_layer_sizes	（35，20）	第 i 个数表示第 i 层的神经元个数
activation	logistic	采用逻辑激活函数，即 sigmod()函数
solver	lbfgs	准牛顿方法族的优化器
alpha	0.0001	L2 正则化参数
其他参数	default	采用默认参数

2.3 SVM 算法

SVC（C-Support Vector Classification）参数设定如下：

参数名	参数设定值	说明
kernel	rbf	算法采用核函数类型
C	600	错误项的惩罚系数
其他参数	default	采用默认参数

3 算法来源

本实验算法均采用 sklearn 机器学习包，采用 python 语言进行实验。

六、实验方法

1 实验方法

本实验采用交叉验证法进行，设置参数 k = 10，即采取 10 折交叉验证法进行实验，最后取十次的平均准确率作为最终的性能指标。

2 假设检验方法

本实验采用 Friedman test 方法，对实验算法结果进行假设检验。

弗里德曼(Friedman test)检验即 “弗里德曼双向秩方差分析”，是多个(相关)样本齐一性的统计检验。 该方法是弗里德曼 (M.Friedman) 1973 年提出的。弗里德曼检验前提要求：1.顺序级数据；2.三个或更多组；3.相关的小组；4.从搭配的数值中随机地抽取样本。

具体检验过程参照《机器学习》(周志华) P42 - P43。

七、实验结果

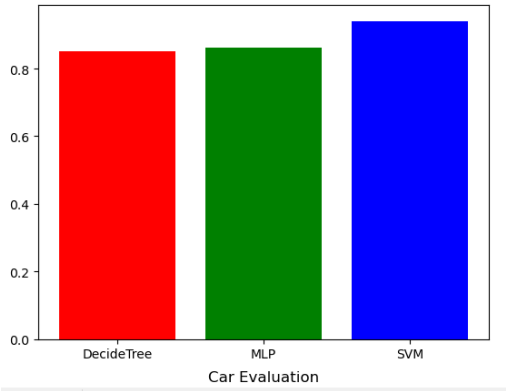
1 精度结果图表

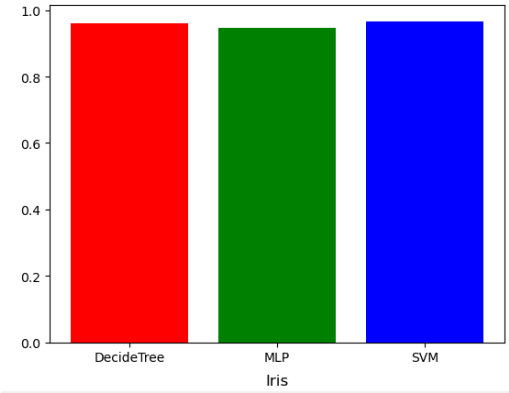
（因部分结果相接近，所以结果保留 5 位小数）

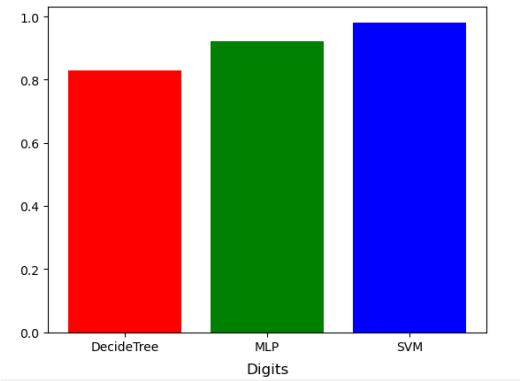
数据集	决策树	MLP 多层感知机网络	SVM 算法
Car Evaluation	0.85196	0.86174	0.94102
Iris	0.96000	0.94667	0.96667
Digits	0.82916	0.92207	0.98163
Wine	0.86470	0.81601	0.86013
部分统计量			
均值	0.87646	0.88662	0.93736
方差	0.00249	0.00262	0.00219

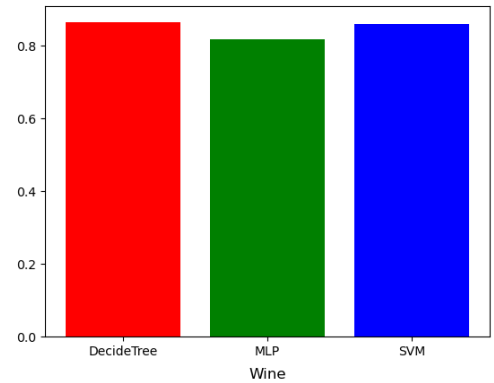
从精度结果来看，SVM 算法在多个数据集中都表现出了最好的分类精度结果，并且平均精度也取得了最高分数，其次是 MLP 多层感知机网络。

从精度结果的方差来看，SVM 算法最为稳定，方差最小，MLP 多层感知机网络的方差最大，结果最不稳定。









2 Friedman 检验

根据实验精度结果表，采用 Friedman 检验方法，构造出下面的检验表。

数据集	决策树	MLP 多层感知机网络	SVM 算法
Car Evaluation	3	2	1
Iris	2	3	1
Digits	3	2	1
Wine	1	3	2
平均值	2.25	2.5	1.25

从表中可以看出，SVM 的平均序值最小，MLP 的平均序值最大，由平均序值，下面进行假设检验。

下面进行假设检验部分，假设 “所有的算法性能均相同”。

$$\tau_{\chi^2} = \frac{k-1}{k} \cdot \frac{12N}{k^2-1} \cdot \sum_{i=1}^{k} \left(r_i - \frac{k+1}{2} \right)^2 = \frac{12N}{k(k+1)} \left(\sum_{i=1}^{k} r_i^2 - \frac{k(k+1)^2}{4} \right) \tag{3}$$

$$\tau_F = \frac{(N-1)\tau_{\chi^2}}{N(k-1) - \tau_{\chi^2}} \tag{4}$$

其中，N 表示数据集个数，k 表示算法个数，r_i表示第 i 个算法的平均序值。

在本实验中，经计算，$\tau_F = 2.333$ 。

因为，τ_F 服从自由度为$k-1$和$(k-1)(N-1)$的 F 分布。根据查表可得，$\mathrm{F}(4,3) = 5.143$，所以$\tau_F < \mathrm{F}(4,3)$，因此可认为三个算法的性能大致相同。