

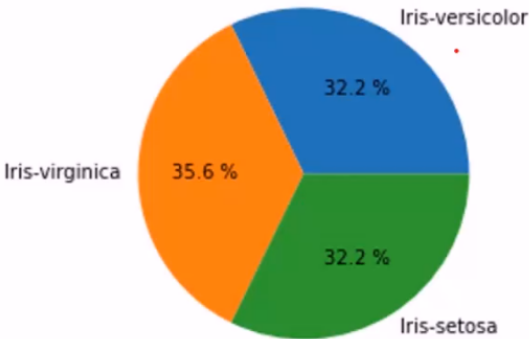
机器学习 上机1 鸢尾花分类

姓名： 张泽群 学号： 19049100002 班级： 2班

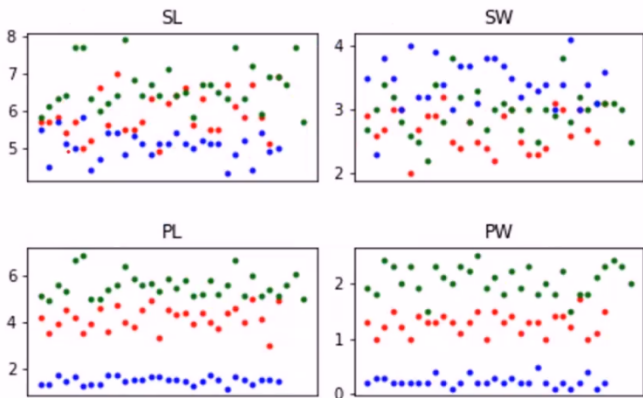
1. 数据集分析与预处理

1.1 数据集分析

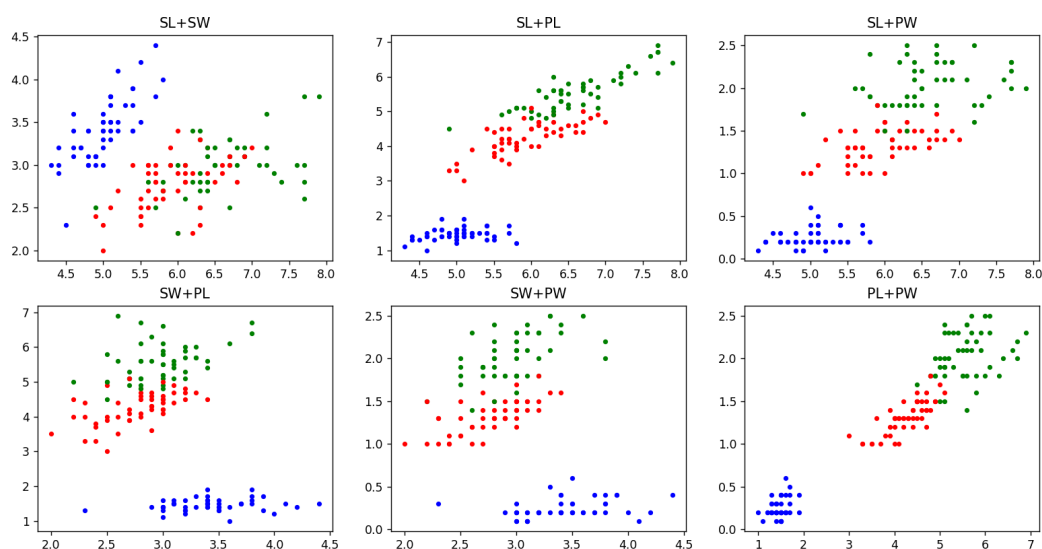
本次实验所用数据集是鸢尾花数据集，数据具有萼片长度、萼片宽度、花瓣长度和花瓣宽度四个数据特征以及3个类别，这三个类别在样本数据集中的分布如下图所示，较为均衡，因此无需做出调整样本比列的数据处理操作。其中，数据集划分为训练、验证、测试集，样本数量分别为90、30、30。



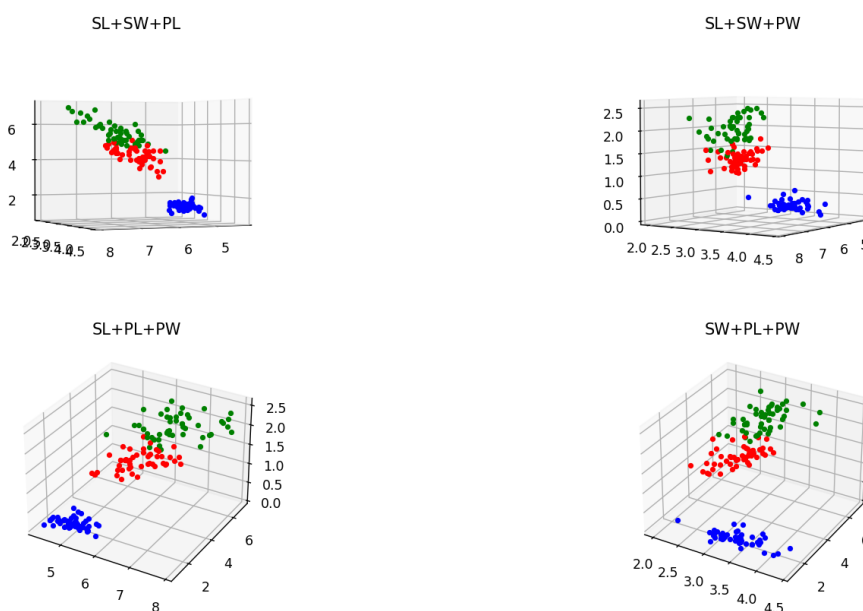
对于数据特征,对于单一维度的特征，我们可以看到三种类别在萼片长度和萼片宽度区分得不是很好，而在花瓣长度和花瓣宽度区分得较好。



对于组合两个特征维度的六种情况，我们可以看到三种类别在除了SL+SW外的其余组合都有明显的区别度，但仍然有明显的边界混合情况。



对于组合三个特征维度的四种情况，通过旋转坐标轴，我们可以看到三种类别在所有组合都有明显的区别度，可见数据集本身的不同类别的差异较好，易于得到较好的分类效果。



1.2 数据预处理

首先，通过1.1中的数据集分析可以看出，原始数据集的不同类别的差异较好，因此无需进行数据标准化以及特征选择也能得到较好的效果。其次，在编程实现的过程中发现进行0-1标准化反而会降低分类预测的精准度，因此本次实验没有进行特殊的数据预处理。

2. 模型训练

首先应用的模型是SVM模型，为此调用了sklearn库中的svm类，并且建立支持向量分类模型SVC，该模型适用于当样例数少于10000时的二元和多元分类。

其主要需要调整的超参数如下：

- **C** 误差项的惩罚参数，C越大，相当于惩罚松弛变量，希望松弛变量接近0，即对误分类的惩罚增大，趋向于对训练集全分对的情况，这样会出现训练集测试时准确率很高，但泛化能力弱；C值小，对误分类的惩罚减小，容错能力增强，泛化能力较强。
- **kernel** svc中指定核函数的类型，可以是：'linear'，'poly'，'rbf'，'sigmoid'，'precomputed' 或者自己指定。默认使用'rbf'。
- **gamma**：kernel='rbf' 时，为高斯核函数，gamma值越小，分类界面越连续；gamma值越大，分类界面越“散”，分类效果越好，但有可能会过拟合。

基于SVC模型，其训练的主要过程如下：

第一种方法是利用for循环进行查找，此种方法实现简单，但效率和全面性较低。此处指定了两个超参数kernel，C以及调整的范围，找出在验证集上具有最高准确率的超参数。

```
# 最优超参数选择(网络搜索) for循环
kernel = ['linear', 'rbf', 'poly', 'sigmoid']
C = np.arange(0.01, 1.01, 0.01)

max_val_score = 0 # 选取验证集上具有最高准确率的超参数
p_1, p_2 = 0, 0

for i in range(len(kernel)):
    for j in range(100):
        model = SVC(kernel=kernel[i], C=C[j])
        model.fit(train_data, train_label)
        val_pred = model.predict(validation_data)
        val_score = metrics.accuracy_score(validation_label, val_pred)
        if val_score > max_val_score:
            max_val_score = val_score
            p_1, p_2 = i, j

print()
print('网络搜索 for循环法')
print('kernel = ', kernel[p_1], ' C = ', C[p_2])
print('Validation Set Accuracy = ', max_val_score)
```

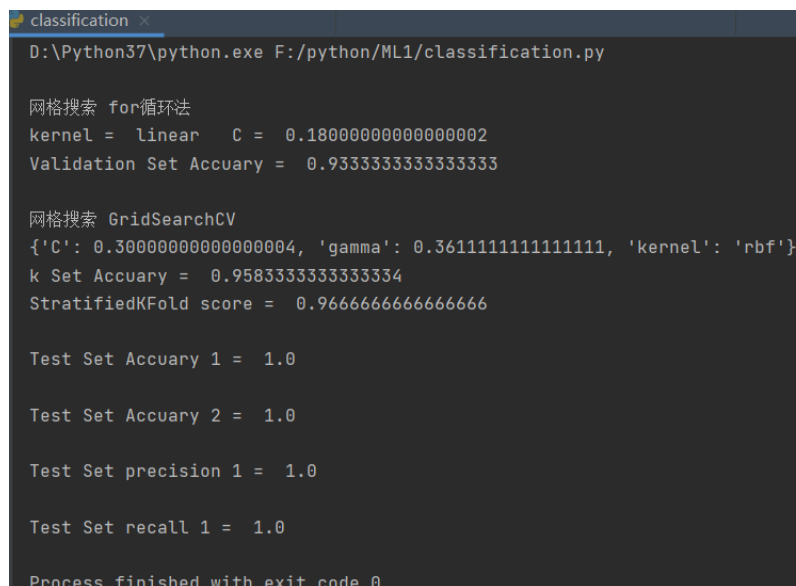
第二种方法是利用sklearn库的GridSearchCV函数进行超参数的调整，此种方法相对来说效率和全面性更高。此处指定了三个超参数kernel, C, gamma以及调整的范围，结合验证集与数据集，利用交叉验证选择最优超参数。

```
print('网格搜索 GridSearchCV') # 通过交叉验证确定最佳效果参数
# 超参数
parameters = {'kernel': ['linear', 'rbf', 'poly', 'sigmoid'], 'C':
np.linspace(0.1, 1, 10), 'gamma': np.linspace(0.25, 1.25, 10)}
svc = SVC()
# 结合验证集与数据集，利用交叉验证选择参数
k_data = np.vstack((validation_data, train_data))
k_label = np.concatenate((validation_label, train_label))
# 模型训练
grid = GridSearchCV(svc, parameters, cv=10, scoring='accuracy')
grid.fit(k_data, k_label)
print(grid.best_params_)
model1 = grid.best_estimator_
k_pred1 = model1.predict(k_data)
print('k Set Accuracy = ', metrics.accuracy_score(k_label, k_pred1))
print('StratifiedKFold score = ', grid.best_score_) # 平均交叉验证分数
```

利用此训练好的模型对测试集进行预测，所得的精确度为1，说明测试集上的所有样本都被成功的预测了；除了SVM模型，应用其他模型进行测试仍然得到了精确度为1的结果，因此在此处就不多赘述，这一点可能是因为数据集的样本分布较为均匀，不同类别在数据特征的差别较为明显，使得分类的精确度非常之高。

3. 实验结果

运行结果：



```
classification x
D:\Python37\python.exe F:/python/ML1/classification.py

网格搜索 for循环法
kernel = linear C = 0.18000000000000002
Validation Set Accuracy = 0.9333333333333333

网格搜索 GridSearchCV
{'C': 0.30000000000000004, 'gamma': 0.3611111111111111, 'kernel': 'rbf'}
k Set Accuracy = 0.9583333333333334
StratifiedKFold score = 0.9666666666666666

Test Set Accuracy 1 = 1.0

Test Set Accuracy 2 = 1.0

Test Set precision 1 = 1.0

Test Set recall 1 = 1.0

Process finished with exit code 0
```

这里显示了使用两种网格搜索调整超参数的方式，第一种for循环法所得的超参数在验证集上的准确率为0.933，也就是说有2个类别预测不正确；第二种利用sklearn的函数GridSearchCV，选择调整了C，gamma和kernel三个参数，最终获得的超参数在验证集上的准确率提升到0.966，也就是说只有1个类别预测不正确。

但是使用两种方法在测试集的精确度都能达到 $Accuracy = 1$ ，这一点可能是因为数据集的样本分布较为均匀，不同类别在数据特征的差别较为明显。

同时利用joblib导出模型为model_1.model。

```
# 导出模型
joblib.dump(model1, 'model_1.model')
```

4. 讨论与结论

4.1 评价指标的讨论

分类算法中常用的评价指标是：accuracy、precision、recall。

(1). 准确率 (accuracy)

分类器**正确分类的样本数** 与 **总样本数**之比。

$$accuracy = \frac{TP+TN}{P+N}$$

(2). 精准率 (precision)

模型预测对的正样本个数 与 **模型预测出的正样本数** 之比。

$$precision = \frac{TP}{TP+FP}$$

(3). 召回率 (recall)

模型预测对的正样本数 与 **所有正样本数** 之比

$$recall = \frac{TP}{P}$$

其中 precision 和 recall 衡量机器学习模型性能的重要指标，特别是**数据集分布不平衡**的案例中。