数据科学 实验二

何尉宁 2021213599 2021219106班

任务列表

必做：
复现课件中线性SVM、决策树、朴素贝叶斯分类的示例，并相对课件代码作出如下作图修改

- 设定支持向量分类器的惩罚为0.05
- 对朴素贝叶斯分类器的先验概率进行设定（可随机设定）
- 在每张结果图上展示图例
- 修改散点颜色为黄和绿

测试结果的正确率保留三位小数展示

选做：
自主选取其他的数据集，采用上述三类分类器进行分类，展示分类结果
探究分类器的参数对于分类结果的影响并进行文字分析（选做）
如：

- DecisionTreeClassifier(max_depth=5)中max_depth设置对于结果的影响（如过拟合或者欠拟合）
- 朴素贝叶斯分类器的先验概率修改对于分类的影响
- 支持向量分类器不同核函数对于结果的影响

参数不限制于课件中代码所用到的参数，可以探究其他的参数
其他分类方法的效果的对比分析（K近邻，随机森林等）

1. 代码复现

1.1 修改惩罚系数与先验概率

1. Set the penalty to 0.5
2. Set the class prior to a random number

```
names = ["Linear_SVM","Decision_Tree","Naive_Bayes"]
# 设置随机数用于先验概率
random_number = np.random.randint(0,100)
random_number /= 100
print(random_number)

classifiers = [
    SVC(kernel = "linear", C = 0.05),
    DecisionTreeClassifier(random_state = 44, max_depth = 5),
    GaussianNB(priors=[random_number, 1-random_number]),
]
```

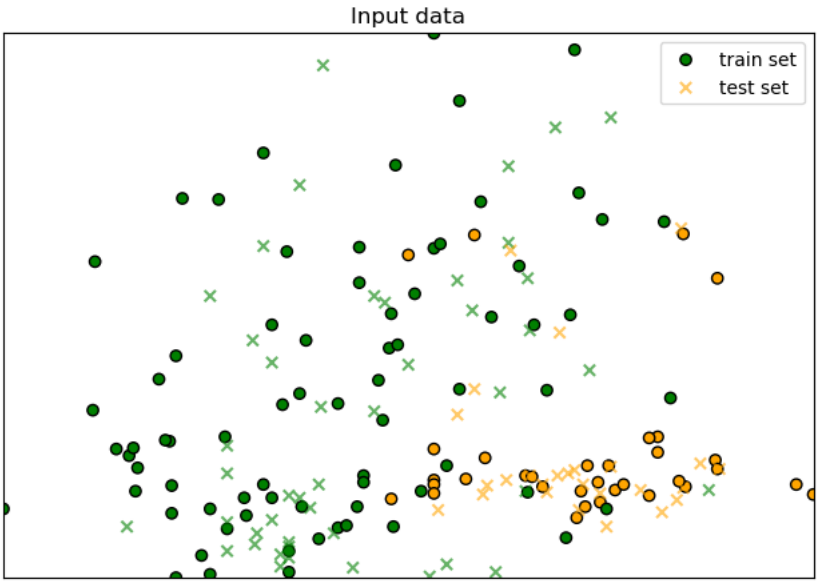
1.2 随机噪声

1. RandomNoise
2. GaussianNoise

```
X,y=make_classification(n_features=2,n_redundant=0,n_informative=2,
                        random_state=1,n_clusters_per_class=1)

random_num = np.random.RandomState(5) # 设置一个伪随机数种子
# 随机扰动噪声
RandomNoise = random_num.uniform(low = -1, high = 1, size = X.shape)
# 高斯噪声
GaussianNoise = random_num.normal(loc = 0, scale = 1, size = X.shape)

# X += 2*RandomNoise
X += 2*GaussianNoise

linearly_separable=(X,y) # 将上述得到的x，y够作为一个线性可分的数据集

datasets=[make_moons(noise=0.1,random_state=np.random.RandomState(5)),
          make_circles(noise=0.1,factor=0.5,random_state=1),
          linearly_separable
          ]
```

1.3 绘图

修改了主题颜色与图案



2. 选做

数据集介绍

Wine

> Donated on 6/30/1991
> Using chemical analysis to determine the origin of wines

Dataset Characteristics Tabular
Subject Area Physics and Chemistry
Associated Tasks Classification
Feature Type Integer, Real
Instances 178
Features 13

2.2 数据处理

```
wine_data = pd.DataFrame(wine_data)
wine_data.columns = ["Class","Alcohol","Malic acid","Ash","Alcalinity of ash","Magnesium","Total phenols",
                     "Flavanoids","Nonflavanoid phenols","Proanthocyanins","Color intensity","Hue",
                     "OD280/OD315 of diluted wines","Proline"]
wine_data.iloc[:,1:] = StandardScaler().fit_transform(wine_data.iloc[:,1:])
wine_data.head()
```

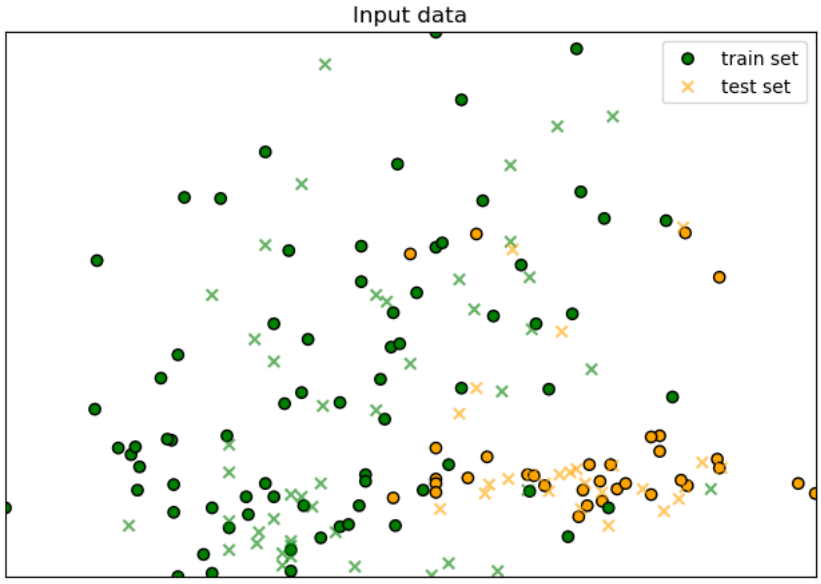
```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	Class	Alcohol	Malic acid	Ash	Alcalinity of ash	Magnesium	Total phenols	Flavanoids	Nonflavanoid phenols	Proanthocyanins	Color intensity	Hue
0	1.0	1.518613	-0.562250	0.232053	-1.169593	1.913905	0.808997	1.034819	-0.659563	1.224884	0.251717	0.36217
1	1.0	0.246290	-0.499413	-0.827996	-2.490847	0.018145	0.568648	0.733629	-0.820719	-0.544721	-0.293321	0.40605
2	1.0	0.196879	0.021231	1.109334	-0.268738	0.088358	0.808997	1.215533	-0.498407	2.135968	0.269020	0.31830
3	1.0	1.691550	-0.346811	0.487926	-0.809251	0.930918	2.491446	1.466525	-0.981875	1.032155	1.186068	-0.4275
4	1.0	0.295700	0.227694	1.840403	0.451946	1.281985	0.808997	0.663351	0.226796	0.401404	-0.319276	0.36217

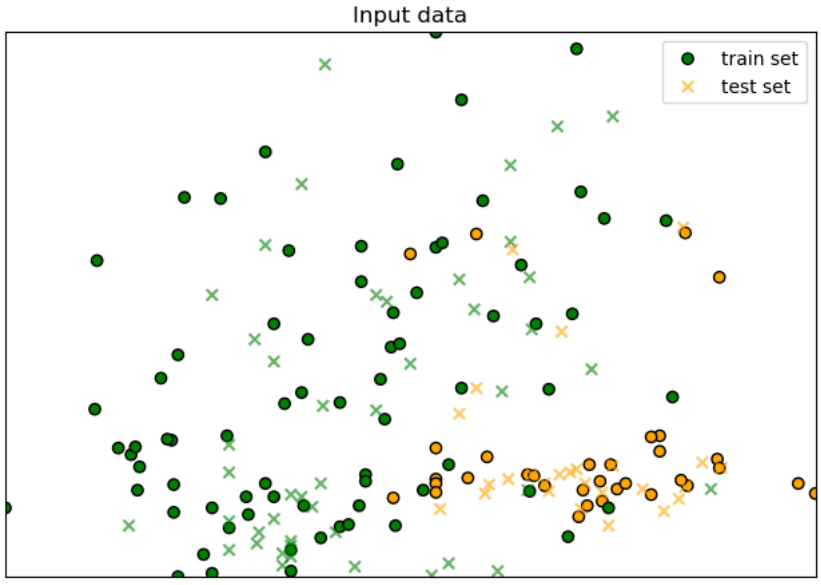
2.3 不同深度的决策树

因为这次选择的wine数据集本身规模不大，所以在深度选择上只使用了1和10来进行对比。
1: 明显欠拟合，score非常低
10: 拟合很好，基本上已经到达score上限

Max_Depth = 1
Decision Tree score: 0.625



Max_Depth = 10
Decision Tree score: 0.9444444444444444



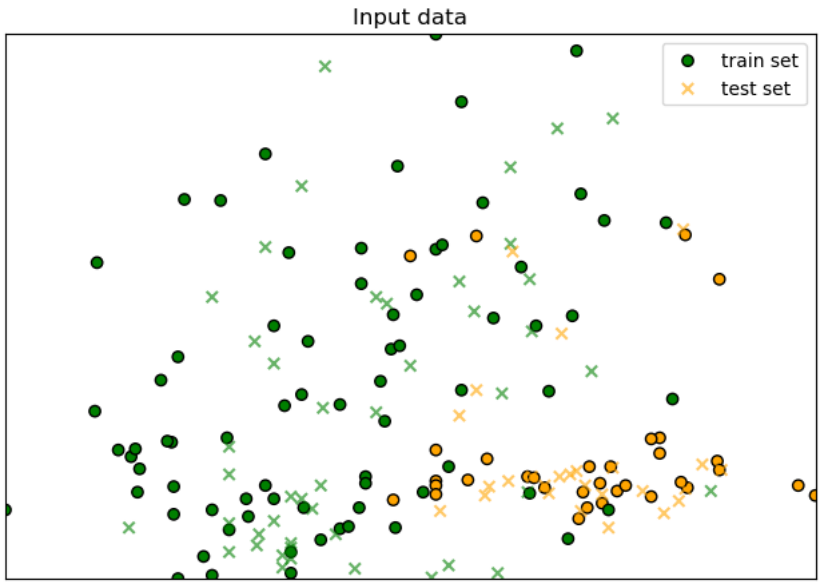

```
# 使用决策树进行分类
X = wine_data.iloc[:,1:]
y = wine_data.iloc[:,0]
X_train, X_test, y_train, y_test = \
        train_test_split(X, y, test_size=0.4, random_state=42)
clf = DecisionTreeClassifier(random_state = 44, max_depth = 1)
clf.fit(X_train, y_train)
score = clf.score(X_test, y_test)
print("Decision Tree score: ", score)

figure = plt.figure(figsize=(36, 18))
i = 1
x_min, x_max = X_train.iloc[:,0].min() - 0.5, X_train.iloc[:,0].max() + 0.5
y_min, y_max = X_train.iloc[:,1].min() - 0.5, X_train.iloc[:,1].max() + 0.5
h = 0.02
xx, yy = np.meshgrid(
    np.linspace(X_train.iloc[:,0].min(), X_train.iloc[:,0].max(), 100),
    np.linspace(X_train.iloc[:,1].min(), X_train.iloc[:,1].max(), 100)
)
cm = ListedColormap((['yellow', 'green']))
```

```
cm_bright = ListedColormap(['#FFA500', '#008000'])

ax = plt.subplot(len(datasets), len(classifiers) + 1, i)
ax.set_title("Input data")
ax.scatter(X_train.iloc[:,0], X_train.iloc[:,1], c=y_train, cmap=cm_bright,
           edgecolors='k', marker='o', label='train set')
ax.scatter(X_test.iloc[:,0], X_test.iloc[:,1], c=y_test, cmap=cm_bright, alpha=0.6,
           facecolor='k', marker='x', label='test set')
ax.set_xlim(xx.min(), xx.max())
ax.set_ylim(yy.min(), yy.max())
ax.set_xticks(())
ax.set_yticks(())
plt.legend()
```

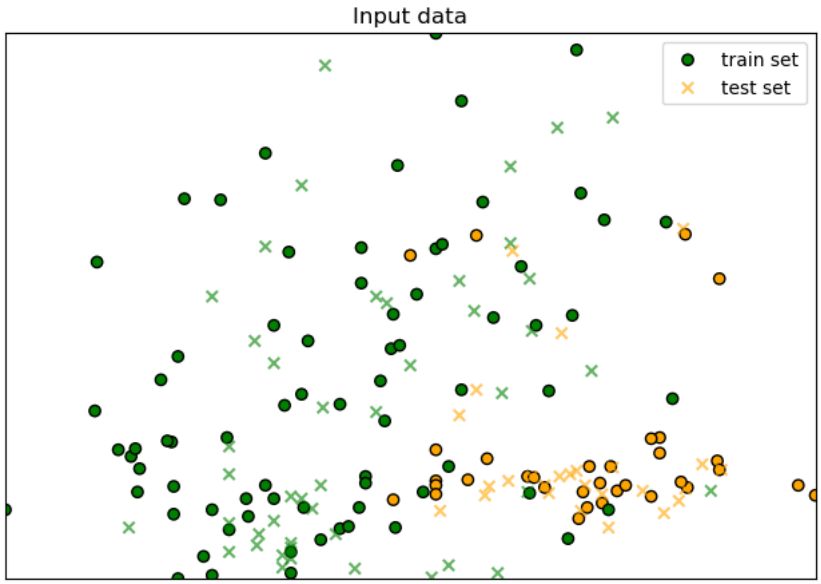
2.4 朴素贝叶斯中不同先验概率的选择

选取了features中相关系数最高的两项特征进行分类，选取default先验概率与[0.2, 0.3, 0.5]两种

priors = defalut
Naive Bayes score: 0.7777777777777778



priors = [0.2, 0.3, 0.5]
Naive Bayes score: 0.8194444444444444



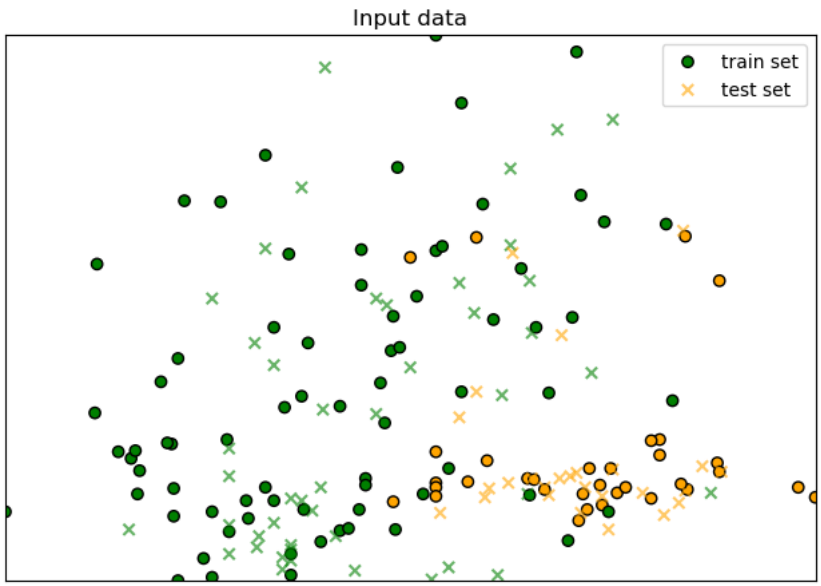

```
wine = np.loadtxt(file_path + "/wine.data", delimiter=",")
wine = pd.DataFrame(wine)
wine.columns = ["Class","Alcohol","Malic acid","Ash","Alcalinity of ash","Magnesium","Total phenols",
                "Flavanoids","Nonflavanoid phenols","Proanthocyanins","Color intensity","Hue",
                "OD280/OD315 of diluted wines","Proline"]
```

```
selected_features = ["Alcohol", "Malic acid"]
wine_subset = wine[["Class"] + selected_features]

X = wine_subset[selected_features]
y = wine_subset["Class"]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=42)

clf = GaussianNB(priors=[0.2, 0.3, 0.5])
clf.fit(X_train, y_train)
score = clf.score(X_test, y_test)
print("Naive Bayes score: ", score)

figure = plt.figure(figsize=(36, 18))
i = 1
x_min, x_max = X_train.iloc[:, 0].min() - 0.5, X_train.iloc[:, 0].max() + 0.5
y_min, y_max = X_train.iloc[:, 1].min() - 0.5, X_train.iloc[:, 1].max() + 0.5
h = 0.02
xx, yy = np.meshgrid(
    np.linspace(X_train.iloc[:, 0].min(), X_train.iloc[:, 0].max(), 100),
    np.linspace(X_train.iloc[:, 1].min(), X_train.iloc[:, 1].max(), 100)
)
cm = ListedColormap((['yellow', 'green']))
cm_bright = ListedColormap(['#FFA500', '#008000'])

ax = plt.subplot(len(datasets), len(classifiers) + 1, i)
ax.set_title("Input data")
ax.scatter(X_train.iloc[:, 0], X_train.iloc[:, 1], c=y_train, cmap=cm_bright,
           edgecolors='k', marker='o', label='train set')
ax.scatter(X_test.iloc[:, 0], X_test.iloc[:, 1], c=y_test, cmap=cm_bright, alpha=0.6,
           facecolor='k', marker='x', label='test set')
ax.set_xlim(xx.min(), xx.max())
ax.set_ylim(yy.min(), yy.max())
ax.set_xticks(())
ax.set_yticks(())
plt.legend()
```

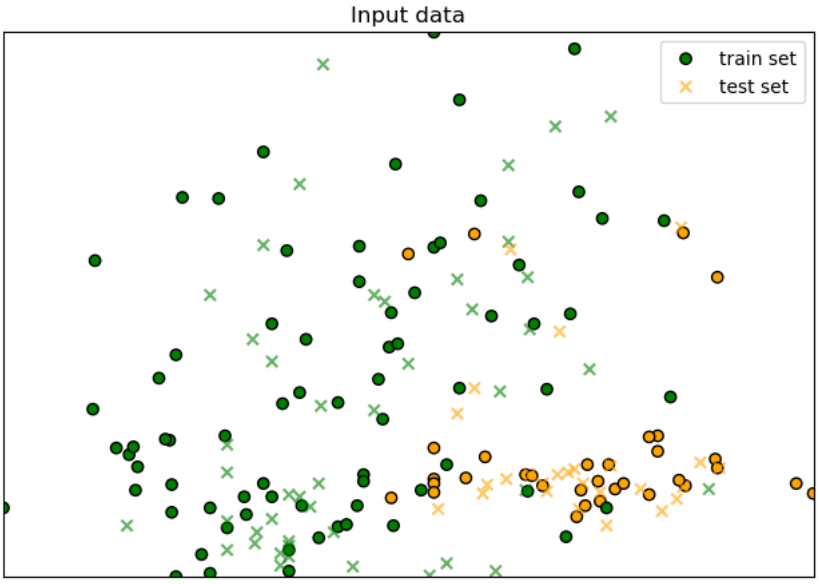
2.5 SVC不同kernel选择

分别使用了linear和poly核函数去拟合特征分布，但多项式的结果较为一般，因为这个简单分类问题基本上是线性的，多项式有点多此一举了。
同时**需要修改惩罚系数，否则准确率会大幅下降，非常欠拟合**

linear
SVC score: 0.9722222222222222



poly
SVC score: 0.9027777777777778




```
# 使用SVC进行分类
X = wine_data.iloc[:,1:]
y = wine_data.iloc[:,0]
X_train, X_test, y_train, y_test = \
        train_test_split(X, y, test_size=0.4, random_state=42)
clf = SVC(kernel = "poly", C = 1)
clf.fit(X_train, y_train)
score = clf.score(X_test, y_test)
print("SVC score: ", score)

figure = plt.figure(figsize=(36, 18))
i = 1
x_min, x_max = X_train.iloc[:,0].min() - 0.5, X_train.iloc[:,0].max() + 0.5
y_min, y_max = X_train.iloc[:,1].min() - 0.5, X_train.iloc[:,1].max() + 0.5
h = 0.02
xx, yy = np.meshgrid(
    np.linspace(X_train.iloc[:,0].min(), X_train.iloc[:,0].max(), 100),
    np.linspace(X_train.iloc[:,1].min(), X_train.iloc[:,1].max(), 100)
)
cm = ListedColormap((['yellow', 'green']))
cm_bright = ListedColormap(['#FFA500', '#008000'])

ax = plt.subplot(len(datasets), len(classifiers) + 1, i)
ax.set_title("Input data")
ax.scatter(X_train.iloc[:,0], X_train.iloc[:,1], c=y_train, cmap=cm_bright,
           edgecolors='k', marker='o', label='train set')
ax.scatter(X_test.iloc[:,0], X_test.iloc[:,1], c=y_test, cmap=cm_bright, alpha=0.6,
           facecolor='k', marker='x', label='test set')
ax.set_xlim(xx.min(), xx.max())
ax.set_ylim(yy.min(), yy.max())
ax.set_xticks(())
ax.set_yticks(())
plt.legend()
```