

鸢尾花(iris)数据集分析

数据集简介



Iris 鸢尾花数据集是一个经典数据集，在统计学习和机器学习领域都经常被用作示例。数据集内包含 3 类共 150 条记录，每类各 50 个数据，每条记录都有 4 项特征：

花萼长度 (Sepal-Length) 、花萼宽度 (Sepal-Width)

花瓣长度 (Petal-Length) 、花瓣宽度 (Petal-Width)

可以通过这 4 个特征预测鸢尾花卉属于 (**iris-setosa**、**iris-versicolour**、**iris-virginica**) 中的哪一品种。

1 准备数据

*****切换到对应环境安装以下包，包安装教程在 [anaconda 配置文档](#)中*****

Scikit-learn: 基于 python 的开源机器学习工具包

Matplotlib: matplotlib 是 python 的一个绘图库，与 numpy、pandas 共享数据科学三剑客的美誉

Seaborn: 一个基于 matplotlib 进行高级封装的可视化库，相比之下，绘制图表更为集成化、绘图风格具有更高的定制性

2 探索性分析（基于图形可视化）

下面对 iris 进行探索性分析，首先导入相关包和数据集：

```
import numpy as np
import pandas as pd
from pandas import plotting
import matplotlib.pyplot as plt
plt.style.use('seaborn')
import seaborn as sns
sns.set_style("whitegrid")
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn import svm
from sklearn import metrics
# 导入数据集 把 Iris.csv 放入同一路径内
iris = pd.read_csv('Iris.csv', usecols=[1, 2, 3, 4, 5])
```

查看数据集信息：

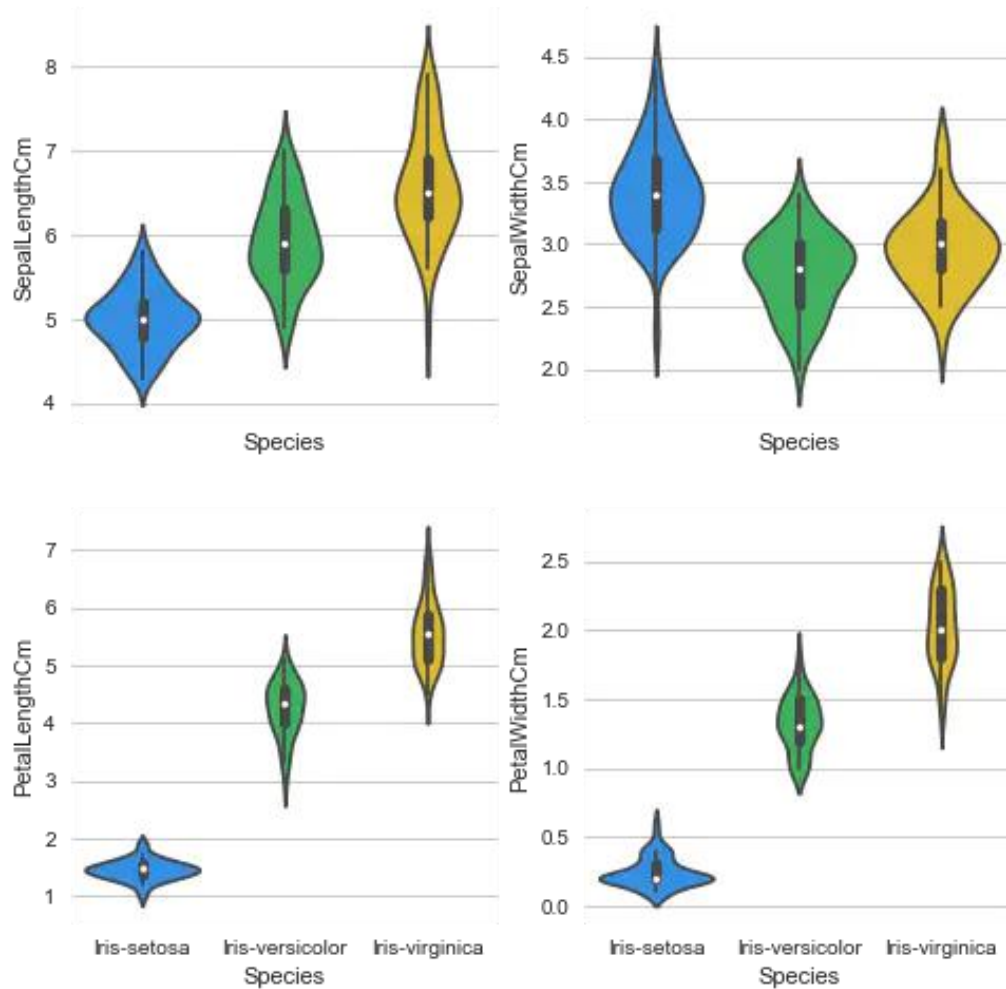
iris.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
#   Column      Non-Null Count  Dtype  
---  -
0    0           150 non-null    float64
1    1           150 non-null    float64
2    2           150 non-null    float64
3    3           150 non-null    float64
4    4           150 non-null    object  
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

2.1 通过 Violin plot (小钢琴图) 和 Pointplot (点图) , 分别从数据分布和斜率, 观察各特征与品种之间的关系:

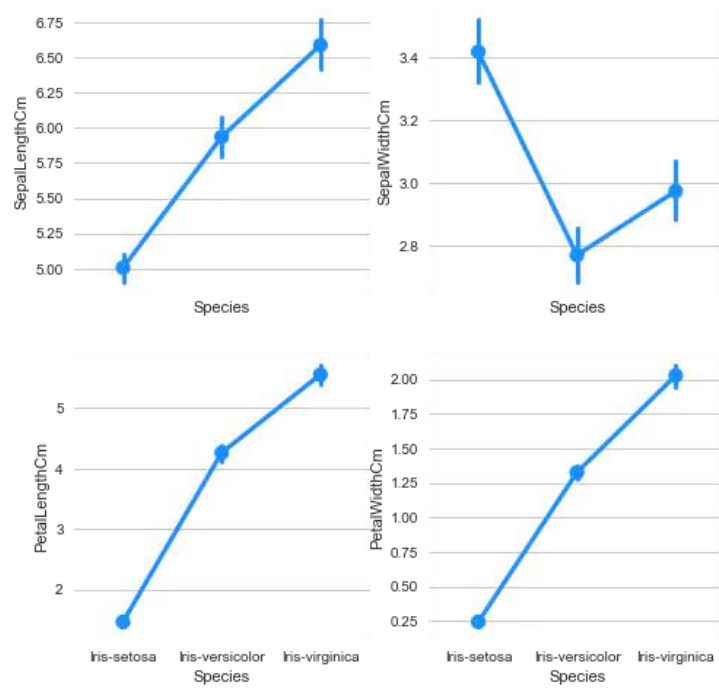
```
# 设置颜色主题
antV = ['#1890FF', '#2FC25B', '#FACC14']
# 绘制 Violinplot 小钢琴图
f, axes = plt.subplots(2, 2, figsize=(8, 8), sharex=True)
sns.despine(left=True)
sns.violinplot(x='Species', y='SepalLengthCm', data=iris, palette=antV,
ax=axes[0, 0])
sns.violinplot(x='Species', y='SepalWidthCm', data=iris, palette=antV,
ax=axes[0, 1])
sns.violinplot(x='Species', y='PetalLengthCm', data=iris, palette=antV,
ax=axes[1, 0])
sns.violinplot(x='Species', y='PetalWidthCm', data=iris, palette=antV,
ax=axes[1, 1])

plt.show()
```



显示数据分布及其概率密度

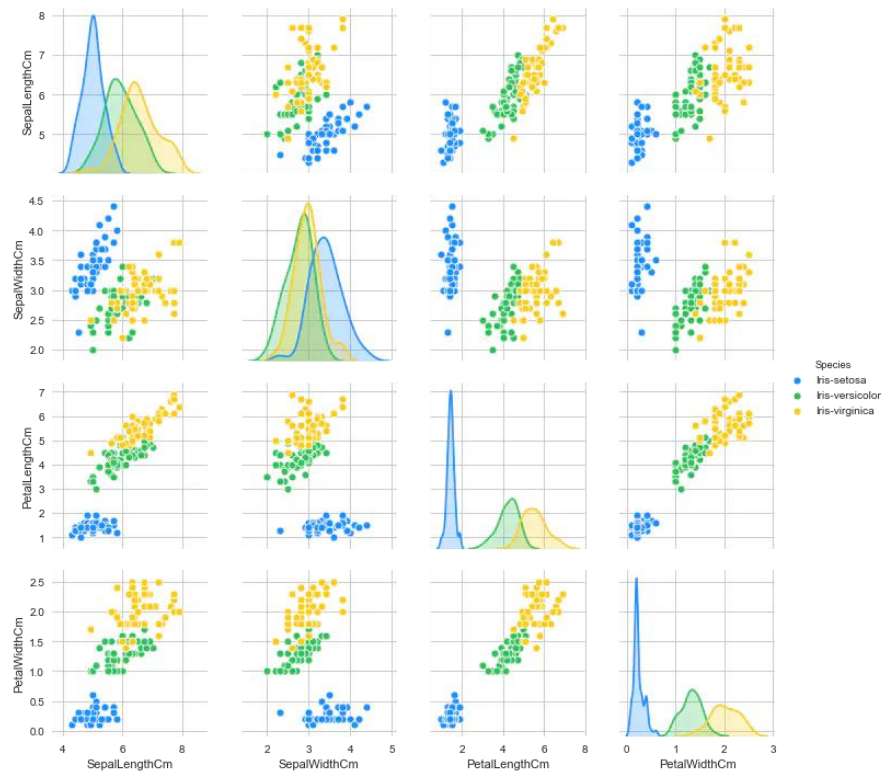
```
# 绘制 pointplot 点图
f, axes = plt.subplots(2, 2, figsize=(8, 8), sharex=True)
sns.despine(left=True)
sns.pointplot(x='Species', y='SepalLengthCm', data=iris, color=antV[0],
ax=axes[0, 0])
sns.pointplot(x='Species', y='SepalWidthCm', data=iris, color=antV[0],
ax=axes[0, 1])
sns.pointplot(x='Species', y='PetalLengthCm', data=iris, color=antV[0],
ax=axes[1, 0])
sns.pointplot(x='Species', y='PetalWidthCm', data=iris, color=antV[0],
ax=axes[1, 1])
plt.show()
```



聚焦一个或多个分类变量的不同级别之间的比较

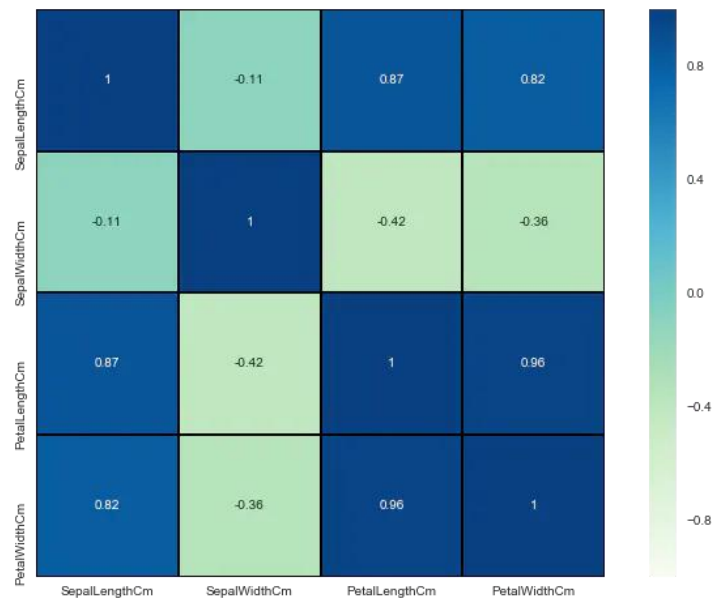
2.2 生成各特征之间关系的矩阵图：

```
sns.pairplot(data=iris, palette=antV, hue= 'Species')
plt.show()
```



2.3 最后，通过热图找出数据集中不同特征之间的相关性，高正值或负值表明特征具有高度相关性：

```
fig=plt.gcf()
fig.set_size_inches(12, 8)
fig=sns.heatmap(iris.corr(), annot=True, cmap='GnBu', linewidths=1,
linecolor='k', square=True, mask=False, vmin=-1, vmax=1,
cbar_kws={"orientation": "vertical"}, cbar=True)
plt.show()
```



从热图可看出，花萼的宽度和长度不相关，而花瓣的宽度和长度则高度相关。

任务 1：使用 seaborn 下的 Implot()方法分别基于花萼和花瓣做线性回归的可视化

(参数: data=iris, x="", y="", palette=antV, hue='Species')

2.4 接下来，通过机器学习，以花萼和花瓣的尺寸为根据，预测其品种

在进行机器学习之前，将数据集拆分为训练和测试数据集。首先，使用标签编码将 3 种鸢尾花的品种名称转换为分类值 (0, 1, 2) 。

```
# 载入特征和标签集
X = iris[['SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm',
          'PetalWidthCm']]
y = iris['Species']
# 对标签集进行编码
encoder = LabelEncoder()
y = encoder.fit_transform(y)
# 查看编码后的标签值
# print(y)
```

接着，将数据集以 7: 3 的比例，拆分为训练数据和测试数据：

```
train_X, test_X, train_y, test_y = train_test_split(X, y, test_size = 0.3,
                                                    random_state = 101)
print(train_X.shape, train_y.shape, test_X.shape, test_y.shape)
```

检查不同模型的准确性：

```
# Support Vector Machine
model = svm.SVC()
model.fit(train_X, train_y)
prediction = model.predict(test_X)
print('The accuracy of the SVM is:
{0}'.format(metrics.accuracy_score(prediction, test_y)))

# Logistic Regression
```

```
model = LogisticRegression()
model.fit(train_X, train_y)
prediction = model.predict(test_X)
print('The accuracy of the Logistic Regression is:
{0}'.format(metrics.accuracy_score(prediction,test_y)))
```

上面使用了数据集的所有特征，下面将分别使用花瓣和花萼的尺寸：

```
petal = iris[['PetalLengthCm', 'PetalWidthCm', 'Species']]
train_p,test_p=train_test_split(petal,test_size=0.3,random_state=0)
train_x_p=train_p[['PetalWidthCm','PetalLengthCm']]
train_y_p=train_p.Species
test_x_p=test_p[['PetalWidthCm','PetalLengthCm']]
test_y_p=test_p.Species
```

```
sepal = iris[['SepalLengthCm', 'SepalWidthCm', 'Species']]
train_s,test_s=train_test_split(sepal,test_size=0.3,random_state=0)
train_x_s=train_s[['SepalWidthCm','SepalLengthCm']]
train_y_s=train_s.Species
test_x_s=test_s[['SepalWidthCm','SepalLengthCm']]
test_y_s=test_s.Species
```

```
# Support Vector Machine
model=svm.SVC()
model.fit(train_x_p,train_y_p)
prediction=model.predict(test_x_p)
print('The accuracy of the SVM using Petals is:
{0}'.format(metrics.accuracy_score(prediction,test_y_p)))
```

```
model.fit(train_x_s,train_y_s)
prediction=model.predict(test_x_s)
print('The accuracy of the SVM using Sepal is:
{0}'.format(metrics.accuracy_score(prediction,test_y_s)))
```



```
# Logistic Regression
model = LogisticRegression()
model.fit(train_x_p, train_y_p)
prediction = model.predict(test_x_p)
print('The accuracy of the Logistic Regression using Petals is:
{0}'.format(metrics.accuracy_score(prediction,test_y_p)))

model.fit(train_x_s, train_y_s)
prediction = model.predict(test_x_s)
print('The accuracy of the Logistic Regression using Sepals is:
{0}'.format(metrics.accuracy_score(prediction,test_y_s)))
```

从中不难看出，使用花瓣的尺寸来训练数据较花萼更准确。正如在探索性分析的热图中所看到的那样，花萼的宽度和长度之间的相关性非常低，而花瓣的宽度和长度之间的相关性非常高。

3 特征选择（基于数理统计）

导入相关包和数据集

```
import pandas as pd
import numpy as np
from sklearn.feature_selection import VarianceThreshold
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
```

```
iris = pd.read_csv('Iris.csv')
```

载入特征和标签集

```
X = iris[['SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm',
'PetalWidthCm']]
```

```
y = iris['Species']
# 对标签集进行编码 标签编码将 3 种鸢尾花的品种名称转换为分类值 (0, 1, 2)
encoder = LabelEncoder()
y = encoder.fit_transform(y)
# print(y)
```

```
train_X, test_X, train_y, test_y = train_test_split(X, y, test_size=0.3,
random_state=42)
print(train_X.shape, train_y.shape, test_X.shape, test_y.shape)
```

打印数据集中的特征数和每个特征的方差

```
print('原数据集中的特征数: \n', X.shape[1])
print('原数据集中不同特征的方差: \n', np.var(X, axis=0), '\n')
```

使用 VarianceThreshold 来过滤掉方差在 0.6 以下的特征

```
selector = VarianceThreshold(threshold=0.6)
X_new = selector.fit_transform(X)
```

打印新数据集的特征数

```
print('方差阈值法选择的特征数: \n', X_new.shape[1])
print('新数据集中不同特征的方差: \n', np.var(X_new, axis=0), '\n')
```

```
model = LogisticRegression()
```

原数据集

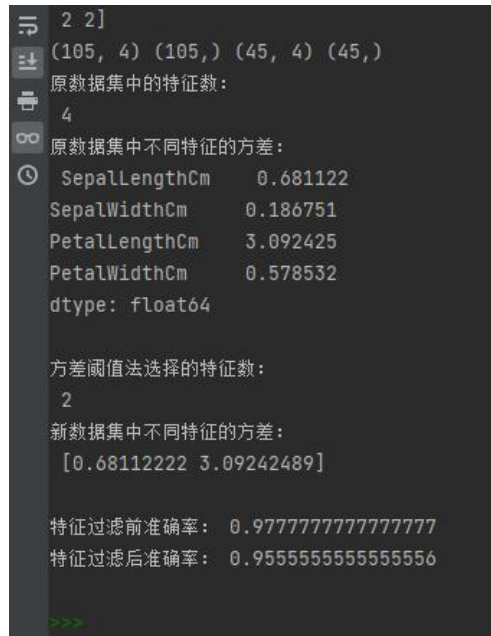
```
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.7,
random_state=0)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
acc = accuracy_score(y_test, y_pred)
print('特征过滤前准确率: ', acc)
```

方差过滤后的新数据集

```

X_train, X_test, y_train, y_test = train_test_split(X_new, y, train_size=0.7,
random_state=0)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
acc = accuracy_score(y_test, y_pred)
print('特征过滤后准确率: ', acc)

```



```

2 2]
(105, 4) (105,) (45, 4) (45,)
原数据集中的特征数:
4
原数据集中不同特征的方差:
SepalLengthCm    0.681122
SepalWidthCm     0.186751
PetalLengthCm    3.092425
PetalWidthCm     0.578532
dtype: float64

方差阈值法选择的特征数:
2
新数据集中不同特征的方差:
[0.68112222 3.09242489]

特征过滤前准确率: 0.9777777777777777
特征过滤后准确率: 0.9555555555555556
>>>

```

仅保留了两个特征的情况下，准确率与原数据集差距不大。

事实上，在这个例子中，这种通过牺牲模型性能来换取计算性能的操作意义不大，因为这个数据集仅有 150 个样本、4 个特征，但是当我们面临成千上万的特征、数以亿计的样本时，我们就有必要进行权衡。

任务 2:

在鸢尾花数据集上使用任意 2 种[特征选择方法](#)实现特征选择

要求:

1. 使用 SVM 进行分类预测
2. 添加 precision、recall 指标来评价分类性能。

```
from sklearn.metrics import precision_score, recall_score
```