# 鸢尾花数据集分析

对数据降维，训练分类模型进行分类，同时评价模型指标

核心代码

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])

print(iris.info())

# # 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()

# 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()

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

plt.show()

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()

sns.lmplot(data=iris, x='SepalWidthCm', y='SepalLengthCm', palette=antV, hue='Species')

sns.lmplot(data=iris, x='PetalLengthCm', y='PetalWidthCm', palette=antV, hue='Species')

plt.show()

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)

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)

# 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)))

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)))

# 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')

# 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)

# 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)

#移除低方差特征

from sklearn.feature\_selection import VarianceThreshold

import random

total = []

for i in range(0, len(iris)):

    temp = []

    for j in range(0, 4):

        temp.append(iris.values[i][j])

    total.append(temp)

sel = VarianceThreshold(threshold=(.8 \* (1 - .8)))

sel.fit\_transform(total)

# print(total)

#单变量特征选择？

from sklearn.feature\_selection import SelectKBest

from sklearn.feature\_selection import chi2

X = total

y = []

for i in range(0, len(iris)):

    y.append(iris.values[i][4])

X\_new = SelectKBest(chi2, k =2).fit\_transform(X, y)

np.random.seed(0)

# c=list(zip(X\_new,y))     #选择特征之后的

# random.shuffle(c)

# X\_new[:],y[:] = zip(\*c)

# iris\_x = X\_new

# iris\_y = y

# iris\_x\_train = iris\_x[:-10]

# iris\_y\_train = iris\_y[:-10]

# iris\_x\_test = iris\_x[-10:]

# iris\_y\_test = iris\_y[-10:]

#

# #使用线性核svc是分类向量机的意思，另外svr是回归支持向量机

# clf = svm.SVC(kernel = 'linear')

# clf.fit(iris\_x\_train,iris\_y\_train)  #学习

#

# #预测

# iris\_y\_predict = clf.predict(iris\_x\_test)

# score = clf.score(iris\_x\_test,iris\_y\_test,sample\_weight=None)

# print('iris\_y\_predict = ',iris\_y\_predict)

# print('iris\_y\_test = ', iris\_y\_test)

# print('Accuracy: ', score)

c=list(zip(X,y))    #未选择特征

random.shuffle(c)

X[:],y[:] = zip(\*c)

iris\_x = X

iris\_y = y

iris\_x\_train = iris\_x[:-10]

iris\_y\_train = iris\_y[:-10]

iris\_x\_test = iris\_x[-10:]

iris\_y\_test = iris\_y[-10:]

#使用线性核svc是分类向量机的意思，另外svr是回归支持向量机

clf = svm.SVC(kernel = 'linear')

clf.fit(iris\_x\_train,iris\_y\_train)  #学习

#预测

iris\_y\_predict = clf.predict(iris\_x\_test)

score = clf.score(iris\_x\_test,iris\_y\_test,sample\_weight=None)

print('iris\_y\_predict = ',iris\_y\_predict)

print('iris\_y\_test = ', iris\_y\_test)

print('Accuracy: ', score)

from sklearn.metrics import precision\_score, recall\_score, f1\_score

p = precision\_score(iris\_y\_test[-10:], iris\_y\_predict, average='micro')

p2 = precision\_score(iris\_y\_test[-10:], iris\_y\_predict, average='macro')

p3 = precision\_score(iris\_y\_test[-10:], iris\_y\_predict, average='weighted')

# p4 = precision\_score(actual, predicted, average='samples')

r = recall\_score(iris\_y\_test[-10:], iris\_y\_predict, average='micro')

r2 = recall\_score(iris\_y\_test[-10:], iris\_y\_predict, average='macro')

r3 = recall\_score(iris\_y\_test[-10:], iris\_y\_predict, average='weighted')

f1score = f1\_score(iris\_y\_test[-10:], iris\_y\_predict, average='micro')

f1score2 = f1\_score(iris\_y\_test[-10:], iris\_y\_predict, average='macro')

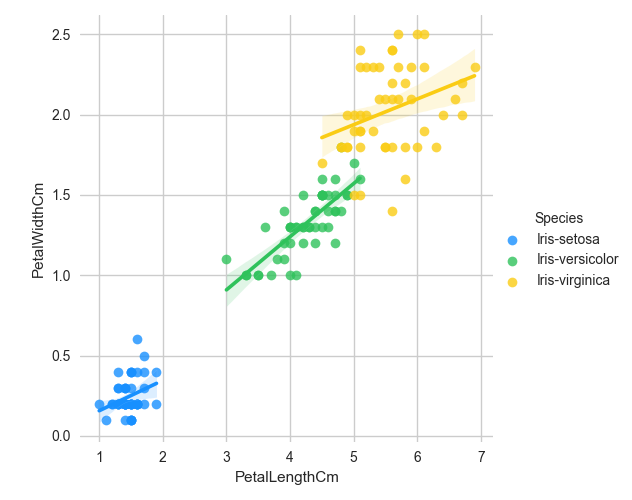
f1score3 = f1\_score(iris\_y\_test[-10:], iris\_y\_predict, average='weighted')

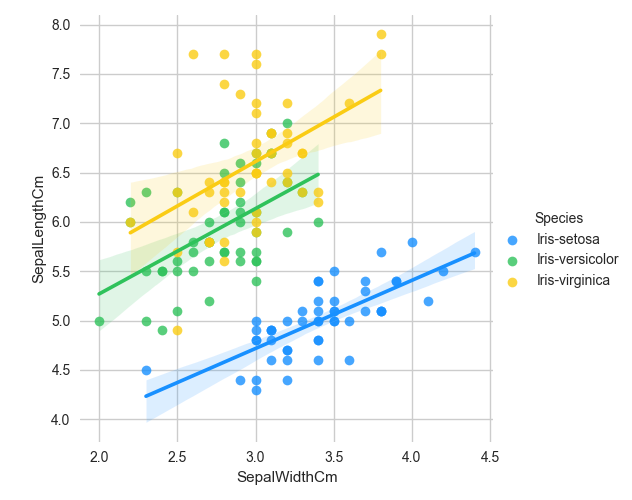
print(p,p2,p3)

print(r,r2,r3)

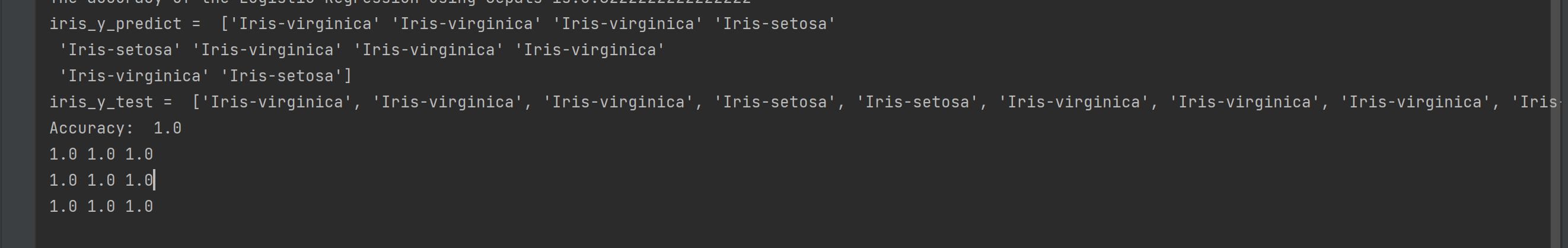
print(f1score,f1score2,f1score3)

**分别基于花萼和花瓣做线性回归的可视化：**

****

****

**b. 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，给出实验结果。要求:使用任一分类器进行预测，添加precision、recall 指标来评价分类性能**

****