1．实验目的

（1）掌握多种常见分类模型的使用方法

（2）能够用accuray，precision，recall，F1-score等多种评价指标比较分类模型

（3）掌握ROC曲线和AUC评价方法和可视化方法

（4）能够综合使用多种评价方法分析分类模型的优劣，筛选最佳模型

2. 任务描述

2.1分类模型比较与评价

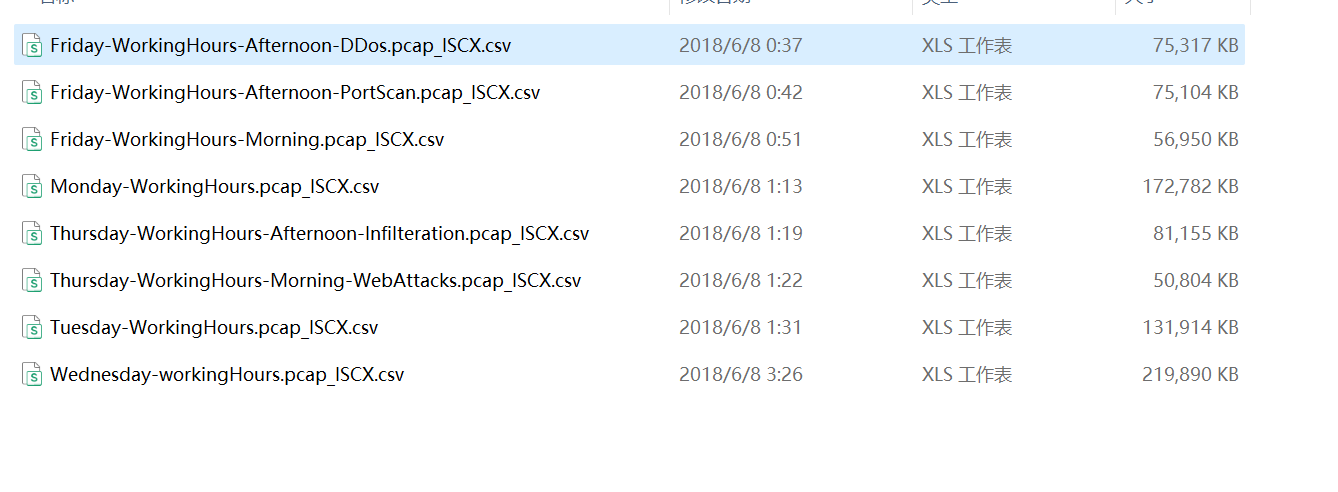
随着加密技术的发展，互联网上加密流量的数量和比例不断增加。根据谷歌在2020年发布的透明度报告，已经有80%以上的网络数据在传输过程中使用了加密技术。如何在加密流量中准确有效的识别出恶意流量，已经成为网络安全领域急需解决的问题。

本次实验基于CIC-IDS-2017开源数据集，该数据集包含多种恶意加密流量和网络攻击。本次任务要求利用多种不同的机器学习方法，在给定的数据集上分别做训练和调参，比较和评价分类模型的在本次任务中的优劣。

2.2数据集介绍

本次实验采用CIC-IDS-2017 开源数据集，该数据集的采集时间为一个星期内的不同时间段，具体为2017年7月3日(星期一)至2017年7月7日（星期五）下午5时，共计5天。星期一数据集是正常的一天，所以只包含正常的流量。其余几天均为恶意流量，具体类型包括暴力 FTP、暴力 SSH、Heartbleed、Web 攻击、恶意渗透、僵尸网络和 DDoS等。

其数据集目录如下图所示：



请注意，本次实验只选取该数据集中的两个子集（Monday和Tuesday）。

3.相关知识

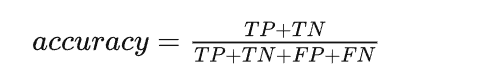
3.1模型评价指标

需要先了解下面几个概念：

TP：正确预测正分类结果 FP：错误预测正分类结果

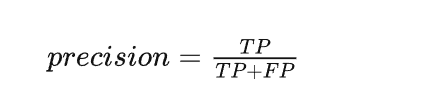
TN：正确预测负分类结果 FN：错误预测负分类结果

Accuracy，指的是“预测正确的样本数÷样本数总数”。计算公式为：



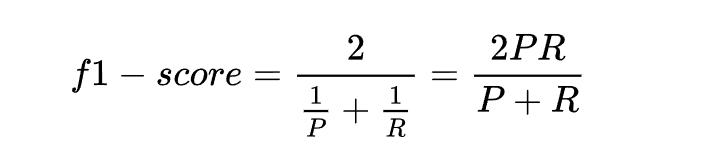
准确率的优点是比较简单直观。通常我们在如下情况的时候使用Accuracy：

Precision，指的是在我们预测为True的样本里面，有多少确实为True的。在信息检索领域，precision也被称为“查准率”。其公式为：



Recall，中文是召回率，指的是，实际上为True的样本有多少被我们挑出来了。在信息检索领域，recall也被称为“查全率”。其公式为：TP/(TP+FN)

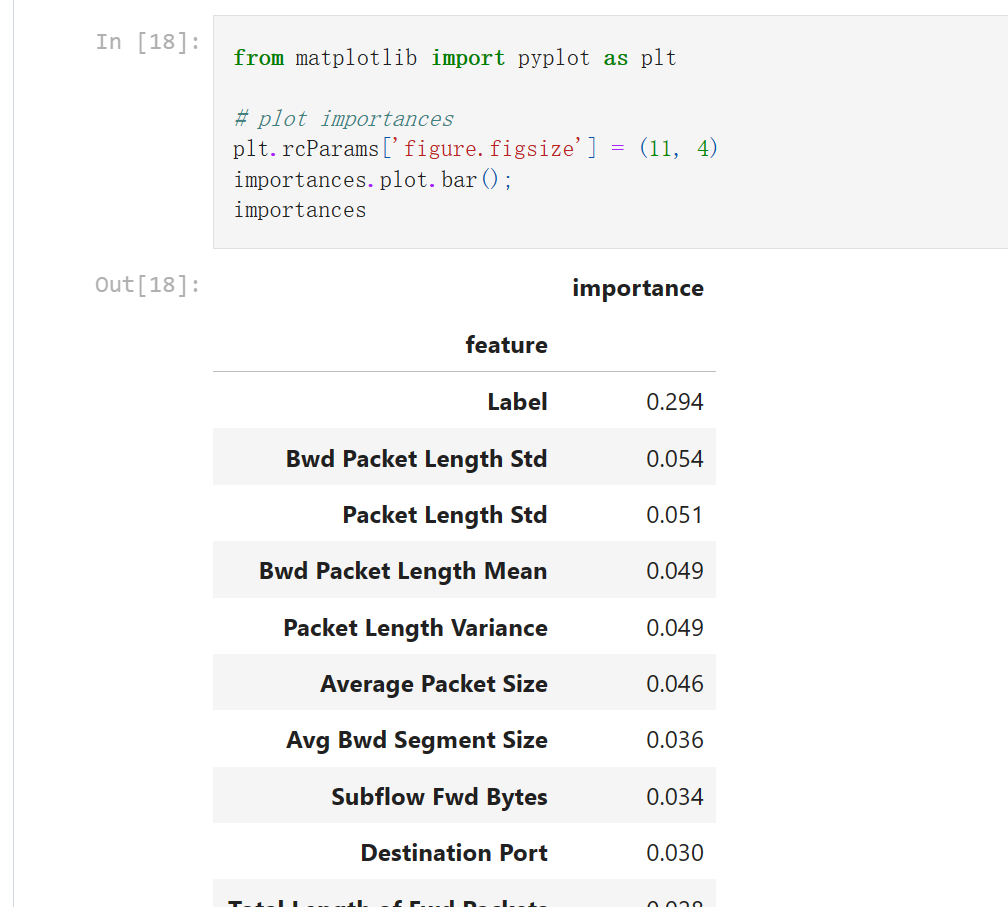
F1值是个综合考虑precision值和recall值的指标。F1 score综合考虑了precision和recall两方面的因素，做到了对于两者的调和，即：既要“求精”也要“求全”，做到不偏科。



3.2 关键特征字段介绍

本次数据集部分字段的介绍：ack\_cnt（带有 ACK 的包数量）、down\_up\_ratio（下载和上传的比例）、pkt\_size\_avg（数据包的平均大小）、

利用随机森林进行特征提取和筛选，得到特征关键字段重要性分布，得到对分类影响较大的特征字段为：Avg Bwd Segment Size、Subflow Fwd Bytes和Destination Port等。



4．实验要求和步骤

4.1读取数据

将原CSV文件变为DataFrame格式的文件，示例代码如下：

df\_monday = pd.read\_csv (filename)

pandas提供了pd.read\_csv()方法可以读取其中的数据并且转换成DataFrame数据格式，一般只需要将读取数据集的目录传参给filename即可完成函数的调用。

4.2必要的数据处理

将脏数据 Inf/NaN 值的列标记为 -1，示例代码如下：

pandas提供的replace方法可以将dataframe里的值替换成自己想要的值，具体语法是replace(to\_replace, value) ，前面是需要替换的值，后面是替换后的值

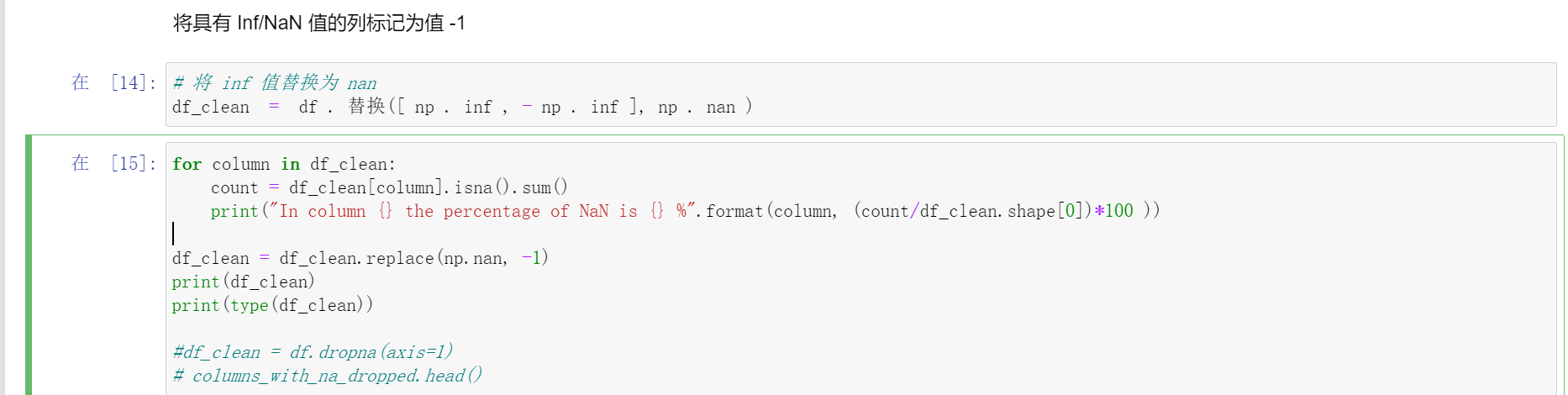
df\_clean = df.replace([np.inf, -np.inf], np.nan)

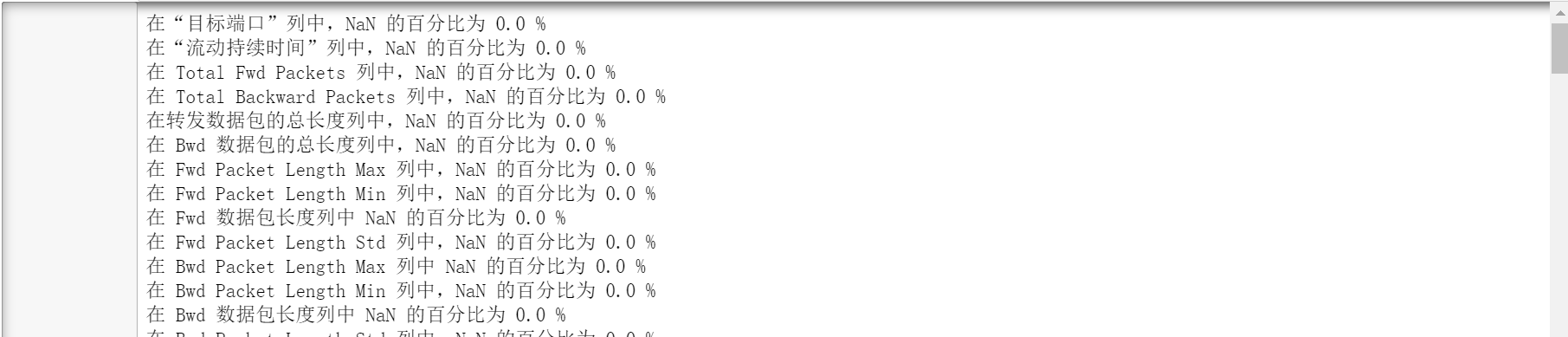
for column in df\_clean:

count = df\_clean[column].isna().sum()

print("在 {} 列中，NaN值的百分比为 {} %".format(column, (count/df\_clean.shape[0])\*100 ))

df\_clean = df\_clean.replace(np.nan, -1)





删除任何列/特征中至少包含一个 inf 或 nan 值的行

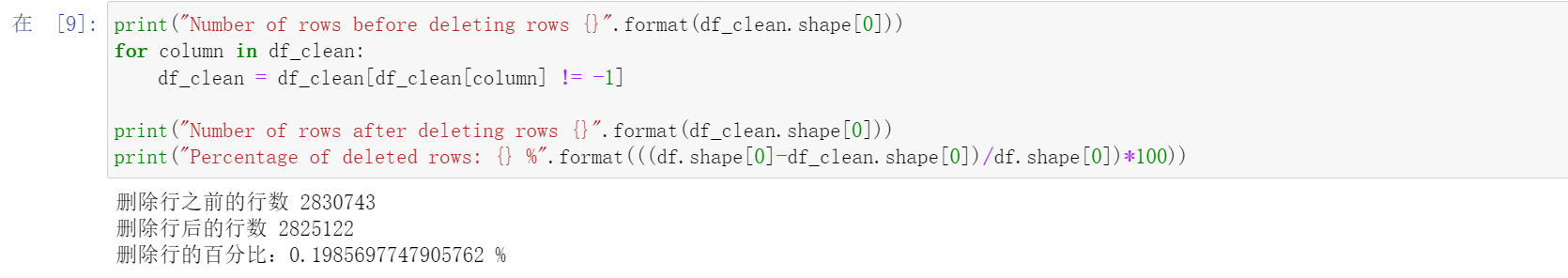
print("删除之前的行数是： {}".format(df\_clean.shape[0]))

for column in df\_clean:

df\_clean = df\_clean[df\_clean[column] != -1]

print("删除后的行数为： {}".format(df\_clean.shape[0]))

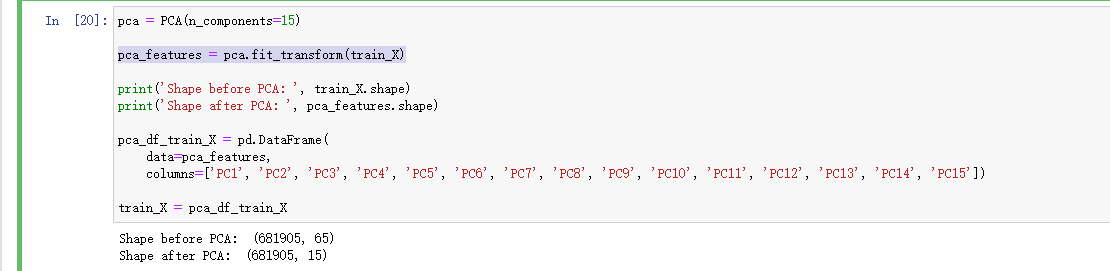
print("删除行的百分比: {} %".format(((df.shape[0]-df\_clean.shape[0])/df.shape[0])\*100))



4.3对提取的特征进行PCA降维，示例代码如下：

pca\_features = pca.fit\_transform(train\_X)

用train\_X训练数据来训练PCA模型，同时返回降维后的数据。



4.4训练集和测试集的划分，示例代码如下：

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=1)

其中，X为还未划分的数据集，y为还为划分的标签。

4.5 生成多个分类模型，包括但不限于决策树，k-近邻（KNN），逻辑回归，Random Forest等，训练模型并优化各模型参数。

(1)利用随机森林提取特征调用分类器及提取重要特征,示例代码如下：

#特征选择

rfc = RandomForestClassifier();

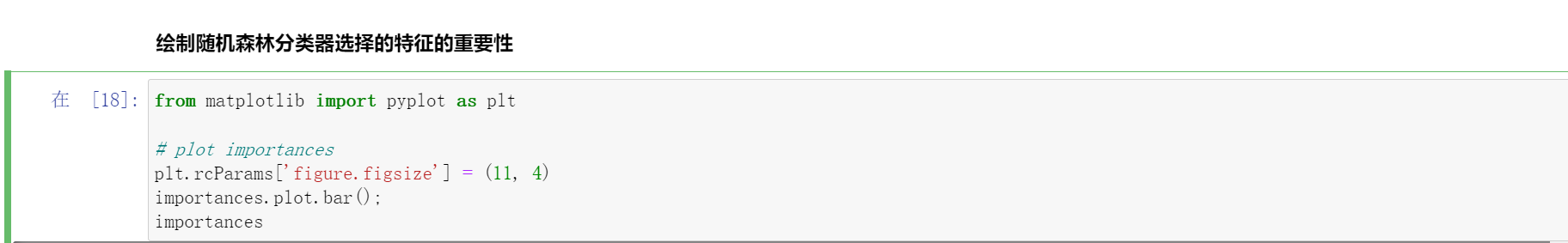
rfc.fit(test\_X, test\_y);

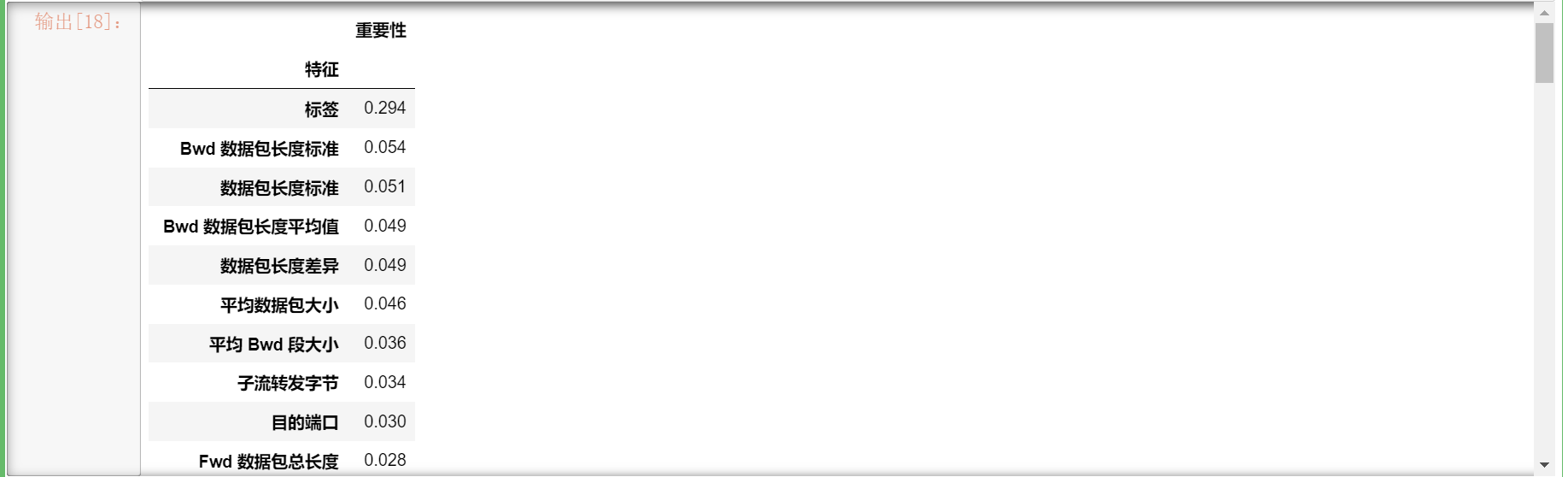
#提取重要特征

score = np.round(rfc.feature\_importances\_,3)

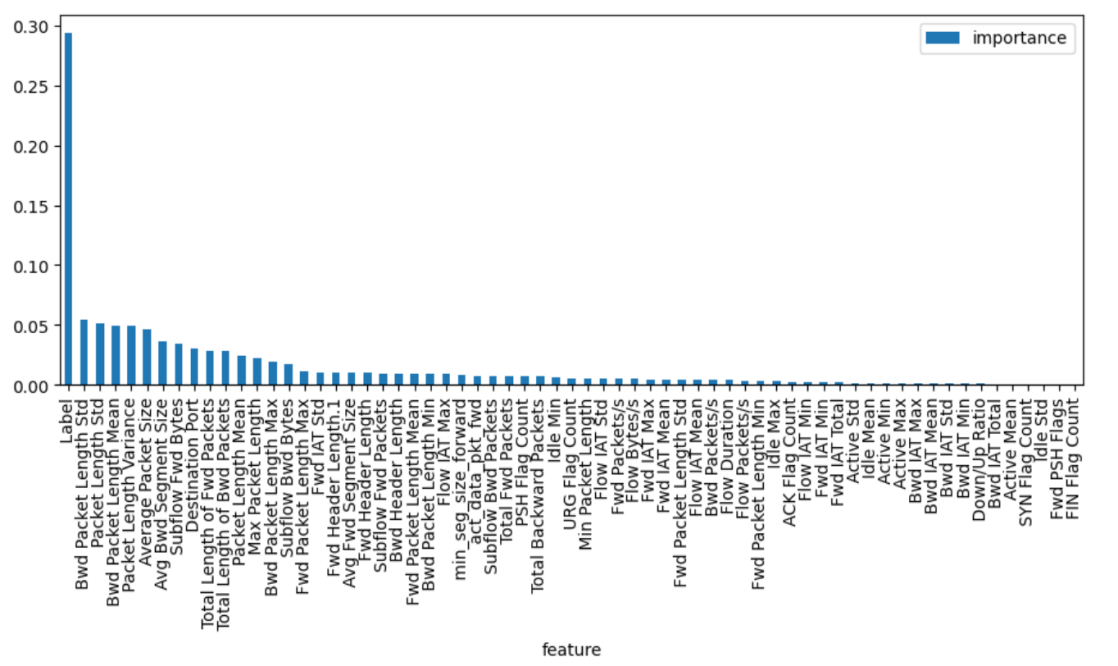
importances = pd.DataFrame({'feature':train\_X.columns,'importance':score})

importances = importances.sort\_values('importance',ascending=False).set\_index('feature')





绘制随机森林分类器选择的特征的重要性图表



(2)高斯朴素贝叶斯训练及调参

分类器定义示例代码如下：

BNB\_Classifier = BernoulliNB()

BNB\_Classifier.fit(train\_X, train\_y)

模型评价如下：

(3)决策树训练和调参

分类器定义示例代码如下：

DTC\_Classifier =

tree.DecisionTreeClassifier(criterion='entropy',random\_state=0)

DTC\_Classifier.fit(train\_X, train\_y)

(4) KNN训练和调参

分类器定义示例代码如下：

KNN\_Classifier = KNeighborsClassifier(n\_jobs=-1)

KNN\_Classifier.fit(train\_X, train\_y)

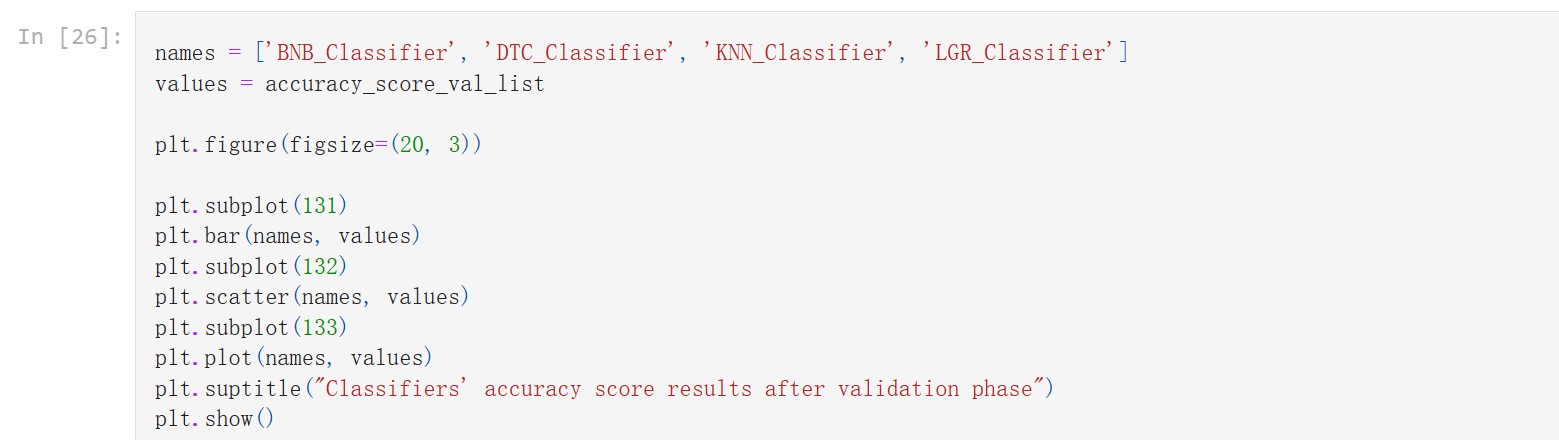
(5)逻辑回归训练和调参

分类器定义示例代码如下：

LGR\_Classifier = LogisticRegression(n\_jobs=-1, random\_state=0)

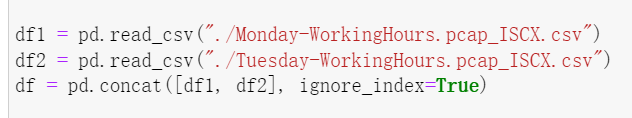
LGR\_Classifier.fit(train\_X, train\_y);

(6)对比各模型的accuray，precision，recall，F1-score。用图表绘制每个分类器的结果



5.测试说明

（1）数据集读取

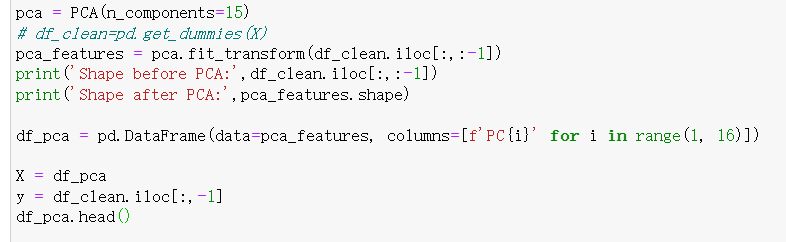


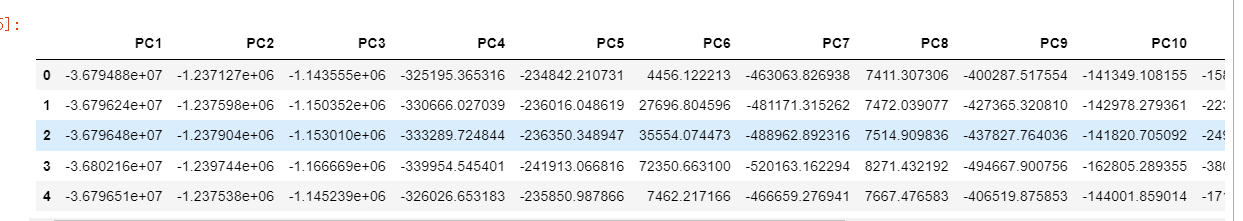
1. 数据清理（将Inf/NaN 值的列标记为 -1）

（3）用随机森林提取重要特征，并可视化

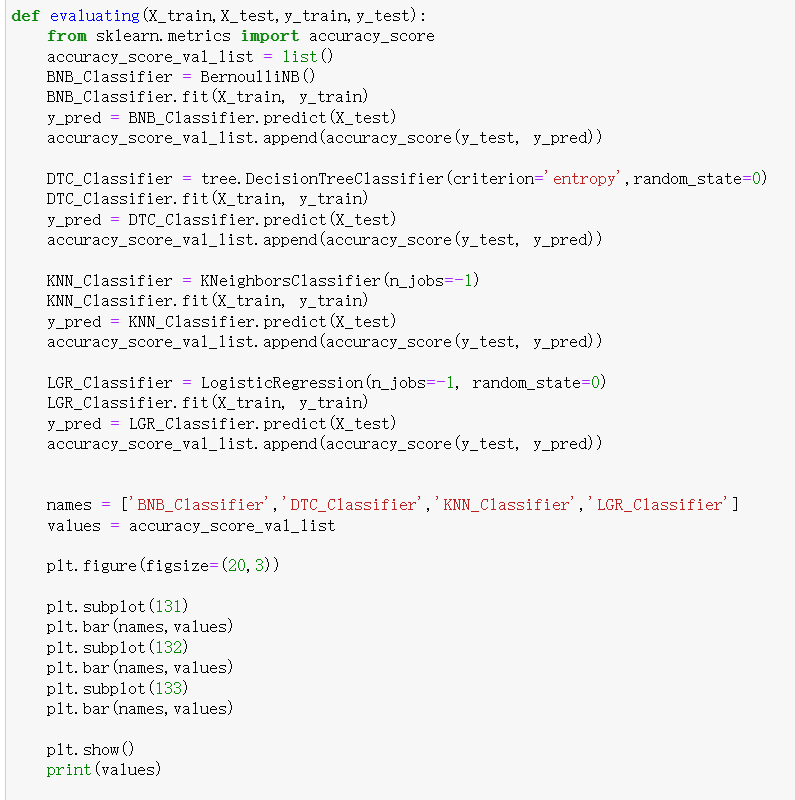


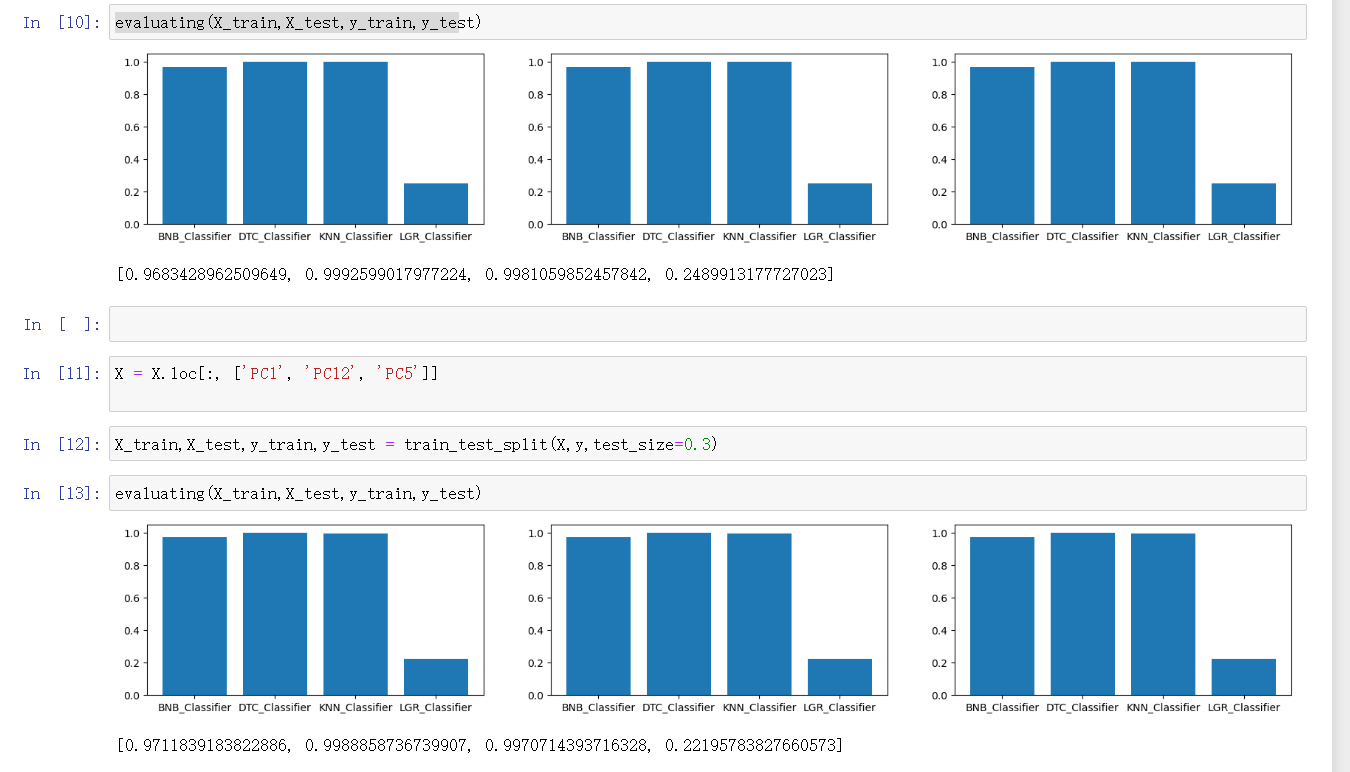
（4）对特征进行PCA降维，并可视化显示降维后各主成分方差值。





（5）各个模型验证的结果，并可视化





6 分析与讨论

（1）填充缺失值可以采取哪些其他策略？

均值中值众数

（2）何时需要对数据做规范化？本实验数据，适合哪种规范化方法？

数据维度过高，本实验PCA

（3）总结、讨论组合方法是否可以改进分类性能。

肯定可以的，可以试试先RFC再PCA