一．诊断模型的训练和部署

1、数据说明和使用方法

1.1训练集数据说明：

文件夹中有多个数据文件，文件名含义如下：

B代表故障发生在Ball位置，同理IR代表故障发生在inner race位置，OR代表故障发生在outer race位置，NORMAL代表数据文件是正常数据文件；

每个数据文件可能包含如下多维信号（部分文件可能其中不包括某些维度的信号）：

DE\_time:驱动端加速度数据

FE\_time:风扇端加速度数据

BA\_time:基本加速度数据

RPM:每分钟转速数据，在提取时实际上RPM对于每个文件只有一个值，但为了文件格式整齐，扩展成了一列，即实际上这一列都是同一个值，代表该文件的RPM。

1.2测试集数据说明：

数据中共包含142个文件，文件名为TEST1.csv-TEST142.csv，每个文件数据共有三列，分别为DE\_time、FE\_time和RPM。

文件中数据列的含义如下：

DE\_time:驱动端加速度数据

FE\_time:风扇端加速度数据

RPM:每分钟转速数据，在提取时实际上RPM对于每个文件只有一个值，但为了文件格式整齐，扩展成了一列，即实际这一列都是同一个值，代表该文件的RPM。

#####2、数据预处理及数据清洗

2.1训练集：

首先，由于BA\_time在整整NORMAL一类全部缺失，且训练集中并没有这一属性的数据，所以将这一属性删除。而后以1024为时间窗长度，256为移动步长，将每个时间窗中的DE\_time，,FE\_time的1024个值作为一条记录，将所有文件中的记录整合，并将每条记录打上标签label，约定normal(NORMAL), ball(B), outer race(OR), inner race(IR)的预测输出标签为0, 1, 2, 3。整合结果如图1所示。

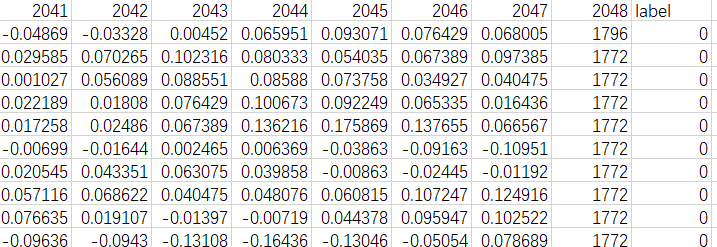


图1 训练集整合结果示意图（部分）

该步骤中并没有进行数据均衡，因为通过实验结果的反馈，数据均衡使测试集的f1\_score和准确率不升反降，而且滑动时间窗某种程度上也起到了过采样的效果。该步骤也没有采用数据清洗，因为原始数据较为干净，清洗原始数据可能会引起时序关系混乱或使关键信息丢失，而且最终预测整个文件的类型时，采用了依据时间窗分类结果出现频率统一预测结果的方式，也起到了数据清洗的效果。

2.2测试集：

基本同训练集，区别在于每条记录的标签为记录所在文件名filename。

3、特征工程

3.1特征提取

经过查阅相关文献和数据探索可知，所给数据在时域、频域、时频域上均有丰富的特征。对于所有训练集、测试集时间窗，分别提取18个时域特征，9个频域特征，5个时频域（小波变换:窗长：200，阶次：4）特征，训练集需要额外保留标签，测试集需要额外保留文件名。

3.2特征选择

特征选择的原则为：不同类间区分度大，同一类间相似性高。

经过分析与实验比较，我们选取了6个时域特征：标准差、峰峰值、偏度、波形因子、峰值因子、脉冲值，9个频域特征：fft变换后，频谱前三的最大值点和标准化前三的最大值，5个时频域特征：一个低频的近似分量和四个高频的细节分量。

有三种特征选择的组合效果较好：1.时域+频域特征，2.时频域特征，3.时域+频域+时频域特征。

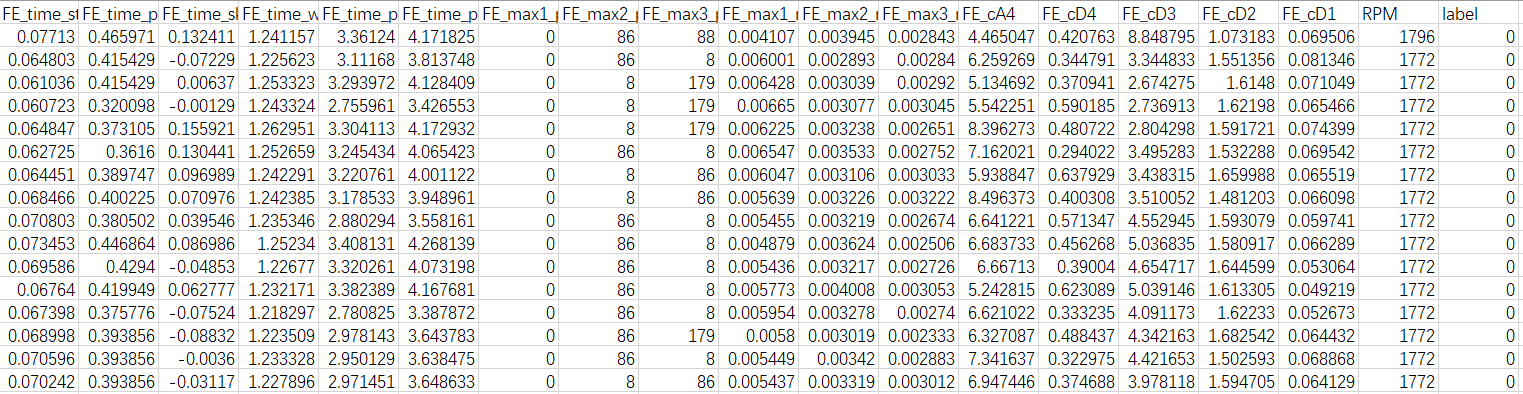


图2 训练集特征选择结果示意图（部分）

4、模型的训练和验证

4.1模型训练

分类模型选用随机森林模型，因为随机森林模型具有使用简易、善于处理高维数据，分类准确度高、泛化能力强的优点。

我们将3.2中提到的三种特征组合选择1，2，3均进行了训练，得到对应三种模型1，2，3。

4.2模型验证

利用三种模型，预测测试集三种特征组合的分类结果，能够有较好的效果。

5、结果的分析和调优

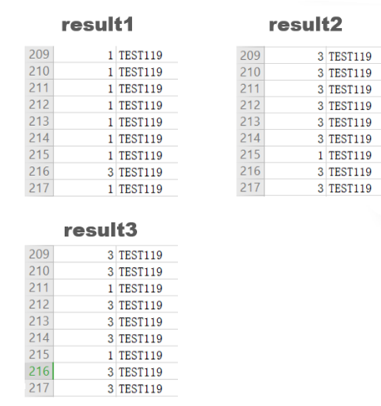
5.1模型调优

我们使用了sklearn中的随机森林模型，有两个重要的调优参数：

1. n\_estimators: 最大的弱学习器的个数。一般情况下，越大越好，但是会减慢计算速度，而且增大到一定程度后提升效果不明显。本例中n\_estimators=100
2. oob\_score :是否采用袋外样本来评估模型的好坏。袋外分数反应了一个模型拟合后的泛化能力，设置为True。

5.2结果整合

先进行横向整合，即将三份预测结果的每一行进行整合，最后取出现次数最多的值作为该时间窗的预测结果。如图3所示。



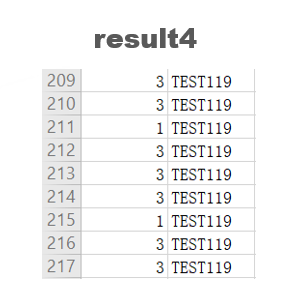


图3 横向整合示意图

再进行纵向整合，对横向整合后的结果，取同一文件中出现频率最高的分类作为整个文件的预测结果。

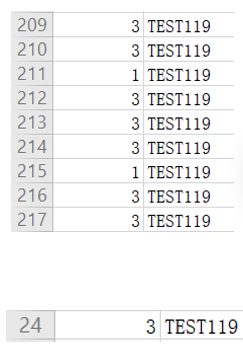


图4 纵向整合示意图

6、模型部署

首先建立code、datagena、dataset、datatw、feature、model、result几个文件夹，code中放置代码。如图5所示

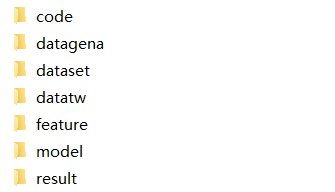


图5 文件部署示意

代码文件包括：

data\_tw.py、test\_tw.py:训练集、测试集数据加窗。

datagena.py、testgena.py:训练集、测试集数据整合。

allfeature\_train、allfeature\_test:训练集、测试集特征提取。

train\_select\_t\_f、test\_select\_t\_f: 训练集、测试集特征选择：时域、频域。

train\_select\_wt、test\_select\_wt: 训练集、测试集特征选择：时频域。

train\_select\_all、test\_select\_all: 训练集、测试集特征选择：时域、频域、时频域。

model中存放分类模型、result中存放分类结果。其余文件存放中间过程的csv文件.

二、核心代码及其注释

1滑动时间窗（出自data\_tw.py）

def timewindow(path\_machine,out\_machine,tw\_size=1024,step\_size=256):

‘’’

读取训练集各文件内容并加时间窗

path\_machine、out\_machine分别为输入、输出文件路径，tw\_size,step\_size为窗长、步长

‘’’

…

sample\_num=df\_machine.shape[0] #行数

tw\_num=math.floor((sample\_num-tw\_size)/step\_size)

#整合后的时间窗数=（采样总数-窗长）/步长，再向下取整

for k in range(tw\_num):

np\_temp=np.array(df\_machine.ix[k\*step\_size:k\*step\_size+tw\_size-1,:])

#第k+1个时间窗对应的采样切片

np\_tw=np\_temp.reshape((1,-1),order='F')

#将三列转制并合并成1行

np\_tw=np\_tw[0,0:(2\*tw\_size+1)]

#只保留1个RPM即可

np\_tw=np\_tw.reshape((1,-1),order='F')

df\_temp=pd.DataFrame(np\_tw)

df\_out=pd.concat([df\_out,df\_temp],axis=0,ignore\_index=True)

#将新得到的时间窗与之前的拼接合并

2 频域特征提取（出自allfeature\_train.py）

#提取频域特征倍频能量以及能量占比

plist\_raw = np.fft.fft(list(df\_line), n=1024)

#快速傅里叶变换

plist = np.abs(plist\_raw)

plist512=plist[0:512]

#由fft对称性，实信号观察一半频域样点即可

plist\_energy = (np.square(plist)).sum()

#前三最大值点与能量占比

max1\_point=np.argmax(plist512)

#最大值点

max1\_ratio=plist[max1\_point]/plist\_energy

#标准化的最大能量值

max1\_value=plist[max1\_point]

#未标准化最大能量值（未使用）

plist[max1\_point]=0

#将该点值清零，以便寻找第二最大值点

max2\_point=np.argmax(plist512)

#寻找第二最大值点

…

3时频域特征提取（出自allfeature\_train.py）

#时频域特征

params = {}

params['len\_piece']=200 #小的时间窗长

params['wave\_layer']=4 #小波阶次

params['wave\_win']=38\*pow(2,params['wave\_layer']-1)-1

def one\_row(arr):

result\_list = []

arr\_add=arr.iloc[:,]

for j in range(int(params['wave\_win']/params['len\_piece'])):

arr\_add =arr\_add.append(arr,ignore\_index=True)

cD=wavedec(arr\_add,'db10',level=params['wave\_layer'])

#离散小波变换 返回CAn：n级平均分解系数，CDn：细节系数

for i in range(params['wave\_layer']+1):

ener\_cD = np.square(cD[i]).sum()

list\_para = [ener\_cD]

result\_list.extend(list\_para)

return result\_list

4随机森林模型训练（出自train\_rfc.py）

def train(train\_data):

train\_data\_y = train\_data['label']

#标签

train\_data\_x = train\_data.drop(['label','RPM'], axis=1)

# 除去标签的所有列就是特征

model\_rfc=RandomForestClassifier(random\_state=0,n\_estimators=100)

# 模型初始化，设置random\_state保证可复现性，便于观察优化

# 模型训练

model\_rfc.fit(train\_data\_x, train\_data\_y)

#保存模型

joblib.dump(model\_rfc, '../model/rfc\_t\_f\_model.model')

5 横向整合（出自selectpre.py）

for i in range(0,pre1.shape[0]):

if(pre1[i] == pre2[i]):

#result1和result2相等，则result1的预测结果即为多数结果

pre.append(1)

pre[i] = pre1[i]

elif((pre1[i] == pre3[i])or(pre2[i] == pre3[i])):

#result2和result3或result1和result3相等，则result3的预测结果即为多数结果

pre.append(1)

pre[i] = pre3[i]

else:

“””

若以上情况均不满足，证明result1、result2、result3三者均不相等，取单独测试情况下得分最高的result3为结果

“””

pre.append(1)

pre[i] = pre3[i]

6纵向整合

import pandas as pd

tw\_result=pd.read\_csv('../result/result4.csv')

unify\_result=tw\_result.groupby('filename').agg(lambda x: x.value\_counts().index[0]).reset\_index()

#按filename分组，将每个分组中不同值的出现次数进行统计，找出最大者，即众数。

columns=['label','filename']

unify\_result.to\_csv('../result/result.csv',index=False,columns=columns)