# 目录

[目录 1](#_Toc67930774)

[1 导入模块说明 1-7](#_Toc67930775)

[1.1 Qt界面模块 1-7](#_Toc67930776)

[1.1.1 QtWidgets 1-7](#_Toc67930777)

[1.1.2 QtGui 1-7](#_Toc67930778)

[1.1.3 QtCore 1-8](#_Toc67930779)

[1.1.4 MyPlatform 1-8](#_Toc67930780)

[1.2 特征提取模块 1-8](#_Toc67930781)

[1.3 特征预测模块 1-9](#_Toc67930782)

[1.4 其他模块 1-10](#_Toc67930783)

[2 登录界面 2-11](#_Toc67930784)

[2.1 \_\_init\_\_ 2-11](#_Toc67930785)

[2.2 checkUsernameIsTrue 2-11](#_Toc67930786)

[2.3 checkPasswordIsTrue 2-11](#_Toc67930787)

[2.4 on\_sure\_Button\_clicked 2-11](#_Toc67930788)

[2.5 \_\_del\_\_ 2-11](#_Toc67930789)

[3 主界面 3-12](#_Toc67930790)

[3.1 枚举类型 3-12](#_Toc67930791)

[3.1.1 class cellType(Enum) 3-12](#_Toc67930792)

[3.1.2 class columnIndex(Enum) 3-12](#_Toc67930793)

[3.2 前端函数 3-12](#_Toc67930794)

[3.2.1 \_\_init\_\_ 3-12](#_Toc67930795)

[3.2.2 init\_comboBox\_text\_icon 3-13](#_Toc67930796)

[3.2.3 init\_waveform 3-13](#_Toc67930797)

[3.2.4 init\_menuAction\_buttonClicked 3-13](#_Toc67930798)

[3.2.5 init\_other\_properties 3-14](#_Toc67930799)

[3.2.6 init\_widget\_menu 3-14](#_Toc67930800)

[3.2.7 init\_myplatform 3-15](#_Toc67930801)

[3.2.8 init\_table\_time\_features 3-15](#_Toc67930802)

[3.2.9 init\_current\_tableWidget 3-16](#_Toc67930803)

[3.2.10 init\_vibration\_tableWidget 3-16](#_Toc67930804)

[3.2.11 on\_pushButton\_clicked 3-16](#_Toc67930805)

[3.2.12 on\_start\_training\_clicked 3-17](#_Toc67930806)

[3.2.13 on\_cancel\_training\_clicked 3-17](#_Toc67930807)

[3.2.14 on\_comboBox\_currentIndexChanged 3-17](#_Toc67930808)

[3.2.15 on\_tableWidget\_currentCellChanged 3-18](#_Toc67930809)

[3.2.16 on\_current\_predict\_tabWidget\_currentCellChanged 3-18](#_Toc67930810)

[3.2.17 on\_vbriation\_predict\_tabWidget\_currentCellChanged 3-18](#_Toc67930811)

[3.2.18 set\_waveform\_parameters 3-18](#_Toc67930812)

[3.2.19 convert\_faultNumber\_intToString 3-20](#_Toc67930813)

[3.2.20 signWindowIsSuccess 3-20](#_Toc67930814)

[3.2.21 signDialogIsSuccess 3-20](#_Toc67930815)

[3.2.22 \_\_del\_\_ 3-20](#_Toc67930816)

[3.3 后端函数 3-21](#_Toc67930817)

[3.3.1 读取波形函数 3-21](#_Toc67930818)

[3.3.1.1 read\_current\_waveform\_data 3-21](#_Toc67930819)

[3.3.1.2 spilit\_current\_waveform\_data 3-21](#_Toc67930820)

[3.3.1.3 read\_current\_waveform\_pca\_features 3-21](#_Toc67930821)

[3.3.1.4 split\_current\_pca\_features\_data\_label 3-21](#_Toc67930822)

[3.3.1.5 spilit\_current\_pca\_features\_data 3-21](#_Toc67930823)

[3.3.1.6 read\_vibration\_waveform\_data 3-22](#_Toc67930824)

[3.3.1.7 spilit\_vibration\_waveform\_data 3-22](#_Toc67930825)

[3.3.1.8 read\_vibration\_waveform\_pca\_features 3-22](#_Toc67930826)

[3.3.1.9 split\_vibration\_pca\_features\_data\_label 3-22](#_Toc67930827)

[3.3.1.10 spilit\_vibration\_pca\_features\_data 3-22](#_Toc67930828)

[3.3.1.11 read\_txtdata\_to\_series 3-22](#_Toc67930829)

[3.3.1.12 read\_all\_txtdata\_to\_dataframe 3-23](#_Toc67930830)

[3.3.1.13 read\_one\_dataframe 3-23](#_Toc67930831)

[3.3.2 数据处理函数 3-23](#_Toc67930832)

[3.3.2.1 calculate\_waveform\_characteristics 3-23](#_Toc67930841)

[3.3.2.2 set\_table\_values 3-23](#_Toc67930842)

[3.3.2.3 return\_cmaps 3-23](#_Toc67930843)

[3.3.2.4 data\_normalization 3-23](#_Toc67930844)

[3.3.2.5 fft\_data 3-24](#_Toc67930845)

[3.3.2.6 correlation\_coefficient 3-24](#_Toc67930846)

[3.3.2.7 imf\_signal\_correlation 3-24](#_Toc67930847)

[3.3.2.8 signal\_correlation\_entropy 3-24](#_Toc67930848)

[3.3.2.9 calculate\_all\_dataframe\_features 3-24](#_Toc67930849)

[3.3.2.10 compress\_features\_to\_pca 3-25](#_Toc67930850)

[3.3.3 数据预测函数 3-26](#_Toc67930851)

[3.3.3.1 start\_svm\_training 3-26](#_Toc67930852)

[3.3.3.2 return\_current\_svm\_model 3-26](#_Toc67930853)

[3.3.3.3 return\_current\_predict\_label\_svm 3-26](#_Toc67930854)

[3.3.3.4 return\_current\_predict\_proba\_svm 3-27](#_Toc67930855)

[3.3.3.5 return\_current\_predict\_mean\_proba\_svm 3-27](#_Toc67930856)

[3.3.3.6 return\_current\_kfold\_predict\_proba\_svm 3-27](#_Toc67930857)

[3.3.3.7 return\_current\_maxiter\_predict\_proba\_svm 3-27](#_Toc67930858)

[3.3.3.8 return\_current\_maxiter\_kfold\_predict\_proba\_svm 3-27](#_Toc67930859)

[3.3.3.9 start\_mlp\_training 3-28](#_Toc67930860)

[3.3.3.10 return\_current\_mlp\_model 3-28](#_Toc67930861)

[3.3.3.11 return\_current\_predict\_label\_mlp 3-28](#_Toc67930862)

[3.3.3.12 return\_current\_predict\_proba\_mlp 3-29](#_Toc67930863)

[3.3.3.13 return\_current\_predict\_mean\_proba\_mlp 3-29](#_Toc67930864)

[3.3.3.14 return\_current\_kfold\_predict\_proba\_mlp 3-29](#_Toc67930865)

[3.3.3.15 return\_current\_maxiter\_predict\_proba\_mlp 3-29](#_Toc67930866)

[3.3.3.16 return\_current\_maxiter\_kfold\_predict\_proba\_mlp 3-29](#_Toc67930867)

[3.3.3.17 return\_current\_model\_traingtime 3-30](#_Toc67930868)

[3.3.3.18 return\_vibration\_svm\_model 3-30](#_Toc67930869)

[3.3.3.19 return\_vibration\_predict\_label\_svm 3-30](#_Toc67930870)

[3.3.3.20 return\_vibration\_predict\_proba\_svm 3-30](#_Toc67930871)

[3.3.3.21 return\_vibration\_predict\_mean\_proba\_svm 3-30](#_Toc67930872)

[3.3.3.22 return\_vibration\_kfold\_predict\_proba\_svm 3-31](#_Toc67930873)

[3.3.3.23 return\_vibration\_maxiter\_predict\_proba\_svm 3-31](#_Toc67930874)

[3.3.3.24 return\_vibration\_maxiter\_kfold\_predict\_proba\_svm 3-31](#_Toc67930875)

[3.3.3.25 return\_vibration\_mlp\_model 3-31](#_Toc67930876)

[3.3.3.26 return\_vibration\_predict\_label\_mlp 3-31](#_Toc67930877)

[3.3.3.27 return\_vibration\_predict\_proba\_mlp 3-31](#_Toc67930878)

[3.3.3.28 return\_vibration\_predict\_mean\_proba\_mlp 3-32](#_Toc67930879)

[3.3.3.29 return\_vibration\_kfold\_predict\_proba\_mlp 3-32](#_Toc67930880)

[3.3.3.30 return\_vibration\_maxiter\_predict\_proba\_mlp 3-32](#_Toc67930881)

[3.3.3.31 return\_vibration\_maxiter\_kfold\_predict\_proba\_mlp 3-32](#_Toc67930882)

[3.3.3.32 return\_vibration\_model\_traingtime 3-32](#_Toc67930883)

[3.3.3.33 return\_mlp\_neural\_network 3-32](#_Toc67930884)

[3.3.3.34 get\_mlp\_acu 3-33](#_Toc67930885)

[3.3.3.35 get\_mlp\_acu\_trainsize 3-33](#_Toc67930886)

[3.3.3.36 ChangeSovler 3-33](#_Toc67930887)

[3.3.3.37 ChangeActivation 3-33](#_Toc67930888)

[3.3.3.38 ChangeAlpha 3-33](#_Toc67930889)

[3.3.3.39 ChangeLearingRate 3-34](#_Toc67930890)

[3.3.3.40 ChangeConstantLearningRateInit 3-34](#_Toc67930891)

[3.3.3.41 ChangeInvscalingLearningRateInit 3-34](#_Toc67930892)

[3.3.3.42 ChangeInvscalingPowert 3-34](#_Toc67930893)

[3.3.3.43 ChangeTrainingSampleRatio 3-34](#_Toc67930894)

[3.3.3.44 get\_svm\_acu\_onetraining 3-34](#_Toc67930895)

[3.3.3.45 get\_mlp\_acu\_onetraining 3-35](#_Toc67930896)

[3.3.3.46 get\_svm\_acu\_trainsize 3-35](#_Toc67930897)

[3.3.3.47 get\_mlp\_acu\_trainsize 3-35](#_Toc67930898)

[3.3.3.48 ChangeMLPTrainsize 3-35](#_Toc67930899)

[3.3.3.49 ChangeSVMTrainsize 3-35](#_Toc67930900)

[3.3.3.50 get\_svm\_acu\_trainsize\_maxiter 3-35](#_Toc67930901)

[3.3.3.51 get\_mlp\_acu\_trainsize\_maxiter 3-36](#_Toc67930902)

[3.3.3.52 one\_training 3-36](#_Toc67930903)

[3.3.3.53 maxiter\_training 3-36](#_Toc67930904)

[3.3.3.54 fixed\_testsize 3-36](#_Toc67930905)

[3.3.3.55 fixed\_training\_num 3-36](#_Toc67930906)

[3.3.3.56 return\_leave\_p\_method\_acu\_training\_num 3-36](#_Toc67930907)

[3.3.3.57 leave\_p\_method 3-36](#_Toc67930908)

[3.3.3.58 make\_meshgrid 3-37](#_Toc67930909)

[3.3.4 可视化函数 3-37](#_Toc67930910)

[3.3.4.1 plot 3-37](#_Toc67930911)

[3.3.4.2 plot\_imfs 3-37](#_Toc67930912)

[3.3.4.3 plot\_energy\_svd\_corr\_imf 3-37](#_Toc67930921)

[3.3.4.4 plot\_energy\_svd\_corr 3-37](#_Toc67930922)

[3.3.4.5 compare\_energy 3-38](#_Toc67930923)

[3.3.4.6 compare\_svd 3-38](#_Toc67930924)

[3.3.4.7 compare\_corr 3-38](#_Toc67930925)

[3.3.4.8 my\_learning\_curve 3-38](#_Toc67930926)

[3.3.4.9 plot\_my\_learning\_curve 3-38](#_Toc67930927)

[3.3.4.10 print\_evaluation\_index 3-38](#_Toc67930928)

[3.3.4.11 plot\_kernel\_density\_estimation 3-39](#_Toc67930929)

[3.3.4.12 compare\_nfold\_cross\_validation\_score 3-39](#_Toc67930930)

[3.3.4.13 compare\_gamma\_cross\_validation\_score 3-39](#_Toc67930931)

[3.3.4.14 plot\_rbf\_sampler\_nystroem 3-39](#_Toc67930932)

[3.3.4.15 PlotACU 3-39](#_Toc67930933)

[3.3.4.16 PlotAlpha 3-40](#_Toc67930934)

[3.3.4.17 PlotConstantLearningRateInit 3-40](#_Toc67930935)

[3.3.4.18 PlotInvscalingLearningRateInit 3-40](#_Toc67930936)

[3.3.4.19 PlotInvscalingPowert 3-40](#_Toc67930937)

[3.3.4.20 PlotTrainingSampleRatio 3-40](#_Toc67930938)

[3.3.4.21 plot\_svm\_acu\_trainsize 3-40](#_Toc67930939)

[3.3.4.22 plot\_mlp\_acu\_trainsize 3-40](#_Toc67930940)

[3.3.4.23 plot\_svm\_acu\_maxiter 3-40](#_Toc67930941)

[3.3.4.24 plot\_mlp\_acu\_maxiter 3-41](#_Toc67930942)

[3.3.4.25 plot\_acu\_fixed\_testsize 3-41](#_Toc67930943)

[3.3.4.26 plot\_acu\_fixed\_training\_num 3-41](#_Toc67930944)

[3.3.4.27 plot\_leave\_p\_method\_acu 3-41](#_Toc67930945)

[3.3.4.28 plot\_contours 3-41](#_Toc67930946)

# 导入模块说明

## Qt界面模块

### QtWidgets

*from PyQt5.QtWidgets import(*

*QMainWindow,QAction,QApplication,QMessageBox,*

*QTableWidgetItem,QAbstractItemView,QLabel)*

QtWidgets模块是PyQt5的顶级父类，仅次于Object类，下方主要继承了QDialog、QMainWindow、QWidget几大窗体模块，这些模块是包含了界面的相关属性；

QAction用于动态创建动作，例如菜单栏动作或者按钮点击的动作，这些动作通过clicked或者triggered等方式触发槽函数；

QMessageBox是对话框类，它的父类是Dialog，主要用于产生消息、警告、错误等对话框，在登陆界面中使用到；

QTableWidgetItem是描述table表格的项，创建项、设置项的类型和属性以及添加项都需要这个模块；

QAbstractItemView用于提供项的可视化，例如表头的设置、单元格的宽度高度等属性，具体可查阅文档；

QLabel用于设置文字标签，或者状态栏，状态栏文字需要一个载体，这个载体必须被QLabel动态创建，也可以设置图片、图标等

QApplication是一个应用程序类，在运行主界面之前，必须说明是应用进程，即app = QApplication(sys.argv)，然后再运行界面my\_platform = myPlatform()，最后界面结束运行时要执行关闭程序sys.exit(app.exec\_())释放内存

### QtGui

*from PyQt5.QtGui import(QIcon, QPainter, QFont, QPen, QColor,*

*QBrush, QPalette, QPixmap)*

QIcon用于设置图标；

QPainter用于设置绘图，绘图需要绘制基本的图形元素，主要有3个属性，QPen用于控制线条颜色、宽度、线型等；QBrush用于设置区域的填充特性，可以填充颜色、填充方式、渐变特性等；QFont是一个格式对象，设置文字的样式、大小等属性；

设置颜色还可能用到颜色板QPalette和QColor；

QPixmap则是处理图片，例如背景图片、QLabel用的图片等

### QtCore

*from PyQt5.QtCore import Qt, pyqtSlot, QTime, QSize*

Qt一般用于显示一些枚举成员，例如坐标轴Qt.Axis，对齐方式Qt. Alignment等等；

pyqtSlot用于指定过载信号或自定义信号携带的参数类型

QTime是定时器，但不是一个UI界面可见的组件，主要属性为interval表示定时中断的周期，单位默认ms，主要的输出信号是timeout()，定时中断时可以发射此信号，定时中断做出响应则需要编写槽函数与信号connect；

QSize主要用来设置大小，例如图标的宽度、高度等

### MyPlatform

这是一个项目文件夹，主要包括4个文件，即myPlatform、faultPlatform、mySignWindow 、mySignDialog，分别是主界面的业务逻辑类和UI界面框架类，以及两个登陆界面，区别是mySignDialog继承于QDialog，mySignWindow继承于QMainWindow，本平台用任何一个都可以，这里用的是mySignWindow

*from MyPlatform.mySignDialog import mySignDialog*

*from MyPlatform.mySignWindow import mySignWindow*

*from MyPlatform import faultPlatform*

## 特征提取模块

*from Feature\_extraction.Features import (emd\_entropy ,*

*svd\_entropy, permutation\_entropy)*

*from pyentrp import entropy as ent*

*from vectorizedsampleentropy import vectapen*

Feature\_extraction.Features是Feature\_extraction文件夹下的文件Features，定义了5个类，本平台用到了求取能量熵、奇异值熵和排列熵的类

pyentrp 和vectorizedsampleentropy 是Github推荐的计算库，ent是近似熵，vectapen是样本熵；

## 特征预测模块

*from sklearn import svm*

*from sklearn.decomposition import PCA*

*from sklearn.neural\_network import MLPClassifier*

svm、MLPClassifier 、PCA分别是机器学习库的支持向量机、多层感知机和主成分降维的包；

*from sklearn.model\_selection import validation\_curve # 模型评估 : 不同内核系数gamma*

*from sklearn.model\_selection import learning\_curve # 模型评估 : 学习率曲线*

*from sklearn.model\_selection import cross\_val\_score # 模型评估 : 不同正则化系数C*

*from sklearn.model\_selection import LeavePOut # 留P法, K折交叉验证中K=n-p的情况,即剩下p个测试集*

*from sklearn.model\_selection import LeaveOneOut # 留1法, K折交叉验证中K=n(样本数)的情况*

*from sklearn.model\_selection import KFold # K折交叉验证*

*from sklearn.model\_selection import train\_test\_split # 留出法的分割方式*

*from sklearn.model\_selection import ShuffleSplit# ShuffleSplit方法，可以随机的把数据打乱，然后分为训练集和测试集*

上述来自于*sklearn.model\_selection*的包是模型用到的一些评估参数和方法；

*from sklearn.kernel\_approximation import (RBFSampler, Nystroem) # 内核近似*

*from sklearn import pipeline*

内核近似包括奈斯特罗估计和蒙特卡洛估计，用于衡量SVM算法的运行时间和准确率，使用内核近似需要管道机制，引入*pipeline*

*from sklearn import metrics*

metrics是评价参数表，提供例如精准率、召回率、召回率、模糊矩阵等评价指标；

*from sklearn.preprocessing import LabelEncoder*

独热编码*LabelEncoder*用于将数据标签自动编码，例如中文的标签欠压、过压、卡涩等可以自动编码成0，1，2，但需要注意的是未必按照顺序编号，也可能是3，1，2，这涉及到在后续界面得到预测概率后对概率位置进行调整

## 其他模块

*from enum import Enum*

枚举类，用于定义单元格的类型使用，即两个类的定义，class cellType(Enum)和class columnIndex(Enum)；

*import random*

随机数模块，用于随机打乱数据集，同时按相同规律打乱标签；

*import os*

*import sys*

系统模块和路径模块，一般获取相对路径进行使用

*import numpy as np*

*import pandas as pd*

*import math*

*import time*

基本的数组模块numpy；读取文件的模块pandas；数学模块math；时间模块，用来计算模型训练时间；

*import seaborn as sns*

用于绘制核密度估计图使用的绘图库；

*from scipy.fftpack import fft*

用于频谱分析和EMD分解用到的库；

*from matplotlib.font\_manager import FontProperties*

*import matplotlib.pyplot as plt*

*from matplotlib.pylab import mpl*

*plt.rcParams['font.sans-serif'] = ['Times New Roman'] # 设置字体风格,必须在前然后设置显示中文*

*mpl.rcParams['font.size'] = 10.5 # 图片字体大小*

*mpl.rcParams['font.sans-serif'] = ['SimHei'] # 显示中文的命令*

*mpl.rcParams['axes.unicode\_minus'] = False # 显示负号的命令*

*mpl.rcParams['agg.path.chunksize'] = 10000*

*plt.rcParams['figure.figsize'] = (7.8, 3.8) # 设置figure\_size尺寸*

*plt.rcParams['savefig.dpi'] = 600 # 图片像素*

*plt.rcParams['figure.dpi'] = 600 # 分辨率*

*font\_set = FontProperties(fname=r"C:\Windows\Fonts\simsun.ttc", size=10.5)*

上述代码用于定义输出的图片格式、输出精度、尺寸等

# 登录界面

## \_\_init\_\_

初始化函数，主要实现：

1. 密码的加密模式*self.ui.password.setEchoMode(QLineEdit.Password)*
2. 设置背景图片和颜色*pix = QPixmap(path)*，*palette = QPalette()*，path为图片的相对路径
3. 定义登陆状态的发射信号*signState = pyqtSignal(bool,bool)*，用于发射登陆用户名和密码正确与否的bool值，在槽函数on\_sure\_Button\_clicked当中发射，即界面点击确定按钮就会发射信号，在主界面中进行接收，这个接收的槽函数只能在主界面编写，不能写在登陆界面中

## checkUsernameIsTrue

检查用户名是否正确。

bool函数，正确返回True，错误返回Fasle

## checkPasswordIsTrue

检查登录密码是否正确。

bool函数，正确返回True，错误返回Fasle

## on\_sure\_Button\_clicked

*self.signState.emit(self.checkPasswordIsTrue(), self.checkUsernameIsTrue())*

登录界面，确定按钮点击时发射自定义的信号，该信号携带了用户名和密码是否正确的信息

## \_\_del\_\_

连接close函数，关闭界面

*self.close()*

# 主界面

## 枚举类型

枚举类型事先需要导入枚举类*from enum import Enum*，自定义枚举类型需要继承*Enum*

### class cellType(Enum)

*parameter\_name\_type = 1*

*parameter\_value\_type = 2*

表格有添加项的操作，这个项的类型有2种，即参数名字项类型和参数大小项类型，表格创建项的函数为*QTableWidgetItem*，第1个参数是传入的数据，例如参数名字类型就传入均值、方差等，参数值类型就传入具体的数字，第2个参数需要指定创建的项类型，也就是本类中定义的枚举类型

### class columnIndex(Enum)

*parameter\_name\_num = 0*

*parameter\_value\_num = 1*

虽然规定了创建项的类型，但是对于同一行而言，存在2列，所以还需要指定该项添加的位置，否则可能会在参数值列添加了参数名字类型的项，这种情况是要避免的，在*setItem*函数中，第1个参数指定行数，第2个参数指定列数，所以只能定义0和1，定义1和2表格就会添加项到2，3列而导致出错，第3个参数指定项类型，项类型没有规定有区分度即可

## 前端函数

### \_\_init\_\_

初始化函数，主要实现：

1. 调用初始化comboBox的函数init\_comboBox\_text\_icon，主要用于实现设置下拉条的文字、图标等
2. 调用初始化波形显示的函数init\_waveform，用于登录主界面时就能显示相应的波形，默认显示正常分合闸220V的电流波形，init\_waveform函数中调用了设置波形参数的函数set\_waveform\_parameters，此函数决定了导入的波形类型为电流或是振动波形
3. 调用初始化菜单栏和动作的关系函数init\_menuAction\_buttonClicked，动作是动态创建的，需要将动作的triggered信号和pushbutton的按钮clicked动作联系起来
4. 调用初始化其它界面组件属性init\_other\_properties函数对有关属性进行初始化
5. 调用动态初始化菜单函数init\_widget\_menu，希望在主界面刚出现时重载菜单是禁用状态，在点击过训练或者加载过波形后才能使用，这要求实现有个全局动作属性*self.init\_action*
6. 初始化波形时域特征的表格

### init\_comboBox\_text\_icon

函数编号：1

初始化ComboBox的文字内容和图标

*path = ":/images/" + str(i + 1) + ".jpg"* 使用资源文件中的路径，图标文件已按顺序0~15进行命名，循环导入路径的图标即可

*self.ui.comboBox.setCurrentText("线圈正常220V")*

默认界面初始化的线圈正常220V的波形，文字也显示为当前项

### init\_waveform

函数编号：2

此函数用于设置界面出现时显示的正常电流波形，*option*已经指定为*"220"*

进入函数首先进行判断，利用set\_waveform\_parameters函数成功时返回的*True*或*False*提示相应的信息

同时还将提示信息显示到状态栏*statusbar*上，状态栏是一个容器，具体的载体需要*QLabel*定义，然后*self.ui.statusbar.addWidget(label)*即可将载体显示化在窗体的状态栏上

### init\_menuAction\_buttonClicked

函数编号：18

此函数用于初始化菜单栏和动作的关系，动作本身可以在ui界面设计时就设置好槽机制，也可以动态的设置动作。函数中初始化了4个动作，对应菜单栏中命令栏的加载波形、开始训练和取消训练，以及退出栏中的退出窗口，这些动作的triggered信号需要与按钮pushbutton的clicked动作connect

*self.ui.load\_action.triggered.connect(self.ui.pushButton.clicked)*

### init\_other\_properties

函数编号：9

此函数用于初始化其它界面上所有的组件属性。

1. 初始化两个算法选择按钮SVM和MLP的使能为True，同时设置MLP的Checked状态为True，SVM的为False，即默认使用MLP算法
2. 训练次数设置最小1次，最大10000次，默认训练次数为10次；最大交叉验证次数不能低于2，不能大于数据集本身的大小，例如电流只有241组数据集，最多交叉验证241次；交叉验证次数默认0或1次，这意味着不能使用交叉验证，而改用留出法即train\_test\_spilit方法分割训练测试集
3. 初始化进度条，最大值为当前界面输入的训练次数，最小值为0，采用百分比显示进度，即*self.ui.progressBar.setFormat("%p%")*，初始化的当前值为50%
4. 初始化按钮图标，包括2个算法选择按钮、3个pushbutton按钮(开始训练、取消训练和加载波形)、4个菜单栏的动态动作，以及主界面本身的图标
5. 初始化菜单栏格式，包括字体风格、大小等
6. 初始化主界面的背景图片和格式(背景透明)，主界面的标题
7. 由于加入了背景图片，界面QLabel的相关文字都会透明而看不到，还需要设置QLabel的文字颜色，*self.ui.kfold\_label.setStyleSheet(*

*"color:rgb({},{},{},255)".format(255, 255, 255))*

1. 除了颜色以外也可以设置其他属性，代码类似于html的风格，这些可以在ui界面上找到QLabel的StyleSheet手动设置属性，然后生成代码，如果希望动态初始化只需要将这些代码复制即可，最后还初始化了QLabel的文字说明

### init\_widget\_menu

函数编号：19

动态初始化菜单，希望在主界面刚出现时重载菜单是禁用状态，在点击过训练或者加载过波形后才能使用；

初始化的动作必须在\_\_init\_\_中定义，这个动作的triggered信号与界面的槽函数init\_myplatform进行connect，这个函数可以析构掉当前窗口并初始化新窗口，但动作不能在本函数中定义；

原因说明如下：

1. 如果在本函数中定义了动作action，那么也需要在本函数中先设置action的enabled和checked为False，才能保证刚进入本函数时是禁用状态；然后想要根据事先定义的属性*self.have\_loaded\_waveform*和*self.have\_trained\_model*判定是否改变禁用状态，但是问题在于界面点击了训练或加载波形按钮之后，两个属性虽然发生了改变，但是这种改变无法传递到本函数，因为本函数只在\_\_init\_\_被执行一次，之后没有再执行，进而也无法影响到本函数内部的动态菜单是否禁用的代码；
2. 两个属性的改变想要影响到动态动作的使能，就必须找到某种方式重复执行本函数来改变动作，或者这个动作就必须是*self.ui*下的实体动作，而不能是本函数的局部动态动作；但对于第一个解决方案，本函数重复执行创建的动态动作与第一次执行创建的动作并不是一个，所以最终还是要求有一个实体的动作，保证在其他函数改变的是同一个动作的使能，即*init\_action*
3. 有了全局意义的实体动作之后，就可以根据两个属性的变化设置其使能状态，为了让动作能跟随属性变化，需要在开始训练函数on\_start\_training\_clicked和加载波形函数on\_pushButton\_clicked中进行判断，以on\_pushButton\_clicked为例，一旦加载波形就立刻将属性*self.have\_loaded\_waveform = True*，然后将重载菜单栏解除禁用，on\_start\_training\_clicked同样，一旦开始点击开始训练进入函数就立刻将属性*self.have\_trained\_model = True*，然后解除重载菜单栏的禁用

### init\_myplatform

函数编号：20

析构掉当前窗口，初始化新窗口，作为重载菜单栏动作动作的槽函数

### init\_table\_time\_features

函数编号：6

初始化计算波形时域特征的表格

在ui界面中这个表格的名字为*self.ui.tableWidget*，首先对*tableWidget*组件进行表格属性的相关设置，包括交替行背景、行高、列宽，其中设置列宽需要依赖于表格的枚举类型，这是因为表格有2列，即参数名字列和参数大小列，例如均值和0.5，对于同一行来说两列的区分需要枚举类型，枚举类型在class columnIndex(Enum)中定义，然后设置表格列数和表头以及表格是否可编辑，设置表头的核心代码为*self.ui.tableWidget.setHorizontalHeaderItem(i, headerItem)*

表格设置行数和行内容，行数取决于计算的波形参数，这里计算了15个，从均值(mean)到峰值因子(crest\_factor)，得到*features*

由于是用来第一次初始化表格的时域参数，故调用计算波形特征的函数calculate\_waveform\_characteristics时传入的option为220，type为电流

然后对表格的每行进行循环，使用的循环条件为表格的函数，即*self.ui.tableWidget.rowCount()*，每行要创建2种类型的项，同时指定项创建的位置，这两个设置需要用到class columnIndex(Enum)和class cellType(Enum)，创建项用到*QTableWidgetItem(index,column,type)*，表格添加项需要用到函数*setItem(index,column,item)*，其中type由class cellType(Enum)指定，column由class columnIndex(Enum)指定，同时参数名字类型的项还存储了个人用户数据，即参数名称，参数值类型的项则是存储了*features*对应的值

同时本函数还要检测时域特征表格初始化是否成功，使用最后一行存储的信息是否正确进行判断，成功要将初始化属性*self.\_init\_tableWidget = True*并返回*True*，如果不修改属性就会导致comboBox每次选择时波形重复初始化失败返回*False*，并有提示信息

### init\_current\_tableWidget

函数编号：10

与init\_table\_time\_features只有2个区别，一个是调用了计算预测概率的函数return\_current\_predict\_mean\_proba\_mlp而不是计算特征features，另一个设置修改的属性为*self.\_init\_current\_tableWidget = True*，性质与*self.\_init\_tableWidget = True*是相同的，原因不再赘述

另外由于是初始化，故使用的预测函数为默认的MLP算法

### init\_vibration\_tableWidget

函数编号：11

与函数init\_current\_tableWidget是类似的，区别在于是初始化振动调用的函数为return\_vibration\_predict\_mean\_proba\_mlp，修改的表格初始化属性为*self.\_init\_vibration\_tableWidget = True*，内部考虑到设置的标签不是按顺序，*y = [6, 5, 1, 7, 0, 4, 3, 2]*，所以需要对预测的概率位置进行调整

### on\_pushButton\_clicked

函数编号：5

加载波形的点击按钮函数

1. 进入本函数后首先要注意在init\_widget\_menu提到的注意事项，即进入函数后就立刻先将全局属性*self.have\_loaded\_waveform = True*。
2. 然后开始检测当前comboBox的当前项编号，利用了下拉栏的触发信号*self.ui.comboBox.currentIndex()*，判断加载的波形类型代号，这里是整型代号，由于在spilit\_current\_waveform\_data和spilit\_vibration\_waveform\_data已经定义了波形代号*option*是个字符串，所以需要先转换整型编号到字符串编号，利用了convert\_faultNumber\_intToString函数实现
3. 转换后的字符串编号根据整型编号分为电流和振动两类，转换成功时然后进入不同的判断，然后将字符串编号作为*option*参数传入set\_waveform\_parameters函数，相应的加载的波形标题也根据相应的*self.ui.comboBox.currentText()*进行设置，同时状态栏显示信息加载波形成功；如果一开始字符串编号就转换失败，则提示波形加载失败

### on\_start\_training\_clicked

函数编号：16

开始训练的点击按钮函数

进入本函数后首先将属性*self.have\_trained\_model = True*，还要将属性*self.cancel\_training\_flag = False*，即一旦点击了开始训练说明取消训练的状态为假，该属性在每一次训练中都会进行检测，如果点击了取消训练这个属性就为真，那么就会停止训练；点击取消训练on\_cancel\_training\_clicked后这个属性还是为真的状态，如果想能够重新开始训练，就必须进入的时候将这个属性先置为假，否则进入训练就退出

然后检测radioButton的checked状态，依据此决定使用何种算法，这两个算法分别对应自定义的槽函数start\_svm\_training()和start\_mlp\_training()

### on\_cancel\_training\_clicked

函数编号：17

*self.cancel\_training\_flag = True*

点击了取消后就将当前属性改为真，那么训练中扫描到会退出

### on\_comboBox\_currentIndexChanged

函数编号：4

comboBox的当前项索引改变的信号，根据项的变化显示到状态栏；同时索引的整型编号转换为字符型编号，调用函数set\_table\_values用于设置时域特征表格的值，此函数需要type和option指定，正好都可以由comboBox的索引变化来决定，其中type由索引在0~7或8~15来决定，option则是由convert\_faultNumber\_intToString转换后的字符串编号决定，然后更新时域特征表格的数据

此函数最重要的注意事项是*self.\_init\_tableWidget = True*，即要将初始化表格的属性设置为真，因为一旦初始化过，不要再进行初始化，故当comboBox当前项变化时也要保证*self.\_init\_tableWidget = True*

在\_\_init\_\_函数中开始初始化时域特征表格时，定义了*self.\_init\_tableWidget = False*，这是默认没有进行过初始化的意思，没有初始化时才会调用初始化时域特征表格的函数init\_table\_time\_features，初始化过后这个函数会将*self.\_init\_tableWidget = True*

### on\_tableWidget\_currentCellChanged

函数编号：7

用于检测鼠标放在时域特征表格的位置，该过载信号携带了当前单元格行列和前一行列的4个位置信息，然后利用位置信息获取当前单元格的存储的信息，即参数名或参数值，然后显示到状态栏上

进入本函数会首先判断是否初始化过表格，没有则进行初始化，否则利用初始化后的信息显示到状态栏

### on\_current\_predict\_tabWidget\_currentCellChanged

函数编号：12

类似于on\_tableWidget\_currentCellChanged函数，也用于显示状态栏信息，区别在于判断的属性为*self.\_init\_current\_tableWidget*

### on\_vbriation\_predict\_tabWidget\_currentCellChanged

函数编号：13

类似于on\_tableWidget\_currentCellChanged函数，也用于显示状态栏信息，区别在于判断的属性为*self.\_init\_vibration\_tableWidget*

### set\_waveform\_parameters

函数编号：3

设置波形参数函数，参数有*type, title, option=None, init=False*

:param init: 函数进入先一级判断是否是初始化波形函数init\_waveform中调用，如果不是初始化，则设置*init=False*

:param type: 一级判断结束后，继续二级判断*type == "电流"或type == "振动"*，选择调用的波形类型，特别的在*init=True*时不需要再写振动信号的相关代码，只需要*pass*即可

:param option:二级判断结束后，还要选择什么样的故障类型，*option*为string型，默认规定的*option*在拆分不同故障电流spilit\_current\_waveform\_data函数中的字典keys值规定为*keys = ["160", "180", "200", "220", "age\_10", "age\_50", "stuck", "240"]*，拆分不同故障振动spilit\_vibration\_waveform\_data函数中同样规定了键值*keys = ["正常合闸","正常分闸", "虚假合闸", "分闸不彻底","储能弹簧卡涩","操作机构卡涩","合闸螺栓松动", "分闸螺栓松动"]*

:param title:图表标题，将comboBox当前项显示的文字加上振动或者电流信号，*title=self.ui.comboBox.currentText()+"振动信号"或+"电流信号"*

主要实现：

1. 首先设置通用的图表主题、图例文字大小和位置、图表动画效果，曲线的线型、颜色、宽度等格式
2. 当*init=True*时，且当*type == "电流"*时调用spilit\_current\_waveform\_data函数得到指定*option="220"*的正常电流波形，即*current\_normal*，由于是初始化函数调用故只需要显示1个波形，之后以指定的步长循环添加点point到线型类的实例*y = QLineSeries()*，由于电流为50ms，所以设置步长为0.05/1100，然后设置曲线的范围，文字大小和类型等，返回*True*
3. 当*init=True*时，且当*type == "振动"*时*pass*掉，如果*type*既不是振动也不是电流会有提示信息*print("类型type只能输入<电流>或<振动>！")*，并返回*False*
4. 当*init=False*时，且当*type == "电流"*时，需要同时显示故障电流波形和正常电流波形方便比对，所以定义了新的线型类实例*y0 = QLineSeries()*，同样的方法调用spilit\_current\_waveform\_data函数得到指定*option*的故障电流波形，返回*True*
5. 当*init=False*时，且当*type == "振动"*时，由于振动信号的正常状态有正常分闸和正常合闸两种，所以不需要同时显示正常和故障振动信号，使用实例*y = QLineSeries()*即可，在这里调用的是spilit\_vibration\_waveform\_data函数，设置显示的x轴范围取决于读取的波形点数和采样率，振动采样率为100k，所以当波形长度为10k时，x轴范围对应的时间为0.1s，那么步长只需要设置为i/N即可，到最后一个点恰好是0.1s，最后同样返回*True*

### convert\_faultNumber\_intToString

函数编号：26

转换整型编号0~15至字符串编号

"160","180","200","220","stuck","age\_10","age\_50","240","正常合闸","正常分闸","虚假合闸","分闸不彻底","储能弹簧卡涩","操作机构卡涩","合闸螺栓松动","分闸螺栓松动"

转换成功则返回转换后的字符串，转换失败则返回0

### signWindowIsSuccess

函数编号：22

*from MyPlatform.mySignWindow import mySignWindow*

在主界面的\_\_init\_\_函数中，主界面继承了登录窗口类

*self.mySignWindow = mySignWindow(self)*

同时在登陆界面的\_\_init\_\_函数已经定义了信号*signState = pyqtSignal(bool,bool)*，这个信号随着登录界面的继承也一并继承，那么这个信号需要连接槽函数，这个函数必须在主界面的程序中定义，不能再登陆界面的程序定义，否则信号不能响应

即*self.mySignWindow.signState.connect(self.signWindowIsSuccess)*

该信号连接了槽函数，也就是本函数signWindowIsSuccess，此函数是过载信号，携带了用户名和密码是否正确的信息，都正确则析构掉登录窗口并弹出成功消息对话框，之后利用*self.show()*让主界面出现，并提示到主界面的状态栏信息登陆成功；有错误则弹出错误对话框，则析构掉当前登录窗口重新实例化弹出新的登录窗口，并提示到登陆界面的状态栏登陆失败，对话框由对话框类*QMessageBox*进行设置

### signDialogIsSuccess

函数编号：23

与signWindowIsSuccess的区别是这个登录界面继承的是Dialog，而前者继承的QMainWindow，但是这里只用了第一个登录界面，使用后者也是可以的

### \_\_del\_\_

函数编号：21

严格来说不是内存析构函数，只是关闭了界面

*self.close()*

## 后端函数

### 读取波形函数

#### read\_current\_waveform\_data

函数编号：24

读取项目文件夹下的电流波形数据，此数据的存在形式为1100×8，即每一种类型的电流波形

*path = os.getcwd() + "/MyPlatform/current\_waveform\_data.csv"*

#### spilit\_current\_waveform\_data

函数编号：25

用到了read\_current\_waveform\_data函数，利用enumerate函数(可以同时给出索引和取值)，将keys中的名称key和读取的波形每列数据作为value进行对应，存储到字典当中，利用字典是因为可以方便的返回option指定的波形，而option就是key，利用key就可以得到对应的波形数据value

#### read\_current\_waveform\_pca\_features

函数编号：28

读取电流的PCA数据current\_pca\_features\_data.csv，注意要以gbk编码读取，使用的路径为相对路径，利用了*os.getcwd()*

#### split\_current\_pca\_features\_data\_label

函数编号：29

调用read\_current\_waveform\_pca\_features函数，分离数据和标签得到X，y

#### spilit\_current\_pca\_features\_data

函数编号：30

继续分离电流信号的pca特征的数据得到8种故障类型的pca用于模型预测，split\_current\_pca\_features\_data\_label函数只是分离数据和标签，本函数则是对数据继续分离得到8种类型的PCA数据，存储到字典当中

#### read\_vibration\_waveform\_data

函数编号：46

读取项目文件夹下的振动波形数据，此数据的存在形式为\*×8，即每一种类型的振动波形，\*是因为有的类型是4500个点，有的10000个点，不是电流信号统一的1100个点

*path = os.getcwd() + "/MyPlatform/vibration\_waveform\_data.csv"*

#### spilit\_vibration\_waveform\_data

函数编号：47

同spilit\_current\_waveform\_data函数，不赘述

#### read\_vibration\_waveform\_pca\_features

函数编号：48

同read\_current\_waveform\_pca\_features函数，不赘述

#### split\_vibration\_pca\_features\_data\_label

函数编号：49

逻辑等同于split\_current\_pca\_features\_data\_label，不再赘述

#### spilit\_vibration\_pca\_features\_data

函数编号：50

逻辑等同于spilit\_current\_pca\_features\_data，不再赘述

#### read\_txtdata\_to\_series

函数编号：69

*:param path:* txt文件路径

*:param num:*文件名的数字编号，一般调用此函数前先将文件夹下所有txt文件名改为顺序数字

*:return:* series

本函数是读取指定路径的文件名为数字的文本文件，因为振动信号当初得到的文件是txt文件，所以必须先读取txt，然后转换成方便处理的series格式

此函数在read\_all\_txtdata\_to\_dataframe函数中调用

#### read\_all\_txtdata\_to\_dataframe

函数编号：70

此函数是基于read\_all\_txtdata\_to\_dataframe函数进一步封装的函数，可以同时处理多个文本文件，然后依次拼接得到的Series即可，然后得到dataframe格式，最后可以进行保存

#### read\_one\_dataframe

函数编号：73

此函数是为了读取某个故障类型的某组波形的emd分解结果，如正常合闸有10个波形，每个波形都会生成对应的emd文件，一般而言分解成6~8列IMF分量，这些csv文件已经按照数字编号进行了编号

本函数的参数是指定路径path和文件编号num，

### 数据处理函数



#### calculate\_waveform\_characteristics

函数编号：27

计算波形的时域特征，有option和type两个参数，指定振动或电流的某个故障，然后调用获取波形的函数传入option参数，spilit\_current\_waveform\_data或spilit\_vibration\_waveform\_data，然后计算波形的特征，返回列表features

#### set\_table\_values

函数编号：8

有2个参数，option和type，调用calculate\_waveform\_characteristics计算指定波形的时域特征并返回features

#### return\_cmaps

函数编号：66

返回颜色的RGB值，在绘图函数中被使用

#### data\_normalization

函数编号：68

数据归一化函数，参数选择有按行归一化，按列归一化，即mode=0或1

，如果需要反归一化则需要给定每个行或者每个列的最大和最小值组成的行向量，这是因为反归一化可能有无穷多种可能，只有限定了最值范围才是唯一的

#### fft\_data

函数编号：71

频谱分析函数，根据采样率和波形点数的变化得到对应的频谱图，主要是为了处理振动信号的频谱，同时鉴于振动信号开始时的突峰可能是异常的，所以在绘制时*plt.plot(half\_f[10:-1], half\_abs\_y[10:-1] / (N / 2)*头一些点是不要的

#### correlation\_coefficient

函数编号：74

计算两个Series的互相关系数，在函数imf\_signal\_correlation中被调用，用于计算原始信号和每个IMF分量之间的互相关系数

#### imf\_signal\_correlation

函数编号：75

计算原始信号和imf分量之间的互相关系数 得到相关系数corr的列表，列表长度6~8，论文统一使用6个，在调用本函数的calculate\_all\_dataframe\_features中会统一进行处理

#### signal\_correlation\_entropy

函数编号：76

计算某个故障波形的的互相关熵，输入参数为imf\_signal\_correlation函数得到的相关系数corr，由于得到的相关系数有可能出现为0的情况，那么计算互相关熵会用到ln(corr[i])，这可能会造成程序错误，对其的处理是先进行判断，如果为0跳过，不为0才进行计算

本函数也在calculate\_all\_dataframe\_features函数中被调用

#### calculate\_all\_dataframe\_features

函数编号：79

:param filepath: 存放csv文件夹的路径

:param csvpath: 存放原始波形的csv文件路径

:param title: 可视化energy、svd、corr的图标题

:return: 某个故障类型所有样本的特征, 所有样本svd、energy、corr平均值

首先列举指定filepath路径下所有的文件，代码为files = os.listdir(filepath)，这个路径举例存放了正常合闸10组样本对应的10组emd分解后的csv文件，而csvpath是正常合闸波形csv文件的路径，source\_data = pd.read\_csv(csvpath)，这个作为原始数据，存放了10组原始波形，用于后续计算与emd分解后的不同IMF分量的相关系数

然后开始对列举的每个csv文件进行循环，得到正常合闸第1组样本的原始波形数据为*column\_data = source\_data.iloc[:, i]*，得到该原始波形数据的emd分解结果为*emd\_data = self.read\_one\_dataframe(filepath, i + 1)*，调用了读取1个csv文件的函数read\_one\_dataframe，这样就得到了一个波形样本的原始数据和对应的emd结果

然后开始计算相关特征。

调用imf\_signal\_correlation函数计算互相关系数，这里注意不要残差，所以计算互相关系数占比时使用*proba\_corr = corrlation[0:-1] / sum(corrlation[0:-1])*，再将互相关系数传入signal\_correlation\_entropy函数，计算互相关系数熵。

考虑到统一只要前6个IMF分量，所以需要判断计算的互相关系数长度，等于6则直接存储，超过6则只要前6个，这样可以保证后续调用compare\_energy、compare\_svd和compare\_corr函数时传入的参数长度是一致的。

对于能量熵和奇异熵是类似的，也要判断长度与6的关系。

计算近似熵使用*vectapen.apen*

计算样本熵使用函数*ent.sample\_entropy*，但是会得到多个结果，取第1个结果为样本熵*sampleEntropy = sampleEntropys[0]*

计算排列熵使用*permutation\_entropy*

然后再计算相关的时域特征，存储到features\_dataframe，同时添加表头，在外部保存特征矩阵数据即可

#### compress\_features\_to\_pca

函数编号：83

主成分降维函数，可以得到方差贡献率和主成分散点图

主成分降维的核心代码语句为生成模型语句*model = PCA(n\_components=2)*和生成压缩数据语句*X\_new = model.fit\_transform(X\_Features)*

之后利用*sns.scatterplot*方法可以生成按类别的散点图，同时为了体现维度的方差贡献率还生成了4维的数据，并绘制条形图，核心语句是方差贡献率在柱子上的文字标注

### 数据预测函数

#### start\_svm\_training

函数编号：14

进入函数后，依据comboBox当前项的索引所处的范围利用函数convert\_faultNumber\_intToString转换为振动或电流的字符串编号，然后根据界面的输入*maxiter\_spinBox*和*kfold\_spinBox*的值，决定训练次数和交叉验证次数

预测的概率和花费时间，对于电流信号，首先调用了函数return\_current\_model\_traingtime，这是一个电流信号的通用函数，，主要是为了区分SVM和MLP算法，调用该函数会进入到两个对应的子函数分别返回SVM和MLP的概率和模型训练时间

然后将预测的概率按照索引去更新电流预测概率或振动预测该概率表格的值，也更新用户数据，同时还更新状态栏

状态栏信息需要用到故障类型和故障概率，这些信息取决于comboBox选择的当前的波形类型

对于振动调用的是return\_vibration\_model\_traingtime函数，类似于电流信号，有同样的逻辑关系，需要注意的是振动的预测概率要调换顺序，具体原因在init\_vibration\_tableWidget当中说明过，最后将信息打印到状态栏上

#### return\_current\_svm\_model

函数编号：31

返回电流信号的svm的普通训练模型，使用的留出法*train\_test\_split*，此函数是为了将得到模型的程序和利用模型预测的程序分开，降低代码之间的耦合性，这样可以在多个函数中都可以调用产生SVM模型的函数，而不必重复书写产生模型的代码

#### return\_current\_predict\_label\_svm

函数编号：32

此函数没有用上

根据选项option返回电流信号的svm预测的标签，即使用的*clf.predict*而非*clf.predict\_proba*函数，得到的预测的标签而不是概率，正好与函数编号33 return\_current\_predict\_proba\_svm相反

#### return\_current\_predict\_proba\_svm

函数编号：33

此函数没有用上

根据选项option返回电流信号的svm预测的概率矩阵，即使用的clf.predict\_proba而非clf.predict函数，得到的预测的概率而不是标签，与函数编号32return\_current\_predict\_label\_svm相反

#### return\_current\_predict\_mean\_proba\_svm

函数编号：34

此函数没有用上

对返回预测的电流信号的svm概率矩阵求平均，调用了return\_current\_predict\_proba\_svm函数

#### return\_current\_kfold\_predict\_proba\_svm

函数编号：35

用于内层循环

首先调用了分离电流PCA二维数据的函数spilit\_current\_pca\_features\_data，得到指定option的pca数据，再调用split\_current\_pca\_features\_data\_label函数得到分离后的数据和对应标签X和y

而交叉验证次数kfold是界面输入的参数传入的，在指定交叉验证次数下得到每一次的概率，然后存储到列表中，但是返回的是数组格式的按行平均值

注意的是为了保证随机性，需要引入代码*state = np.random.get\_state()，np.random.shuffle(X)*来获得数据集的状态并洗牌，同时以相同的规律对标签y进行洗牌

#### return\_current\_maxiter\_predict\_proba\_svm

函数编号：36

带有maxiter的svm的电流信号的普通训练模型，使用的划分测试训练集方法为*train\_test\_split*，用于交叉验证次数=0和1时调用时返回，

#### return\_current\_maxiter\_kfold\_predict\_proba\_svm

函数编号：37

用于外层循环

本函数用于计算电流信号下，maxiter次(外循环)，kfold次交叉验证(内循环)的预测概率，在进入训练时需要先判断kfold的大小，如果大于2说明使用带交叉验证的算法函数return\_current\_kfold\_predict\_proba\_svm，否则使用不带交叉验证的，即使用留出法进行划分测试训练集的算法函数return\_current\_maxiter\_predict\_proba\_svm，这个函数可以看出区别在于没有kfold参数，既不使用交叉验证选项

在循环中注意的是利用训练次数才显示进度条的当前值，并循环扫描属性*self.cancel\_training\_flag*检测是否点击了取消训练，如果点击就提示状态栏信息并返回；其次进度条引入了计时器，*counter = QTime()*，在每次训练时都会*counter.start()*计数一次，但是界面会产生卡顿，卡顿的解决方案是添加代码*QApplication.processEvents()*，可以持续刷新界面

最后要注意return\_current\_kfold\_predict\_proba\_svm返回的预测概率*predict\_kfold\_proba\_mean*本身虽然是一个行向量，但它是kfold次下交叉验证得到的概率矩阵的按行平均值，另外由于还要进行外循环maxiter次，所以被添加到*Predict\_proba*当中，这个是maxiter次下kfold次交叉验证的平均值，实际上还是个概率矩阵，故返回的时候还要继续按行平均

简单来说真正的单次练是一次训练的一次交叉验证，那么总共训练maxiter\*kfold次，为了最终显示到状态栏只能是行向量，那么就必须先被平均kfold次，再被平均maxiter次

平均的时候使用*np.array(Predict\_proba).mean(axis=0)*，必须先转换成数组，因为列表没有直接mean的方法

#### start\_mlp\_training

函数编号：15

函数的逻辑和start\_svm\_training是相同的，在电流和振动分别用到的函数也一样，即return\_current\_model\_traingtime和return\_vibration\_model\_traingtime

#### return\_current\_mlp\_model

函数编号：38

与return\_current\_svm\_model函数相同，区别返回的是电流信号的mlp的普通模型

#### return\_current\_predict\_label\_mlp

函数编号：39

函数没用上

与函数return\_current\_predict\_label\_svm相同，根据选项option返回电流信号的mlp预测的标签，与return\_current\_predict\_proba\_mlp不同，返回的是预测标签

#### return\_current\_predict\_proba\_mlp

函数编号：40

与return\_current\_predict\_proba\_svm函数相同

与return\_current\_predict\_label\_mlp函数相反

根据选项option返回电流信号的预测的mlp概率矩阵

#### return\_current\_predict\_mean\_proba\_mlp

函数编号：41

调用了return\_current\_predict\_proba\_mlp函数得到的概率矩阵求平均，本函数在初始化电流预测表格init\_current\_tableWidget当中被调用，这是因为初始化默认使用的MLP算法

基于此，所以相比于return\_current\_predict\_mean\_proba\_svm函数没被使用，本函数使用到了

#### return\_current\_kfold\_predict\_proba\_mlp

函数编号：42

用于内层循环

与return\_current\_kfold\_predict\_proba\_svm函数逻辑关系一致，区别是使用的MLP算法

#### return\_current\_maxiter\_predict\_proba\_mlp

函数编号：43

带有maxiter的mlp的电流信号的普通训练模型，使用的划分测试训练集方法为*train\_test\_split*，用于交叉验证次数=0和1时调用，

与return\_current\_maxiter\_predict\_proba\_svm函数逻辑是一致的

#### return\_current\_maxiter\_kfold\_predict\_proba\_mlp

函数编号：44

用于外层循环

此函数的逻辑与return\_current\_maxiter\_kfold\_predict\_proba\_svm是完全一致的，区别只在于使用的算法是MLP算法

交叉验证次数kfold大于等于2时调用函数return\_current\_kfold\_predict\_proba\_mlp，否则调用没有kfold参数的函数return\_current\_maxiter\_kfold\_predict\_proba\_mlp

#### return\_current\_model\_traingtime

函数编号：45

此函数有4个参数，*model, maxiter, kfold, option*，分别表示在电流信号的前提下，选择SVM或MLP模型、训练次数、交叉验证次数和预测的波形类型，这些参数在model的判断下分别进入到两个函数中计算预测的概率，即

return\_current\_maxiter\_kfold\_predict\_proba\_svm

和return\_current\_maxiter\_kfold\_predict\_proba\_mlp

返回用MLP和SVM算法得到的maxiter次训练次数(外循环)且kfold次交叉验证(内循环)下的预测概率

#### return\_vibration\_svm\_model

函数编号：51

类似于return\_current\_svm\_model函数，逻辑相似，不再赘述

#### return\_vibration\_predict\_label\_svm

函数编号：52

函数用不上

类似于return\_current\_predict\_label\_svm函数，得到的是预测标签，不同于return\_vibration\_predict\_proba\_svm函数，逻辑相似，不再赘述

#### return\_vibration\_predict\_proba\_svm

函数编号：53

函数用不上

类似于return\_current\_predict\_proba\_svm函数，得到的是预测概率，不同于return\_vibration\_predict\_label\_svm函数，逻辑相似，不再赘述

#### return\_vibration\_predict\_mean\_proba\_svm

函数编号：54

函数用不上

类似于return\_current\_predict\_mean\_proba\_svm函数，逻辑相似，不再赘述

#### return\_vibration\_kfold\_predict\_proba\_svm

函数编号：55

用于内层循环

与函数编号35对应的函数return\_current\_kfold\_predict\_proba\_svm逻辑是一致的，区别在于是返回振动信号的概率，内部调用的分离PCA数据和标签X，y的函数分别是

spilit\_vibration\_pca\_features\_data和split\_vibration\_pca\_features\_data\_label

#### return\_vibration\_maxiter\_predict\_proba\_svm

函数编号：56

与函数编号36对应的函数return\_current\_maxiter\_predict\_proba\_svm逻辑是一致的

返回带有maxiter的svm的振动信号的普通训练模型，用于交叉验证次数=0和1时调用，不再赘述

#### return\_vibration\_maxiter\_kfold\_predict\_proba\_svm

函数编号：57

用于外层循环

与return\_current\_maxiter\_kfold\_predict\_proba\_svm函数逻辑一致，区别是对振动信号进行预测

#### return\_vibration\_mlp\_model

函数编号：58

与return\_current\_predict\_label\_mlp函数类似，逻辑相似，不再赘述

#### return\_vibration\_predict\_label\_mlp

函数编号：59

与return\_current\_predict\_label\_mlp函数类似，逻辑相似，不再赘述

#### return\_vibration\_predict\_proba\_mlp

函数编号：60

与return\_current\_predict\_proba\_mlp函数类似，逻辑相似，不再赘述

#### return\_vibration\_predict\_mean\_proba\_mlp

函数编号：61

与return\_current\_predict\_mean\_proba\_mlp函数类似，逻辑相似，在初始化振动波形预测概率函数init\_vibration\_tableWidget中用到，因为默认使用的MLP算法不再赘述

#### return\_vibration\_kfold\_predict\_proba\_mlp

函数编号：62

用于内层循环

与return\_current\_kfold\_predict\_proba\_mlp函数类似，逻辑相似，不再赘述

#### return\_vibration\_maxiter\_predict\_proba\_mlp

函数编号：63

与return\_current\_maxiter\_predict\_proba\_mlp函数类似，带有maxiter的mlp的振动信号的普通训练模型用于交叉验证次数=0和1时调用返回，逻辑相似，不再赘述

#### return\_vibration\_maxiter\_kfold\_predict\_proba\_mlp

函数编号：64

用于外层循环

与return\_current\_maxiter\_kfold\_predict\_proba\_mlp函数类似，逻辑相似，不再赘述

#### return\_vibration\_model\_traingtime

函数编号：65

与return\_current\_model\_traingtime的逻辑一致，区别是返回振动信号的预测概率和训练时间，两类不同算法分别调用的函数是

return\_vibration\_maxiter\_kfold\_predict\_proba\_svm

和return\_vibration\_maxiter\_kfold\_predict\_proba\_mlp

#### return\_mlp\_neural\_network

函数编号：91

得到某个训练集比例的mlp基础模型预测概率，这样可以避免代码无意义的重复，如果需要得到模型，只需要传入参数 *clf, X, y, trainsize=0.7*即可

在get\_mlp\_acu和get\_mlp\_acu\_trainsize中调用，是最基本的函数

#### get\_mlp\_acu

函数编号：92

得到maxiter次下固定训练比例的mlp的预测准确率，这是因为对于return\_mlp\_neural\_network函数是没有参数maxiter，本函数在此函数基础上添加了maxiter参数，作为基础函数用于比较不同训练次数使用

#### get\_mlp\_acu\_trainsize

函数编号：93

此函数是在return\_mlp\_neural\_network函数基础上，添加了训练比例transize参数，为了比较不同训练比例的准确率，也是基础函数

#### ChangeSovler

函数编号：94

改变梯度优化器lbfgs、adam、sgd

使用了get\_mlp\_acu函数获得预测准确率，调用了PlotACU函数绘制不同优化器的准确率

#### ChangeActivation

函数编号：95

改变激活函数 identity、logistic、tanh、relu

使用了get\_mlp\_acu函数获得预测准确率，调用了PlotACU函数绘制不同优化器的准确率

#### ChangeAlpha

函数编号：96

改变正则化系数*alpha*

*Alphas = [1e-5,1e-4,1e-3,1e-2,1e-1,0.5,1,5,10,20,50,80,100]*

使用了get\_mlp\_acu函数获得预测准确率，调用了PlotACU函数绘制不同优化器的准确率

#### ChangeLearingRate

函数编号：97

改变学习方法 constant or invscaling

使用了get\_mlp\_acu函数获得预测准确率，调用了PlotACU函数绘制不同优化器的准确率

#### ChangeConstantLearningRateInit

函数编号：98

固定学习率方式下*constant* 改变学习率

*Learning\_rate\_inits = [1e-5,1e-4,1e-3,1e-2,1e-1,0.5,1,5,10,20,50,80,100]*

使用了get\_mlp\_acu函数获得预测准确率，调用了PlotACU函数绘制不同优化器的准确率

#### ChangeInvscalingLearningRateInit

函数编号：99

逆学习率时*invscaling* 改变学习率

使用了get\_mlp\_acu函数获得预测准确率，调用了PlotACU函数绘制不同优化器的准确率

#### ChangeInvscalingPowert

函数编号：100

逆学习率时*invscaling* 改变逆缩放指数*power\_t*

*power\_ts = [1e-5,1e-4,1e-3,1e-2,1e-1,0.5,1,5,10,20,50,80]*

使用了get\_mlp\_acu函数获得预测准确率，调用了PlotACU函数绘制不同优化器的准确率

#### ChangeTrainingSampleRatio

函数编号：101

改变训练集比例，调用了get\_mlp\_acu\_trainsize函数获得预测准确率，调用了PlotTrainingSampleRatio函数绘制不同优化器的准确率

#### get\_svm\_acu\_onetraining

函数编号：108

svm模型固定maxiter次某个训练比例下的概率，在获取不同训练及比例概率get\_svm\_acu\_trainsize函数中调用，本函数默认训练100次

#### get\_mlp\_acu\_onetraining

函数编号：109

mlp模型固定maxiter次某个训练比例下的概率，在获取不同训练及比例概率get\_mlp\_acu\_trainsize函数中调用，本函数默认训练100次

#### get\_svm\_acu\_trainsize

函数编号：110

获得svm模型每个比例100次准确率的平均值、平均时间、该比例对应训练集个数，调用了get\_svm\_acu\_onetraining函数

#### get\_mlp\_acu\_trainsize

函数编号：111

获得mlp模型在maxiter次改变训练集比例，即某个transize下的准确率acu，反复调用基础函数get\_mlp\_acu\_onetraining得到

#### ChangeMLPTrainsize

函数编号：112

改变MLP模型的训练集比例 调用get\_mlp\_acu\_trainsize函数得到准确率平均值，然后调用plot\_mlp\_acu\_trainsize函数绘制不同训练集比例MLP模型的准确率变化

#### ChangeSVMTrainsize

函数编号：113

改变SVM模型的训练集比例 调用get\_svm\_acu\_trainsize函数得到准确率平均值，然后调用plot\_svm\_acu\_trainsize函数绘制不同训练集比例MLP模型的准确率变化

#### get\_svm\_acu\_trainsize\_maxiter

函数编号：116

svm模型改变训练次数(固定训练集比例)

#### get\_mlp\_acu\_trainsize\_maxiter

函数编号：117

mlp模型改变训练次数(固定训练集比例)

#### one\_training

函数编号：120

比较留出法的基础函数，一次训练得到的准确率，可以用来作为基础函数被反复调用

#### maxiter\_training

函数编号：121

比较留出法的基础函数，获得某个transize下的准确率ACU，作为基础函数反复调用，也调用了one\_training函数

#### fixed\_testsize

函数编号：122

比较留出法固定测试集比例的情况，调用了maxiter\_training函数获得平均准确率，然后调用plot\_acu\_fixed\_testsize函数进行绘制

#### fixed\_training\_num

函数编号：123

比较留出法固定训练次数的情况，调用了maxiter\_training函数获得平均准确率，然后调用plot\_acu\_fixed\_training\_num函数进行绘制

#### return\_leave\_p\_method\_acu\_training\_num

函数编号：126

返回留1法和留2法当前训练次数和当前的准确率变化，在leave\_p\_method函数中被调用生成训练次数和准确率

#### leave\_p\_method

函数编号：128

是return\_leave\_p\_method\_acu\_training\_num和plot\_leave\_p\_method\_acu函数的主函数，调用这两个函数得到留1或留2的准确率变化趋势图

#### make\_meshgrid

函数编号：129

比较不同核函数的pca数据边界，本函数用于生成曲面，在plot\_contours函数中被调用

### 可视化函数

#### plot

函数编号：67

函数不使用

一个统一的绘图格式

#### plot\_imfs



函数编号：72

用于可视化EMD分解后的IMF分量，本论文使用的是matlab进行emd分解，然后将分解后的结果导入，使用本函数进行可视化

在matlab中已经将每组波形按照类型得到了csv文件，读取这些csv文件的函数为read\_one\_dataframe

#### plot\_energy\_svd\_corr\_imf

函数编号：77

函数未用上

函数本意类似于plot\_energy\_svd\_corr，不过是绘制三种熵的，但是后来考虑到想把所有故障波形种类的熵都进行绘制，所以此函数弃用，而是使用了3个函数去进行比较，分别是compare\_energy、compare\_svd和compare\_corr，传入的参数为不同故障类型各自的10个样本波形熵的平均值，这样更具有客观性

#### plot\_energy\_svd\_corr

函数编号：78

可视化不同imf分解程度的emd能量、svd奇异值、corr相关系数，也就是说归一化emd能量随着IMF的关系、归一化svd奇异值随IMF关系、归一化corr相关系数随IMF的关系，只不过是画在一张图上

本函数在calculate\_all\_dataframe\_features函数中被调用，传入的值是某个故障类型所有波形样本都计算了emd、svd、corr之后并取平均的结果，这样更客观，而不是选取某一个样本进行绘制

#### compare\_energy

函数编号：80

样本参数为8种故障波形能量熵的平均值，通过给定不同的故障波形文件夹路径，调用函数calculate\_all\_dataframe\_features得到的返回值就是3种熵，将能量熵传入给本函数即可比较不同故障类型各自能量熵的平均值

#### compare\_svd

函数编号：81

样本参数为8种故障波形奇异熵的平均值，通过给定不同的故障波形文件夹路径，调用函数calculate\_all\_dataframe\_features得到的返回值就是3种熵，将能量熵传入给本函数即可比较不同故障类型各自奇异熵的平均值

#### compare\_corr

函数编号：82

样本参数为8种故障波形互相关熵的平均值，通过给定不同的故障波形文件夹路径，调用函数calculate\_all\_dataframe\_features得到的返回值就是3种熵，将能量熵传入给本函数即可比较不同故障类型各自互相关熵的平均值

#### my\_learning\_curve

函数编号：84

绘制SVM模型学习率曲线的函数，目的是计算指定的学习模型estimator在不同大小的训练集经过交叉验证后的训练的分和测试得分，为了得到较好的图像展示效果横轴的训练比例只进行五等分，即*train\_size=np.linspace(0.1, 1.0, 5)*

#### plot\_my\_learning\_curve

函数编号：85

调用函数84，即my\_learning\_curve函数，这是为了其它模型也能调用，否则要重复更改参数写代码

#### print\_evaluation\_index

函数编号：86

返回svm模型的评价指标表和混淆矩阵，由于交叉验证的缘故，每次都会生成混淆矩阵会造成内存不够用，所以使用参数*confusion\_matrix=False*进行控制，评价参数则全部来自于*metrics*模块

评价参数表可以选择是否保存，取决于*isSave*参数

#### plot\_kernel\_density\_estimation

函数编号：87

使用留出法绘制核密度估计图，也即最普通的*train\_test\_split*方法，一般默认训练比例为0.7，训练次数100次

#### compare\_nfold\_cross\_validation\_score

函数编号：88

调用了*cross\_val\_score*模块，比较不同正则化系数*alpha*交叉验证得分，一般固定交叉验证次数*n\_folds=10*，横坐标需要使用对数刻度*plt.semilogx*

为了解决刻度不能显示负幂的问题，解决方案是

*ax = plt.gca()*

*locs, labels = plt.yticks()*

*plt.yticks(locs, list(map(lambda x: "%g" % x, locs)))*

*for tick in ax.xaxis.get\_major\_ticks():*

*tick.label1.set\_fontproperties('stixgeneral')*

以上代码只需要复制粘贴即可，什么也不需要做

#### compare\_gamma\_cross\_validation\_score

函数编号：89

比较不同gamma系数的交叉验证得分

类似于88号函数compare\_nfold\_cross\_validation\_score，区别是调用的模块为*validation\_curve*

#### plot\_rbf\_sampler\_nystroem

函数编号：90

绘制奈斯特罗和蒙特卡洛估计

#### PlotACU

函数编号：102

输入参数只有迭代次数maxiter和准确率ACU，以及两个标题使用的字符串title1和title2

#### PlotAlpha

函数编号：103

绘制不同正则化系数的准确率变化趋势

#### PlotConstantLearningRateInit

函数编号：104

绘制固定学习率方式不同学习率的准确率变化趋势

#### PlotInvscalingLearningRateInit

函数编号：105

绘制逆学习率方式不同学习率的准确率变化趋势

#### PlotInvscalingPowert

函数编号：106

绘制逆学习率方式不同逆缩放指数的准确率变化趋势

#### PlotTrainingSampleRatio

函数编号：107

比较不同训练集比率，传入的参数为准确率，花费时间，训练代数和训练集的个数作为横坐标，在ChangeTrainingSampleRatio函数中被调用

#### plot\_svm\_acu\_trainsize

函数编号：114

绘制svm模型随训练比例的准确率变化

#### plot\_mlp\_acu\_trainsize

函数编号：115

绘制mlp模型随训练比例的准确率变化

#### plot\_svm\_acu\_maxiter

函数编号：118

绘制svm模型随训练次数的准确率变化

#### plot\_mlp\_acu\_maxiter

函数编号：119

绘制mlp模型随训练次数的准确率变化

#### plot\_acu\_fixed\_testsize

函数编号：124

绘制留出法，固定测试集比例的acu

#### plot\_acu\_fixed\_training\_num

函数编号：125

绘制留出法，固定训练次数的acu

#### plot\_leave\_p\_method\_acu

函数编号：127

绘制留1法或留2法的准确率变化图，在leave\_p\_method函数中被调用生成图像

#### plot\_contours

函数编号：130

比较不同核函数的pca数据边界热力图，调用了生成边界的make\_meshgrid函数