Escuela Universitaria
Politécnica - La Almunia
Centro adscrito
Universidad Zaragoza

ESCUELA UNIVERSITARIA POLITÉCNICA
DE LA ALMUNIA DE DOÑA GODINA (ZARAGOZA)

ANEXOS

Caracterización de baterías a partir de datos abiertos mediante técnicas de inteligencia artificial

Battery Characterization on the Basis Of Open Data Using Artificial Intelligence Techniques

425.21.86

Autor: Aida Martín Lacosta

Director: Tomás Cortés Arcos

Fecha: 06 2022

INDICE DE CONTENIDO

1. CÓDIGOS

1.1. BATERIAS.PY

```
# -*- coding: utf-8 -*-
"""
Created on Sat Jul 24 18:25:27 2021

@author: Aida Martin Lacosta
"""
#===========#
# Librerias #
#===========#

from pycaret.classification import *
import pandas as pd
import functools
import numpy as np
import matplotlib.pyplot as plt
from sklearn import svm
import itertools

#=============================#
# Inicializacion de Variables #
#=============================#

carpeta = "C:/Users/Aida/Documents/TFG IOI/Programas"

fichero1 = "/INR/11_05_2015_SP20-2_DST_50SOC.xls"
fichero2 = "/INR/11_05_2015_SP20-2_DST_80SOC.xls"
fichero3 = "/A123/A1-007-DST-US06-FUDS-25-20120827.xlsx"
fichero4 = "/A123/A1-008-DST-US06-FUDS-25-20120827.xlsx"
fichero5 = "/INR/11_09_2015_SP20-2_FUDS_50SOC.xls"
fichero6 = "/INR/11_06_2015_SP20-2_FUDS_80SOC.xls"

#===========#
# Funciones #
#===========#

def concatenar_lista_dataframes(lista):
    df = functools.reduce(lambda df1, df2: df1.append(df2, ignore_index =
True), lista).drop_duplicates()
    return pd.concat( [df, pd.DataFrame()], ignore_index = True)

def concatenar_hojas_excel(fichero, lst_indices_hojas):
    ex= pd.ExcelFile(fichero)
    return concatenar_lista_dataframes(ex.parse(ex.sheet_names[i]) for i
in lst_indices_hojas )

#=========================================================================#
# Leer el fichero y concatenar las hojas de los excel en un DataFrame #
#=========================================================================#

lista1 = concatenar_hojas_excel(carpeta+fichero1, [1]) #<- Fichero INR
```

```
lista2 = concatenar_hojas_excel(carpeta+fichero2, [1])
lista5 = concatenar_hojas_excel(carpeta+fichero5, [1])
lista6 = concatenar_hojas_excel(carpeta+fichero6, [1])
lista3 = concatenar_hojas_excel(carpeta+fichero3, [2]) #<- Fichero A123
lista4 = concatenar_hojas_excel(carpeta+fichero4, [2])

listaIRN = [lista1, lista2, lista4, lista5]
listaA123 = [lista3, lista5, lista6]

label_lista=['Date_Time','Step_Time(s)','Step_Index','Cycle_Index','Charg
e_Energy(Wh)','Discharge_Energy(Wh)','dV/dt(V/s)','Is_FC_Data','AC_Impeda
nce(Ohm)','ACI_Phase_Angle(Deg)']
label_lista2=['Date_Time','Step_Time(s)','Step_Index','Cycle_Index','Char
ge_Energy(Wh)','Discharge_Energy(Wh)','dV/dt(V/s)','Is_FC_Data','AC_Imped
ance(Ohm)','ACI_Phase_Angle(Deg)','Temperature (C)_1']

# ================================= #
# Valores para cada bateria:       #   *En caso de que sea necesario
# ================================= #
#     INR 18650-20R -> 1           #
#     A123 ----------> 2           #
#     CS2 -----------> 3  *        #
#     K2_016 --------> 4  *        #
#     K2_039 --------> 5  *        #
#     CX2 -----------> 6  *        #
#     PL ------------> 7  *        #
# ================================= #

#
================================================================================
=========== #
# Eliminar las columnas no deseadas del DataFrame y añadir columna del
tipo de bateria #
#
================================================================================
=========== #

lista1.drop(label_lista,1,inplace=True)
lista1.insert(7, 'Battery', 1)
lista2.drop(label_lista,1,inplace=True)
lista2.insert(7, 'Battery', 1)
lista3.drop(label_lista2,1,inplace=True)
lista3.insert(7, 'Battery', 2)
lista4.drop(label_lista2,1,inplace=True)
lista4.insert(7, 'Battery', 2)
lista5.drop(label_lista,1,inplace=True)
lista5.insert(7, 'Battery', 1)
lista6.drop(label_lista,1,inplace=True)
lista6.insert(7, 'Battery', 1)

# ================================================ #
# Concatenacion de todos los archivos de cada bateria #
# ================================================ #

IRN = pd.concat(listaIRN)
A123 = pd.concat(listaA123)
listaBat = [IRN, A123]
```

```
# ================================================================== #
# Eliminar las columnas no deseadas del DataFrame y añadir columna del #
# tipo de bateria (para lista concatenada de cada bateria)           #
# ================================================================== #
# IRN.drop(label_lista,1,inplace=True)                               #
# IRN.insert(7, 'Battery', 1)                                        #
# IRN = IRN.iloc[::15, :]                                            #
# A123.drop(label_lista2,1,inplace=True)                             #
# A123.insert(7, 'Battery', 2)                                       #
# A123 = A123.iloc[::15, :]                                          #
# ================================================================== #

# ================================================ #
# Concatenar todas las baterias en un solo Dataframe #
# ================================================ #

bat = pd.concat(listaBat)
print(bat.shape)
print(bat.columns.values)
Baterias = bat.sample(frac=0.95, random_state=786)

data_unseen = bat.drop(Baterias.index)
Baterias.reset_index(inplace=True, drop=True)
data_unseen.reset_index(inplace=True, drop=True)

# ====================================== #
# Plotear los Dataframes de los archivos #
# ====================================== #

plt.title('IRN 18650-20R DST 50% Battery Level at 25°C')
plt.plot(lista1['Test_Time(s)'], lista1['Current(A)'], c='red', label =
'Current(A)', marker = ',')
plt.plot(lista1['Test_Time(s)'], lista1['Voltage(V)'], c='orange', label
= 'Voltage(V)')
plt.plot(lista1['Test_Time(s)'], lista1['Charge_Capacity(Ah)'],
c='yellow', label = 'Charge_Capacity(Ah)')
plt.plot(lista1['Test_Time(s)'], lista1['Discharge_Capacity(Ah)'],
c='green', label = 'Discharge_Capacity(Ah)')
plt.plot(lista1['Test_Time(s)'], lista1['Internal_Resistance(Ohm)'],
c='blue', label = 'Internal_Resistance(Ohm)')
plt.plot(lista1['Test_Time(s)'], lista1['Battery'], c='purple', label =
'Battery Type')
plt.legend(loc = 3)
plt.show()

plt.title('IRN 18650-20R DST 80% Battery Level at 25°C')
plt.plot(lista2['Test_Time(s)'], lista2['Current(A)'], c='red', label =
'Current(A)')
plt.plot(lista2['Test_Time(s)'], lista2['Voltage(V)'], c='orange', label
= 'Voltage(V)')
plt.plot(lista2['Test_Time(s)'], lista2['Charge_Capacity(Ah)'],
c='yellow', label = 'Charge_Capacity(Ah)')
plt.plot(lista2['Test_Time(s)'], lista2['Discharge_Capacity(Ah)'],
c='green', label = 'Discharge_Capacity(Ah)')
plt.plot(lista2['Test_Time(s)'], lista2['Internal_Resistance(Ohm)'],
c='blue', label = 'Internal_Resistance(Ohm)')
plt.plot(lista2['Test_Time(s)'], lista2['Battery'], c='purple', label =
'Battery Type')
plt.legend(loc = 3)
```

```
plt.show()

plt.title('A123 DST at 25°C 1')
plt.plot(lista3['Test_Time(s)'], lista3['Current(A)'], c='red', label =
'Current(A)')
plt.plot(lista3['Test_Time(s)'], lista3['Voltage(V)'], c='orange', label
= 'Voltage(V)')
plt.plot(lista3['Test_Time(s)'], lista3['Charge_Capacity(Ah)'],
c='yellow', label = 'Charge_Capacity(Ah)')
plt.plot(lista3['Test_Time(s)'], lista3['Discharge_Capacity(Ah)'],
c='green', label = 'Discharge_Capacity(Ah)')
plt.plot(lista3['Test_Time(s)'], lista3['Internal_Resistance(Ohm)'],
c='blue', label = 'Internal_Resistance(Ohm)')
plt.plot(lista3['Test_Time(s)'], lista3['Battery'], c='purple', label =
'Battery Type')
plt.legend(loc = 3)
plt.show()

plt.title('A123 DST at 25°C 2')
plt.plot(lista4['Test_Time(s)'], lista4['Current(A)'], c='red', label =
'Current(A)')
plt.plot(lista4['Test_Time(s)'], lista4['Voltage(V)'], c='orange', label
= 'Voltage(V)')
plt.plot(lista4['Test_Time(s)'], lista4['Charge_Capacity(Ah)'],
c='yellow', label = 'Charge_Capacity(Ah)')
plt.plot(lista4['Test_Time(s)'], lista4['Discharge_Capacity(Ah)'],
c='green', label = 'Discharge_Capacity(Ah)')
plt.plot(lista4['Test_Time(s)'], lista4['Internal_Resistance(Ohm)'],
c='blue', label = 'Internal_Resistance(Ohm)')
plt.plot(lista4['Test_Time(s)'], lista4['Battery'], c='purple', label =
'Battery Type')
plt.legend(loc = 3)
plt.show()

plt.title('IRN 18650-20R FUDS 50% Battery Level at 25°C')
plt.plot(lista5['Test_Time(s)'], lista5['Current(A)'], c='red', label =
'Current(A)')
plt.plot(lista5['Test_Time(s)'], lista5['Voltage(V)'], c='orange', label
= 'Voltage(V)')
plt.plot(lista5['Test_Time(s)'], lista5['Charge_Capacity(Ah)'],
c='yellow', label = 'Charge_Capacity(Ah)')
plt.plot(lista5['Test_Time(s)'], lista5['Discharge_Capacity(Ah)'],
c='green', label = 'Discharge_Capacity(Ah)')
plt.plot(lista5['Test_Time(s)'], lista5['Internal_Resistance(Ohm)'],
c='blue', label = 'Internal_Resistance(Ohm)')
plt.plot(lista5['Test_Time(s)'], lista5['Battery'], c='purple', label =
'Battery Type')
plt.legend(loc = 3)
plt.show()

plt.title('IRN 18650-20R FUDS 80% Battery Level at 25°C')
plt.plot(lista6['Test_Time(s)'], lista6['Current(A)'], c='red', label =
'Current(A)')
plt.plot(lista6['Test_Time(s)'], lista6['Voltage(V)'], c='orange', label
= 'Voltage(V)')
plt.plot(lista6['Test_Time(s)'], lista6['Charge_Capacity(Ah)'],
c='yellow', label = 'Charge_Capacity(Ah)')
plt.plot(lista6['Test_Time(s)'], lista6['Discharge_Capacity(Ah)'],
c='green', label = 'Discharge_Capacity(Ah)')
```

```
plt.plot(lista6['Test_Time(s)'], lista6['Internal_Resistance(Ohm)'],
c='blue', label = 'Internal_Resistance(Ohm)')
plt.plot(lista6['Test_Time(s)'], lista6['Battery'], c='purple', label =
'Battery Type')
plt.legend(loc = 3)
plt.show()


# ========================================================= #
# Setup de nuestro modelo  y comparar distintos modelos #
# ========================================================= #

clasificacion = setup(Baterias, target='Battery', train_size = 0.3)
best = compare_models()
best = pull()
print(best)

# ================================================================== #
# Crear el setup para nuestro modelo  y comparar distintos modelos #
# ================================================================== #

svm = create_model('svm')
svm_data = pull()

# ============================================ #
# Plotear los errores y evaluacion del modelo #
# ============================================ #

tuned_svm = tune_model(svm)
print(tuned_svm)
tuned_svm_score = pull()

plot_model(tuned_svm, plot = 'error')
plot_model(tuned_svm, plot='feature')
plot_model(tuned_svm, plot='confusion_matrix')
evaluate_model(tuned_svm)
svm_score = pull()

# ==================== #
# Modelo de prediccion #
# ==================== #

predict_model(tuned_svm)
predict_data = pull()

final_svm = finalize_model(tuned_svm)
print(final_svm)


predict_model(final_svm);
final_svm_data = pull()
unseen_predictions = predict_model(final_svm, data=data_unseen)
unseen_predicitons_score = pull()

print(unseen_predictions.head())

save_model(final_svm, 'battery_svm')
```

1.2. TENSION.PY

```
#===========#
# Librerias #
#===========#

from pycaret.regression import *
import pandas as pd
import functools
from sklearn import linear_model
from sklearn.metrics import r2_score
import numpy as np
import matplotlib.pyplot as plt

#===========#
# Funciones #
#===========#

def concatenar_lista_dataframes(lista):
    df = functools.reduce(lambda df1, df2: df1.append(df2, ignore_index =
True), lista).drop_duplicates()
    return pd.concat( [df, pd.DataFrame()], ignore_index = True,
names=None)

def concatenar_hojas_excel(fichero, lst_indices_hojas):
    ex= pd.ExcelFile(fichero)
    return concatenar_lista_dataframes(ex.parse(ex.sheet_names[i]) for i
in lst_indices_hojas )

#============================#
# Inicializacion de Variables #
#============================#

carpeta = "C:/Users/Aida/Documents/TFG IOI/Programas"

fichero = "/INR/11_05_2015_SP20-2_DST_50SOC.xls"

#=================================================================#
# Leer el fichero y concatenar las hojas de los excel en un DataFrame #
#=================================================================#

df = concatenar_hojas_excel(carpeta+fichero, [1])

label_lista =
['Charge_Energy(Wh)','Discharge_Energy(Wh)','dV/dt(V/s)','ACI_Phase_Angle
(Deg)','Is_FC_Data','AC_Impedance(Ohm)','Internal_Resistance(Ohm)',
'Date_Time']

df.drop(label_lista,1,inplace=True)

#=================================================================#
# Coger subsets del dataframe para crear un nuevo dataframe de puntos #
#=================================================================#
tension = df.sample(frac=0.95, random_state=786)

data_unseen = df.drop(tension.index)
tension.reset_index(inplace=True, drop=True)
data_unseen.reset_index(inplace=True, drop=True)
```

```
nd = tension.iloc[::3, :] #Coge una fila cada 3 posiciones
indice = tension.index

#==========================================================================#
# Coger subsets del dataframe para crear un nuevo dataframe de puntos #
#==========================================================================#


regression = setup(nd, target='Voltage(V)', train_size = 0.3)
best = compare_models()
best_data = pull()
print(best_data)

rf = create_model('rf')
rf_score = pull()

tuned_rf = tune_model(rf)
tuned_rf_score = pull()

plot_model(tuned_rf)
plot_model(tuned_rf, plot = 'error')
plot_model(tuned_rf, plot='feature')


evaluate_model(tuned_rf)
eval_rf_score = pull()

rf_data = predict_model(tuned_rf)
rf_data_score = pull()

final_rf = finalize_model(tuned_rf)
print(final_rf)

final_rf_data = predict_model(final_rf);
final_rf_score = pull()

unseen_predictions = predict_model(final_rf, data=data_unseen)
unseen_predictions_score = pull()
print(unseen_predictions.head())
```

Relación de documentos

(_) Memoria NN páginas
(X) Anexos NN páginas

La Almunia, a 22 de Junio de 2022

Firmado: Aida Martín Lacosta