ANEXO A. MODELO COMPUTACIONAL ESTRUCTURADO EN LENGUAJE DE PROGRAMACIÓN PYTHON 3.7

1. MÓDULO DE ACCESO

```
"""
ESCUELA POLITÉCNICA NACIONAL
AUTOR: ALEXIS GUAMAN FIGUEROA
1_MÓDULO DE ACCESO
"""

#==================================== LOGIN ================================

from tkinter import *
import tkinter.ttk as ttk
from tkinter import messagebox

root = Tk()

root.geometry('350x230+700+300')
root.title('LOGIN')
root.resizable(0,0)
root.iconbitmap('epn.ico')

txq_1 = Label(root, text = 'Inicio de sesión ', fg='#616A6B',font=('Arial',12)).place(x=0,y=0)
Label(root, text = 'AGF ®', fg='#616A6B',font=('Tw Cen MT',9)).place(x=300,y=200)

Label(root, text = 'Usuario: '    , fg='#616A6B',font=('Arial',12)).place(x=10,y=50)
Label(root, text = 'Contraseña: ', fg='#616A6B',font=('Arial',12)).place(x=10,y=100)

# Usuario por defecto
default_user = StringVar(root, value='1720282571')
US = Entry(root,textvariable=default_user, fg='#616A6B',font=('Arial',11)).place(x=150,y=  50)
PS = Entry(root,show ='*',fg='#616A6B',font=('Arial',11))
PS.place(x=150,y= 100)

def ingreso(self):
    pasw = PS.get()

    if pasw == '152125':     #================ CONTRASEÑA ====================
        root.destroy()
        print(' ')
        print(' ')
        print('**   Contraseña correcta...     EJECUTANDO....')
        print(' ')
        print(' ')
        runfile('C:/Users/ALEXIS GUAMAN/Dropbox/TESIS_1/0_3_ANALIZADOR DE GRÁFICAS/GRAPHIC ANALIZER.py', wdir='C:/Users/ALEXIS GUAMAN/Dropbox/TESIS_1/0_3_ANALIZADOR DE GRÁFICAS')

    else:
        incorr_ps = messagebox.askretrycancel('ERROR','Contraseña incorrecta...!')
        if incorr_ps == False:
            root.destroy()
```

```
48.
49.     #Button(root, text = '  INGRESAR AL SISTEMA  ',bg='#2ECC71',fg='#1A5276',command
       = lambda:ingreso(self)).place(x=110,y=150)
50.     Label(root, text = 'Presione ENTER para INGRESAR: ', bg='#2ECC71',fg='#1A5276',fo
       nt=('Arial',12)).place(x=50,y=150)
51.     PS.bind('<Return>', ingreso)
52.
53.     root.mainloop()
```

2. MÓDULO DE GRÁFICAS RÁPIDAS

```
54.     """
55.     ESCUELA POLITÉCNICA NACIONAL
56.     AUTOR: ALEXIS GUAMAN FIGUEROA
57.     2_MÓDULO DE GRÁFICAS RÁPIDAS
58.     """
59.     #**********************************************************************************
       *************
60.     #************************************ GRAPHIC ANALIZER ************************
       *************
61.     #**********************************************************************************
       *************
62.
63.     print (' ')
64.     print (' ')
65.     print ('**  Compilando... ')
66.     print (' ')
67.     print (' ')
68.
69.     from tkinter import *
70.     import tkinter.ttk as ttk
71.     from tkinter import messagebox
72.
73.     #------------------------ PARÁMETROS INICIALES, CREACIÓN DE LA VENTANA RAIZ ----
       ------------------
74.
75.     # Se genera la raiz o ventana principal
76.     raiz = Tk()
77.     # Se establece el tamaño  y posición de la ventana principal RAIZ
78.     raiz.geometry('600x520+100+100')
79.     # TÍTULO DE LA VENTANA PRINCIPAL
80.     raiz.title('Análisis de datos')
81.     # impide redimensionar la ventana principal RAIZ
82.     raiz.resizable(0,0)
83.     # Se estable un ícono para la ventana principal RAIZ se llama con el nombre del a
       rchivo
84.     raiz.iconbitmap('epn.ico')
85.
86.     # Se establece el color de fondo de la ventana principal RAIZ
87.     #raiz.config(bg='#34495E')
88.
89.
90.     #===>Texto de ayuda (Acerca de)
91.     def infoAdicional():
92.         messagebox.showinfo('Acerca de este programa','PROYECTO PREVIO A LA OBTENCIÓN
        DEL TÍTULO DE INGENIERO ELÉCTRICO\n\nEste Programa fue realizado por Alexis Guam
       án Figueroa')
93.
94.     #===>Ejecuta archivo de proyección semanal
95.     def proySem():
96.         raiz.destroy()
97.         runfile('C:/Users/ALEXIS GUAMAN/Dropbox/TESIS_1/0_3_ANALIZADOR DE GRÁFICAS/03
       _proyeccion_semanal.py', wdir='C:/Users/ALEXIS GUAMAN/Dropbox/TESIS_1/0_3_ANALIZA
       DOR DE GRÁFICAS')
```

```
98.
99.     #===>Cierra el proyecto
100.    def salirProyecto():
101.        PregCierre = messagebox.askquestion('Cerrar programa','Desea cerrar el progra
        ma ?')
102.        if PregCierre == 'yes':
103.            raiz.destroy()
104.
105.    #================================================================================
106.    #=========================SUBRUTINA PARA RESET ==================================
107.    #================================================================================
108.    def boton_reset():
109.        global raiz
110.        raiz.destroy()
111.        runfile('C:/Users/ALEXIS GUAMAN/Dropbox/TESIS_1/0_3_ANALIZADOR DE GRÁFICAS/GR
        APHIC ANALIZER.py', wdir='C:/Users/ALEXIS GUAMAN/Dropbox/TESIS_1/0_3_ANALIZADOR D
        E GRÁFICAS')
112.
113.    b_res = Button(raiz, text = '  RESET  ',bg='#A81C1C',fg='#FFFFFF',command = boton
        _reset)
114.    b_res.place(x=445,y=445)
115.
116.    #================================================================================
117.    #====================== Se establece el menú de la raíz =========================
118.    #================================================================================
119.
120.    barraMenu = Menu(raiz)
121.    raiz.config(menu=barraMenu)
122.
123.    archivoMenu = Menu(barraMenu, tearoff = 0)
124.
125.    archivoMenu.add_command(label = 'Nuevo')
126.    archivoMenu.add_command(label = 'Guardar')
127.    archivoMenu.add_command(label = 'Guardar Como')
128.    archivoMenu.add_separator()
129.    archivoMenu.add_command(label = 'Nueva Proyección Semanal',command = proySem)
130.    archivoMenu.add_separator()
131.    archivoMenu.add_command(label = 'Reiniciar', command = boton_reset)
132.    archivoMenu.add_command(label = 'Salir', command = salirProyecto)
133.
134.    archivoEdicion = Menu(barraMenu, tearoff = 0)
135.    archivoEdicion.add_command(label = 'Copiar')
136.    archivoEdicion.add_command(label = 'Cortar')
137.    archivoEdicion.add_command(label = 'Pegar')
138.
139.    archivoHerramientas = Menu(barraMenu, tearoff = 0)
140.
141.    archivoAyuda = Menu(barraMenu, tearoff = 0)
142.    archivoAyuda.add_command(label = 'Acerca de...', command = infoAdicional)
143.
144.    barraMenu.add_cascade(label = 'Archivo', menu = archivoMenu)
145.    barraMenu.add_cascade(label = 'Edición', menu = archivoEdicion)
146.    barraMenu.add_cascade(label = 'Herramientas', menu = archivoHerramientas)
147.    barraMenu.add_cascade(label = 'Ayuda', menu = archivoAyuda)
148.
149.    #================================================================================
150.    #================================================================================
151.
152.
153.
154.    #------------------------ Ingreso de imágenes --------------------------------
155.
156.    imag_eeasa = PhotoImage(file='eeasa_peq.png')
157.    Label(raiz, image = imag_eeasa).place(x=140,y=20)
158.
159.    imag_pronost = PhotoImage(file='pronostico_peq.png')
```

```
160.    Label(raiz, image = imag_pronost).place(x=10,y=390)
161.
162.
163.    #------------------------- Ingreso de textos ---------------------------------
164.
165.    txq_1 = Label(raiz, text = 'Seleccione el tipo de gráficas a generar: ', fg='#616
        A6B',font=('Arial',12)).place(x=20,y=150)
166.    Label(raiz, text = 'AGF ®', fg='#616A6B',font=('Tw Cen MT',9)).place(x=545,y=475)
167.
168.    #-------------------------Ingreso de MENÚ 1-----------------------------------
169.
170.    cbx = ttk.Combobox(values=['RÁPIDA','ACUMULATIVA','HISTÓRICA'],state="readonly")
171.    cbx.place(x=330,y=150)
172.
173.
174.
175.    #------------------------- Ingreso de BOTON CHECK 1 --------------------------
176.
177.    b1 = Button(raiz, text = '  Ok  ',command = lambda:boton_1())
178.    b1.place(x=500,y=150)
179.
180.    #------------------------- LISTA DE FUNCIONES ----------------------
181.
182.    #-----------------------Función 1, PARA EL BOTON CHECK 1----------------------
183.
184.    def boton_1():
185.
186.        pos_menu_1 = cbx.current( )
187.
188.
189.
190.
191.        if pos_menu_1 == 0:
192.            txt_2 = Label(raiz, text = 'GRÁFICA RÁPIDA', fg='#616A6B',font=('Arial',1
        2)).place(x=20,y=200)
193.            txt_3 = Label(raiz, text = 'Seleccione el tipo de gráfica: ', fg='#616A6B
        ',font=('Arial',10)).place(x=20,y=230)
194.
195.            def boton_2():
196.                global pos_menu_2
197.
198.                pos_menu_2 = cbx_tipograf.current()
199.                po_menu_2 = cbx_tipograf.get()
200.
201.                varOpcion = IntVar()
202.
203.                if pos_menu_2 == -1:
204.                    messagebox.showinfo(message="  Debe seleccionar el tipo de gráfic
        as a generar ! ", title="Faltan datos")
205.                    print('  Debe seleccionar el tipo de gráficas a generar ! ')
206.                else:
207.                    print('**  Gráfica del tipo: ',po_menu_2)
208.
209.
210.
211.                def imp1():
212.                    global val_check
213.                    val_check = varOpcion.get()
214.                    if val_check == 1:
215.                        print('**  Ha seleccionado SCATTER')
216.                    else:
217.                        print('**  Sin SCATTER')
218.
```

```
219.    #=============================================================================

220.    #================================= GRÁFICA ANUAL =============================
221.    #=============================================================================
222.                if   pos_menu_2==0:
223.                    b2.place_forget()
224.
225.                    Label(raiz, text = 'AÑO: ', fg='#616A6B',font=('Arial',10)).place
        (x=350,y=230)
226.                    cbx_anio = ttk.Combobox(values=['2013','2014','2015','2016','2017
        ','2018'],state="readonly")
227.                    cbx_anio.place(x=400,y=230)
228.
229.                    Label(raiz, text = 'S / E: ', fg='#616A6B',font=('Arial',10)).pla
        ce(x=350,y=260)
230.                    cbx_se = ttk.Combobox(values=['AMBATO','BAÑOS','PUYO','TENA','TOT
        ORAS'],state="readonly")
231.                    cbx_se.place(x=400,y=260)
232.
233.                    #Radiobutton(raiz,text = 'GENERAR SCATTER', fg='#616A6B',font=('A
        rial',10), variable = varOpcion, value=1, command=imp1).place(x=350,y=290)
234.                    Checkbutton(raiz, text = 'GENERAR SCATTER', fg='#616A6B',font=('A
        rial',10), variable = varOpcion, onvalue = 1, offvalue = 0, command=imp1).place(x
        =350,y=290)
235.
236.                    b_go = Button(raiz, text = '  GRAFICAR  ',bg='#2ECC71',fg='#1A527
        6',command = lambda:boton_go())
237.                    b_go.place(x=505,y=445)
238.
239.                    def boton_go():
240.
241.                        global pos_se, po_anio, po_se
242.
243.                        pos_anio = cbx_anio.current()
244.                        pos_se  = cbx_se.current()
245.
246.                        po_anio = cbx_anio.get()
247.                        po_se  = cbx_se.get()
248.
249.                        if pos_anio == -1:
250.                            messagebox.showinfo(message="Ingrese Año", title="Faltan
        datos")
251.                        elif pos_se == -1:
252.                            messagebox.showinfo(message="Ingrese la Subestación", tit
        le="Faltan datos")
253.                        else:
254.                            print ('****    Se ha seleccionado el año', po_anio, 'Par
        a la subestación', po_se)
255.                            runfile('C:/Users/ALEXIS GUAMAN/Dropbox/TESIS_1/0_3_ANALI
        ZADOR DE GRÁFICAS/01_plots.py', wdir='C:/Users/ALEXIS GUAMAN/Dropbox/TESIS_1/0_3_
        ANALIZADOR DE GRÁFICAS')
256.                            #runfile('C:/Users/ALEXIS GUAMAN/Dropbox/TESIS_1/0_3_ANAL
        IZADOR DE GRÁFICAS/hola_borrar.py', wdir='C:/Users/ALEXIS GUAMAN/Dropbox/TESIS_1/
        0_3_ANALIZADOR DE GRÁFICAS')
257.
258.    #=============================================================================

259.    #================================= GRÁFICA MENSUAL ===========================
260.    #=============================================================================

261.                elif pos_menu_2==1:
262.                    b2.place_forget()
263.
264.                    Label(raiz, text = 'AÑO: ', fg='#616A6B',font=('Arial',10)).place
        (x=350,y=230)
```

```
265.                    cbx_anio = ttk.Combobox(values=['2013','2014','2015','2016','2017
    ','2018'],state="readonly")
266.                    cbx_anio.place(x=400,y=230)
267.
268.                    Label(raiz, text = 'MES: ', fg='#616A6B',font=('Arial',10)).place
    (x=350,y=260)
269.                    cbx_mes = ttk.Combobox(values=['ENERO','FEBRERO','MARZO','ABRIL',
    'MAYO','JUNIO','JULIO','AGOSTO','SEPTIEMBRE','OCTUBRE','NOVIEMBRE','DICIEMBRE'],s
    tate="readonly")
270.                    cbx_mes.place(x=400,y=260)
271.
272.                    Label(raiz, text = 'S / E: ', fg='#616A6B',font=('Arial',10)).pla
    ce(x=350,y=290)
273.                    cbx_se = ttk.Combobox(values=['AMBATO','BAÑOS','PUYO','TENA','TOT
    ORAS'],state="readonly")
274.                    cbx_se.place(x=400,y=290)
275.
276.                    Checkbutton(raiz, text = 'GENERAR SCATTER', fg='#616A6B',font=('A
    rial',10), variable = varOpcion, onvalue = 1, offvalue = 0, command=imp1).place(x
    =350,y=320)
277.
278.                    b_go = Button(raiz, text = '  GRAFICAR  ',bg='#2ECC71',fg='#1A527
    6',command = lambda:boton_go())
279.                    b_go.place(x=505,y=445)
280.
281.                    def boton_go():
282.
283.                        global pos_se, pos_mes, po_anio, po_se, po_mes
284.
285.                        pos_anio = cbx_anio.current()
286.                        pos_mes = cbx_mes.current()
287.                        pos_se  = cbx_se.current()
288.
289.                        po_anio = cbx_anio.get()
290.                        po_mes = cbx_mes.get()
291.                        po_se  = cbx_se.get()
292.
293.                        if pos_anio == -1:
294.                            messagebox.showinfo(message="Ingrese Año", title="Faltan
    datos")
295.                        elif pos_mes == -1:
296.                            messagebox.showinfo(message="Ingrese Mes", title="Faltan
    datos")
297.                        elif pos_se == -1:
298.                            messagebox.showinfo(message="Ingrese la Subestación", tit
    le="Faltan datos")
299.                        else:
300.                            print ('****    Se ha seleccionado el año', po_anio, ',co
    n mes', po_mes,'para la subestación',po_se)
301.                            runfile('C:/Users/ALEXIS GUAMAN/Dropbox/TESIS_1/0_3_ANALI
    ZADOR DE GRÁFICAS/01_plots.py', wdir='C:/Users/ALEXIS GUAMAN/Dropbox/TESIS_1/0_3_
    ANALIZADOR DE GRÁFICAS')
302.
303.    #=============================================================================

304.    #================================= GRÁFICA SEMANAL ===========================
305.    #=============================================================================

306.                elif pos_menu_2==2:
307.                    b2.place_forget()
308.
309.                    Label(raiz, text = 'AÑO: ', fg='#616A6B',font=('Arial',10)).place
    (x=350,y=230)
310.                    cbx_anio = ttk.Combobox(values=['2013','2014','2015','2016','2017
    ','2018'],state="readonly")
311.                    cbx_anio.place(x=400,y=230)
```

```
312.
313.                     Label(raiz, text = 'MES: ', fg='#616A6B',font=('Arial',10)).place
    (x=350,y=260)
314.                     cbx_mes = ttk.Combobox(values=['ENERO','FEBRERO','MARZO','ABRIL',
    'MAYO','JUNIO','JULIO','AGOSTO','SEPTIEMBRE','OCTUBRE','NOVIEMBRE','DICIEMBRE'],s
    tate="readonly")
315.                     cbx_mes.place(x=400,y=260)
316.
317.                     Label(raiz, text = 'SEM: ', fg='#616A6B',font=('Arial',10)).place
    (x=350,y=290)
318.                     cbx_sem = ttk.Combobox(values=['1','2','3','4','5','6'],state="re
    adonly")
319.                     cbx_sem.place(x=400,y=290)
320.
321.                     Label(raiz, text = 'S / E: ', fg='#616A6B',font=('Arial',10)).pla
    ce(x=350,y=320)
322.                     cbx_se = ttk.Combobox(values=['AMBATO','BAÑOS','PUYO','TENA','TOT
    ORAS'],state="readonly")
323.                     cbx_se.place(x=400,y=320)
324.
325.                     Checkbutton(raiz, text = 'GENERAR SCATTER', fg='#616A6B',font=('A
    rial',10), variable = varOpcion, onvalue = 1, offvalue = 0, command=imp1).place(x
    =350,y=350)
326.
327.                     b_go = Button(raiz, text = '  GRAFICAR  ',bg='#2ECC71',fg='#1A527
    6',command = lambda:boton_go())
328.                     b_go.place(x=505,y=445)
329.
330.                     def boton_go():
331.
332.                         global pos_se, pos_mes, pos_sem, po_anio, po_mes, po_sem, po_
    se
333.
334.
335.                         pos_anio = cbx_anio.current()
336.                         pos_mes = cbx_mes.current()
337.                         pos_sem = cbx_sem.current()
338.                         pos_se  = cbx_se.current()
339.
340.                         po_anio = cbx_anio.get()
341.                         po_mes = cbx_mes.get()
342.                         po_sem = cbx_sem.get()
343.                         po_se  = cbx_se.get()
344.
345.                         def verif_semana():
346.                             import calendar
347.                             num_semanas = len(calendar.monthcalendar(int(po_anio),pos
    _mes+1))
348.     #                         print(int(po_sem))
349.     #                         print(num_semanas)
350.
351.                             if int(po_sem) <= num_semanas:
352.
353.                                 if (calendar.monthcalendar(int(po_anio),pos_mes+1))[p
    os_sem][0] == 0:
354.                                     messagebox.showinfo(message="La semana "+po_sem+'
     para el mes ' +po_mes + ' de '+po_anio + ' No posee suficientes días para el aná
    lisis', title="Seleccione otra semana")
355.
356.                                 elif (calendar.monthcalendar(int(po_anio),pos_mes+1))
    [pos_sem][-1] == 0:
357.                                     messagebox.showinfo(message="La semana "+po_sem+'
     para el mes ' +po_mes + ' de '+po_anio + ' No posee suficientes días para el aná
    lisis', title="Seleccione otra semana")
358.
359.                                 else:
```

```
360.                                     global dias_restantes
361.                                     inicio_semana = 0
362.                                     for j in range (7):
363.                                         if calendar.monthcalendar(int(po_anio),pos_me
    s+1)[0][j] == 0:
364.                                             inicio_semana = inicio_semana + 1
365.                                     dias_restantes = 7 - inicio_semana
366.                                     print('días restantes son: ',dias_restantes)
367.
368.                                     runfile('C:/Users/ALEXIS GUAMAN/Dropbox/TESIS_1/0
    _3_ANALIZADOR DE GRÁFICAS/01_plots.py', wdir='C:/Users/ALEXIS GUAMAN/Dropbox/TESI
    S_1/0_3_ANALIZADOR DE GRÁFICAS')
369.
370.
371.                             else:
372.                                 messagebox.showinfo(message="No existe semana "+po_se
    m+' para el mes ' +po_mes + ' de '+po_anio, title="Seleccione otra semana")
373.
374.
375.                         if pos_anio == -1:
376.                             messagebox.showinfo(message="Ingrese Año", title="Faltan
    datos")
377.                         elif pos_mes == -1:
378.                             messagebox.showinfo(message="Ingrese Mes", title="Faltan
    datos")
379.                         elif pos_sem == -1:
380.                             messagebox.showinfo(message="Ingrese Semana", title="Falt
    an datos")
381.                         elif pos_se == -1:
382.                             messagebox.showinfo(message="Ingrese la Subestación", tit
    le="Faltan datos")
383.                         else:
384.                             print ('****    Se ha seleccionado el año', po_anio, ',co
    n mes', po_mes, ',semana N°', po_sem,'para la subestación',po_se)
385.                             verif_semana()
386.
387.
388.     #=============================================================================

389.     #================================ GRÁFICA DIARIA ============================
390.     #=============================================================================

391.
392.                 elif pos_menu_2==3:
393.                     b2.place_forget()
394.
395.                     Label(raiz, text = 'AÑO: ', fg='#616A6B',font=('Arial',10)).place
    (x=350,y=230)
396.                     cbx_anio = ttk.Combobox(values=['2013','2014','2015','2016','2017
    ','2018'],state="readonly")
397.                     cbx_anio.place(x=400,y=230)
398.
399.                     Label(raiz, text = 'MES: ', fg='#616A6B',font=('Arial',10)).place
    (x=350,y=260)
400.                     cbx_mes = ttk.Combobox(values=['ENERO','FEBRERO','MARZO','ABRIL',
    'MAYO','JUNIO','JULIO','AGOSTO','SEPTIEMBRE','OCTUBRE','NOVIEMBRE','DICIEMBRE'],s
    tate="readonly")
401.                     cbx_mes.place(x=400,y=260)
402.
403.                     Label(raiz, text = 'DÍA: ', fg='#616A6B',font=('Arial',10)).place
    (x=350,y=290)
404.                     cbx_dia = ttk.Combobox(values=['1','2','3','4','5','6','7','8','9
    ','10','11','12','13','14','15','16','17','18','19','20','21','22','23','24','25'
    ,'26','27','28','29','30','31'],state="readonly")
405.                     cbx_dia.place(x=400,y=290)
406.
```

```
                    Label(raiz, text = 'S / E: ', fg='#616A6B',font=('Arial',10)).pla
    ce(x=350,y=320)
                    cbx_se = ttk.Combobox(values=['AMBATO','BAÑOS','PUYO','TENA','TOT
    ORAS'],state="readonly")
                    cbx_se.place(x=400,y=320)

                    Checkbutton(raiz, text = 'GENERAR SCATTER', fg='#616A6B',font=('A
    rial',10), variable = varOpcion, onvalue = 1, offvalue = 0, command=imp1).place(x
    =350,y=350)

                    b_go = Button(raiz, text = '  GRAFICAR  ',bg='#2ECC71',fg='#1A527
    6',command = lambda:boton_go())
                    b_go.place(x=505,y=445)

                    def boton_go():

                        global pos_se, pos_mes, pos_sem, po_anio, po_mes, po_sem, po_
    se, pos_dia, po_dia

                        pos_anio = cbx_anio.current()
                        pos_mes = cbx_mes.current()
                        pos_dia = cbx_dia.current()
                        pos_se  = cbx_se.current()

                        po_anio = cbx_anio.get()
                        po_mes = cbx_mes.get()
                        po_dia = cbx_dia.get()
                        po_se  = cbx_se.get()

    #                     def verif_dia():


                        from calendar import monthrange

                        if pos_anio == -1:
                            messagebox.showinfo(message="Ingrese Año", title="Faltan
    datos")
                        elif pos_mes == -1:
                            messagebox.showinfo(message="Ingrese Mes", title="Faltan
    datos")
                        elif pos_dia == -1:
                            messagebox.showinfo(message="Ingrese Día", title="Faltan
    datos")
                        elif pos_se == -1:
                            messagebox.showinfo(message="Ingrese la Subestación", tit
    le="Faltan datos")
                        elif int(po_dia) > monthrange(int(po_anio), pos_mes+1)[1]:
                            messagebox.showinfo(message='No existe día '+po_dia +' pa
    ra el mes ' +po_mes + ' de '+ po_anio, title="Dato erróneo")
                        else:

                            runfile('C:/Users/ALEXIS GUAMAN/Dropbox/TESIS_1/0_3_ANALI
    ZADOR DE GRÁFICAS/01_plots.py', wdir='C:/Users/ALEXIS GUAMAN/Dropbox/TESIS_1/0_3_
    ANALIZADOR DE GRÁFICAS')


    #                         from calendar import monthrange
                            print ('****    Se ha seleccionado el año', po_anio, ',co
    n mes', po_mes, ',día', po_dia,'para la subestación',po_se)
    #                         verif_dia()
```

```
459.            cbx_tipograf = ttk.Combobox(values=['Anual','Mensual','Semanal','Diario']
    ,state="readonly")
460.            cbx_tipograf.place(x=200,y=230)
461.
462.    #------------------------ Ingreso de BOTON CHECK 2 ----------------------------
    -------------------------------
463.
464.            b2 = Button(raiz, text = '  Ok  ',command = lambda:boton_2())
465.            b2.place(x=350,y=230)
466.
467.            b1.place_forget()       #   Elimina el botón pulsado
468.            print('**  Ha seleccionado gráfica RÁPIDA...')
469.
470.
471.        elif  pos_menu_1 == 1:
472.            txt_2 = Label(raiz, text = 'GRÁFICAS ACUMULTIVAS ', fg='#616A6B',font=('A
    rial',12)).place(x=20,y=200)
473.            b1.place_forget()
474.            print('**  Ha seleccionado gráfica ACUMULATIVA...')
475.
476.    #=============================================================================

477.    #================================= GRÁFICA ANUAL ==============================
478.    #=============================================================================
479.
480.            Label(raiz, text = 'AÑO: ', fg='#616A6B',font=('Arial',10)).place(x=350,y
    =230)
481.            cbx_anio = ttk.Combobox(values=['2013','2014','2015','2016','2017','2018'
    ],state="readonly")
482.            cbx_anio.place(x=400,y=230)
483.
484.            Label(raiz, text = 'S / E: ', fg='#616A6B',font=('Arial',10)).place(x=350
    ,y=260)
485.            cbx_se = ttk.Combobox(values=['AMBATO','BAÑOS','PUYO','TENA','TOTORAS'],s
    tate="readonly")
486.            cbx_se.place(x=400,y=260)
487.
488.
489.            b_go = Button(raiz, text = '  GRAFICAR  ',bg='#2ECC71',fg='#1A5276',comma
    nd = lambda:boton_go())
490.            b_go.place(x=505,y=445)
491.
492.            def boton_go():
493.
494.                global pos_se, po_anio, po_se
495.
496.                pos_anio = cbx_anio.current()
497.                pos_se  = cbx_se.current()
498.
499.                po_anio = cbx_anio.get()
500.                po_se  = cbx_se.get()
501.
502.                if pos_anio == -1:
503.                    messagebox.showinfo(message="Ingrese Año", title="Faltan datos")

504.                elif pos_se == -1:
505.                    messagebox.showinfo(message="Ingrese la Subestación", title="Falt
    an datos")
506.                else:
507.                    print ('****    Se ha seleccionado el año', po_anio, 'Para la sub
    estación', po_se)
508.                    runfile('C:/Users/ALEXIS GUAMAN/Dropbox/TESIS_1/0_3_ANALIZADOR DE
     GRÁFICAS/02_plots_acum.py', wdir='C:/Users/ALEXIS GUAMAN/Dropbox/TESIS_1/0_3_ANA
    LIZADOR DE GRÁFICAS')
509.
510.
```

```
511.
512.        elif  pos_menu_1 == 2:
513.                txt_2 = Label(raiz, text = 'GRÁFICAS ACUMULATIVA HISTÓRICA', fg='#616
    A6B',font=('Arial',12)).place(x=20,y=200)
514.                b1.place_forget()
515.                print('**  Ha seleccionado gráfica HISTÓRICA...')
516.
517.    #=============================================================================

518.    #================================= GRÁFICA HISTÓRICA ============================
    ==
519.    #=============================================================================
520.
521.                Label(raiz, text = 'AÑO: ', fg='#616A6B',font=('Arial',10)).place(x=3
    50,y=230)
522.                cbx_anio = ttk.Combobox(values=['2013','2014','2015','2016','2017','2
    018'],state="readonly")
523.                cbx_anio.place(x=400,y=230)
524.
525.                Label(raiz, text = 'S / E: ', fg='#616A6B',font=('Arial',10)).place(x
    =350,y=260)
526.                cbx_se = ttk.Combobox(values=['AMBATO','BAÑOS','PUYO','TENA','TOTORAS
    '],state="readonly")
527.                cbx_se.place(x=400,y=260)
528.
529.
530.                b_go = Button(raiz, text = '  GRAFICAR  ',bg='#2ECC71',fg='#1A5276',c
    ommand = lambda:boton_go())
531.                b_go.place(x=505,y=445)
532.
533.                def boton_go():
534.
535.                    global pos_se, po_anio, po_se
536.
537.                    pos_anio = cbx_anio.current()
538.                    pos_se  = cbx_se.current()
539.
540.                    po_anio = cbx_anio.get()
541.                    po_se  = cbx_se.get()
542.
543.                    if pos_anio == -1:
544.                        messagebox.showinfo(message="Ingrese Año", title="Faltan dato
    s")
545.                    if pos_se == -1:
546.                        messagebox.showinfo(message="Ingrese la Subestación", title="
    Faltan datos")
547.                    else:
548.                        print ('****    Se ha seleccionado el año', po_anio, 'Para la
     subestación', po_se)
549.                        runfile('C:/Users/ALEXIS GUAMAN/Dropbox/TESIS_1/0_3_ANALIZADO
    R DE GRÁFICAS/02_plots_acum.py', wdir='C:/Users/ALEXIS GUAMAN/Dropbox/TESIS_1/0_3
    _ANALIZADOR DE GRÁFICAS')
550.
551.
552.        else:
553.                print('  Debe seleccionar el tipo de gráficas a generar ! ')
554.
555.                messagebox.showinfo(message="  Debe seleccionar el tipo de gráficas a
     generar ! ", title="Faltan datos")
556.
557.      # Encierra en la ventana principal a la raiz----------------------------
558.    raiz.mainloop()
559.
560.
561.    print (' ')
562.    print (' ')
```

```
563.    print ('**  COMPLETADO...!')
564.    print (' ')
565.    print (' ')
566.    #importa librería pandas
567.    import pandas as pd
568.    import matplotlib.pyplot as plt
569.    import numpy as np
570.
571.
572.    #============================== VAR GLOBALES ================================
573.    global pos_menu_2
574.    global pos_se, po_anio, po_se, val_check,po_dia
575.    anio = po_anio
576.    sube = po_se + ' '
577.
578.    #============================================================================
579.
580.    print(' ')
581.    print(' ')
582.    print('Procesing Plots...')
583.    print(' ')
584.
585.    doc_clima = 'HIS_CLIMA_' + anio + '.xls'
586.    doc_demand = 'HIS_POT_' + anio + '.xls'
587.
588.    df1 = pd.read_excel(doc_clima, sheetname='Hoja1',  header=None)
589.
590.    hora = df1.iloc[:,2].values.tolist()
591.    hora.pop(0)
592.
593.    temp = df1.iloc[:,4].values.tolist()
594.    temp.pop(0)
595.
596.    mes = df1.iloc[:,0].values.tolist()
597.    mes.pop(0)
598.
599.    dia = df1.iloc[:,1].values.tolist()
600.    dia.pop(0)
601.
602.    #===> Reemplaza valores (0) con (nan)
603.    for i in range(len(temp)):
604.        if temp[i] == 0:
605.            temp[i] = float('nan')
606.    #============================================================================
607.    df2 = pd.read_excel(doc_demand, sheetname='Hoja1',  header=None)
608.
609.    amba = df2.iloc[:,4].values.tolist()
610.    amba.pop(0)
611.
612.    toto = df2.iloc[:,5].values.tolist()
613.    toto.pop(0)
614.
615.    puyo = df2.iloc[:,6].values.tolist()
616.    puyo.pop(0)
617.
618.    tena = df2.iloc[:,7].values.tolist()
619.    tena.pop(0)
620.
621.    bani = df2.iloc[:,8].values.tolist()
622.    bani.pop(0)
623.
624.    #===> Reemplaza valores (0) con (nan)
625.    for i in range(len(temp)):
626.        if amba[i] == 0:
627.            amba[i] = float('nan')
628.        if toto[i] == 0:
```

```
            toto[i] = float('nan')
        if puyo[i] == 0:
            puyo[i] = float('nan')
        if tena[i] == 0:
            tena[i] = float('nan')
        if bani[i] == 0:
            bani[i] = float('nan')
    #===> Se establece una matriz con los datos importados
    data = np.column_stack((temp,amba,toto, puyo, tena, bani))

    #=============================================================================
    #================================ GRÁFICA ANUAL ==============================
    #=============================================================================
    if   pos_menu_2==0:

    #===> Tamaño de la ventana de la gráfica
        plt.subplots(figsize=(15, 8))
    #===> Título general superior
        plt.suptitle(u' GRÁFICA ANUAL ', fontsize=16, fontweight='bold')


        import calendar
    #===> Creamos una lista con las posiciones del primer día de cada mes
        dias = [mes.index('ENERO'), mes.index('FEBRERO'), mes.index('MARZO'),
                mes.index('ABRIL'), mes.index('MAYO'),    mes.index('JUNIO'),
                mes.index('JULIO'), mes.index('AGOSTO'),  mes.index('SEPTIEMBRE'),
                mes.index('OCTUBRE'), mes.index('NOVIEMBRE'), mes.index('DICIEMBRE')]

    #===> Creamos una lista con los nombres de los meses
        meses = calendar.month_name[1:13]

        plt.subplot(1,1,1)
    #===> Tiempo para la gráfica, considera un año al tomar todos los datos
        plt.xlim(0,8759)

    #===> Subtítulo de la gráfica
        plt.title(u'Carga vs. Temperatura S/E '+sube +anio, fontweight='bold',
                  style='italic')

    #=== > Etiq. del eje x (meses del año) se marcan según el primer dia de c/mes
        plt.xticks(dias, meses, size = 'small', color = 'b', rotation = 45)

    #===> Condicional de gráficas, var generales: se, col
        if pos_se == 0:
            se = amba
            col = '#DCD037'
        elif pos_se == 1:
            se = bani
            col = '#4ACB71'
        elif pos_se == 2:
            se = puyo
            col = '#349A9D'
        elif pos_se == 3:
            se = tena
            col = '#CC8634'
        elif pos_se == 4:
            se = toto
            col = '#CD336F'

        plt.plot(se, col, label = 'Potencia')
        plt.legend()
        plt.xlabel(u'TIEMPO [ meses ]')
        plt.ylabel(u'CARGA  [ kW ]')
        plt.twinx()
        g2, = plt.plot(temp, 'b', label = 'Temperatura en °C')
```

```
694.        plt.ylabel(u'TEMPERATURA  [ °C ]')
695.        plt.legend(handles=[g2], loc=4)
696.        plt.grid(color='#C8C8C8', linestyle='-', linewidth=0.5)
697.
698.    #================================ SCATTER ANUAL ================================
699.
700.    #===> Si se activa el check, genera el scatter
701.        if val_check == 1:
702.
703.            fig2, (ax1) = plt.subplots(nrows=1,ncols=1,figsize=(15, 8))
704.            plt.suptitle(u'SCATTER ANUAL',fontweight='bold',
705.                         style='italic')
706.            ax1.scatter(x=temp, y=se, marker='.', c=col)
707.            ax1.set_title('$Temperatura$ vs. $Potencia$ $S/E$ '+sube+'$año $: ' +anio
    )
708.            ax1.set_ylabel('$Carga$  $[$ $kW$ $]$')
709.            ax1.set_xlabel('$Temperatura$ $en$ $°C$')
710.
711.    #================================================================================
712.    #================================================================================
713.
714.
715.
716.    #================================================================================
717.    #================================ GRÁFICA MENSUAL ================================
718.    #================================================================================
719.    if   pos_menu_2==1:
720.
721.        global pos_mes, po_mes
722.        mesi = ' / '+po_mes+ ' '
723.
724.        import calendar
725.
726.    #===> Tamaño de la ventana de la gráfica
727.        plt.subplots(figsize=(15, 8))
728.    #===> Título general superior
729.        plt.suptitle(u' GRÁFICA MENSUAL ', fontsize=16, fontweight='bold')
730.
731.    #===> Subrutina, genera posiciones de los meses según el mes seleccionado
732.        x_meses = [mes.index('ENERO'), mes.index('FEBRERO'), mes.index('MARZO'),
733.                   mes.index('ABRIL'), mes.index('MAYO'),    mes.index('JUNIO'),
734.                   mes.index('JULIO'), mes.index('AGOSTO'),  mes.index('SEPTIEMBRE'),

735.                   mes.index('OCTUBRE'), mes.index('NOVIEMBRE'), mes.index('DICIEMBRE
    ')]
736.        for i in range (12):
737.            if pos_mes == i:
738.                if pos_mes == 11:
739.                    val_ini = x_meses[i]
740.                    val_fin = 8759
741.                else:
742.                    val_ini = x_meses[i]
743.                    val_fin = x_meses[i+1]-1
744.
745.        cant_datos_mes = val_fin - val_ini
746.
747.    #===> Creamos una lista con los números de día del mes seleccionado (etiquetas)
748.        x_dias = calendar.monthcalendar(int(anio),pos_mes+1)
749.        dias = []
750.        for i in range (len(x_dias)):
751.            for j in range (len(x_dias[i])):
752.                if x_dias[i][j] != 0:
753.                    dias.append(x_dias[i][j])
754.        dias_str = []
755.        for i in range (len(dias)):
756.            dias_str.append(str(dias[i]))
```

```
757.
758.    #===> Creamos una lista con las escalas correspondientes a los num de días
759.        i = val_ini
760.        can_dias = []
761.        while i <= val_fin:
762.            can_dias.append(i)
763.            i = i + (cant_datos_mes / len(dias))
764.
765.        dias_int = []
766.        for i in range (len(can_dias)):
767.            dias_int.append(int(can_dias[i])+1)
768.
769.        plt.subplot(1,1,1)
770.    #===> Tiempo para la gráfica, considera posiciones del mes seleccionado
771.        plt.xlim(val_ini,val_fin)
772.
773.    #===> Subtítulo de la gráfica
774.        plt.title(u'Carga vs. Temperatura S/E '+sube +mesi +anio, fontweight='bold',
775.                  style='italic')
776.
777.    #=== > Etiq. del eje x (días del mes seleccionado)
778.        plt.xticks(dias_int, dias_str, size = 'small', color = 'b', rotation = 45)
779.
780.    #===> Condicional de gráficas, var generales: se, col
781.        if pos_se == 0:
782.            se = amba
783.            col = '#DCD037'
784.        elif pos_se == 1:
785.            se = bani
786.            col = '#4ACB71'
787.        elif pos_se == 2:
788.            se = puyo
789.            col = '#349A9D'
790.        elif pos_se == 3:
791.            se = tena
792.            col = '#CC8634'
793.        elif pos_se == 4:
794.            se = toto
795.            col = '#CD336F'
796.
797.        plt.plot(se, col, label = 'Potencia')
798.        plt.legend()
799.        plt.xlabel(u'TIEMPO [ días ]')
800.        plt.ylabel(u'CARGA  [ kW ]')
801.        plt.twinx()
802.        g2, = plt.plot(temp, 'b', label = 'Temperatura en °C')
803.        plt.ylabel(u'TEMPERATURA  [ °C ]')
804.        plt.legend(handles=[g2], loc=4)
805.        plt.grid(color='#C8C8C8', linestyle='-', linewidth=0.5)
806.
807.    #================================ SCATTER MENSUAL ================================
808.
809.    #===> Si se activa el check, genera el scatter
810.        if val_check == 1:
811.
812.            for i in range (val_ini):
813.                se.pop(0)
814.                temp.pop(0)
815.
816.            i = cant_datos_mes + 1
817.            inicio = len(se)
818.            for i in range (inicio - cant_datos_mes - 1):
819.                se.pop(cant_datos_mes + 1)
820.                temp.pop(cant_datos_mes + 1)
```

```
821.
822.
823.            fig2, (ax1) = plt.subplots(nrows=1,ncols=1,figsize=(15, 8))
824.
825.            ax1.scatter(x=temp, y=se, marker='.',linewidth=3, c=col)
826.            ax1.set_title('Scatter MENSUAL: $Temperatura$ vs. $Potencia$ $S/E$ '+sube
       )
827.            ax1.set_ylabel('$Carga$  $[$ $kW$ $]$')
828.            ax1.set_xlabel('$Temperatura$ $en$ $°C$')
829.
830.    #=============================================================================
831.    #=============================================================================
832.
833.
834.
835.
836.    #=============================================================================
837.    #============================== GRÁFICA SEMANAL ==============================
838.    #=============================================================================
839.    if   pos_menu_2==2:
840.
841.        #global pos_mes, po_mes
842.        global po_sem, dias_restantes
843.        mesi = ' / '+po_mes+ ' '
844.
845.        import calendar
846.
847.    #===> Tamaño de la ventana de la gráfica
848.        plt.subplots(figsize=(15, 8))
849.    #===> Título general superior
850.        plt.suptitle(u' GRÁFICA SEMANAL ', fontsize=16, fontweight='bold')
851.
852.    #===> Subrutina, genera posiciones de los días según el mes y semana seleccionada

853.        x_meses = [mes.index('ENERO'), mes.index('FEBRERO'), mes.index('MARZO'),
854.                   mes.index('ABRIL'), mes.index('MAYO'),    mes.index('JUNIO'),
855.                   mes.index('JULIO'), mes.index('AGOSTO'),  mes.index('SEPTIEMBRE'),

856.                   mes.index('OCTUBRE'), mes.index('NOVIEMBRE'), mes.index('DICIEMBRE
       ')]
857.        for i in range (12):
858.            if pos_mes == i:
859.                if pos_mes == 11:
860.                    val_ini = x_meses[i]
861.                    #val_fin = 8759
862.                else:
863.                    val_ini = x_meses[i]
864.                    #val_fin = x_meses[i+1]-1
865.
866.        if int(po_sem) == 1:
867.            val_ini_sem = val_ini
868.        elif int(po_sem) == 2:
869.            val_ini_sem = val_ini + 24 * dias_restantes
870.        elif int(po_sem) == 3:
871.            val_ini_sem = val_ini + 24 * dias_restantes + 24 * 7
872.        elif int(po_sem) == 4:
873.            val_ini_sem = val_ini + 24 * dias_restantes + 24 * 14
874.        elif int(po_sem) == 5:
875.            val_ini_sem = val_ini + 24 * dias_restantes + 24 * 21
876.        elif int(po_sem) == 6:
877.            val_ini_sem = val_ini + 24 * dias_restantes + 24 * 28
878.
879.
880.        val_fin = val_ini_sem + 167
881.        cant_datos_mes = val_fin - val_ini_sem
882.
```

```
883.    #===> Creamos una lista con los números de día del mes seleccionado (etiquetas)
884.
885.        dias_semana =['Lunes ','Martes ','Miércoles ','Jueves ','Viernes ','Sabado ',
        'Domingo ']
886.        numero_dias = []
887.        for i in range (dia[val_ini_sem], dia[val_ini_sem] + 7):
888.            numero_dias.append(i)
889.        dias_str = []
890.        for i in range (7):
891.            dias_str.append(dias_semana[i]+str(numero_dias[i]))
892.
893.    #===> Creamos una lista con las escalas correspondientes a los num de días
894.        i = val_ini_sem
895.        can_dias = []
896.        while i <= val_fin:
897.            can_dias.append(i)
898.            i = i + (cant_datos_mes / len(dias_str))
899.
900.        dias_int = []
901.        for i in range (len(can_dias)):
902.            dias_int.append(int(can_dias[i])+1)
903.
904.        plt.subplot(1,1,1)
905.    #===> Tiempo para la gráfica, considera posiciones del mes y SEMANA seleccionada

906.        plt.xlim(val_ini_sem,val_fin)
907.
908.    #===> Subtítulo de la gráfica
909.        plt.title(u'Carga vs. Temperatura S/E '+sube +' / Semana N° '+po_sem +mesi +a
        nio, fontweight='bold',
910.                  style='italic')
911.
912.    #=== > Etiq. del eje x (días del mes seleccionado)
913.        plt.xticks(dias_int, dias_str, size = 'small', color = 'b', rotation = 45)
914.
915.    #===> Condicional de gráficas, var generales: se, col
916.        if pos_se == 0:
917.            se = amba
918.            col = '#DCD037'
919.        elif pos_se == 1:
920.            se = bani
921.            col = '#4ACB71'
922.        elif pos_se == 2:
923.            se = puyo
924.            col = '#349A9D'
925.        elif pos_se == 3:
926.            se = tena
927.            col = '#CC8634'
928.        elif pos_se == 4:
929.            se = toto
930.            col = '#CD336F'
931.
932.        plt.plot(se, col, label = 'Potencia')
933.        plt.legend()
934.        plt.xlabel(u'TIEMPO [ días ]')
935.        plt.ylabel(u'CARGA  [ kW ]')
936.        plt.twinx()
937.        g2, = plt.plot(temp, 'b', label = 'Temperatura en °C')
938.        plt.ylabel(u'TEMPERATURA  [ °C ]')
939.        plt.legend(handles=[g2], loc=4)
940.        plt.grid(color='#C8C8C8', linestyle='-', linewidth=0.5)
941.
942.    #================================ SCATTER SEMANAL ================================

943.
944.    #===> Si se activa el check, genera el scatter
```

```
945.        if val_check == 1:
946.
947.            for i in range (val_ini_sem):
948.                se.pop(0)
949.                temp.pop(0)
950.
951.            i = cant_datos_mes + 1
952.            inicio = len(se)
953.            for i in range (inicio - cant_datos_mes - 1):
954.                se.pop(cant_datos_mes + 1)
955.                temp.pop(cant_datos_mes + 1)
956.
957.
958.            fig2, (ax1) = plt.subplots(nrows=1,ncols=1,figsize=(15, 8))
959.
960.            ax1.scatter(x=temp, y=se, marker='.',linewidth=3, c=col)
961.            ax1.set_title('Scatter SEMANAL: $Temperatura$ vs. $Potencia$ $S/E$ '+sube
    )
962.            ax1.set_ylabel('$Carga$  $[$ $kW$ $]$')
963.            ax1.set_xlabel('$Temperatura$ $en$ $°C$')
964.
965.    #=============================================================================
966.    #=============================================================================
967.
968.
969.
970.
971.    #=============================================================================
972.    #================================ GRÁFICA DIARIA ==============================
973.    #=============================================================================
974.    if   pos_menu_2==3:
975.
976.        #global pos_mes, po_mes
977.        global pos_dia, po_dia
978.
979.    #    mesi = ' / '+po_mes+ ' '
980.
981.        import calendar
982.
983.    #===> Tamaño de la ventana de la gráfica
984.        plt.subplots(figsize=(15, 8))
985.
986.    #===> Título general superior
987.        plt.suptitle(u' GRÁFICA DIARIA ', fontsize=16, fontweight='bold')
988.
989.    #===> Subrutina, genera posiciones de los días según el mes y semana seleccionada

990.        x_meses = [mes.index('ENERO'), mes.index('FEBRERO'), mes.index('MARZO'),
991.                   mes.index('ABRIL'), mes.index('MAYO'),    mes.index('JUNIO'),
992.                   mes.index('JULIO'), mes.index('AGOSTO'),  mes.index('SEPTIEMBRE'),

993.                   mes.index('OCTUBRE'), mes.index('NOVIEMBRE'), mes.index('DICIEMBRE
    ')]
994.        for i in range (12):
995.            if pos_mes == i:
996.                if pos_mes == 11:
997.                    val_ini = x_meses[i]
998.                else:
999.                    val_ini = x_meses[i]
1000.
1001.       val_ini = val_ini + pos_dia * 24
1002.       val_fin = val_ini + 24
1003.
1004.       print(val_ini)
1005.       print(val_fin)
1006.
```

```
1007.
1008.  #===> Creamos una lista con los números de horas del día seleccionado (etiquetas)
1009.
1010.      horas_dia = []
1011.      horas_str = []
1012.      for i in range (1,25):
1013.          horas_dia.append(i)
1014.          horas_str.append(str(horas_dia[i-1])+':00:00')
1015.
1016.  #===> Creamos una lista con las escalas correspondientes a los num de horas
1017.
1018.      i = val_ini
1019.      can_horas = []
1020.      while i <= val_fin:
1021.          can_horas.append(i)
1022.          i = i + (24 / len(horas_str))
1023.
1024.
1025.      plt.subplot(1,1,1)
1026.  #===> Tiempo para la gráfica, considera posiciones del mes y SEMANA seleccionada
1027.      plt.xlim(val_ini,val_fin)
1028.
1029.  #===> Subtítulo de la gráfica
1030.      plt.title(u'Carga vs. Temperatura S/E '+sube +' / '+po_dia +' de '+po_mes +'
      de '+anio, fontweight='bold',
1031.                  style='italic')
1032.
1033.  #=== > Etiq. del eje x (días del mes seleccionado)
1034.      plt.xticks(can_horas, horas_str, size = 'small', color = 'b', rotation = 45)
1035.
1036.  #===> Condicional de gráficas, var generales: se, col
1037.      if pos_se == 0:
1038.          se = amba
1039.          col = '#DCD037'
1040.      elif pos_se == 1:
1041.          se = bani
1042.          col = '#4ACB71'
1043.      elif pos_se == 2:
1044.          se = puyo
1045.          col = '#349A9D'
1046.      elif pos_se == 3:
1047.          se = tena
1048.          col = '#CC8634'
1049.      elif pos_se == 4:
1050.          se = toto
1051.          col = '#CD336F'
1052.
1053.      plt.plot(se, col, label = 'Potencia')
1054.      plt.legend()
1055.      plt.xlabel(u'TIEMPO [ Horas ]')
1056.      plt.ylabel(u'CARGA  [ kW ]')
1057.      plt.twinx()
1058.      g2, = plt.plot(temp, 'b', label = 'Temperatura en °C')
1059.      plt.ylabel(u'TEMPERATURA  [ °C ]')
1060.      plt.legend(handles=[g2], loc=4)
1061.      plt.grid(color='#C8C8C8', linestyle='-', linewidth=0.5)
1062.
1063.  #================================ SCATTER DIARIO ================================
1064.
1065.  #===> Si se activa el check, genera el scatter
1066.      if val_check == 1:
1067.
```

```
            for i in range (val_ini):
                se.pop(0)
                temp.pop(0)

            i = 24 + 1
            inicio = len(se)
            for i in range (inicio - 24 - 1):
                se.pop(24 + 1)
                temp.pop(24 + 1)


            fig2, (ax1) = plt.subplots(nrows=1,ncols=1,figsize=(15, 8))

            ax1.scatter(x=temp, y=se, marker='.',linewidth=3, c=col)
            ax1.set_title('Scatter DIARIO: $Temperatura$ vs. $Potencia$ $S/E$ '+sube)

            ax1.set_ylabel('$Carga$  $[$ $kW$ $]$')
            ax1.set_xlabel('$Temperatura$ $en$ $°C$')

    #=============================================================================
    #=============================================================================

    print(' ')
    print('Completed...                    OK...!')
    print(' ')
```

3. MÓDULO DE GRÁFICAS ACUMULADAS

```
"""
ESCUELA POLITÉCNICA NACIONAL
AUTOR: ALEXIS GUAMAN FIGUEROA
3_MÓDULO DE GRÁFICAS ACUMULADAS
"""

#importa librería pandas
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np


#============================== VAR GLOBALES ================================
global pos_menu_2
global pos_se, po_anio, po_se, val_check,po_dia
anio = po_anio
sube = po_se + ' '

#============================================================================

print(' ')
print(' ')
print('Procesing Plots...')
print(' ')

doc_clima = 'HIS_CLIMA_' + anio + '.xls'
doc_demand = 'HIS_POT_' + anio + '.xls'

df1 = pd.read_excel(doc_clima, sheetname='Hoja1',  header=None)

mes = df1.iloc[:,0].values.tolist()
mes.pop(0)

dia = df1.iloc[:,1].values.tolist()
```

```
dia.pop(0)

hora = df1.iloc[:,2].values.tolist()
hora.pop(0)

temp = df1.iloc[:,4].values.tolist()
temp.pop(0)



#===> Reemplaza valores (0) con (nan)
for i in range(len(temp)):
    if temp[i] == 0:
        temp[i] = float('nan')
#=============================================================================
df2 = pd.read_excel(doc_demand, sheetname='Hoja1',  header=None)

amba = df2.iloc[:,4].values.tolist()
amba.pop(0)

toto = df2.iloc[:,5].values.tolist()
toto.pop(0)

puyo = df2.iloc[:,6].values.tolist()
puyo.pop(0)

tena = df2.iloc[:,7].values.tolist()
tena.pop(0)

bani = df2.iloc[:,8].values.tolist()
bani.pop(0)

#===> Reemplaza valores (0) con (nan)
for i in range(len(temp)):
    if amba[i] == 0:
        amba[i] = float('nan')
    if toto[i] == 0:
        toto[i] = float('nan')
    if puyo[i] == 0:
        puyo[i] = float('nan')
    if tena[i] == 0:
        tena[i] = float('nan')
    if bani[i] == 0:
        bani[i] = float('nan')
#===> Se establece una matriz con los datos importados
data = np.column_stack((temp,amba,toto, puyo, tena, bani))



#===> Condicional de gráficas, var generales: se, col
if pos_se == 0:
    se = amba
    col = '#DCD037'
elif pos_se == 1:
    se = bani
    col = '#4ACB71'
elif pos_se == 2:
    se = puyo
    col = '#349A9D'
elif pos_se == 3:
    se = tena
    col = '#CC8634'
elif pos_se == 4:
    se = toto
    col = '#CD336F'
```

```
1194.
1195.
1196.
1197. for i in range (len(dia)):
1198.     if mes[i] == 'ENERO':
1199.         mes[i] = '01'
1200.     elif mes [i] == 'FEBRERO':
1201.         mes[i] = '02'
1202.     elif mes [i] == 'MARZO':
1203.         mes[i] = '03'
1204.     elif mes [i] == 'ABRIL':
1205.         mes[i] = '04'
1206.     elif mes [i] == 'MAYO':
1207.         mes[i] = '05'
1208.     elif mes [i] == 'JUNIO':
1209.         mes[i] = '06'
1210.     elif mes [i] == 'JULIO':
1211.         mes[i] = '07'
1212.     elif mes [i] == 'AGOSTO':
1213.         mes[i] = '08'
1214.     elif mes [i] == 'SEPTIEMBRE':
1215.         mes[i] = '09'
1216.     elif mes [i] == 'OCTUBRE':
1217.         mes[i] = '10'
1218.     elif mes [i] == 'NOVIEMBRE':
1219.         mes[i] = '11'
1220.     elif mes [i] == 'DICIEMBRE':
1221.         mes[i] = '12'
1222.
1223. dia_str = []
1224. for i in range (len(dia)):
1225.     dia_str.append( str (dia[i]))
1226.     if dia[i] < 10:
1227.         dia_str[i] = '0'+dia_str[i]
1228.
1229. data_2 = np.column_stack((mes,dia_str,se,hora))
1230.
1231. df_general = pd.DataFrame(np.array(data_2))
1232.
1233. df_general = df_general.assign(idx = (anio + '-' + df_general[0] + '-' +
1234.                                       df_general[1]))
1235.
1236. df_general = df_general.set_index('idx')
1237.
1238. del df_general[0]
1239. del df_general[1]
1240.
1241. df_general = df_general.astype(np.float)
1242.
1243. df_general_pivot = df_general.pivot(columns=3)
1244.
1245.
1246. df_general_pivot.T.plot(figsize=(15,8), legend=False, color=col, alpha=0.02)
1247.
1248. #===> Título general superior
1249. plt.suptitle(u' GRÁFICA DIARIA ACUMULADA ', fontsize=16, fontweight='bold')
1250.
1251. #===> Subtítulo de la gráfica
1252. plt.title(u'Comparativa para la S/E '+sube +', año '+anio, fontweight='bold',
1253.           style='italic')
1254.
1255.
1256.
1257. plt.ylabel(u'CARGA  [ kW ]')
1258. plt.xlabel(u'TIEMPO [ Horas ]')
1259.
```

```
plt.grid(color='#C8C8C8', linestyle='-', linewidth=0.5)


#===> Creamos una lista con los números de horas del día seleccionado (etiquetas)

horas_dia = []
horas_str = []
for i in range (1,25):
    horas_dia.append(i)
    horas_str.append(str(horas_dia[i-1])+':00:00')

#===> Creamos una lista con las escalas correspondientes a los num de horas

can_horas = []
for i in range (24):
    can_horas.append(i)

#=== > Etiq. del eje x (días del mes seleccionado)
plt.xticks(can_horas, horas_str, size = 'small', color = 'b', rotation = 45)
```

4. MÓDULO DE PROYECCIÓN SEMANAL

```
"""
ESCUELA POLITÉCNICA NACIONAL
AUTOR: ALEXIS GUAMAN FIGUEROA
4_MÓDULO DE PROYECCIÓN SEMANAL
"""
#*****************************************************************************
#**************************GENERADOR DE PROYECCIÓN **************************
#*****************************************************************************

print (' ')
print (' ')
print ('**  Programa de proyección semanal... ')
print (' ')
print (' ')

from tkinter import *
import tkinter.ttk as ttk
from tkinter import messagebox

#--------------- PARÁMETROS INICIALES, CREACIÓN DE LA VENTANA RAIZ ------------

# Se genera la raiz o ventana principal
raiz = Tk()
# Se establece el tamaño y posición de la ventana principal raiz
raiz.geometry('600x520+100+100') #+600+200
# TÍTULO DE LA VENTANA PRINCIPAL
raiz.title('Proyección Semanal')
# impide redimensionar la ventana principal raiz
raiz.resizable(0,0)
# Se estable un ícono para la ventana principal
raiz.iconbitmap('epn.ico')

# Se establece el color de fondo de la ventana principal RAIZ
#raiz.config(bg='#34495E')


#===>Texto de ayuda (Acerca de)
def infoAdicional():
```

```
1320.     messagebox.showinfo('Acerca de este programa','PROYECTO PREVIO A LA OBTENCIÓN D
       EL TÍTULO DE INGENIERO ELÉCTRICO\n\nEste Programa fue realizado por Alexis Guamán
        Figueroa')
1321. 
1322. #===>Ejecuta archivo de proyección semanal
1323. def proySem():
1324.     raiz.destroy()
1325.     runfile('C:/Users/ALEXIS GUAMAN/Dropbox/TESIS_1/0_3_ANALIZADOR DE GRÁFICAS/GRAP
       HIC ANALIZER.py', wdir='C:/Users/ALEXIS GUAMAN/Dropbox/TESIS_1/0_3_ANALIZADOR DE
       GRÁFICAS')
1326. 
1327. #===>Cierra el proyecto
1328. def salirProyecto():
1329.     PregCierre = messagebox.askquestion('Cerrar programa','Desea cerrar el programa
       ?')
1330.     if PregCierre == 'yes':
1331.         raiz.destroy()
1332. 
1333. #=============================================================================
1334. #=========================SUBRUTINA PARA RESET ================================
1335. #=============================================================================
1336. def boton_reset():
1337.     global raiz
1338.     raiz.destroy()
1339.     runfile('C:/Users/ALEXIS GUAMAN/Dropbox/TESIS_1/0_3_ANALIZADOR DE GRÁFICAS/GRAP
       HIC ANALIZER.py', wdir='C:/Users/ALEXIS GUAMAN/Dropbox/TESIS_1/0_3_ANALIZADOR DE
       GRÁFICAS')
1340. 
1341. 
1342. #=============================================================================
1343. #====================== Se establece el menú de la raíz ======================
1344. #=============================================================================
1345. 
1346. barraMenu = Menu(raiz)
1347. raiz.config(menu=barraMenu)
1348. 
1349. archivoMenu = Menu(barraMenu, tearoff = 0)
1350. 
1351. #archivoMenu.add_command(label = 'Nuevo')
1352. #archivoMenu.add_command(label = 'Guardar')
1353. #archivoMenu.add_command(label = 'Guardar Como')
1354. #archivoMenu.add_separator()
1355. archivoMenu.add_command(label = 'Nuevo Análiss de Datos',command = proySem)
1356. archivoMenu.add_separator()
1357. archivoMenu.add_command(label = 'Reiniciar', command = boton_reset)
1358. archivoMenu.add_command(label = 'Salir', command = salirProyecto)
1359. 
1360. #archivoEdicion = Menu(barraMenu, tearoff = 0)
1361. #archivoEdicion.add_command(label = 'Copiar')
1362. #archivoEdicion.add_command(label = 'Cortar')
1363. #archivoEdicion.add_command(label = 'Pegar')
1364. 
1365. #archivoHerramientas = Menu(barraMenu, tearoff = 0)
1366. 
1367. archivoAyuda = Menu(barraMenu, tearoff = 0)
1368. archivoAyuda.add_command(label = 'Acerca de...', command = infoAdicional)
1369. 
1370. barraMenu.add_cascade(label = 'Archivo', menu = archivoMenu)
1371. #barraMenu.add_cascade(label = 'Edición', menu = archivoEdicion)
1372. #barraMenu.add_cascade(label = 'Herramientas', menu = archivoHerramientas)
1373. barraMenu.add_cascade(label = 'Ayuda', menu = archivoAyuda)
1374. 
1375. #=============================================================================
1376. #=============================================================================
1377. 
1378. 
```

```
#------------------------ Ingreso de imágenes -------------------------------

imag_eeasa_2 = PhotoImage(file='eeasa_peq_2.png')
Label(raiz, image = imag_eeasa_2).place(x=140,y=20)

imag_pronost_2 = PhotoImage(file='energy.png')
Label(raiz, image = imag_pronost_2).place(x=10,y=390)

#------------------------ Ingreso de textos ---------------------------------

txt_1 = Label(raiz, text = 'LOAD FORECASTING', fg='#616A6B',
              font=('Arial',12)).place(x=200,y=150)

txq_2 = Label(raiz, text = 'Seleccione la metodología: ',fg='#616A6B',
              font=('Arial',12)).place(x=30,y=200)
txq_3 = Label(raiz, text = 'Realizado por: ',fg='#616A6B',
              font=('Arial',12)).place(x=30,y=230)
txq_4 = Label(raiz, text = 'Generar gráficas: ',fg='#616A6B',
              font=('Arial',12)).place(x=30,y=260)

txt_5 = Label(raiz, text = 'SELECCIONE LAS FECHAS: ', fg='#616A6B',
              font=('Arial',12)).place(x=20,y=310)
#------------------------ DATOS DE PROYECCIÓN -------------------------------
txq_6 = Label(raiz, text = 'Datos\nProyección: ',fg='#616A6B',
              font=('Arial',12)).place(x=215,y=340)

Label(raiz, text = 'AÑO: ', fg='#616A6B',font=('Arial',10)).place(x=140,y=390)
cbx_anio_p = ttk.Combobox(values=['2013','2014','2015','2016','2017','2018'],state=
  "readonly")
cbx_anio_p.place(x=190,y=390)

Label(raiz, text = 'MES: ', fg='#616A6B',font=('Arial',10)).place(x=140,y=420)
cbx_mes_p = ttk.Combobox(values=['ENERO','FEBRERO','MARZO','ABRIL','MAYO','JUNIO','
  JULIO','AGOSTO','SEPTIEMBRE','OCTUBRE','NOVIEMBRE','DICIEMBRE'],state="readonly")

cbx_mes_p.place(x=190,y=420)

Label(raiz, text = 'DÍA: ', fg='#616A6B',font=('Arial',10)).place(x=140,y=450)
cbx_dia_p = ttk.Combobox(values=['1','2','3','4','5','6','7','8','9','10','11','12'
  ,'13','14','15','16','17','18','19','20','21','22','23','24','25','26','27','28',
  '29','30','31'],state="readonly")
cbx_dia_p.place(x=190,y=450)

#------------------------ DATOS DE COMPARACIÓN ------------------------------
txq_7 = Label(raiz, text = 'Datos\nComparación: ',fg='#616A6B',
              font=('Arial',12)).place(x=410,y=340)

Label(raiz, text = 'AÑO: ', fg='#616A6B',font=('Arial',10)).place(x=350,y=390)
cbx_anio_c = ttk.Combobox(values=['2013','2014','2015','2016','2017','2018'],state=
  "readonly")
cbx_anio_c.place(x=400,y=390)

Label(raiz, text = 'MES: ', fg='#616A6B',font=('Arial',10)).place(x=350,y=420)
cbx_mes_c = ttk.Combobox(values=['ENERO','FEBRERO','MARZO','ABRIL','MAYO','JUNIO','
  JULIO','AGOSTO','SEPTIEMBRE','OCTUBRE','NOVIEMBRE','DICIEMBRE'],state="readonly")

cbx_mes_c.place(x=400,y=420)

Label(raiz, text = 'DÍA: ', fg='#616A6B',font=('Arial',10)).place(x=350,y=450)
cbx_dia_c = ttk.Combobox(values=['1','2','3','4','5','6','7','8','9','10','11','12'
  ,'13','14','15','16','17','18','19','20','21','22','23','24','25','26','27','28',
  '29','30','31'],state="readonly")
cbx_dia_c.place(x=400,y=450)


```

```
1434. Label(raiz, text = 'AGF ®', fg='#616A6B',font=('Tw Cen MT',9)).place(x=545,y=475)
1435. 
1436. #----------------------Ingreso de MENÚ txq_2 METODOLOGÍA ----------------------
1437. cbx_txq_2 = ttk.Combobox(values=['Promedios (CECON)', 'REGRESIÓN LINEAL (LR)','S.V.
       M.','M.L.P.'],state="readonly")
1438. cbx_txq_2.place(x=250,y=200)
1439. 
1440. #----------------------Ingreso de MENÚ txq_3 REALIZD POR ----------------------
1441. cbx_txq_3 = ttk.Combobox(values=['A.G','B.M','A.T','C.G','L.T','E.Ñ','E.T','Otro'],
       state="readonly")
1442. cbx_txq_3.place(x=250,y=230)
1443. 
1444. #----------------------Ingreso de MENÚ txq_4 COMPARACIÓN ----------------------
1445. cbx_txq_4 = ttk.Combobox(values=['SI','NO'],state="readonly")
1446. cbx_txq_4.place(x=250,y=260)
1447. 
1448. 
1449. #=============================================================================
1450. #============================== BOTÓN GENERAR ================================
1451. #=============================================================================
1452. b_go = Button(raiz, text = '  GENERAR PROYECCIÓN  ',bg='#2ECC71',fg='#1A5276',
1453.                width=20, height=5, command = lambda:boton_Proyeccion())
1454. b_go.place(x=420,y=200)
1455. 
1456. def boton_Proyeccion():
1457. 
1458. #=============================================================================
1459. #======================== DECLARACIÓN DE VARIABLES ===========================
1460. #=============================================================================

1461.     global pos_Metodologia, pos_Realizado, pos_Comparacion
1462.     pos_Metodologia = cbx_txq_2.current()
1463.     pos_Realizado   = cbx_txq_3.current()
1464.     pos_Comparacion = cbx_txq_4.current()
1465. 
1466.     global pos_anio_p, pos_mes_p, pos_dia_p
1467.     pos_anio_p = cbx_anio_p.current()
1468.     pos_mes_p  = cbx_mes_p. current()
1469.     pos_dia_p  = cbx_dia_p. current()
1470. 
1471.     global pos_anio_c, pos_mes_c, pos_dia_c
1472.     pos_anio_c = cbx_anio_c.current()
1473.     pos_mes_c  = cbx_mes_c. current()
1474.     pos_dia_c  = cbx_dia_c. current()
1475. #*****************************************************************************
1476.     global po_Metodologia, po_Realizado, po_Comparacion
1477.     po_Metodologia = cbx_txq_2.get()
1478.     po_Realizado   = cbx_txq_3.get()
1479.     po_Comparacion = cbx_txq_4.get()
1480. 
1481.     global po_anio_p, po_mes_p, po_dia_p
1482.     po_anio_p = cbx_anio_p.get()
1483.     po_mes_p  = cbx_mes_p. get()
1484.     po_dia_p  = cbx_dia_p. get()
1485. 
1486.     global po_anio_c, po_mes_c, po_dia_c
1487.     po_anio_c = cbx_anio_c.get()
1488.     po_mes_c  = cbx_mes_c. get()
1489.     po_dia_c  = cbx_dia_c. get()
1490. #=============================================================================
1491. #=============================================================================
1492. 
1493. 
1494. #=============================================================================
1495. #======================== VERIFICA DATOS FALTANTES ===========================
```

```
1496. #================================================================================

1497.     from calendar import monthrange
1498. 
1499.     if pos_Metodologia == -1:
1500.         messagebox.showinfo(message="Ingrese la Metodología", title="Faltan datos")

1501.     elif pos_Realizado == -1:
1502.         messagebox.showinfo(message="Ingrese Responsable", title="Faltan datos")
1503.     elif pos_Comparacion == -1:
1504.         messagebox.showinfo(message="Ingrese Comparación", title="Faltan datos")
1505. #*********************************************************************************

1506.     elif pos_anio_p == -1:
1507.         messagebox.showinfo(message="Ingrese Año de Proyección", title="Faltan dato
       s")
1508.     elif pos_mes_p == -1:
1509.         messagebox.showinfo(message="Ingrese Mes de Proyección", title="Faltan dato
       s")
1510.     elif pos_dia_p == -1:
1511.         messagebox.showinfo(message="Ingrese Día de Proyección", title="Faltan dato
       s")
1512. 
1513.     elif int(po_dia_p) > monthrange(int(po_anio_p), pos_mes_p+1)[1]:
1514.         messagebox.showinfo(message='No existe día '+po_dia_p +' para el mes ' +po_
       mes_p + ' de '+ po_anio_p, title="Dato erróneo")
1515. #*********************************************************************************

1516.     elif pos_anio_c == -1:
1517.         messagebox.showinfo(message="Ingrese Año de Comparación", title="Faltan dat
       os")
1518.     elif pos_mes_c == -1:
1519.         messagebox.showinfo(message="Ingrese Mes de Comparación", title="Faltan dat
       os")
1520.     elif pos_dia_c == -1:
1521.         messagebox.showinfo(message="Ingrese Día de Comparación", title="Faltan dat
       os")
1522. 
1523.     elif int(po_dia_c) > monthrange(int(po_anio_c), pos_mes_c+1)[1]:
1524.         messagebox.showinfo(message='No existe día '+po_dia_c +' para el mes ' +po_
       mes_c + ' de '+ po_anio_c, title="Dato erróneo")
1525. #*********************************************************************************

1526. 
1527.     else:
1528.         if pos_Metodologia == 0:
1529.             print(' ')
1530.             print ('EJECUTADO PROYECCIÓN POR METOLOGÍA PROMEDIOS...')
1531.             print(' ')
1532.             runfile('C:/Users/ALEXIS GUAMAN/Dropbox/TESIS_1/0_3_ANALIZADOR DE GRÁFI
       CAS/04_proyeccion_promedios.py', wdir='C:/Users/ALEXIS GUAMAN/Dropbox/TESIS_1/0_3
       _ANALIZADOR DE GRÁFICAS')
1533.         elif pos_Metodologia == 1:
1534.             print(' ')
1535.             print ('EJECUTADO PROYECCIÓN POR METOLOGÍA REGRESIÓN LINEAL...')
1536.             print(' ')
1537.             runfile('C:/Users/ALEXIS GUAMAN/Dropbox/TESIS_1/0_3_ANALIZADOR DE GRÁFI
       CAS/06_RL.py', wdir='C:/Users/ALEXIS GUAMAN/Dropbox/TESIS_1/0_3_ANALIZADOR DE GRÁ
       FICAS')
1538.         elif pos_Metodologia == 2:
1539.             print(' ')
1540.             print ('EJECUTADO PROYECCIÓN POR METOLOGÍA MÁQUINA DE VECTOR SOPORTE...
       ')
1541.             print(' ')
```

```
            runfile('C:/Users/ALEXIS GUAMAN/Dropbox/TESIS_1/0_3_ANALIZADOR DE GRÁFICAS/07_SVM.py', wdir='C:/Users/ALEXIS GUAMAN/Dropbox/TESIS_1/0_3_ANALIZADOR DE GRÁFICAS')
        elif pos_Metodologia == 3:
            print(' ')
            print ('EJECUTADO PROYECCIÓN POR METOLOGÍA PERCEPTRÓN MULTICAPA...')
            print(' ')
            runfile('C:/Users/ALEXIS GUAMAN/Dropbox/TESIS_1/0_3_ANALIZADOR DE GRÁFICAS/08_MLP.py', wdir='C:/Users/ALEXIS GUAMAN/Dropbox/TESIS_1/0_3_ANALIZADOR DE GRÁFICAS')


#================================================================================
#================================================================================
raiz.mainloop()

print (' ')
print (' ')
print ('**  COMPLETADO...!')
print (' ')
print (' ')
```

5. MÓDULO DE PROYECCIÓN CON ALGORITMO ARIMA

```
"""
ESCUELA POLITÉCNICA NACIONAL
AUTOR: ALEXIS GUAMAN FIGUEROA
5_MÓDULO DE PROYECCIÓN CON ALGORITMO ARIMA
"""
from time import time
def test():


    #importa librerías
    import pandas as pd
    import matplotlib.pyplot as plt
    import numpy as np


    from pandas import ExcelWriter

    from xlrd import open_workbook
    from xlutils.copy import copy


    from tkinter import messagebox

    import calendar
    from datetime import datetime,timedelta
    #================================ VAR GLOBALES ================================

    global pos_Metodologia, pos_Realizado, pos_Comparacion
    global pos_anio_p, pos_mes_p, pos_dia_p
    global pos_anio_c, pos_mes_c, pos_dia_c
    #**********************************************************************************

    global po_Metodologia, po_Realizado, po_Comparacion
    global po_anio_p, po_mes_p, po_dia_p
    global po_anio_c, po_mes_c, po_dia_c
    #================================================================================


    doc_Proy = 'HIS_POT_' + po_anio_p + '.xls'
```

```
1595.     doc_Comp = 'HIS_POT_' + po_anio_c + '.xls'
1596.
1597.
1598.     df_Proy = pd.read_excel(doc_Proy, sheetname='Hoja1',  header=None)
1599.     df_Comp = pd.read_excel(doc_Comp, sheetname='Hoja1',  header=None)
1600.
1601.     #=============================================================================
1602.     #========================== SEMANA DE PROYECCIÓN ==============================
1603.     #=============================================================================
1604.
1605.     mes_p = df_Proy.iloc[:,0].values.tolist()
1606.     mes_p.pop(0)
1607.
1608.     dia_p = df_Proy.iloc[:,1].values.tolist()
1609.     dia_p.pop(0)
1610.
1611.     hora_p = df_Proy.iloc[:,2].values.tolist()
1612.     hora_p.pop(0)
1613.
1614.     amba_p = df_Proy.iloc[:,4].values.tolist()
1615.     amba_p.pop(0)
1616.
1617.     toto_p = df_Proy.iloc[:,5].values.tolist()
1618.     toto_p.pop(0)
1619.
1620.     puyo_p = df_Proy.iloc[:,6].values.tolist()
1621.     puyo_p.pop(0)
1622.
1623.     tena_p = df_Proy.iloc[:,7].values.tolist()
1624.     tena_p.pop(0)
1625.
1626.     bani_p = df_Proy.iloc[:,8].values.tolist()
1627.     bani_p.pop(0)
1628.
1629.     ##===> Reemplaza valores (0) con (nan)
1630.     #for i in range(len(mes_p)):
1631.     #    if amba_p[i] == 0:
1632.     #        amba_p[i] = float('nan')
1633.     #    if toto_p[i] == 0:
1634.     #        toto_p[i] = float('nan')
1635.     #    if puyo_p[i] == 0:
1636.     #        puyo_p[i] = float('nan')
1637.     #    if tena_p[i] == 0:
1638.     #        tena_p[i] = float('nan')
1639.     #    if bani_p[i] == 0:
1640.     #        bani_p[i] = float('nan')
1641.
1642.     #===> Reemplaza valores (0) con (el inmediato siguiente)
1643.     for i in range(len(mes_p)):
1644.         if amba_p[i] == 0:
1645.             amba_p[i] = amba_p[i+1]
1646.         if toto_p[i] == 0:
1647.             toto_p[i] = toto_p[i+1]
1648.         if puyo_p[i] == 0:
1649.             puyo_p[i] = puyo_p[i+1]
1650.         if tena_p[i] == 0:
1651.             tena_p[i] = tena_p[i+1]
1652.         if bani_p[i] == 0:
1653.             bani_p[i] = bani_p[i+1]
1654.
1655.
1656.
1657.
```

```
1658.
1659.     #===> Se establece una matriz con los datos importados
1660.     #data_p = np.column_stack((amba_p, toto_p, puyo_p, tena_p, bani_p))
1661.
1662.     #================================================================================

1663.     #================================================================================

1664.     #===> Día de inicio de la semana de proyección
1665.
1666.     if po_mes_p == 'ENERO':
1667.         if int(po_dia_p) < 28:
1668.
1669.             doc_Proy_EXTRA_p = 'HIS_POT_' + str(int(po_anio_p)-1) + '.xls'
1670.             df_Proy_EXTRA_p = pd.read_excel(doc_Proy_EXTRA_p, sheetname='Hoja1',  h
      eader=None)
1671.             df_Proy_Extra_p = df_Proy_EXTRA_p[-(24*31):]
1672.
1673.             mess_extra = df_Proy_Extra_p.iloc[:,0].values.tolist()
1674.             amba_extra = df_Proy_Extra_p.iloc[:,4].values.tolist()
1675.             toto_extra = df_Proy_Extra_p.iloc[:,5].values.tolist()
1676.             puyo_extra = df_Proy_Extra_p.iloc[:,6].values.tolist()
1677.             tena_extra = df_Proy_Extra_p.iloc[:,7].values.tolist()
1678.             bani_extra = df_Proy_Extra_p.iloc[:,8].values.tolist()
1679.             #===> Obtengo únicamente el mes de diciembre del año pasado
1680.             mess_extra = mess_extra[-(24*31):]
1681.             #===> Elimino el mes de diciembre del año actual
1682.             mes_p = mes_p[:-(24*31)]
1683.             #===> Nueva lista con el Dic año pasado mas resto de datos año actual
1684.             mes_p = mess_extra + mes_p
1685.
1686.             amba_p = amba_extra + amba_p
1687.             toto_p = toto_extra + toto_p
1688.             puyo_p = puyo_extra + puyo_p
1689.             tena_p = tena_extra + tena_p
1690.             bani_p = bani_extra + bani_p
1691.
1692.
1693.     #===> APUNTA AL MES SELECCIONADO DE PROYECCIÓN
1694.     x_meses_p = [mes_p.index('ENERO'),   mes_p.index('FEBRERO'),   mes_p.index('MAR
      ZO'),
1695.                  mes_p.index('ABRIL'),   mes_p.index('MAYO'),      mes_p.index('JUN
      IO'),
1696.                  mes_p.index('JULIO'),   mes_p.index('AGOSTO'),    mes_p.index('SEP
      TIEMBRE'),
1697.                  mes_p.index('OCTUBRE'), mes_p.index('NOVIEMBRE'), mes_p.index('DIC
      IEMBRE')]
1698.
1699.     for i in range (12):
1700.         if pos_mes_p == i:
1701.             if pos_mes_p == 11:
1702.                 ubic_mes = x_meses_p[i]
1703.             else:
1704.                 ubic_mes = x_meses_p[i] # VARIABLE Posición del mes seleccionado
1705.
1706.     #===> DEFINE LA UBICACIÓN DEL DÍA SELECCIONADO DE PROYECCIÓN
1707.     print(' ')
1708.     print ('Fecha de Proyección: ',po_dia_p,' de ',mes_p[ubic_mes],' de ',po_anio_p
      )
1709.     if int(po_dia_p) == 1:
1710.         ubicacion = ubic_mes
1711.     else:
1712.         ubicacion = (int(po_dia_p)-1) * 24 + ubic_mes
1713. #    print(' ')
1714. #    print (ubicacion) #============================================================>
      >> VARIABLE DE UBICACIÓN EXACTA EN LISTA DE DATOS
```

```
1715.
1716.     amba_p = amba_p[:ubicacion]
1717.     toto_p = toto_p[:ubicacion]
1718.     puyo_p = puyo_p[:ubicacion]
1719.     tena_p = tena_p[:ubicacion]
1720.     bani_p = bani_p[:ubicacion]
1721.
1722.
1723.     #===> Valores inicio y fin para lista de datos a ser usada
1724.     val_ini = ubicacion - 24 * 7 * 4 #Posición del valor inicial de 4 semanas
1725.     val_fin = ubicacion - 1          #Posición del valor final
1726.     #=============================================================================

1727.     #=============================================================================

1728.
1729.     #===> Proyección semananal S/E Ambato
1730.     ProyAmba = []
1731.     for i in range (24*7):
1732.         ProyAmba.append((amba_p[val_ini + i + 7*24*0] +
1733.                          amba_p[val_ini + i + 7*24*1] +
1734.                          amba_p[val_ini + i + 7*24*2] +
1735.                          amba_p[val_ini + i + 7*24*3])/4)
1736.     #===> Proyección semananal S/E Totoras
1737.     ProyToto = []
1738.     for i in range (24*7):
1739.         ProyToto.append((toto_p[val_ini + i + 7*24*0] +
1740.                          toto_p[val_ini + i + 7*24*1] +
1741.                          toto_p[val_ini + i + 7*24*2] +
1742.                          toto_p[val_ini + i + 7*24*3])/4)
1743.     #===> Proyección semananal S/E Puyo
1744.     ProyPuyo = []
1745.     for i in range (24*7):
1746.         ProyPuyo.append((puyo_p[val_ini + i + 7*24*0] +
1747.                          puyo_p[val_ini + i + 7*24*1] +
1748.                          puyo_p[val_ini + i + 7*24*2] +
1749.                          puyo_p[val_ini + i + 7*24*3])/4)
1750.     #===> Proyección semananal S/E Tena
1751.     ProyTena = []
1752.     for i in range (24*7):
1753.         ProyTena.append((tena_p[val_ini + i + 7*24*0] +
1754.                          tena_p[val_ini + i + 7*24*1] +
1755.                          tena_p[val_ini + i + 7*24*2] +
1756.                          tena_p[val_ini + i + 7*24*3])/4)
1757.     #===> Proyección semananal S/E Baños
1758.     ProyBani = []
1759.     for i in range (24*7):
1760.         ProyBani.append((bani_p[val_ini + i + 7*24*0] +
1761.                          bani_p[val_ini + i + 7*24*1] +
1762.                          bani_p[val_ini + i + 7*24*2] +
1763.                          bani_p[val_ini + i + 7*24*3])/4)
1764.
1765.     #=============================================================================

1766.     #========================== SEMANA DE COMPARACIÓN ============================

1767.     #=============================================================================

1768.
1769.     mes_c = df_Comp.iloc[:,0].values.tolist()
1770.     mes_c.pop(0)
1771.
1772.     dia_c = df_Comp.iloc[:,1].values.tolist()
1773.     dia_c.pop(0)
1774.
1775.     hora_c = df_Comp.iloc[:,2].values.tolist()
```

```
    hora_c.pop(0)

    amba_c = df_Comp.iloc[:,4].values.tolist()
    amba_c.pop(0)

    toto_c = df_Comp.iloc[:,5].values.tolist()
    toto_c.pop(0)

    puyo_c = df_Comp.iloc[:,6].values.tolist()
    puyo_c.pop(0)

    tena_c = df_Comp.iloc[:,7].values.tolist()
    tena_c.pop(0)

    bani_c = df_Comp.iloc[:,8].values.tolist()
    bani_c.pop(0)

    #===> Reemplaza valores (0) con (nan)
    for i in range(len(mes_p)):
        if amba_c[i] == 0:
            amba_c[i] = float('nan')
        if toto_c[i] == 0:
            toto_c[i] = float('nan')
        if puyo_c[i] == 0:
            puyo_c[i] = float('nan')
        if tena_c[i] == 0:
            tena_c[i] = float('nan')
        if bani_c[i] == 0:
            bani_c[i] = float('nan')

    #===> Se establece una matriz con los datos importados
    #data_c = np.column_stack((amba_c, toto_c, puyo_c, tena_c, bani_c))

    #================================================================================

    #================================================================================

    if po_mes_c == 'ENERO':
        if int(po_dia_c) < 8:

            doc_Proy_EXTRA_c = 'HIS_POT_' + str(int(po_anio_c)-1) + '.xls'
            df_Proy_EXTRA_c = pd.read_excel(doc_Proy_EXTRA_c, sheetname='Hoja1',  h
eader=None)
            df_Proy_EXTRA_c = df_Proy_EXTRA_c[-(24*31):]

            mess_extra_c = df_Proy_EXTRA_c.iloc[:,0].values.tolist()
            amba_extra_c = df_Proy_EXTRA_c.iloc[:,4].values.tolist()
            toto_extra_c = df_Proy_EXTRA_c.iloc[:,5].values.tolist()
            puyo_extra_c = df_Proy_EXTRA_c.iloc[:,6].values.tolist()
            tena_extra_c = df_Proy_EXTRA_c.iloc[:,7].values.tolist()
            bani_extra_c = df_Proy_EXTRA_c.iloc[:,8].values.tolist()
            #===> Obtengo únicamente 7 días de diciembre del año pasado
            mess_extra_c = mess_extra_c[-(24*31):]
            #===> Elimino el mes de diciembre del año actual
            mes_c = mes_c[:-(24*31)]
            #===> Nueva lista con 7 días de Dic del año pasado mas resto de datos a
ño actual
            mes_c = mess_extra_c + mes_c

            amba_c = amba_extra_c + amba_c
            toto_c = toto_extra_c + toto_c
            puyo_c = puyo_extra_c + puyo_c
            tena_c = tena_extra_c + tena_c
            bani_c = bani_extra_c + bani_c

    #===> APUNTA AL MES SELECCIONADO DE COMPARACIÓN
```

```
1838.     x_meses_c = [mes_c.index('ENERO'),   mes_c.index('FEBRERO'),   mes_c.index('MAR
      ZO'),
1839.                  mes_c.index('ABRIL'),   mes_c.index('MAYO'),      mes_c.index('JUN
      IO'),
1840.                  mes_c.index('JULIO'),   mes_c.index('AGOSTO'),    mes_c.index('SEP
      TIEMBRE'),
1841.                  mes_c.index('OCTUBRE'), mes_c.index('NOVIEMBRE'), mes_c.index('DIC
      IEMBRE')]
1842.
1843.     for i in range (12):
1844.         if pos_mes_c == i:
1845.             if pos_mes_c == 11:
1846.                 ubic_mes_c = x_meses_c[i]
1847.             else:
1848.                 ubic_mes_c = x_meses_c[i] # VARIABLE Posición del mes seleccionado

1849.
1850.
1851.     #===> DEFINE LA UBICACIÓN DEL DÍA SELECCIONADO DE COMPARACIÓN
1852.     print(' ')
1853.     print ('Fecha Comparación: ',po_dia_c,' de ',mes_c[ubic_mes_c],' de ',po_anio_c
      )
1854.     if int(po_dia_c) == 1:
1855.         ubicacion_c = ubic_mes_c
1856.     else:
1857.         ubicacion_c = (int(po_dia_c)-1) * 24 + ubic_mes_c
1858.
1859. #    print(' ')
1860. #    print (ubicacion_c) #============================================================
      =>>> VARIABLE DE UBICACIÓN EXACTA EN LISTA DE DATOS de comparación
1861.
1862.     amba_c = amba_c[:ubicacion_c]
1863.     toto_c = toto_c[:ubicacion_c]
1864.     puyo_c = puyo_c[:ubicacion_c]
1865.     tena_c = tena_c[:ubicacion_c]
1866.     bani_c = bani_c[:ubicacion_c]
1867.
1868.     #===> Valores inicio y fin para lista de datos a ser usada
1869.     val_ini_c = ubicacion_c - 24 * 7 #Posición del valor inicial
1870.     val_fin_c = ubicacion_c - 1      #Posición del valor final
1871.
1872.     #===> Comparación semananal S/E Ambato
1873.     CompAmba = []
1874.     for i in range (24*7):
1875.         CompAmba.append(amba_c[val_ini_c + i ])
1876.     #===> Comparación semananal S/E Totoras
1877.     CompToto = []
1878.     for i in range (24*7):
1879.         CompToto.append(toto_c[val_ini_c + i ])
1880.     #===> Comparación semananal S/E Puyo
1881.     CompPuyo = []
1882.     for i in range (24*7):
1883.         CompPuyo.append(puyo_c[val_ini_c + i  ])
1884.     #===> Comparación semananal S/E Tena
1885.     CompTena = []
1886.     for i in range (24*7):
1887.         CompTena.append(tena_c[val_ini_c + i ])
1888.     #===> Comparación semananal S/E Baños
1889.     CompBani = []
1890.     for i in range (24*7):
1891.         CompBani.append(bani_c[val_ini_c + i ])
1892.
1893.      #===> Cuadro informativo de la semana proyectada y comparativa
1894.
1895.     x_proy = datetime(int(po_anio_p),(pos_mes_p+1),int(po_dia_p))
1896.     x_comp = datetime(int(po_anio_c),(pos_mes_c+1),int(po_dia_c))
```

```
1897.
1898.     dias_proy = timedelta(days = 28)
1899.     dias_comp = timedelta(days =  7)
1900.
1901.     fecha_ini_proy = x_proy - dias_proy
1902.     fecha_fin_proy = x_proy - timedelta(days = 1)
1903.
1904.     fecha_ini_comp = x_comp - dias_comp
1905.     fecha_fin_comp = x_comp - timedelta(days = 1)
1906.
1907.     sem_ini = x_proy
1908.     sem_fin = x_proy + timedelta(days = 6)
1909.
1910.     s_i_1 = fecha_ini_proy
1911.     s_i_2 = s_i_1 + timedelta(days = 7)
1912.     s_i_3 = s_i_2 + timedelta(days = 7)
1913.     s_i_4 = s_i_3 + timedelta(days = 7)
1914.
1915.     s_f_1 = fecha_ini_proy + timedelta(days = 6)
1916.     s_f_2 = s_f_1 + timedelta(days = 7)
1917.     s_f_3 = s_f_2 + timedelta(days = 7)
1918.     s_f_4 = s_f_3 + timedelta(days = 7)
1919.
1920.
1921.     #===> Salida de cuadro de texto informativo
1922.
1923.     messagebox.showinfo(message='Semana de Proyección:\n   del '+str(sem_ini.day)+'
       / '+
1924.                         str(sem_ini.month) +' / '+str(sem_ini.year)+'\tal '+
1925.                         str(sem_fin.day)+' / '+str(sem_fin.month)+' / '+str(sem_fin
       .year)+
1926.                         '\n\nSemana de Comparación:\n   del '+ str(fecha_ini_comp.d
       ay)+' / '+
1927.                         str(fecha_ini_comp.month)+' / '+str(fecha_ini_comp.year)+'\
       t al '+
1928.                         str(fecha_fin_comp.day)+' / '+str(fecha_fin_comp.month)+' /
        '+
1929.                         str(fecha_fin_comp.year)+
1930.                         '\n\nSemanas Promediadas:\n\n   Semana 1 : del '+str(s_i_1.
       day)+' / '+
1931.                         str(s_i_1.month)+ ' / '+str(s_i_1.year)+' al '+str(s_f_1.da
       y)+ ' / '+
1932.                         str(s_f_1.month)+ ' / '+str(s_f_1.year)+
1933.                         '\n   Semana 2 : del '+str(s_i_2.day)+' / '+str(s_i_2.month
       )+ ' / '+
1934.                         str(s_i_2.year)+' al '+str(s_f_2.day)+ ' / '+str(s_f_2.mont
       h)+ ' / '+
1935.                         str(s_f_2.year)+
1936.                         '\n   Semana 3 : del '+str(s_i_3.day)+' / '+str(s_i_3.month
       )+ ' / '+
1937.                         str(s_i_3.year)+' al '+str(s_f_3.day)+ ' / '+str(s_f_3.mont
       h)+ ' / '+
1938.                         str(s_f_3.year)+
1939.                         '\n   Semana 4 : del '+str(s_i_4.day)+' / '+str(s_i_4.month
       )+ ' / '+
1940.                         str(s_i_4.year)+' al '+str(s_f_4.day)+ ' / '+str(s_f_4.mont
       h)+ ' / '+
1941.                         str(s_f_4.year)
1942.                         ,title="Información")
1943.
1944.     #=============================================================================

1945.     #======================== RUTINA GENERA EXCEL ================================

1946.     #=============================================================================
```

```
1947.     df = pd.DataFrame([' '])
1948.     writer = ExcelWriter('01_PROYECCION_PROMEDIOS.xls')
1949.     df.to_excel(writer, 'Salida_Proyección_CECON', index=False)
1950.     df.to_excel(writer, 'Salida_Comparación_CECON', index=False)
1951.     df.to_excel(writer, 'Salida_ERRORES_CECON', index=False)
1952.     writer.save()
1953.
1954.     #abre el archivo de excel plantilla
1955.     rb = open_workbook('01_PROYECCION_PROMEDIOS.xls')
1956.     #crea una copia del archivo plantilla
1957.     wb = copy(rb)
1958.     #se ingresa a la hoja 1 de la copia del archivo excel
1959.     ws = wb.get_sheet(0)
1960.     ws_c = wb.get_sheet(1)
1961.     ws_e = wb.get_sheet(2)
1962.     #============================Hoja 1 Proyección ================================

1963.     #ws.write(0,0,'MES')
1964.     #ws.write(0,1,'DÍA')
1965.     #ws.write(0,2,'#')
1966.     ws.write(0,0,'METODOLOGÍA DE PROMEDIOS  (A.R.I.M.A.)')
1967.     ws.write(1,0,'PROYECCIÓN SEMANAL DE CARGA')
1968.     ws.write(2,0,'Semana de Proyección: del '+str(sem_ini.day)+'/'
1969.              +str(sem_ini.month) +'/'+str(sem_ini.year)+'\t al '+
1970.              str(sem_fin.day)+'/'+str(sem_fin.month)+'/'+str(sem_fin.year))
1971.
1972.     ws.write(3,0,'Realizado por: '+str(po_Realizado))
1973.
1974.     # Define lista de títulos
1975.     titulos = ['HORA','S/E AMBATO','S/E TOTORAS','S/E PUYO','S/E TENA','S/E BAÑOS']

1976.
1977.     for i in range(len(titulos)):
1978.         ws.write(9,i,titulos[i])
1979.
1980.     Aux = 10
1981.     Aux2 = 0
1982.     for i in range (7):
1983.         for j in range (24):
1984.             ws.write(Aux,0,j+1)
1985.             ws.write(Aux,1,ProyAmba[j + Aux2])
1986.             ws.write(Aux,2,ProyToto[j + Aux2])
1987.             ws.write(Aux,3,ProyPuyo[j + Aux2])
1988.             ws.write(Aux,4,ProyTena[j + Aux2])
1989.             ws.write(Aux,5,ProyBani[j + Aux2])
1990.             Aux = Aux + 1
1991.         Aux2 = 24 * (i+1)
1992.         Aux = Aux + 1
1993.     #============================Hoja 2 Comparación ================================

1994.
1995.     #===> Fecha
1996.     ws_c.write(0,0,'Semana de Proyección: del '+str(sem_ini.day)+'/'
1997.                +str(sem_ini.month) +'/'+str(sem_ini.year)+'\t al '+
1998.                str(sem_fin.day)+'/'+str(sem_fin.month)+'/'+str(sem_fin.year))
1999.     ws_c.write(2,0,'PROYECCIÓN SEMANAL')
2000.
2001.     for i in range(len(titulos)):
2002.         ws_c.write(9,i,titulos[i])
2003.
2004.     Aux = 10
2005.     Aux2 = 0
2006.     for i in range (7):
2007.         for j in range (24):
2008.             ws_c.write(Aux,0,j+1)
2009.             ws_c.write(Aux,1,ProyAmba[j + Aux2])
```

```
            ws_c.write(Aux,2,ProyToto[j + Aux2])
            ws_c.write(Aux,3,ProyPuyo[j + Aux2])
            ws_c.write(Aux,4,ProyTena[j + Aux2])
            ws_c.write(Aux,5,ProyBani[j + Aux2])
            Aux = Aux + 1
        Aux2 = 24 * (i+1)
        Aux = Aux + 1

    #ws_c.write(0,6,po_dia_c+'/'+po_mes_c+'/'+po_anio_c)
    ws_c.write(2,6,'SEMANA DE COMPARACIÓN')

    for i in range(len(titulos)):
        ws_c.write(9,i+6,titulos[i])

    Aux = 10
    Aux2 = 0
    for i in range (7):
        for j in range (24):
            ws_c.write(Aux,6,j+1)
            ws_c.write(Aux,7,CompAmba[j + Aux2])
            ws_c.write(Aux,8,CompToto[j + Aux2])
            ws_c.write(Aux,9,CompPuyo[j + Aux2])
            ws_c.write(Aux,10,CompTena[j + Aux2])
            ws_c.write(Aux,11,CompBani[j + Aux2])
            Aux = Aux + 1
        Aux2 = 24 * (i+1)
        Aux = Aux + 1


    # ==========================   Cálculo de errores    ==========================

    ws_c.write(2,12,'CÁLCULO DE ERRORES')
    ws_c.write(3,12,'PORCENTAJE DE ERROR MEDIO ABSOLUTO (PEMA)')

    for i in range(len(titulos)):
        ws_c.write(9,i+12,titulos[i])


    errAmba = []
    errToto = []
    errPuyo = []
    errTena = []
    errBani = []

    for i in range(len(ProyAmba)):
        errAmba.append((abs( ProyAmba[i] - CompAmba[i] ) / CompAmba[i] )*100)
        errToto.append((abs( ProyToto[i] - CompToto[i] ) / CompToto[i] )*100)
        errPuyo.append((abs( ProyPuyo[i] - CompPuyo[i] ) / CompPuyo[i] )*100)
        errTena.append((abs( ProyTena[i] - CompTena[i] ) / CompTena[i] )*100)
        errBani.append((abs( ProyBani[i] - CompBani[i] ) / CompBani[i] )*100)

    Aux = 10
    Aux2 = 0
    for i in range (7):
        for j in range (24):
            ws_c.write(Aux,12,j+1)
            ws_c.write(Aux,13,errAmba[j + Aux2])
            ws_c.write(Aux,14,errToto[j + Aux2])
            ws_c.write(Aux,15,errPuyo[j + Aux2])
            ws_c.write(Aux,16,errTena[j + Aux2])
            ws_c.write(Aux,17,errBani[j + Aux2])
            Aux = Aux + 1
        Aux2 = 24 * (i+1)
        Aux = Aux + 1
    # Suma de los valores de las listas
```

```
    SumAmba = 0
    SumToto = 0
    SumPuyo = 0
    SumTena = 0
    SumBani = 0
    for i in range (len(ProyAmba)):
        SumAmba = errAmba[i] + SumAmba
        SumToto = errToto[i] + SumToto
        SumPuyo = errPuyo[i] + SumPuyo
        SumTena = errTena[i] + SumTena
        SumBani = errBani[i] + SumBani
    # Almaceno en una lista los resultados de las sumas
    Sumas = [SumAmba, SumToto, SumPuyo, SumTena, SumBani]

    # Imprime los resultados en la fila correspondiente
    ws_c.write(Aux+1,11,'SUMATORIA TOTAL')
    for i in range (len(Sumas)):
         ws_c.write(Aux+1,i+13,Sumas[i])

    # Cálculo de los promedios de las sumas
    ws_c.write(Aux+2,11,'ERROR MEDIO ABSOLUTO')
    for i in range(len(Sumas)):
        ws_c.write(Aux+2,i+13,(Sumas[i]/len(errAmba)))

    # Cálculo de la exactitud de la proyección
    ws_c.write(Aux+3,11,'EXACTITUD DE LA PROYECCIÓN')
    for i in range(len(Sumas)):
        ws_c.write(Aux+3,i+13,(100-(Sumas[i]/len(errAmba))))


    for i in range(len(titulos)-1):
        ws_c.write(Aux,i+13,titulos[i+1])

    # ==========================   SOLO   errores    ==========================

    # ==================================> cálculo de et = Comp - Proy
    for i in range(len(titulos)-1):
        ws_e.write(9,i,titulos[i+1])
    ws_e.write(7,0,'et = Comp - Proy')

    et_Amba = []
    et_Toto = []
    et_Puyo = []
    et_Tena = []
    et_Bani = []

    for i in range (len(ProyAmba)):
        et_Amba.append(CompAmba[i] - ProyAmba[i])
        et_Toto.append(CompToto[i] - ProyToto[i])
        et_Puyo.append(CompPuyo[i] - ProyPuyo[i])
        et_Tena.append(CompTena[i] - ProyTena[i])
        et_Bani.append(CompBani[i] - ProyBani[i])

        ws_e.write(i+10,0, (et_Amba[i]))
        ws_e.write(i+10,1, (et_Toto[i]))
        ws_e.write(i+10,2, (et_Puyo[i]))
        ws_e.write(i+10,3, (et_Tena[i]))
        ws_e.write(i+10,4, (et_Bani[i]))

    # ==================================> cálculo de abs(et) = abs(Comp - Proy)
    for i in range(len(titulos)-1):
        ws_e.write(9,i+6,titulos[i+1])
    ws_e.write(7,6,'abs(et) = abs(Comp - Proy)')

    abs_et_Amba = []
    abs_et_Toto = []
```

```
    abs_et_Puyo = []
    abs_et_Tena = []
    abs_et_Bani = []

    for i in range (len(ProyAmba)):
        abs_et_Amba.append(abs(et_Amba[i]))
        abs_et_Toto.append(abs(et_Toto[i]))
        abs_et_Puyo.append(abs(et_Puyo[i]))
        abs_et_Tena.append(abs(et_Tena[i]))
        abs_et_Bani.append(abs(et_Bani[i]))

        ws_e.write(i+10,6,  (abs_et_Amba[i]))
        ws_e.write(i+10,7,  (abs_et_Toto[i]))
        ws_e.write(i+10,8,  (abs_et_Puyo[i]))
        ws_e.write(i+10,9,  (abs_et_Tena[i]))
        ws_e.write(i+10,10, (abs_et_Bani[i]))

    # ==================================> cálculo de et^2
    for i in range(len(titulos)-1):
        ws_e.write(9,i+12,titulos[i+1])
    ws_e.write(7,12,'et^2')

    et_Amba2 = []
    et_Toto2 = []
    et_Puyo2 = []
    et_Tena2 = []
    et_Bani2 = []

    for i in range (len(ProyAmba)):
        et_Amba2.append((et_Amba[i])**2)
        et_Toto2.append((et_Toto[i])**2)
        et_Puyo2.append((et_Puyo[i])**2)
        et_Tena2.append((et_Tena[i])**2)
        et_Bani2.append((et_Bani[i])**2)

        ws_e.write(i+10,12, (et_Amba2[i]))
        ws_e.write(i+10,13, (et_Toto2[i]))
        ws_e.write(i+10,14, (et_Puyo2[i]))
        ws_e.write(i+10,15, (et_Tena2[i]))
        ws_e.write(i+10,16, (et_Bani2[i]))

    # ==================================> cálculo de abs(et) / Comp
    for i in range(len(titulos)-1):
        ws_e.write(9,i+18,titulos[i+1])
    ws_e.write(7,18,'abs(et) / Comp')

    d1_Amba = []
    d1_Toto = []
    d1_Puyo = []
    d1_Tena = []
    d1_Bani = []

    for i in range (len(ProyAmba)):
        d1_Amba.append(abs_et_Amba[i] / CompAmba[i])
        d1_Toto.append(abs_et_Toto[i] / CompToto[i])
        d1_Puyo.append(abs_et_Puyo[i] / CompPuyo[i])
        d1_Tena.append(abs_et_Tena[i] / CompTena[i])
        d1_Bani.append(abs_et_Bani[i] / CompBani[i])

        ws_e.write(i+10,18, (d1_Amba[i]))
        ws_e.write(i+10,19, (d1_Toto[i]))
        ws_e.write(i+10,20, (d1_Puyo[i]))
        ws_e.write(i+10,21, (d1_Tena[i]))
        ws_e.write(i+10,22, (d1_Bani[i]))

    # ==================================> cálculo de et / Comp
```

```
    for i in range(len(titulos)-1):
        ws_e.write(9,i+24,titulos[i+1])
    ws_e.write(7,24,'et / Comp')

    d2_Amba = []
    d2_Toto = []
    d2_Puyo = []
    d2_Tena = []
    d2_Bani = []

    for i in range (len(ProyAmba)):
        d2_Amba.append(et_Amba[i] / CompAmba[i])
        d2_Toto.append(et_Toto[i] / CompToto[i])
        d2_Puyo.append(et_Puyo[i] / CompPuyo[i])
        d2_Tena.append(et_Tena[i] / CompTena[i])
        d2_Bani.append(et_Bani[i] / CompBani[i])

        ws_e.write(i+10,24, (d2_Amba[i]))
        ws_e.write(i+10,25, (d2_Toto[i]))
        ws_e.write(i+10,26, (d2_Puyo[i]))
        ws_e.write(i+10,27, (d2_Tena[i]))
        ws_e.write(i+10,28, (d2_Bani[i]))

    ws_e.write(0,0, 'INDICADORES')
    # ==================================> Cálculo DAM
    DAM_Amba = 0
    DAM_Toto = 0
    DAM_Puyo = 0
    DAM_Tena = 0
    DAM_Bani = 0

    for i in range (len(ProyAmba)):
        DAM_Amba = abs_et_Amba[i] + DAM_Amba
        DAM_Toto = abs_et_Toto[i] + DAM_Toto
        DAM_Puyo = abs_et_Puyo[i] + DAM_Puyo
        DAM_Tena = abs_et_Tena[i] + DAM_Tena
        DAM_Bani = abs_et_Bani[i] + DAM_Bani

    DAM_Amba = DAM_Amba / (len(ProyAmba))
    DAM_Toto = DAM_Toto / (len(ProyAmba))
    DAM_Puyo = DAM_Puyo / (len(ProyAmba))
    DAM_Tena = DAM_Tena / (len(ProyAmba))
    DAM_Bani = DAM_Bani / (len(ProyAmba))

    ws_e.write(1,0, ('DAM'))
    DAM = [DAM_Amba,DAM_Toto,DAM_Puyo,DAM_Tena,DAM_Bani]
    for i in range(len(DAM)):
        ws_e.write(1,i+1,DAM[i])


    # ==================================> Cálculo EMC
    EMC_Amba = 0
    EMC_Toto = 0
    EMC_Puyo = 0
    EMC_Tena = 0
    EMC_Bani = 0

    for i in range (len(ProyAmba)):
        EMC_Amba = et_Amba2[i] + EMC_Amba
        EMC_Toto = et_Toto2[i] + EMC_Toto
        EMC_Puyo = et_Puyo2[i] + EMC_Puyo
        EMC_Tena = et_Tena2[i] + EMC_Tena
        EMC_Bani = et_Bani2[i] + EMC_Bani

    EMC_Amba = EMC_Amba / (len(ProyAmba))
    EMC_Toto = EMC_Toto / (len(ProyAmba))
```

```
    EMC_Puyo = EMC_Puyo / (len(ProyAmba))
    EMC_Tena = EMC_Tena / (len(ProyAmba))
    EMC_Bani = EMC_Bani / (len(ProyAmba))

    ws_e.write(2,0, ('EMC'))
    EMC = [EMC_Amba,EMC_Toto,EMC_Puyo,EMC_Tena,EMC_Bani]

    for i in range(len(EMC)):
        ws_e.write(2,i+1,EMC[i])

    # ==================================> Cálculo PEMA
    PEMA_Amba = 0
    PEMA_Toto = 0
    PEMA_Puyo = 0
    PEMA_Tena = 0
    PEMA_Bani = 0

    for i in range (len(ProyAmba)):
        PEMA_Amba = (abs_et_Amba[i] / CompAmba [i]) + PEMA_Amba
        PEMA_Toto = (abs_et_Toto[i] / CompToto [i]) + PEMA_Toto
        PEMA_Puyo = (abs_et_Puyo[i] / CompPuyo [i]) + PEMA_Puyo
        PEMA_Tena = (abs_et_Tena[i] / CompTena [i]) + PEMA_Tena
        PEMA_Bani = (abs_et_Bani[i] / CompBani [i]) + PEMA_Bani

    PEMA_Amba = (PEMA_Amba / (len(ProyAmba))) *100
    PEMA_Toto = (PEMA_Toto / (len(ProyAmba))) *100
    PEMA_Puyo = (PEMA_Puyo / (len(ProyAmba))) *100
    PEMA_Tena = (PEMA_Tena / (len(ProyAmba))) *100
    PEMA_Bani = (PEMA_Bani / (len(ProyAmba))) *100

    ws_e.write(3,0, ('PEMA'))
    PEMA = [PEMA_Amba,PEMA_Toto,PEMA_Puyo,PEMA_Tena,PEMA_Bani]

    for i in range(len(PEMA)):
        ws_e.write(3,i+1,PEMA[i])

    # ==================================> Cálculo PME
    PME_Amba = 0
    PME_Toto = 0
    PME_Puyo = 0
    PME_Tena = 0
    PME_Bani = 0

    for i in range (len(ProyAmba)):
        PME_Amba = (et_Amba[i] / CompAmba [i]) + PME_Amba
        PME_Toto = (et_Toto[i] / CompToto [i]) + PME_Toto
        PME_Puyo = (et_Puyo[i] / CompPuyo [i]) + PME_Puyo
        PME_Tena = (et_Tena[i] / CompTena [i]) + PME_Tena
        PME_Bani = (et_Bani[i] / CompBani [i]) + PME_Bani

    PME_Amba = (PME_Amba / (len(ProyAmba))) *100
    PME_Toto = (PME_Toto / (len(ProyAmba))) *100
    PME_Puyo = (PME_Puyo / (len(ProyAmba))) *100
    PME_Tena = (PME_Tena / (len(ProyAmba))) *100
    PME_Bani = (PME_Bani / (len(ProyAmba))) *100

    ws_e.write(4,0, ('PME'))
    PME = [PME_Amba,PME_Toto,PME_Puyo,PME_Tena,PME_Bani]

    for i in range(len(PME)):
        ws_e.write(4,i+1,PME[i])

    # ==================================> Cálculo EXACTITUD DE LA PROYECCION
    ws_e.write(5,0,'EP')
    for i in range(len(PEMA)):
        ws_e.write(5,i+1,(100-PEMA[i]))
```

```
2339.
2340.
2341.     for i in range(len(titulos)-1):
2342.         ws_e.write(0,i+1,titulos[i+1])
2343.
2344.     wb.save('01_PROYECCION_PROMEDIOS.xls')
2345.
2346.     #=============================================================================

2347.     #================================ GRÁFICAS ===================================

2348.     #=============================================================================

2349.
2350.     if po_Comparacion == 'SI':
2351.
2352.     #===> Creamos una lista tipo entero para relacionar con las etiquetas
2353.         can_datos = []
2354.         for i in range(7*24):
2355.             if i%2!=1:
2356.                 can_datos.append(i)
2357. #         print(len(can_datos))
2358.
2359.     #===> Creamos una lista con los números de horas del día seleccionado (etiqueta
       s)
2360.
2361.         horas_dia = []
2362.         horas_str = []
2363.         for i in range (7):
2364.             for i in range (1,25):
2365.                 if i%2!=0:
2366.                     horas_dia.append(i)
2367.         for i in range (len(horas_dia)):
2368.             horas_str.append(str(horas_dia[i]))
2369.
2370.     #===> Tamaño de la ventana de la gráfica
2371.         plt.subplots(figsize=(15, 8))
2372.
2373.     #===> Título general superior
2374.         plt.suptitle(u' PROYECCIÓN SEMANAL DE CARGA\nMETODOLOGÍA DE PROMEDIOS ',fon
       tsize=14, fontweight='bold')
2375.
2376.         plt.subplot(5,1,1)
2377.         plt.plot(CompAmba,'blue', label = 'Comparación')
2378.         plt.plot(ProyAmba,'#DCD037', label = 'Proyección')
2379.         plt.legend(loc='upper left')
2380.         plt.xlabel(u'TIEMPO [ días ]')
2381.         plt.ylabel(u'S/E AMBATO\n\nCARGA  [ kW ]')
2382.         plt.grid(color='#C8C8C8', linestyle='-', linewidth=0.5)
2383.         plt.xticks(can_datos, horas_str, size = 5, color = 'b', rotation = 90)
2384.
2385.         plt.subplot(5,1,2)
2386.         plt.plot(CompToto,'blue', label = 'Comparación')
2387.         plt.plot(ProyToto,'#CD336F', label = 'Proyección')
2388.         plt.legend(loc='upper left')
2389.         plt.xlabel(u'TIEMPO [ días ]')
2390.         plt.ylabel(u'S/E TOTORAS\n\nCARGA  [ kW ]')
2391.         plt.grid(color='#C8C8C8', linestyle='-', linewidth=0.5)
2392.         plt.xticks(can_datos, horas_str, size = 5, color = 'b', rotation = 90)
2393.
2394.         plt.subplot(5,1,3)
2395.         plt.plot(CompPuyo,'blue', label = 'Comparación')
2396.         plt.plot(ProyPuyo,'#349A9D', label = 'Proyección')
2397.         plt.legend(loc='upper left')
2398.         plt.xlabel(u'TIEMPO [ días ]')
2399.         plt.ylabel(u'S/E PUYO\n\nCARGA  [ kW ]')
```

```
2400.         plt.grid(color='#C8C8C8', linestyle='-', linewidth=0.5)
2401.         plt.xticks(can_datos, horas_str, size = 5, color = 'b', rotation = 90)
2402.
2403.         plt.subplot(5,1,4)
2404.         plt.plot(CompTena,'blue', label = 'Comparación')
2405.         plt.plot(ProyTena,'#CC8634', label = 'Proyección')
2406.         plt.legend(loc='upper left')
2407.         plt.xlabel(u'TIEMPO [ días ]')
2408.         plt.ylabel(u'S/E TENA\n\nCARGA  [ kW ]')
2409.         plt.grid(color='#C8C8C8', linestyle='-', linewidth=0.5)
2410.         plt.xticks(can_datos, horas_str, size = 5, color = 'b', rotation = 90)
2411.
2412.         plt.subplot(5,1,5)
2413.         plt.plot(CompBani,'blue', label = 'Comparación')
2414.         plt.plot(ProyBani,'#4ACB71', label = 'Proyección')
2415.         plt.legend(loc='upper left')
2416.         plt.xlabel(u'TIEMPO [ días ]')
2417.         plt.ylabel(u'S/E BAÑOS\n\nCARGA  [ kW ]')
2418.         plt.grid(color='#C8C8C8', linestyle='-', linewidth=0.5)
2419.         plt.xticks(can_datos, horas_str, size = 5, color = 'b', rotation = 90)
2420.
2421.
2422.
2423.
2424.
2425.     #=============================================================================
2426.     #=============================================================================
2427.
2428.     print(' ')
2429.     print(' ')
2430.     print('*** Archivo Excel de Proyección semanal ha sido generado')
2431.     print(' ')
2432.     print('*** Completado !...')
2433.
2434. start_time = time()
2435. test()
2436. elapsed_time = time() - start_time
2437. print(' ')
2438. print(' ')
2439. print('////////////////////////////////////////')
2440. print(' ')
2441. print('ALGORITMO UTILIZADO: ARIMA')
2442. print("Tiempo transcurrido: %.10f segundos." % elapsed_time)
2443. print(' ')
2444. print('////////////////////////////////////////')
```

6. MÓDULO DE PROYECCIÓN CON ALGORITMO LR

```
2445.
2446. """
2447. ESCUELA POLITÉCNICA NACIONAL
2448. AUTOR: ALEXIS GUAMAN FIGUEROA
2449. 6_MÓDULO DE PROYECCIÓN CON ALGORITMO LR
2450. """
2451. from time import time
2452. def test():
2453.     #========================= MODELO DE REGRESIÓN LINEAL =========================
2454.
2455.     #===> LA MUESTRA USARÁ 5952 DATOS HISTÓRICOS
2456.     #===> La variable forecast_out controla el tiempo de predicción a realizar
2457.
2458.     import pandas as pd
```

```
2459.     #import csv, math , datetime
2460.     #import time
2461.     import numpy as np
2462.     from sklearn import preprocessing, cross_validation, svm
2463.     from sklearn.linear_model import LinearRegression, LogisticRegression
2464.     import math
2465.     import matplotlib.pyplot as plt
2466.     from matplotlib import style
2467.     style.use('ggplot')
2468.
2469.
2470.     from pandas import ExcelWriter
2471.
2472.     from xlrd import open_workbook
2473.     from xlutils.copy import copy
2474.
2475.     from datetime import date
2476.     from datetime import datetime,timedelta
2477.
2478.     #============================== VAR GLOBALES ================================
2479.     global pos_Metodologia, pos_Realizado, pos_Comparacion
2480.     global pos_anio_p, pos_mes_p, pos_dia_p
2481.     global pos_anio_c, pos_mes_c, pos_dia_c
2482.     #*******************************************************************************
2483.     global po_Metodologia, po_Realizado, po_Comparacion
2484.     global po_anio_p, po_mes_p, po_dia_p
2485.     global po_anio_c, po_mes_c, po_dia_c
2486.     #===============================================================================
2487.
2488.     doc_Proy       = 'HIS_POT_' + po_anio_p + '.xls'
2489.     doc_Proy_extra = 'HIS_POT_' + str(int(po_anio_p)-1) + '.xls'
2490.     doc_Comp       = 'HIS_POT_' + po_anio_c + '.xls'
2491.
2492.
2493.     # Genera el DataFrame a partir del archivo de datos
2494.     df_Proy       = pd.read_excel(doc_Proy      , sheetname='Hoja1',  header=None)
2495.     df_Proy_extra = pd.read_excel(doc_Proy_extra, sheetname='Hoja1',  header=None)
2496.     df_Comp       = pd.read_excel(doc_Comp      , sheetname='Hoja1',  header=None)
2497.
2498.     ## Se escoge la columna que se desea realizar la prediccion
2499.     #forcast_col = 4
2500.     ## Llena con -99999 las celdas o espacios vacíos del DF
2501.     #df_Proy.fillna(-99999, inplace=True)
2502.
2503.     # Numero total de datos a proyectar, se toma 24hrs x 7 días
2504.     forecast_out = 24*7
2505.
2506.     # Lista de datos de históricos de proyección
2507.     amba_1 = (df_Proy_extra.iloc[:,4].values.tolist())
2508.     toto_1 = (df_Proy_extra.iloc[:,5].values.tolist())
2509.     puyo_1 = (df_Proy_extra.iloc[:,6].values.tolist())
2510.     tena_1 = (df_Proy_extra.iloc[:,7].values.tolist())
2511.     bani_1 = (df_Proy_extra.iloc[:,8].values.tolist())
2512.
2513.     amba_2 = (df_Proy.iloc[:,4].values.tolist())
2514.     toto_2 = (df_Proy.iloc[:,5].values.tolist())
2515.     puyo_2 = (df_Proy.iloc[:,6].values.tolist())
2516.     tena_2 = (df_Proy.iloc[:,7].values.tolist())
2517.     bani_2 = (df_Proy.iloc[:,8].values.tolist())
2518.
```

```
    # Elimino la primera fila de cada DF (TEXTOS TÍTULOS)
    amba_1.pop(0)
    toto_1.pop(0)
    puyo_1.pop(0)
    tena_1.pop(0)
    bani_1.pop(0)

    amba_2.pop(0)
    toto_2.pop(0)
    puyo_2.pop(0)
    tena_2.pop(0)
    bani_2.pop(0)

    amba_p = amba_1 + amba_2
    toto_p = toto_1 + toto_2
    puyo_p = puyo_1 + puyo_2
    tena_p = tena_1 + tena_2
    bani_p = bani_1 + bani_2

    amba_p = pd.DataFrame(amba_p)
    toto_p = pd.DataFrame(toto_p)
    puyo_p = pd.DataFrame(puyo_p)
    tena_p = pd.DataFrame(tena_p)
    bani_p = pd.DataFrame(bani_p)

    #=============================================================================

    #=============================================================================

    #===>UBICACIONES GENERALES de las posiciones para la proyección

    #===> Variables de entrada, ejemplo
    #po_anio_p = '2018'
    #pos_mes_p = 0       # Marzo
    #po_dia_p  = '1'

    #===> Variables internas
    # FECHA SELECCIONADA
    fehca_selec = datetime(int(po_anio_p),(pos_mes_p+1),int(po_dia_p))
    # DÍAS DE PROYECCIÓN
    dias_proy = timedelta(days = 5952/24 + 1)   # 5952 * 24 = 248 días mas un día de salto
    #dias_proy = timedelta(days = 10000/24 + 1)   # 5952 * 24 = 248 días mas un día de salto

    # FECHA DE INICIO
    fecha_inicio = (fehca_selec + timedelta(days = 7)) - dias_proy
    # FECHA DE FIN
    fecha_fin = fehca_selec
    #fecha_fin = fehca_selec + timedelta(days = 7) # 24 horas de un día
    # FECHAS DE INICIO DE CADA AÑO POSICIÓN INICIAL
    enero_1 = datetime(fecha_inicio.year,1,1)
    #enero_2 = datetime(fecha_fin.year   ,1,1)

    ## POSICIÓN INICIAL
    val_ini = (abs(fecha_inicio - enero_1).days) * 24 # Posición del valor inicial 5952 DATOS
    val_fin = (abs(fecha_fin    - enero_1).days) * 24 # Posición del valor final 5952 + 24 = 5976 DATOS

    #print(val_ini)
    #print(val_fin)

    # ELIMINA LOS ÚLTIMOS VALORES DEL DF
    amba_p = amba_p[:val_fin]
    toto_p = toto_p[:val_fin]
```

```
2579.     puyo_p = puyo_p[:val_fin]
2580.     tena_p = tena_p[:val_fin]
2581.     bani_p = bani_p[:val_fin]
2582.     # ELIMINA LOS PRIMEROS VALORES DEL DF ** Queda un DF con 5952 datos
2583.     amba_p = amba_p[val_ini:]
2584.     toto_p = toto_p[val_ini:]
2585.     puyo_p = puyo_p[val_ini:]
2586.     tena_p = tena_p[val_ini:]
2587.     bani_p = bani_p[val_ini:]
2588.     # REINICIA EL INDEX DE CADA DF
2589.     amba_p = amba_p.reset_index(drop=True)
2590.     toto_p = toto_p.reset_index(drop=True)
2591.     puyo_p = puyo_p.reset_index(drop=True)
2592.     tena_p = tena_p.reset_index(drop=True)
2593.     bani_p = bani_p.reset_index(drop=True)
2594.
2595.     #=============================================================================

2596.     #=============================================================================

2597.
2598.     # Crea una nueva columna con la regresion apuntada del forecast_out
2599.     # La nueva columna es una copia y desplazo de valores de la columna volume
2600.     # La copia y desplaza los valores desde la posicion forecast_out.
2601.     amba_p['Prediccion'] = amba_p[0].shift(-forecast_out)
2602.     toto_p['Prediccion'] = toto_p[0].shift(-forecast_out)
2603.     puyo_p['Prediccion'] = puyo_p[0].shift(-forecast_out)
2604.     tena_p['Prediccion'] = tena_p[0].shift(-forecast_out)
2605.     bani_p['Prediccion'] = bani_p[0].shift(-forecast_out)
2606.
2607.     #Se crea un arreglo a partir del DF eliminando la columna de Prediccion
2608.     X_amba = np.array(amba_p.drop(['Prediccion'],1))
2609.     X_toto = np.array(toto_p.drop(['Prediccion'],1))
2610.     X_puyo = np.array(puyo_p.drop(['Prediccion'],1))
2611.     X_tena = np.array(tena_p.drop(['Prediccion'],1))
2612.     X_bani = np.array(bani_p.drop(['Prediccion'],1))
2613.
2614.     # Preprocesa y escala los datos del areglo X
2615.     X_amba = preprocessing.scale(X_amba)
2616.     X_toto = preprocessing.scale(X_toto)
2617.     X_puyo = preprocessing.scale(X_puyo)
2618.     X_tena = preprocessing.scale(X_tena)
2619.     X_bani = preprocessing.scale(X_bani)
2620.
2621.     # Crea nuevo arreglo tomando los (forecast_out = 7*24= 168) Ultimos datos de X

2622.     X_lately_amba = X_amba[-forecast_out:]
2623.     X_lately_toto = X_toto[-forecast_out:]
2624.     X_lately_puyo = X_puyo[-forecast_out:]
2625.     X_lately_tena = X_tena[-forecast_out:]
2626.     X_lately_bani = X_bani[-forecast_out:]
2627.
2628.     #El nuevo arreglo contiende todos los datos, incluso el valor de (forecast_out)

2629.     #Se queda con los primeros datos excepto los 168 últimos
2630.     X_amba = X_amba[:-forecast_out:]
2631.     X_toto = X_toto[:-forecast_out:]
2632.     X_puyo = X_puyo[:-forecast_out:]
2633.     X_tena = X_tena[:-forecast_out:]
2634.     X_bani = X_bani[:-forecast_out:]
2635.
2636.     # Elimina todas las filas que tienen datos vaci-
      os NAN  o sea los (forecast_out) ultimos datos
2637.     amba_p.dropna(inplace=True)
2638.     toto_p.dropna(inplace=True)
2639.     puyo_p.dropna(inplace=True)
```

```
    tena_p.dropna(inplace=True)
    bani_p.dropna(inplace=True)


    # Se crea nuevo arreglo que contiene la columna de Prediccion
    y_amba = np.array(amba_p['Prediccion'])
    y_toto = np.array(toto_p['Prediccion'])
    y_puyo = np.array(puyo_p['Prediccion'])
    y_tena = np.array(tena_p['Prediccion'])
    y_bani = np.array(bani_p['Prediccion'])

    #================================================================================

    #=======================   INICIO ALGORITMO RL    ==============================

    #================================================================================


    clf = LinearRegression()
    # Metodologia de validacion cruzada para los datos obtenidos
    X_train_amba, X_test_amba, y_train_amba, y_test_amba = cross_validation.train_test_split(X_amba, y_amba, test_size=0.3)
    clf.fit(X_train_amba,y_train_amba)
    accuracy_amba = clf.score(X_test_amba,y_test_amba)
    ProyAmba = clf.predict(X_lately_amba)# ============>>>   SALIDA PROYECCIÓN forecast_set

    X_train_toto, X_test_toto, y_train_toto, y_test_toto = cross_validation.train_test_split(X_toto, y_toto, test_size=0.3)
    clf.fit(X_train_toto,y_train_toto)
    accuracy_toto = clf.score(X_test_toto,y_test_toto)
    ProyToto = clf.predict(X_lately_toto)# ============>>>   SALIDA PROYECCIÓN forecast_set

    X_train_puyo, X_test_puyo, y_train_puyo, y_test_puyo = cross_validation.train_test_split(X_puyo, y_puyo, test_size=0.3)
    clf.fit(X_train_puyo,y_train_puyo)
    accuracy_puyo = clf.score(X_test_puyo,y_test_puyo)
    ProyPuyo = clf.predict(X_lately_puyo)# ============>>>   SALIDA PROYECCIÓN forecast_set

    X_train_tena, X_test_tena, y_train_tena, y_test_tena = cross_validation.train_test_split(X_tena, y_tena, test_size=0.3)
    clf.fit(X_train_tena,y_train_tena)
    accuracy_tena = clf.score(X_test_tena,y_test_tena)
    ProyTena = clf.predict(X_lately_tena)# ============>>>   SALIDA PROYECCIÓN forecast_set

    X_train_bani, X_test_bani, y_train_bani, y_test_bani = cross_validation.train_test_split(X_bani, y_bani, test_size=0.3)
    clf.fit(X_train_bani,y_train_bani)
    accuracy_bani = clf.score(X_test_bani,y_test_bani)
    ProyBani = clf.predict(X_lately_bani)# ============>>>   SALIDA PROYECCIÓN forecast_set

    # Datos de comparación
    #CompAmba = (amba_p[0][-forecast_out:]).reset_index(drop=True)
    #CompToto = (toto_p[0][-forecast_out:]).reset_index(drop=True)
    #CompPuyo = (puyo_p[0][-forecast_out:]).reset_index(drop=True)
    #CompTena = (tena_p[0][-forecast_out:]).reset_index(drop=True)
    #CompBani = (bani_p[0][-forecast_out:]).reset_index(drop=True)
    #================================================================================

    #========================== SEMANA DE COMPARACIÓN ==============================
```

```
    #=============================================================================

    mes_c = df_Comp.iloc[:,0].values.tolist()
    mes_c.pop(0)

    dia_c = df_Comp.iloc[:,1].values.tolist()
    dia_c.pop(0)

    hora_c = df_Comp.iloc[:,2].values.tolist()
    hora_c.pop(0)

    amba_c = df_Comp.iloc[:,4].values.tolist()
    amba_c.pop(0)

    toto_c = df_Comp.iloc[:,5].values.tolist()
    toto_c.pop(0)

    puyo_c = df_Comp.iloc[:,6].values.tolist()
    puyo_c.pop(0)

    tena_c = df_Comp.iloc[:,7].values.tolist()
    tena_c.pop(0)

    bani_c = df_Comp.iloc[:,8].values.tolist()
    bani_c.pop(0)

    #===> Reemplaza valores (0) con (nan)
    for i in range(len(dia_c)):
        if amba_c[i] == 0:
            amba_c[i] = float('nan')
        if toto_c[i] == 0:
            toto_c[i] = float('nan')
        if puyo_c[i] == 0:
            puyo_c[i] = float('nan')
        if tena_c[i] == 0:
            tena_c[i] = float('nan')
        if bani_c[i] == 0:
            bani_c[i] = float('nan')

    #===> Se establece una matriz con los datos importados
    #data_c = np.column_stack((amba_c, toto_c, puyo_c, tena_c, bani_c))

    #=============================================================================

    #=============================================================================

    if po_mes_c == 'ENERO':
        if int(po_dia_c) < 8:

            doc_Proy_EXTRA_c = 'HIS_POT_' + str(int(po_anio_c)-1) + '.xls'
            df_Proy_EXTRA_c = pd.read_excel(doc_Proy_EXTRA_c, sheetname='Hoja1',  h
eader=None)
            df_Proy_EXTRA_c = df_Proy_EXTRA_c[-(24*31):]

            mess_extra_c = df_Proy_EXTRA_c.iloc[:,0].values.tolist()
            amba_extra_c = df_Proy_EXTRA_c.iloc[:,4].values.tolist()
            toto_extra_c = df_Proy_EXTRA_c.iloc[:,5].values.tolist()
            puyo_extra_c = df_Proy_EXTRA_c.iloc[:,6].values.tolist()
            tena_extra_c = df_Proy_EXTRA_c.iloc[:,7].values.tolist()
            bani_extra_c = df_Proy_EXTRA_c.iloc[:,8].values.tolist()
            #===> Obtengo únicamente 7 días de diciembre del año pasado
            mess_extra_c = mess_extra_c[-(24*31):]
            #===> Elimino el mes de diciembre del año actual
            mes_c = mes_c[:-(24*31)]
```

```
            #===> Nueva lista con 7 días de Dic del año pasado mas resto de datos a
ño actual
            mes_c = mess_extra_c + mes_c

            amba_c = amba_extra_c + amba_c
            toto_c = toto_extra_c + toto_c
            puyo_c = puyo_extra_c + puyo_c
            tena_c = tena_extra_c + tena_c
            bani_c = bani_extra_c + bani_c

    #===> APUNTA AL MES SELECCIONADO DE COMPARACIÓN
    x_meses_c = [mes_c.index('ENERO'),   mes_c.index('FEBRERO'),   mes_c.index('MAR
ZO'),
                 mes_c.index('ABRIL'),   mes_c.index('MAYO'),      mes_c.index('JUN
IO'),
                 mes_c.index('JULIO'),   mes_c.index('AGOSTO'),    mes_c.index('SEP
TIEMBRE'),
                 mes_c.index('OCTUBRE'), mes_c.index('NOVIEMBRE'), mes_c.index('DIC
IEMBRE')]

    for i in range (12):
        if pos_mes_c == i:
            if pos_mes_c == 11:
                ubic_mes_c = x_meses_c[i]
            else:
                ubic_mes_c = x_meses_c[i] # VARIABLE Posición del mes seleccionado


    #===> DEFINE LA UBICACIÓN DEL DÍA SELECCIONADO DE COMPARACIÓN
    print(' ')
    print ('Fecha Comparación: ',po_dia_c,' de ',mes_c[ubic_mes_c],' de ',po_anio_c
)
    if int(po_dia_c) == 1:
        ubicacion_c = ubic_mes_c
    else:
        ubicacion_c = (int(po_dia_c)-1) * 24 + ubic_mes_c

#    print(' ')
#    print (ubicacion_c) #=========================================================
=>>> VARIABLE DE UBICACIÓN EXACTA EN LISTA DE DATOS de comparación

    amba_c = amba_c[:ubicacion_c]
    toto_c = toto_c[:ubicacion_c]
    puyo_c = puyo_c[:ubicacion_c]
    tena_c = tena_c[:ubicacion_c]
    bani_c = bani_c[:ubicacion_c]

    #===> Valores inicio y fin para lista de datos a ser usada
    val_ini_c = ubicacion_c - 24 * 7 #Posición del valor inicial
    val_fin_c = ubicacion_c - 1      #Posición del valor final

    #===> Comparación semananal S/E Ambato
    CompAmba = []
    for i in range (24*7):
        CompAmba.append(amba_c[val_ini_c + i ])
    #===> Comparación semananal S/E Totoras
    CompToto = []
    for i in range (24*7):
        CompToto.append(toto_c[val_ini_c + i ])
    #===> Comparación semananal S/E Puyo
    CompPuyo = []
    for i in range (24*7):
        CompPuyo.append(puyo_c[val_ini_c + i  ])
    #===> Comparación semananal S/E Tena
    CompTena = []
```

```
2809.     for i in range (24*7):
2810.         CompTena.append(tena_c[val_ini_c + i ])
2811.     #===> Comparación semananal S/E Baños
2812.     CompBani = []
2813.     for i in range (24*7):
2814.         CompBani.append(bani_c[val_ini_c + i ])
2815.
2816.
2817.     #=============================================================================

2818.     #================================ GRÁFICAS ===================================

2819.     #=============================================================================

2820.     if po_Comparacion == 'SI':
2821.     #===> Creamos una lista tipo entero para relacionar con las etiquetas
2822.         can_datos = []
2823.         for i in range(7*24):
2824.             if i%2!=1:
2825.                 can_datos.append(i)
2826.     #===> Creamos una lista con los números de horas del día seleccionado (etiqueta
       s)
2827.         horas_dia = []
2828.         horas_str = []
2829.         for i in range (7):
2830.             for i in range (1,25):
2831.                 if i%2!=0:
2832.                     horas_dia.append(i)
2833.         for i in range (len(horas_dia)):
2834.             horas_str.append(str(horas_dia[i]))
2835.
2836.     #===> Tamaño de la ventana de la gráfica
2837.         plt.subplots(figsize=(15, 8))
2838.
2839.     #===> Título general superior
2840.         plt.suptitle(u' PROYECCIÓN SEMANAL DE CARGA\n REGRESIÓN LINEAL ',fontsize=1
       4, fontweight='bold')
2841.
2842.         plt.subplot(5,1,1)
2843.         plt.plot(CompAmba,'blue', label = 'Comparación')
2844.         plt.plot(ProyAmba,'#DCD037', label = 'Proyección')
2845.         plt.legend(loc='upper left')
2846.         plt.xlabel(u'TIEMPO [ días ]')
2847.         plt.ylabel(u'S/E AMBATO\n\nCARGA  [ kW ]')
2848.         plt.grid(color='#C8C8C8', linestyle='-', linewidth=0.5)
2849.         plt.xticks(can_datos, horas_str, size = 5, color = 'b', rotation = 90)
2850.
2851.         plt.subplot(5,1,2)
2852.         plt.plot(CompToto,'blue', label = 'Comparación')
2853.         plt.plot(ProyToto,'#CD336F', label = 'Proyección')
2854.         plt.legend(loc='upper left')
2855.         plt.xlabel(u'TIEMPO [ días ]')
2856.         plt.ylabel(u'S/E TOTORAS\n\nCARGA  [ kW ]')
2857.         plt.grid(color='#C8C8C8', linestyle='-', linewidth=0.5)
2858.         plt.xticks(can_datos, horas_str, size = 5, color = 'b', rotation = 90)
2859.
2860.         plt.subplot(5,1,3)
2861.         plt.plot(CompPuyo,'blue', label = 'Comparación')
2862.         plt.plot(ProyPuyo,'#349A9D', label = 'Proyección')
2863.         plt.legend(loc='upper left')
2864.         plt.xlabel(u'TIEMPO [ días ]')
2865.         plt.ylabel(u'S/E PUYO\n\nCARGA  [ kW ]')
2866.         plt.grid(color='#C8C8C8', linestyle='-', linewidth=0.5)
2867.         plt.xticks(can_datos, horas_str, size = 5, color = 'b', rotation = 90)
2868.
2869.         plt.subplot(5,1,4)
```

```
2870.         plt.plot(CompTena,'blue', label = 'Comparación')
2871.         plt.plot(ProyTena,'#CC8634', label = 'Proyección')
2872.         plt.legend(loc='upper left')
2873.         plt.xlabel(u'TIEMPO [ días ]')
2874.         plt.ylabel(u'S/E TENA\n\nCARGA  [ kW ]')
2875.         plt.grid(color='#C8C8C8', linestyle='-', linewidth=0.5)
2876.         plt.xticks(can_datos, horas_str, size = 5, color = 'b', rotation = 90)
2877.
2878.         plt.subplot(5,1,5)
2879.         plt.plot(CompBani,'blue', label = 'Comparación')
2880.         plt.plot(ProyBani,'#4ACB71', label = 'Proyección')
2881.         plt.legend(loc='upper left')
2882.         plt.xlabel(u'TIEMPO [ días ]')
2883.         plt.ylabel(u'S/E BAÑOS\n\nCARGA  [ kW ]')
2884.         plt.grid(color='#C8C8C8', linestyle='-', linewidth=0.5)
2885.         plt.xticks(can_datos, horas_str, size = 5, color = 'b', rotation = 90)
2886.
2887.     #graf = []
2888.     #for i in range (52248, 52415):
2889.     #    graf.append(xp[i])
2890.     #plt.legend(loc=1)
2891.
2892.     #==========> Fechas de salida:
2893.     sem_ini = datetime(int(po_anio_p),(pos_mes_p+1),int(po_dia_p))
2894.     sem_fin = sem_ini + timedelta(days = 6)
2895.
2896.     #================================================================================

2897.     #======================== RUTINA GENERA EXCEL ===================================

2898.     #================================================================================

2899.     df = pd.DataFrame([' '])
2900.     writer = ExcelWriter('02_PROYECCION_RL.xls')
2901.     df.to_excel(writer, 'Salida_Proyección_CECON', index=False)
2902.     df.to_excel(writer, 'Salida_Comparación_CECON', index=False)
2903.     df.to_excel(writer, 'Salida_ERRORES_CECON', index=False)
2904.     writer.save()
2905.
2906.     #abre el archivo de excel plantilla
2907.     rb = open_workbook('02_PROYECCION_RL.xls')
2908.     #crea una copia del archivo plantilla
2909.     wb = copy(rb)
2910.     #se ingresa a la hoja 1 de la copia del archivo excel
2911.     ws = wb.get_sheet(0)
2912.     ws_c = wb.get_sheet(1)
2913.     ws_e = wb.get_sheet(2)
2914.     #============================Hoja 1 Proyección ================================

2915.     #ws.write(0,0,'MES')
2916.     #ws.write(0,1,'DÍA')
2917.     #ws.write(0,2,'#')
2918.     ws.write(0,0,'METODOLOGÍA REGRESIÓN LINEAL  (L.R.)')
2919.     ws.write(1,0,'PROYECCIÓN SEMANAL DE CARGA')
2920.     ws.write(2,0,'Semana de Proyección: del '+str(sem_ini.day)+'/'
2921.              +str(sem_ini.month) +'/'+str(sem_ini.year)+'\t al '+
2922.              str(sem_fin.day)+'/'+str(sem_fin.month)+'/'+str(sem_fin.year))
2923.
2924.     ws.write(3,0,'Realizado por: '+str(po_Realizado))
2925.
2926.     # Define lista de títulos
2927.     titulos = ['HORA','S/E AMBATO','S/E TOTORAS','S/E PUYO','S/E TENA','S/E BAÑOS']

2928.
2929.     for i in range(len(titulos)):
2930.         ws.write(9,i,titulos[i])
```

```
2931.
2932.     Aux = 10
2933.     Aux2 = 0
2934.     for i in range (7):
2935.         for j in range (24):
2936.             ws.write(Aux,0,j+1)
2937.             ws.write(Aux,1,ProyAmba[j + Aux2])
2938.             ws.write(Aux,2,ProyToto[j + Aux2])
2939.             ws.write(Aux,3,ProyPuyo[j + Aux2])
2940.             ws.write(Aux,4,ProyTena[j + Aux2])
2941.             ws.write(Aux,5,ProyBani[j + Aux2])
2942.             Aux = Aux + 1
2943.         Aux2 = 24 * (i+1)
2944.         Aux = Aux + 1
2945.     #==========================Hoja 2 Comparación ================================
2946.
2947.     #===> Fecha
2948.     ws_c.write(0,0,'Semana de Proyección: del '+str(sem_ini.day)+'/'
2949.                +str(sem_ini.month) +'/'+str(sem_ini.year)+'\t al '+
2950.                str(sem_fin.day)+'/'+str(sem_fin.month)+'/'+str(sem_fin.year))
2951.     ws_c.write(2,0,'PROYECCIÓN SEMANAL')
2952.
2953.     for i in range(len(titulos)):
2954.         ws_c.write(9,i,titulos[i])
2955.
2956.     Aux = 10
2957.     Aux2 = 0
2958.     for i in range (7):
2959.         for j in range (24):
2960.             ws_c.write(Aux,0,j+1)
2961.             ws_c.write(Aux,1,ProyAmba[j + Aux2])
2962.             ws_c.write(Aux,2,ProyToto[j + Aux2])
2963.             ws_c.write(Aux,3,ProyPuyo[j + Aux2])
2964.             ws_c.write(Aux,4,ProyTena[j + Aux2])
2965.             ws_c.write(Aux,5,ProyBani[j + Aux2])
2966.             Aux = Aux + 1
2967.         Aux2 = 24 * (i+1)
2968.         Aux = Aux + 1
2969.
2970.     #ws_c.write(0,6,po_dia_c+'/'+po_mes_c+'/'+po_anio_c)
2971.     ws_c.write(2,6,'SEMANA DE COMPARACIÓN')
2972.
2973.     for i in range(len(titulos)):
2974.         ws_c.write(9,i+6,titulos[i])
2975.
2976.     Aux = 10
2977.     Aux2 = 0
2978.     for i in range (7):
2979.         for j in range (24):
2980.             ws_c.write(Aux,6,j+1)
2981.             ws_c.write(Aux,7,CompAmba[j + Aux2])
2982.             ws_c.write(Aux,8,CompToto[j + Aux2])
2983.             ws_c.write(Aux,9,CompPuyo[j + Aux2])
2984.             ws_c.write(Aux,10,CompTena[j + Aux2])
2985.             ws_c.write(Aux,11,CompBani[j + Aux2])
2986.             Aux = Aux + 1
2987.         Aux2 = 24 * (i+1)
2988.         Aux = Aux + 1
2989.
2990.
2991.     # ==========================   Cálculo de errores    ==========================
2992.
2993.     ws_c.write(2,12,'CÁLCULO DE ERRORES')
2994.     ws_c.write(3,12,'PORCENTAJE DE ERROR MEDIO ABSOLUTO (PEMA)')
```

```
2995.
2996.     for i in range(len(titulos)):
2997.         ws_c.write(9,i+12,titulos[i])
2998.
2999.
3000.     errAmba = []
3001.     errToto = []
3002.     errPuyo = []
3003.     errTena = []
3004.     errBani = []
3005.
3006.     for i in range(len(ProyAmba)):
3007.         errAmba.append((abs( ProyAmba[i] - CompAmba[i] ) / CompAmba[i] )*100)
3008.         errToto.append((abs( ProyToto[i] - CompToto[i] ) / CompToto[i] )*100)
3009.         errPuyo.append((abs( ProyPuyo[i] - CompPuyo[i] ) / CompPuyo[i] )*100)
3010.         errTena.append((abs( ProyTena[i] - CompTena[i] ) / CompTena[i] )*100)
3011.         errBani.append((abs( ProyBani[i] - CompBani[i] ) / CompBani[i] )*100)
3012.
3013.     Aux = 10
3014.     Aux2 = 0
3015.     for i in range (7):
3016.         for j in range (24):
3017.             ws_c.write(Aux,12,j+1)
3018.             ws_c.write(Aux,13,errAmba[j + Aux2])
3019.             ws_c.write(Aux,14,errToto[j + Aux2])
3020.             ws_c.write(Aux,15,errPuyo[j + Aux2])
3021.             ws_c.write(Aux,16,errTena[j + Aux2])
3022.             ws_c.write(Aux,17,errBani[j + Aux2])
3023.             Aux = Aux + 1
3024.         Aux2 = 24 * (i+1)
3025.         Aux = Aux + 1
3026.     # Suma de los valores de las listas
3027.     SumAmba = 0
3028.     SumToto = 0
3029.     SumPuyo = 0
3030.     SumTena = 0
3031.     SumBani = 0
3032.     for i in range (len(ProyAmba)):
3033.         SumAmba = errAmba[i] + SumAmba
3034.         SumToto = errToto[i] + SumToto
3035.         SumPuyo = errPuyo[i] + SumPuyo
3036.         SumTena = errTena[i] + SumTena
3037.         SumBani = errBani[i] + SumBani
3038.     # Almaceno en una lista los resultados de las sumas
3039.     Sumas = [SumAmba, SumToto, SumPuyo, SumTena, SumBani]
3040.
3041.     # Imprime los resultados en la fila correspondiente
3042.     ws_c.write(Aux+1,11,'SUMATORIA TOTAL')
3043.     for i in range (len(Sumas)):
3044.          ws_c.write(Aux+1,i+13,Sumas[i])
3045.
3046.     # Cálculo de los promedios de las sumas
3047.     ws_c.write(Aux+2,11,'ERROR MEDIO ABSOLUTO')
3048.     for i in range(len(Sumas)):
3049.         ws_c.write(Aux+2,i+13,(Sumas[i]/len(errAmba)))
3050.
3051.     # Cálculo de la exactitud de la proyección
3052.     ws_c.write(Aux+3,11,'EXACTITUD DE LA PROYECCIÓN')
3053.     for i in range(len(Sumas)):
3054.         ws_c.write(Aux+3,i+13,(100-(Sumas[i]/len(errAmba))))
3055.
3056.
3057.     for i in range(len(titulos)-1):
3058.         ws_c.write(Aux,i+13,titulos[i+1])
3059.
3060.     # ==========================   SOLO   errores    ==========================
```

```
3061.
3062.     # ==================================> cálculo de et = Comp - Proy
3063.     for i in range(len(titulos)-1):
3064.         ws_e.write(9,i,titulos[i+1])
3065.     ws_e.write(7,0,'et = Comp - Proy')
3066.
3067.     et_Amba = []
3068.     et_Toto = []
3069.     et_Puyo = []
3070.     et_Tena = []
3071.     et_Bani = []
3072.
3073.     for i in range (len(ProyAmba)):
3074.         et_Amba.append(CompAmba[i] - ProyAmba[i])
3075.         et_Toto.append(CompToto[i] - ProyToto[i])
3076.         et_Puyo.append(CompPuyo[i] - ProyPuyo[i])
3077.         et_Tena.append(CompTena[i] - ProyTena[i])
3078.         et_Bani.append(CompBani[i] - ProyBani[i])
3079.
3080.         ws_e.write(i+10,0, (et_Amba[i]))
3081.         ws_e.write(i+10,1, (et_Toto[i]))
3082.         ws_e.write(i+10,2, (et_Puyo[i]))
3083.         ws_e.write(i+10,3, (et_Tena[i]))
3084.         ws_e.write(i+10,4, (et_Bani[i]))
3085.
3086.     # ==================================> cálculo de abs(et) = abs(Comp - Proy)
3087.     for i in range(len(titulos)-1):
3088.         ws_e.write(9,i+6,titulos[i+1])
3089.     ws_e.write(7,6,'abs(et) = abs(Comp - Proy)')
3090.
3091.     abs_et_Amba = []
3092.     abs_et_Toto = []
3093.     abs_et_Puyo = []
3094.     abs_et_Tena = []
3095.     abs_et_Bani = []
3096.
3097.     for i in range (len(ProyAmba)):
3098.         abs_et_Amba.append(abs(et_Amba[i]))
3099.         abs_et_Toto.append(abs(et_Toto[i]))
3100.         abs_et_Puyo.append(abs(et_Puyo[i]))
3101.         abs_et_Tena.append(abs(et_Tena[i]))
3102.         abs_et_Bani.append(abs(et_Bani[i]))
3103.
3104.         ws_e.write(i+10,6,  (abs_et_Amba[i]))
3105.         ws_e.write(i+10,7,  (abs_et_Toto[i]))
3106.         ws_e.write(i+10,8,  (abs_et_Puyo[i]))
3107.         ws_e.write(i+10,9,  (abs_et_Tena[i]))
3108.         ws_e.write(i+10,10, (abs_et_Bani[i]))
3109.
3110.     # ==================================> cálculo de et^2
3111.     for i in range(len(titulos)-1):
3112.         ws_e.write(9,i+12,titulos[i+1])
3113.     ws_e.write(7,12,'et^2')
3114.
3115.     et_Amba2 = []
3116.     et_Toto2 = []
3117.     et_Puyo2 = []
3118.     et_Tena2 = []
3119.     et_Bani2 = []
3120.
3121.     for i in range (len(ProyAmba)):
3122.         et_Amba2.append((et_Amba[i])**2)
3123.         et_Toto2.append((et_Toto[i])**2)
3124.         et_Puyo2.append((et_Puyo[i])**2)
3125.         et_Tena2.append((et_Tena[i])**2)
3126.         et_Bani2.append((et_Bani[i])**2)
```

```
3127.
3128.             ws_e.write(i+10,12, (et_Amba2[i]))
3129.             ws_e.write(i+10,13, (et_Toto2[i]))
3130.             ws_e.write(i+10,14, (et_Puyo2[i]))
3131.             ws_e.write(i+10,15, (et_Tena2[i]))
3132.             ws_e.write(i+10,16, (et_Bani2[i]))
3133.
3134.         # ==================================> cálculo de abs(et) / Comp
3135.         for i in range(len(titulos)-1):
3136.             ws_e.write(9,i+18,titulos[i+1])
3137.         ws_e.write(7,18,'abs(et) / Comp')
3138.
3139.         d1_Amba = []
3140.         d1_Toto = []
3141.         d1_Puyo = []
3142.         d1_Tena = []
3143.         d1_Bani = []
3144.
3145.         for i in range (len(ProyAmba)):
3146.             d1_Amba.append(abs_et_Amba[i] / CompAmba[i])
3147.             d1_Toto.append(abs_et_Toto[i] / CompToto[i])
3148.             d1_Puyo.append(abs_et_Puyo[i] / CompPuyo[i])
3149.             d1_Tena.append(abs_et_Tena[i] / CompTena[i])
3150.             d1_Bani.append(abs_et_Bani[i] / CompBani[i])
3151.
3152.             ws_e.write(i+10,18, (d1_Amba[i]))
3153.             ws_e.write(i+10,19, (d1_Toto[i]))
3154.             ws_e.write(i+10,20, (d1_Puyo[i]))
3155.             ws_e.write(i+10,21, (d1_Tena[i]))
3156.             ws_e.write(i+10,22, (d1_Bani[i]))
3157.
3158.         # ==================================> cálculo de et / Comp
3159.         for i in range(len(titulos)-1):
3160.             ws_e.write(9,i+24,titulos[i+1])
3161.         ws_e.write(7,24,'et / Comp')
3162.
3163.         d2_Amba = []
3164.         d2_Toto = []
3165.         d2_Puyo = []
3166.         d2_Tena = []
3167.         d2_Bani = []
3168.
3169.         for i in range (len(ProyAmba)):
3170.             d2_Amba.append(et_Amba[i] / CompAmba[i])
3171.             d2_Toto.append(et_Toto[i] / CompToto[i])
3172.             d2_Puyo.append(et_Puyo[i] / CompPuyo[i])
3173.             d2_Tena.append(et_Tena[i] / CompTena[i])
3174.             d2_Bani.append(et_Bani[i] / CompBani[i])
3175.
3176.             ws_e.write(i+10,24, (d2_Amba[i]))
3177.             ws_e.write(i+10,25, (d2_Toto[i]))
3178.             ws_e.write(i+10,26, (d2_Puyo[i]))
3179.             ws_e.write(i+10,27, (d2_Tena[i]))
3180.             ws_e.write(i+10,28, (d2_Bani[i]))
3181.
3182.         ws_e.write(0,0, 'INDICADORES')
3183.         # ==================================> Cálculo DAM
3184.         DAM_Amba = 0
3185.         DAM_Toto = 0
3186.         DAM_Puyo = 0
3187.         DAM_Tena = 0
3188.         DAM_Bani = 0
3189.
3190.         for i in range (len(ProyAmba)):
3191.             DAM_Amba = abs_et_Amba[i] + DAM_Amba
3192.             DAM_Toto = abs_et_Toto[i] + DAM_Toto
```

```
3193.         DAM_Puyo = abs_et_Puyo[i] + DAM_Puyo
3194.         DAM_Tena = abs_et_Tena[i] + DAM_Tena
3195.         DAM_Bani = abs_et_Bani[i] + DAM_Bani
3196.
3197.     DAM_Amba = DAM_Amba / (len(ProyAmba))
3198.     DAM_Toto = DAM_Toto / (len(ProyAmba))
3199.     DAM_Puyo = DAM_Puyo / (len(ProyAmba))
3200.     DAM_Tena = DAM_Tena / (len(ProyAmba))
3201.     DAM_Bani = DAM_Bani / (len(ProyAmba))
3202.
3203.     ws_e.write(1,0, ('DAM'))
3204.     DAM = [DAM_Amba,DAM_Toto,DAM_Puyo,DAM_Tena,DAM_Bani]
3205.     for i in range(len(DAM)):
3206.         ws_e.write(1,i+1,DAM[i])
3207.
3208.
3209.     # ==================================> Cálculo EMC
3210.     EMC_Amba = 0
3211.     EMC_Toto = 0
3212.     EMC_Puyo = 0
3213.     EMC_Tena = 0
3214.     EMC_Bani = 0
3215.
3216.     for i in range (len(ProyAmba)):
3217.         EMC_Amba = et_Amba2[i] + EMC_Amba
3218.         EMC_Toto = et_Toto2[i] + EMC_Toto
3219.         EMC_Puyo = et_Puyo2[i] + EMC_Puyo
3220.         EMC_Tena = et_Tena2[i] + EMC_Tena
3221.         EMC_Bani = et_Bani2[i] + EMC_Bani
3222.
3223.     EMC_Amba = EMC_Amba / (len(ProyAmba))
3224.     EMC_Toto = EMC_Toto / (len(ProyAmba))
3225.     EMC_Puyo = EMC_Puyo / (len(ProyAmba))
3226.     EMC_Tena = EMC_Tena / (len(ProyAmba))
3227.     EMC_Bani = EMC_Bani / (len(ProyAmba))
3228.
3229.     ws_e.write(2,0, ('EMC'))
3230.     EMC = [EMC_Amba,EMC_Toto,EMC_Puyo,EMC_Tena,EMC_Bani]
3231.
3232.     for i in range(len(EMC)):
3233.         ws_e.write(2,i+1,EMC[i])
3234.
3235.     # ==================================> Cálculo PEMA
3236.     PEMA_Amba = 0
3237.     PEMA_Toto = 0
3238.     PEMA_Puyo = 0
3239.     PEMA_Tena = 0
3240.     PEMA_Bani = 0
3241.
3242.     for i in range (len(ProyAmba)):
3243.         PEMA_Amba = (abs_et_Amba[i] / CompAmba [i]) + PEMA_Amba
3244.         PEMA_Toto = (abs_et_Toto[i] / CompToto [i]) + PEMA_Toto
3245.         PEMA_Puyo = (abs_et_Puyo[i] / CompPuyo [i]) + PEMA_Puyo
3246.         PEMA_Tena = (abs_et_Tena[i] / CompTena [i]) + PEMA_Tena
3247.         PEMA_Bani = (abs_et_Bani[i] / CompBani [i]) + PEMA_Bani
3248.
3249.     PEMA_Amba = (PEMA_Amba / (len(ProyAmba))) *100
3250.     PEMA_Toto = (PEMA_Toto / (len(ProyAmba))) *100
3251.     PEMA_Puyo = (PEMA_Puyo / (len(ProyAmba))) *100
3252.     PEMA_Tena = (PEMA_Tena / (len(ProyAmba))) *100
3253.     PEMA_Bani = (PEMA_Bani / (len(ProyAmba))) *100
3254.
3255.     ws_e.write(3,0, ('PEMA'))
3256.     PEMA = [PEMA_Amba,PEMA_Toto,PEMA_Puyo,PEMA_Tena,PEMA_Bani]
3257.
3258.     for i in range(len(PEMA)):
```

```
3259.         ws_e.write(3,i+1,PEMA[i])
3260.
3261.     # ==================================> Cálculo PME
3262.     PME_Amba = 0
3263.     PME_Toto = 0
3264.     PME_Puyo = 0
3265.     PME_Tena = 0
3266.     PME_Bani = 0
3267.
3268.     for i in range (len(ProyAmba)):
3269.         PME_Amba = (et_Amba[i] / CompAmba [i]) + PME_Amba
3270.         PME_Toto = (et_Toto[i] / CompToto [i]) + PME_Toto
3271.         PME_Puyo = (et_Puyo[i] / CompPuyo [i]) + PME_Puyo
3272.         PME_Tena = (et_Tena[i] / CompTena [i]) + PME_Tena
3273.         PME_Bani = (et_Bani[i] / CompBani [i]) + PME_Bani
3274.
3275.     PME_Amba = (PME_Amba / (len(ProyAmba))) *100
3276.     PME_Toto = (PME_Toto / (len(ProyAmba))) *100
3277.     PME_Puyo = (PME_Puyo / (len(ProyAmba))) *100
3278.     PME_Tena = (PME_Tena / (len(ProyAmba))) *100
3279.     PME_Bani = (PME_Bani / (len(ProyAmba))) *100
3280.
3281.     ws_e.write(4,0, ('PME'))
3282.     PME = [PME_Amba,PME_Toto,PME_Puyo,PME_Tena,PME_Bani]
3283.
3284.     for i in range(len(PME)):
3285.         ws_e.write(4,i+1,PME[i])
3286.
3287.     # ==================================> Cálculo EXACTITUD DE LA PROYECCION
3288.     ws_e.write(5,0,'EP')
3289.     for i in range(len(PEMA)):
3290.         ws_e.write(5,i+1,(100-PEMA[i]))
3291.
3292.
3293.     for i in range(len(titulos)-1):
3294.         ws_e.write(0,i+1,titulos[i+1])
3295.
3296.     wb.save('02_PROYECCION_RL.xls')
3297.     #=============================================================================
3298.     #=============================================================================
3299.
3300.     print(' ')
3301.     print(' ')
3302.
3303.     print('* Porcentaje de Exactitud de la Regresión Lineal:')
3304.     print(' ')
3305.     print('==> S/E Ambato :',accuracy_amba*100,'%')
3306.     print('==> S/E Totoras:',accuracy_toto*100,'%')
3307.     print('==> S/E Puyo   :',accuracy_puyo*100,'%')
3308.     print('==> S/E Tena   :',accuracy_tena*100,'%')
3309.     print('==> S/E Baños  :',accuracy_bani*100,'%')
3310.     print(' ')
3311.     print(' ')
3312.     print('*** RL Completed...!!!')
3313.     print(' ')
3314.     print(' ')
3315.
3316.
3317. start_time = time()
3318. test()
3319. elapsed_time = time() - start_time
3320. print(' ')
3321. print(' ')
3322. print('////////////////////////////////////////')
```

```
print(' ')
print('ALGORITMO UTILIZADO: LR')
print("Tiempo transcurrido: %.10f segundos." % elapsed_time)
print(' ')
print('////////////////////////////////////////')
```

7. MÓDULO DE PROYECCIÓN CON ALGORITMO SVM

```

"""
Created on Fri Apr  5 11:17:08 2019
ESCUELA POLITÉCNICA NACIONAL
AUTOR: ALEXIS GUAMAN FIGUEROA
7_MÓDULO DE PROYECCIÓN CON ALGORITMO SVM
"""
from time import time
def test():

    #========================= MODELO DE REGRESIÓN LINEAL =========================

    #===> LA MUESTRA USARÁ 5952 DATOS HISTÓRICOS
    #===> La variable forecast_out controla el tiempo de predicción a realizar

    import pandas as pd
    #import csv, math , datetime
    #import time
    import numpy as np
    from sklearn import preprocessing, cross_validation, svm
    from sklearn.linear_model import LinearRegression, LogisticRegression
    import math
    import matplotlib.pyplot as plt
    from matplotlib import style
    style.use('ggplot')


    from pandas import ExcelWriter

    from xlrd import open_workbook
    from xlutils.copy import copy

    from datetime import date
    from datetime import datetime,timedelta

    #============================== VAR GLOBALES ==================================

    global pos_Metodologia, pos_Realizado, pos_Comparacion
    global pos_anio_p, pos_mes_p, pos_dia_p
    global pos_anio_c, pos_mes_c, pos_dia_c
    #**************************************************************************

    global po_Metodologia, po_Realizado, po_Comparacion
    global po_anio_p, po_mes_p, po_dia_p
    global po_anio_c, po_mes_c, po_dia_c
    #==========================================================================


    doc_Proy       = 'HIS_POT_' + po_anio_p + '.xls'
    doc_Proy_extra = 'HIS_POT_' + str(int(po_anio_p)-1) + '.xls'
    doc_Comp       = 'HIS_POT_' + po_anio_c + '.xls'


    # Genera el DataFrame a partir del archivo de datos
```

```
3379.     df_Proy       = pd.read_excel(doc_Proy      , sheetname='Hoja1',  header=None)
3380.     df_Proy_extra = pd.read_excel(doc_Proy_extra, sheetname='Hoja1',  header=None)
3381.     df_Comp       = pd.read_excel(doc_Comp      , sheetname='Hoja1',  header=None)
3382.
3383.     ## Se escoge la columna que se desea realizar la prediccion
3384.     #forcast_col = 4
3385.     ## Llena con -99999 las celdas o espacios vacíos del DF
3386.     #df_Proy.fillna(-99999, inplace=True)
3387.
3388.     # Numero total de datos a proyectar, se toma 24hrs x 7 días
3389.     forecast_out = 24*7
3390.
3391.     # Lista de datos de históricos de proyección
3392.     amba_1 = (df_Proy_extra.iloc[:,4].values.tolist())
3393.     toto_1 = (df_Proy_extra.iloc[:,5].values.tolist())
3394.     puyo_1 = (df_Proy_extra.iloc[:,6].values.tolist())
3395.     tena_1 = (df_Proy_extra.iloc[:,7].values.tolist())
3396.     bani_1 = (df_Proy_extra.iloc[:,8].values.tolist())
3397.
3398.     amba_2 = (df_Proy.iloc[:,4].values.tolist())
3399.     toto_2 = (df_Proy.iloc[:,5].values.tolist())
3400.     puyo_2 = (df_Proy.iloc[:,6].values.tolist())
3401.     tena_2 = (df_Proy.iloc[:,7].values.tolist())
3402.     bani_2 = (df_Proy.iloc[:,8].values.tolist())
3403.
3404.     # Elimino la primera fila de cada DF (TEXTOS TÍTULOS)
3405.     amba_1.pop(0)
3406.     toto_1.pop(0)
3407.     puyo_1.pop(0)
3408.     tena_1.pop(0)
3409.     bani_1.pop(0)
3410.
3411.     amba_2.pop(0)
3412.     toto_2.pop(0)
3413.     puyo_2.pop(0)
3414.     tena_2.pop(0)
3415.     bani_2.pop(0)
3416.
3417.     amba_p = amba_1 + amba_2
3418.     toto_p = toto_1 + toto_2
3419.     puyo_p = puyo_1 + puyo_2
3420.     tena_p = tena_1 + tena_2
3421.     bani_p = bani_1 + bani_2
3422.
3423.     amba_p = pd.DataFrame(amba_p)
3424.     toto_p = pd.DataFrame(toto_p)
3425.     puyo_p = pd.DataFrame(puyo_p)
3426.     tena_p = pd.DataFrame(tena_p)
3427.     bani_p = pd.DataFrame(bani_p)
3428.
3429.     #================================================================================
3430.     #================================================================================
3431.     #===>UBICACIONES GENERALES de las posiciones para la proyección
3432.
3433.     #===> Variables de entrada, ejemplo
3434.     #po_anio_p = '2018'
3435.     #pos_mes_p = 0       # Marzo
3436.     #po_dia_p  = '1'
3437.
3438.     #===> Variables internas
3439.     # FECHA SELECCIONADA
```

```
3440.     fehca_selec = datetime(int(po_anio_p),(pos_mes_p+1),int(po_dia_p))
3441.     # DÍAS DE PROYECCIÓN
3442.     dias_proy = timedelta(days = 5952/24 + 1)   # 5952 * 24 = 248 días mas un día d
       e salto
3443.
3444.     # FECHA DE INICIO
3445.     fecha_inicio = (fehca_selec + timedelta(days = 7)) - dias_proy
3446.     # FECHA DE FIN
3447.     fecha_fin = fehca_selec
3448.     #fecha_fin = fehca_selec + timedelta(days = 7) # 24 horas de un día
3449.     # FECHAS DE INICIO DE CADA AÑO POSICIÓN INICIAL
3450.     enero_1 = datetime(fecha_inicio.year,1,1)
3451.     #enero_2 = datetime(fecha_fin.year   ,1,1)
3452.
3453.     ## POSICIÓN INICIAL
3454.     val_ini = (abs(fecha_inicio - enero_1).days) * 24 # Posición del valor inicial
       5952 DATOS
3455.     val_fin = (abs(fecha_fin    - enero_1).days) * 24 # Posición del valor final 59
       52 + 24 = 5976 DATOS
3456.
3457.     #print(val_ini)
3458.     #print(val_fin)
3459.
3460.     # ELIMINA LOS ÚLTIMOS VALORES DEL DF
3461.     amba_p = amba_p[:val_fin]
3462.     toto_p = toto_p[:val_fin]
3463.     puyo_p = puyo_p[:val_fin]
3464.     tena_p = tena_p[:val_fin]
3465.     bani_p = bani_p[:val_fin]
3466.     # ELIMINA LOS PRIMEROS VALORES DEL DF ** Queda un DF con 5952 datos
3467.     amba_p = amba_p[val_ini:]
3468.     toto_p = toto_p[val_ini:]
3469.     puyo_p = puyo_p[val_ini:]
3470.     tena_p = tena_p[val_ini:]
3471.     bani_p = bani_p[val_ini:]
3472.     # REINICIA EL INDEX DE CADA DF
3473.     amba_p = amba_p.reset_index(drop=True)
3474.     toto_p = toto_p.reset_index(drop=True)
3475.     puyo_p = puyo_p.reset_index(drop=True)
3476.     tena_p = tena_p.reset_index(drop=True)
3477.     bani_p = bani_p.reset_index(drop=True)
3478.
3479.     #=============================================================================
       ==
3480.     #=============================================================================
       ==
3481.
3482.     # Crea una nueva columna con la regresion apuntada del forecast_out
3483.     # La nueva columna es una copia y desplazo de valores de la columna volume
3484.     # La copia y desplaza los valores desde la posicion forecast_out.
3485.     amba_p['Prediccion'] = amba_p[0].shift(-forecast_out)
3486.     toto_p['Prediccion'] = toto_p[0].shift(-forecast_out)
3487.     puyo_p['Prediccion'] = puyo_p[0].shift(-forecast_out)
3488.     tena_p['Prediccion'] = tena_p[0].shift(-forecast_out)
3489.     bani_p['Prediccion'] = bani_p[0].shift(-forecast_out)
3490.
3491.     #Se crea un arreglo a partir del DF eliminando la columna de Prediccion
3492.     X_amba = np.array(amba_p.drop(['Prediccion'],1))
3493.     X_toto = np.array(toto_p.drop(['Prediccion'],1))
3494.     X_puyo = np.array(puyo_p.drop(['Prediccion'],1))
3495.     X_tena = np.array(tena_p.drop(['Prediccion'],1))
3496.     X_bani = np.array(bani_p.drop(['Prediccion'],1))
3497.
3498.     # Preprocesa y escala los datos del areglo X
3499.     X_amba = preprocessing.scale(X_amba)
3500.     X_toto = preprocessing.scale(X_toto)
```

```
3501.     X_puyo = preprocessing.scale(X_puyo)
3502.     X_tena = preprocessing.scale(X_tena)
3503.     X_bani = preprocessing.scale(X_bani)
3504.
3505.     # Crea nuevo arreglo tomando los (forecast_out = 7*24= 168) Ultimos datos de X

3506.     X_lately_amba = X_amba[-forecast_out:]
3507.     X_lately_toto = X_toto[-forecast_out:]
3508.     X_lately_puyo = X_puyo[-forecast_out:]
3509.     X_lately_tena = X_tena[-forecast_out:]
3510.     X_lately_bani = X_bani[-forecast_out:]
3511.
3512.     #El nuevo arreglo contiende todos los datos, incluso el valor de (forecast_out)

3513.     #Se queda con los primeros datos excepto los 168 últimos
3514.     X_amba = X_amba[:-forecast_out:]
3515.     X_toto = X_toto[:-forecast_out:]
3516.     X_puyo = X_puyo[:-forecast_out:]
3517.     X_tena = X_tena[:-forecast_out:]
3518.     X_bani = X_bani[:-forecast_out:]
3519.
3520.     # Elimina todas las filas que tienen datos vaci-
      os NAN  o sea los (forecast_out) ultimos datos
3521.     amba_p.dropna(inplace=True)
3522.     toto_p.dropna(inplace=True)
3523.     puyo_p.dropna(inplace=True)
3524.     tena_p.dropna(inplace=True)
3525.     bani_p.dropna(inplace=True)
3526.
3527.
3528.     # Se crea nuevo arreglo que contiene la columna de Prediccion
3529.     y_amba = np.array(amba_p['Prediccion'])
3530.     y_toto = np.array(toto_p['Prediccion'])
3531.     y_puyo = np.array(puyo_p['Prediccion'])
3532.     y_tena = np.array(tena_p['Prediccion'])
3533.     y_bani = np.array(bani_p['Prediccion'])
3534.
3535.     #=============================================================================

3536.     #=======================   INICIO ALGORITMO SVM    ============================

3537.     #=============================================================================

3538.
3539.     model = svm.SVR(kernel='rbf',gamma='auto',C=10)
3540.
3541.     # Metodologia de validacion cruzada para los datos obtenidos
3542.     X_train_amba, X_test_amba, y_train_amba, y_test_amba = cross_validation.train_t
      est_split(X_amba, y_amba, test_size=0.3)
3543.     model.fit(X_train_amba,y_train_amba)
3544.     accuracy_amba = model.score(X_test_amba,y_test_amba)
3545.     ProyAmba = model.predict(X_lately_amba)# ============>>>   SALIDA PROYECCIÓN
3546.
3547.     X_train_toto, X_test_toto, y_train_toto, y_test_toto = cross_validation.train_t
      est_split(X_toto, y_toto, test_size=0.3)
3548.     model.fit(X_train_toto,y_train_toto)
3549.     accuracy_toto = model.score(X_test_toto,y_test_toto)
3550.     ProyToto = model.predict(X_lately_toto)# ============>>>   SALIDA PROYECCIÓN
            forecast_set
3551.
3552.     X_train_puyo, X_test_puyo, y_train_puyo, y_test_puyo = cross_validation.train_t
      est_split(X_puyo, y_puyo, test_size=0.3)
3553.     model.fit(X_train_puyo,y_train_puyo)
3554.     accuracy_puyo = model.score(X_test_puyo,y_test_puyo)
3555.     ProyPuyo = model.predict(X_lately_puyo)# ============>>>   SALIDA PROYECCIÓN
            forecast_set
```

```
3556.
3557.     X_train_tena, X_test_tena, y_train_tena, y_test_tena = cross_validation.train_t
       est_split(X_tena, y_tena, test_size=0.3)
3558.     model.fit(X_train_tena,y_train_tena)
3559.     accuracy_tena = model.score(X_test_tena,y_test_tena)
3560.     ProyTena = model.predict(X_lately_tena)# ============>>>   SALIDA PROYECCIÓN
              forecast_set
3561.
3562.     X_train_bani, X_test_bani, y_train_bani, y_test_bani = cross_validation.train_t
       est_split(X_bani, y_bani, test_size=0.3)
3563.     model.fit(X_train_bani,y_train_bani)
3564.     accuracy_bani = model.score(X_test_bani,y_test_bani)
3565.     ProyBani = model.predict(X_lately_bani)# ============>>>   SALIDA PROYECCIÓN
              forecast_set
3566.
3567.     # Datos de comparación
3568.     #CompAmba = (amba_p[0][-forecast_out:]).reset_index(drop=True)
3569.     #CompToto = (toto_p[0][-forecast_out:]).reset_index(drop=True)
3570.     #CompPuyo = (puyo_p[0][-forecast_out:]).reset_index(drop=True)
3571.     #CompTena = (tena_p[0][-forecast_out:]).reset_index(drop=True)
3572.     #CompBani = (bani_p[0][-forecast_out:]).reset_index(drop=True)
3573.     #================================================================================

3574.     #========================== SEMANA DE COMPARACIÓN ============================

3575.     #================================================================================

3576.
3577.     mes_c = df_Comp.iloc[:,0].values.tolist()
3578.     mes_c.pop(0)
3579.
3580.     dia_c = df_Comp.iloc[:,1].values.tolist()
3581.     dia_c.pop(0)
3582.
3583.     hora_c = df_Comp.iloc[:,2].values.tolist()
3584.     hora_c.pop(0)
3585.
3586.     amba_c = df_Comp.iloc[:,4].values.tolist()
3587.     amba_c.pop(0)
3588.
3589.     toto_c = df_Comp.iloc[:,5].values.tolist()
3590.     toto_c.pop(0)
3591.
3592.     puyo_c = df_Comp.iloc[:,6].values.tolist()
3593.     puyo_c.pop(0)
3594.
3595.     tena_c = df_Comp.iloc[:,7].values.tolist()
3596.     tena_c.pop(0)
3597.
3598.     bani_c = df_Comp.iloc[:,8].values.tolist()
3599.     bani_c.pop(0)
3600.
3601.     #===> Reemplaza valores (0) con (nan)
3602.     for i in range(len(dia_c)):
3603.         if amba_c[i] == 0:
3604.             amba_c[i] = float('nan')
3605.         if toto_c[i] == 0:
3606.             toto_c[i] = float('nan')
3607.         if puyo_c[i] == 0:
3608.             puyo_c[i] = float('nan')
3609.         if tena_c[i] == 0:
3610.             tena_c[i] = float('nan')
3611.         if bani_c[i] == 0:
3612.             bani_c[i] = float('nan')
3613.
3614.     #===> Se establece una matriz con los datos importados
```

```
3615.     #data_c = np.column_stack((amba_c, toto_c, puyo_c, tena_c, bani_c))
3616.
3617.     #============================================================================
3618.     #============================================================================
3619.     if po_mes_c == 'ENERO':
3620.         if int(po_dia_c) < 8:
3621.
3622.             doc_Proy_EXTRA_c = 'HIS_POT_' + str(int(po_anio_c)-1) + '.xls'
3623.             df_Proy_EXTRA_c = pd.read_excel(doc_Proy_EXTRA_c, sheetname='Hoja1',  h
      eader=None)
3624.             df_Proy_EXTRA_c = df_Proy_EXTRA_c[-(24*31):]
3625.
3626.             mess_extra_c = df_Proy_EXTRA_c.iloc[:,0].values.tolist()
3627.             amba_extra_c = df_Proy_EXTRA_c.iloc[:,4].values.tolist()
3628.             toto_extra_c = df_Proy_EXTRA_c.iloc[:,5].values.tolist()
3629.             puyo_extra_c = df_Proy_EXTRA_c.iloc[:,6].values.tolist()
3630.             tena_extra_c = df_Proy_EXTRA_c.iloc[:,7].values.tolist()
3631.             bani_extra_c = df_Proy_EXTRA_c.iloc[:,8].values.tolist()
3632.             #===> Obtengo únicamente 7 días de diciembre del año pasado
3633.             mess_extra_c = mess_extra_c[-(24*31):]
3634.             #===> Elimino el mes de diciembre del año actual
3635.             mes_c = mes_c[:-(24*31)]
3636.             #===> Nueva lista con 7 días de Dic del año pasado mas resto de datos a
      ño actual
3637.             mes_c = mess_extra_c + mes_c
3638.
3639.             amba_c = amba_extra_c + amba_c
3640.             toto_c = toto_extra_c + toto_c
3641.             puyo_c = puyo_extra_c + puyo_c
3642.             tena_c = tena_extra_c + tena_c
3643.             bani_c = bani_extra_c + bani_c
3644.
3645.     #===> APUNTA AL MES SELECCIONADO DE COMPARACIÓN
3646.     x_meses_c = [mes_c.index('ENERO'),   mes_c.index('FEBRERO'),   mes_c.index('MAR
      ZO'),
3647.                  mes_c.index('ABRIL'),   mes_c.index('MAYO'),      mes_c.index('JUN
      IO'),
3648.                  mes_c.index('JULIO'),   mes_c.index('AGOSTO'),    mes_c.index('SEP
      TIEMBRE'),
3649.                  mes_c.index('OCTUBRE'), mes_c.index('NOVIEMBRE'), mes_c.index('DIC
      IEMBRE')]
3650.
3651.     for i in range (12):
3652.         if pos_mes_c == i:
3653.             if pos_mes_c == 11:
3654.                 ubic_mes_c = x_meses_c[i]
3655.             else:
3656.                 ubic_mes_c = x_meses_c[i] # VARIABLE Posición del mes seleccionado
3657.
3658.
3659.     #===> DEFINE LA UBICACIÓN DEL DÍA SELECCIONADO DE COMPARACIÓN
3660. #    print(' ')
3661. #    print ('Fecha Comparación: ',po_dia_c,' de ',mes_c[ubic_mes_c],' de ',po_anio_
      c)
3662.     if int(po_dia_c) == 1:
3663.         ubicacion_c = ubic_mes_c
3664.     else:
3665.         ubicacion_c = (int(po_dia_c)-1) * 24 + ubic_mes_c
3666.
3667. #    print(' ')
3668. #    print (ubicacion_c) #============================================================
      =>>> VARIABLE DE UBICACIÓN EXACTA EN LISTA DE DATOS de comparación
3669.
```

```
3670.     amba_c = amba_c[:ubicacion_c]
3671.     toto_c = toto_c[:ubicacion_c]
3672.     puyo_c = puyo_c[:ubicacion_c]
3673.     tena_c = tena_c[:ubicacion_c]
3674.     bani_c = bani_c[:ubicacion_c]
3675.
3676.     #===> Valores inicio y fin para lista de datos a ser usada
3677.     val_ini_c = ubicacion_c - 24 * 7 #Posición del valor inicial
3678.     val_fin_c = ubicacion_c - 1      #Posición del valor final
3679.
3680.     #===> Comparación semananal S/E Ambato
3681.     CompAmba = []
3682.     for i in range (24*7):
3683.         CompAmba.append(amba_c[val_ini_c + i ])
3684.     #===> Comparación semananal S/E Totoras
3685.     CompToto = []
3686.     for i in range (24*7):
3687.         CompToto.append(toto_c[val_ini_c + i ])
3688.     #===> Comparación semananal S/E Puyo
3689.     CompPuyo = []
3690.     for i in range (24*7):
3691.         CompPuyo.append(puyo_c[val_ini_c + i  ])
3692.     #===> Comparación semananal S/E Tena
3693.     CompTena = []
3694.     for i in range (24*7):
3695.         CompTena.append(tena_c[val_ini_c + i ])
3696.     #===> Comparación semananal S/E Baños
3697.     CompBani = []
3698.     for i in range (24*7):
3699.         CompBani.append(bani_c[val_ini_c + i ])
3700.
3701.     #=============================================================================

3702.     #================================ GRÁFICAS ===================================

3703.     #=============================================================================

3704.     if po_Comparacion == 'SI':
3705.     #===> Creamos una lista tipo entero para relacionar con las etiquetas
3706.         can_datos = []
3707.         for i in range(7*24):
3708.             if i%2!=1:
3709.                 can_datos.append(i)
3710.     #===> Creamos una lista con los números de horas del día seleccionado (etiqueta
        s)
3711.         horas_dia = []
3712.         horas_str = []
3713.         for i in range (7):
3714.             for i in range (1,25):
3715.                 if i%2!=0:
3716.                     horas_dia.append(i)
3717.         for i in range (len(horas_dia)):
3718.             horas_str.append(str(horas_dia[i]))
3719.
3720.     #===> Tamaño de la ventana de la gráfica
3721.         plt.subplots(figsize=(15, 8))
3722.
3723.     #===> Título general superior
3724.         plt.suptitle(u' PROYECCIÓN SEMANAL DE CARGA\n SUPORT VECTOR MACHINE ',fonts
        ize=14, fontweight='bold')
3725.
3726.         plt.subplot(5,1,1)
3727.         plt.plot(CompAmba,'blue', label = 'Comparación')
3728.         plt.plot(ProyAmba,'#DCD037', label = 'Proyección')
3729.         plt.legend(loc='upper left')
3730.         plt.xlabel(u'TIEMPO [ días ]')
```

```
3731.         plt.ylabel(u'S/E AMBATO\n\nCARGA  [ kW ]')
3732.         plt.grid(color='#C8C8C8', linestyle='-', linewidth=0.5)
3733.         plt.xticks(can_datos, horas_str, size = 5, color = 'b', rotation = 90)
3734.
3735.         plt.subplot(5,1,2)
3736.         plt.plot(CompToto,'blue', label = 'Comparación')
3737.         plt.plot(ProyToto,'#CD336F', label = 'Proyección')
3738.         plt.legend(loc='upper left')
3739.         plt.xlabel(u'TIEMPO [ días ]')
3740.         plt.ylabel(u'S/E TOTORAS\n\nCARGA  [ kW ]')
3741.         plt.grid(color='#C8C8C8', linestyle='-', linewidth=0.5)
3742.         plt.xticks(can_datos, horas_str, size = 5, color = 'b', rotation = 90)
3743.
3744.         plt.subplot(5,1,3)
3745.         plt.plot(CompPuyo,'blue', label = 'Comparación')
3746.         plt.plot(ProyPuyo,'#349A9D', label = 'Proyección')
3747.         plt.legend(loc='upper left')
3748.         plt.xlabel(u'TIEMPO [ días ]')
3749.         plt.ylabel(u'S/E PUYO\n\nCARGA  [ kW ]')
3750.         plt.grid(color='#C8C8C8', linestyle='-', linewidth=0.5)
3751.         plt.xticks(can_datos, horas_str, size = 5, color = 'b', rotation = 90)
3752.
3753.         plt.subplot(5,1,4)
3754.         plt.plot(CompTena,'blue', label = 'Comparación')
3755.         plt.plot(ProyTena,'#CC8634', label = 'Proyección')
3756.         plt.legend(loc='upper left')
3757.         plt.xlabel(u'TIEMPO [ días ]')
3758.         plt.ylabel(u'S/E TENA\n\nCARGA  [ kW ]')
3759.         plt.grid(color='#C8C8C8', linestyle='-', linewidth=0.5)
3760.         plt.xticks(can_datos, horas_str, size = 5, color = 'b', rotation = 90)
3761.
3762.         plt.subplot(5,1,5)
3763.         plt.plot(CompBani,'blue', label = 'Comparación')
3764.         plt.plot(ProyBani,'#4ACB71', label = 'Proyección')
3765.         plt.legend(loc='upper left')
3766.         plt.xlabel(u'TIEMPO [ días ]')
3767.         plt.ylabel(u'S/E BAÑOS\n\nCARGA  [ kW ]')
3768.         plt.grid(color='#C8C8C8', linestyle='-', linewidth=0.5)
3769.         plt.xticks(can_datos, horas_str, size = 5, color = 'b', rotation = 90)
3770.
3771.     #graf = []
3772.     #for i in range (52248, 52415):
3773.     #    graf.append(xp[i])
3774.     #plt.legend(loc=1)
3775.
3776.     #==========> Fechas de salida:
3777.     sem_ini = datetime(int(po_anio_p),(pos_mes_p+1),int(po_dia_p))
3778.     sem_fin = sem_ini + timedelta(days = 6)
3779.
3780.     #================================================================================
3781.     #======================== RUTINA GENERA EXCEL ===================================
3782.     #================================================================================
3783.     df = pd.DataFrame([' '])
3784.     writer = ExcelWriter('03_PROYECCION_SVM.xls')
3785.     df.to_excel(writer, 'Salida_Proyección_CECON', index=False)
3786.     df.to_excel(writer, 'Salida_Comparación_CECON', index=False)
3787.     df.to_excel(writer, 'Salida_ERRORES_CECON', index=False)
3788.     writer.save()
3789.
3790.     #abre el archivo de excel plantilla
3791.     rb = open_workbook('03_PROYECCION_SVM.xls')
3792.     #crea una copia del archivo plantilla
3793.     wb = copy(rb)
```

```
3794.     #se ingresa a la hoja 1 de la copia del archivo excel
3795.     ws = wb.get_sheet(0)
3796.     ws_c = wb.get_sheet(1)
3797.     ws_e = wb.get_sheet(2)
3798.     #============================Hoja 1 Proyección ================================

3799.     #ws.write(0,0,'MES')
3800.     #ws.write(0,1,'DÍA')
3801.     #ws.write(0,2,'#')
3802.     ws.write(0,0,'METODOLOGÍA MÁQUINA DE VECTOR SOPORTE  (S.V.M.)')
3803.     ws.write(1,0,'PROYECCIÓN SEMANAL DE CARGA')
3804.     ws.write(2,0,'Semana de Proyección: del '+str(sem_ini.day)+'/'
3805.              +str(sem_ini.month) +'/'+str(sem_ini.year)+'\t al '+
3806.              str(sem_fin.day)+'/'+str(sem_fin.month)+'/'+str(sem_fin.year))
3807.
3808.     ws.write(3,0,'Realizado por: '+str(po_Realizado))
3809.
3810.     # Define lista de títulos
3811.     titulos = ['HORA','S/E AMBATO','S/E TOTORAS','S/E PUYO','S/E TENA','S/E BAÑOS']

3812.
3813.     for i in range(len(titulos)):
3814.         ws.write(9,i,titulos[i])
3815.
3816.     Aux = 10
3817.     Aux2 = 0
3818.     for i in range (7):
3819.         for j in range (24):
3820.             ws.write(Aux,0,j+1)
3821.             ws.write(Aux,1,ProyAmba[j + Aux2])
3822.             ws.write(Aux,2,ProyToto[j + Aux2])
3823.             ws.write(Aux,3,ProyPuyo[j + Aux2])
3824.             ws.write(Aux,4,ProyTena[j + Aux2])
3825.             ws.write(Aux,5,ProyBani[j + Aux2])
3826.             Aux = Aux + 1
3827.         Aux2 = 24 * (i+1)
3828.         Aux = Aux + 1
3829.     #============================Hoja 2 Comparación ================================

3830.
3831.     #===> Fecha
3832.     ws_c.write(0,0,'Semana de Proyección: del '+str(sem_ini.day)+'/'
3833.                +str(sem_ini.month) +'/'+str(sem_ini.year)+'\t al '+
3834.                str(sem_fin.day)+'/'+str(sem_fin.month)+'/'+str(sem_fin.year))
3835.     ws_c.write(2,0,'PROYECCIÓN SEMANAL')
3836.
3837.     for i in range(len(titulos)):
3838.         ws_c.write(9,i,titulos[i])
3839.
3840.     Aux = 10
3841.     Aux2 = 0
3842.     for i in range (7):
3843.         for j in range (24):
3844.             ws_c.write(Aux,0,j+1)
3845.             ws_c.write(Aux,1,ProyAmba[j + Aux2])
3846.             ws_c.write(Aux,2,ProyToto[j + Aux2])
3847.             ws_c.write(Aux,3,ProyPuyo[j + Aux2])
3848.             ws_c.write(Aux,4,ProyTena[j + Aux2])
3849.             ws_c.write(Aux,5,ProyBani[j + Aux2])
3850.             Aux = Aux + 1
3851.         Aux2 = 24 * (i+1)
3852.         Aux = Aux + 1
3853.
3854.     #ws_c.write(0,6,po_dia_c+'/'+po_mes_c+'/'+po_anio_c)
3855.     ws_c.write(2,6,'SEMANA DE COMPARACIÓN')
3856.
```

```
    for i in range(len(titulos)):
        ws_c.write(9,i+6,titulos[i])

    Aux = 10
    Aux2 = 0
    for i in range (7):
        for j in range (24):
            ws_c.write(Aux,6,j+1)
            ws_c.write(Aux,7,CompAmba[j + Aux2])
            ws_c.write(Aux,8,CompToto[j + Aux2])
            ws_c.write(Aux,9,CompPuyo[j + Aux2])
            ws_c.write(Aux,10,CompTena[j + Aux2])
            ws_c.write(Aux,11,CompBani[j + Aux2])
            Aux = Aux + 1
        Aux2 = 24 * (i+1)
        Aux = Aux + 1


    # ==========================   Cálculo de errores    ==========================

    ws_c.write(2,12,'CÁLCULO DE ERRORES')
    ws_c.write(3,12,'PORCENTAJE DE ERROR MEDIO ABSOLUTO (PEMA)')

    for i in range(len(titulos)):
        ws_c.write(9,i+12,titulos[i])


    errAmba = []
    errToto = []
    errPuyo = []
    errTena = []
    errBani = []

    for i in range(len(ProyAmba)):
        errAmba.append((abs( ProyAmba[i] - CompAmba[i] ) / CompAmba[i] )*100)
        errToto.append((abs( ProyToto[i] - CompToto[i] ) / CompToto[i] )*100)
        errPuyo.append((abs( ProyPuyo[i] - CompPuyo[i] ) / CompPuyo[i] )*100)
        errTena.append((abs( ProyTena[i] - CompTena[i] ) / CompTena[i] )*100)
        errBani.append((abs( ProyBani[i] - CompBani[i] ) / CompBani[i] )*100)

    Aux = 10
    Aux2 = 0
    for i in range (7):
        for j in range (24):
            ws_c.write(Aux,12,j+1)
            ws_c.write(Aux,13,errAmba[j + Aux2])
            ws_c.write(Aux,14,errToto[j + Aux2])
            ws_c.write(Aux,15,errPuyo[j + Aux2])
            ws_c.write(Aux,16,errTena[j + Aux2])
            ws_c.write(Aux,17,errBani[j + Aux2])
            Aux = Aux + 1
        Aux2 = 24 * (i+1)
        Aux = Aux + 1
    # Suma de los valores de las listas
    SumAmba = 0
    SumToto = 0
    SumPuyo = 0
    SumTena = 0
    SumBani = 0
    for i in range (len(ProyAmba)):
        SumAmba = errAmba[i] + SumAmba
        SumToto = errToto[i] + SumToto
        SumPuyo = errPuyo[i] + SumPuyo
        SumTena = errTena[i] + SumTena
        SumBani = errBani[i] + SumBani
```

```
    # Almaceno en una lista los resultados de las sumas
    Sumas = [SumAmba, SumToto, SumPuyo, SumTena, SumBani]

    # Imprime los resultados en la fila correspondiente
    ws_c.write(Aux+1,11,'SUMATORIA TOTAL')
    for i in range (len(Sumas)):
         ws_c.write(Aux+1,i+13,Sumas[i])

    # Cálculo de los promedios de las sumas
    ws_c.write(Aux+2,11,'ERROR MEDIO ABSOLUTO')
    for i in range(len(Sumas)):
        ws_c.write(Aux+2,i+13,(Sumas[i]/len(errAmba)))

    # Cálculo de la exactitud de la proyección
    ws_c.write(Aux+3,11,'EXACTITUD DE LA PROYECCIÓN')
    for i in range(len(Sumas)):
        ws_c.write(Aux+3,i+13,(100-(Sumas[i]/len(errAmba))))


    for i in range(len(titulos)-1):
        ws_c.write(Aux,i+13,titulos[i+1])

    # ==========================   SOLO   errores    ==========================

    # ==================================> cálculo de et = Comp - Proy
    for i in range(len(titulos)-1):
        ws_e.write(9,i,titulos[i+1])
    ws_e.write(7,0,'et = Comp - Proy')

    et_Amba = []
    et_Toto = []
    et_Puyo = []
    et_Tena = []
    et_Bani = []

    for i in range (len(ProyAmba)):
        et_Amba.append(CompAmba[i] - ProyAmba[i])
        et_Toto.append(CompToto[i] - ProyToto[i])
        et_Puyo.append(CompPuyo[i] - ProyPuyo[i])
        et_Tena.append(CompTena[i] - ProyTena[i])
        et_Bani.append(CompBani[i] - ProyBani[i])

        ws_e.write(i+10,0, (et_Amba[i]))
        ws_e.write(i+10,1, (et_Toto[i]))
        ws_e.write(i+10,2, (et_Puyo[i]))
        ws_e.write(i+10,3, (et_Tena[i]))
        ws_e.write(i+10,4, (et_Bani[i]))

    # ==================================> cálculo de abs(et) = abs(Comp - Proy)
    for i in range(len(titulos)-1):
        ws_e.write(9,i+6,titulos[i+1])
    ws_e.write(7,6,'abs(et) = abs(Comp - Proy)')

    abs_et_Amba = []
    abs_et_Toto = []
    abs_et_Puyo = []
    abs_et_Tena = []
    abs_et_Bani = []

    for i in range (len(ProyAmba)):
        abs_et_Amba.append(abs(et_Amba[i]))
        abs_et_Toto.append(abs(et_Toto[i]))
        abs_et_Puyo.append(abs(et_Puyo[i]))
        abs_et_Tena.append(abs(et_Tena[i]))
        abs_et_Bani.append(abs(et_Bani[i]))

```

```
3988.          ws_e.write(i+10,6,  (abs_et_Amba[i]))
3989.          ws_e.write(i+10,7,  (abs_et_Toto[i]))
3990.          ws_e.write(i+10,8,  (abs_et_Puyo[i]))
3991.          ws_e.write(i+10,9,  (abs_et_Tena[i]))
3992.          ws_e.write(i+10,10, (abs_et_Bani[i]))
3993.
3994.      # ==================================> cálculo de et^2
3995.      for i in range(len(titulos)-1):
3996.          ws_e.write(9,i+12,titulos[i+1])
3997.      ws_e.write(7,12,'et^2')
3998.
3999.      et_Amba2 = []
4000.      et_Toto2 = []
4001.      et_Puyo2 = []
4002.      et_Tena2 = []
4003.      et_Bani2 = []
4004.
4005.      for i in range (len(ProyAmba)):
4006.          et_Amba2.append((et_Amba[i])**2)
4007.          et_Toto2.append((et_Toto[i])**2)
4008.          et_Puyo2.append((et_Puyo[i])**2)
4009.          et_Tena2.append((et_Tena[i])**2)
4010.          et_Bani2.append((et_Bani[i])**2)
4011.
4012.          ws_e.write(i+10,12, (et_Amba2[i]))
4013.          ws_e.write(i+10,13, (et_Toto2[i]))
4014.          ws_e.write(i+10,14, (et_Puyo2[i]))
4015.          ws_e.write(i+10,15, (et_Tena2[i]))
4016.          ws_e.write(i+10,16, (et_Bani2[i]))
4017.
4018.      # ==================================> cálculo de abs(et) / Comp
4019.      for i in range(len(titulos)-1):
4020.          ws_e.write(9,i+18,titulos[i+1])
4021.      ws_e.write(7,18,'abs(et) / Comp')
4022.
4023.      d1_Amba = []
4024.      d1_Toto = []
4025.      d1_Puyo = []
4026.      d1_Tena = []
4027.      d1_Bani = []
4028.
4029.      for i in range (len(ProyAmba)):
4030.          d1_Amba.append(abs_et_Amba[i] / CompAmba[i])
4031.          d1_Toto.append(abs_et_Toto[i] / CompToto[i])
4032.          d1_Puyo.append(abs_et_Puyo[i] / CompPuyo[i])
4033.          d1_Tena.append(abs_et_Tena[i] / CompTena[i])
4034.          d1_Bani.append(abs_et_Bani[i] / CompBani[i])
4035.
4036.          ws_e.write(i+10,18, (d1_Amba[i]))
4037.          ws_e.write(i+10,19, (d1_Toto[i]))
4038.          ws_e.write(i+10,20, (d1_Puyo[i]))
4039.          ws_e.write(i+10,21, (d1_Tena[i]))
4040.          ws_e.write(i+10,22, (d1_Bani[i]))
4041.
4042.      # ==================================> cálculo de et / Comp
4043.      for i in range(len(titulos)-1):
4044.          ws_e.write(9,i+24,titulos[i+1])
4045.      ws_e.write(7,24,'et / Comp')
4046.
4047.      d2_Amba = []
4048.      d2_Toto = []
4049.      d2_Puyo = []
4050.      d2_Tena = []
4051.      d2_Bani = []
4052.
4053.      for i in range (len(ProyAmba)):
```

```
4054.          d2_Amba.append(et_Amba[i] / CompAmba[i])
4055.          d2_Toto.append(et_Toto[i] / CompToto[i])
4056.          d2_Puyo.append(et_Puyo[i] / CompPuyo[i])
4057.          d2_Tena.append(et_Tena[i] / CompTena[i])
4058.          d2_Bani.append(et_Bani[i] / CompBani[i])
4059.
4060.          ws_e.write(i+10,24, (d2_Amba[i]))
4061.          ws_e.write(i+10,25, (d2_Toto[i]))
4062.          ws_e.write(i+10,26, (d2_Puyo[i]))
4063.          ws_e.write(i+10,27, (d2_Tena[i]))
4064.          ws_e.write(i+10,28, (d2_Bani[i]))
4065.
4066.      ws_e.write(0,0, 'INDICADORES')
4067.      # ==================================> Cálculo DAM
4068.      DAM_Amba = 0
4069.      DAM_Toto = 0
4070.      DAM_Puyo = 0
4071.      DAM_Tena = 0
4072.      DAM_Bani = 0
4073.
4074.      for i in range (len(ProyAmba)):
4075.          DAM_Amba = abs_et_Amba[i] + DAM_Amba
4076.          DAM_Toto = abs_et_Toto[i] + DAM_Toto
4077.          DAM_Puyo = abs_et_Puyo[i] + DAM_Puyo
4078.          DAM_Tena = abs_et_Tena[i] + DAM_Tena
4079.          DAM_Bani = abs_et_Bani[i] + DAM_Bani
4080.
4081.      DAM_Amba = DAM_Amba / (len(ProyAmba))
4082.      DAM_Toto = DAM_Toto / (len(ProyAmba))
4083.      DAM_Puyo = DAM_Puyo / (len(ProyAmba))
4084.      DAM_Tena = DAM_Tena / (len(ProyAmba))
4085.      DAM_Bani = DAM_Bani / (len(ProyAmba))
4086.
4087.      ws_e.write(1,0, ('DAM'))
4088.      DAM = [DAM_Amba,DAM_Toto,DAM_Puyo,DAM_Tena,DAM_Bani]
4089.      for i in range(len(DAM)):
4090.          ws_e.write(1,i+1,DAM[i])
4091.
4092.
4093.      # ==================================> Cálculo EMC
4094.      EMC_Amba = 0
4095.      EMC_Toto = 0
4096.      EMC_Puyo = 0
4097.      EMC_Tena = 0
4098.      EMC_Bani = 0
4099.
4100.      for i in range (len(ProyAmba)):
4101.          EMC_Amba = et_Amba2[i] + EMC_Amba
4102.          EMC_Toto = et_Toto2[i] + EMC_Toto
4103.          EMC_Puyo = et_Puyo2[i] + EMC_Puyo
4104.          EMC_Tena = et_Tena2[i] + EMC_Tena
4105.          EMC_Bani = et_Bani2[i] + EMC_Bani
4106.
4107.      EMC_Amba = EMC_Amba / (len(ProyAmba))
4108.      EMC_Toto = EMC_Toto / (len(ProyAmba))
4109.      EMC_Puyo = EMC_Puyo / (len(ProyAmba))
4110.      EMC_Tena = EMC_Tena / (len(ProyAmba))
4111.      EMC_Bani = EMC_Bani / (len(ProyAmba))
4112.
4113.      ws_e.write(2,0, ('EMC'))
4114.      EMC = [EMC_Amba,EMC_Toto,EMC_Puyo,EMC_Tena,EMC_Bani]
4115.
4116.      for i in range(len(EMC)):
4117.          ws_e.write(2,i+1,EMC[i])
4118.
4119.      # ==================================> Cálculo PEMA
```

```
4120.     PEMA_Amba = 0
4121.     PEMA_Toto = 0
4122.     PEMA_Puyo = 0
4123.     PEMA_Tena = 0
4124.     PEMA_Bani = 0
4125.
4126.     for i in range (len(ProyAmba)):
4127.         PEMA_Amba = (abs_et_Amba[i] / CompAmba [i]) + PEMA_Amba
4128.         PEMA_Toto = (abs_et_Toto[i] / CompToto [i]) + PEMA_Toto
4129.         PEMA_Puyo = (abs_et_Puyo[i] / CompPuyo [i]) + PEMA_Puyo
4130.         PEMA_Tena = (abs_et_Tena[i] / CompTena [i]) + PEMA_Tena
4131.         PEMA_Bani = (abs_et_Bani[i] / CompBani [i]) + PEMA_Bani
4132.
4133.     PEMA_Amba = (PEMA_Amba / (len(ProyAmba))) *100
4134.     PEMA_Toto = (PEMA_Toto / (len(ProyAmba))) *100
4135.     PEMA_Puyo = (PEMA_Puyo / (len(ProyAmba))) *100
4136.     PEMA_Tena = (PEMA_Tena / (len(ProyAmba))) *100
4137.     PEMA_Bani = (PEMA_Bani / (len(ProyAmba))) *100
4138.
4139.     ws_e.write(3,0, ('PEMA'))
4140.     PEMA = [PEMA_Amba,PEMA_Toto,PEMA_Puyo,PEMA_Tena,PEMA_Bani]
4141.
4142.     for i in range(len(PEMA)):
4143.         ws_e.write(3,i+1,PEMA[i])
4144.
4145.     # ==================================> Cálculo PME
4146.     PME_Amba = 0
4147.     PME_Toto = 0
4148.     PME_Puyo = 0
4149.     PME_Tena = 0
4150.     PME_Bani = 0
4151.
4152.     for i in range (len(ProyAmba)):
4153.         PME_Amba = (et_Amba[i] / CompAmba [i]) + PME_Amba
4154.         PME_Toto = (et_Toto[i] / CompToto [i]) + PME_Toto
4155.         PME_Puyo = (et_Puyo[i] / CompPuyo [i]) + PME_Puyo
4156.         PME_Tena = (et_Tena[i] / CompTena [i]) + PME_Tena
4157.         PME_Bani = (et_Bani[i] / CompBani [i]) + PME_Bani
4158.
4159.     PME_Amba = (PME_Amba / (len(ProyAmba))) *100
4160.     PME_Toto = (PME_Toto / (len(ProyAmba))) *100
4161.     PME_Puyo = (PME_Puyo / (len(ProyAmba))) *100
4162.     PME_Tena = (PME_Tena / (len(ProyAmba))) *100
4163.     PME_Bani = (PME_Bani / (len(ProyAmba))) *100
4164.
4165.     ws_e.write(4,0, ('PME'))
4166.     PME = [PME_Amba,PME_Toto,PME_Puyo,PME_Tena,PME_Bani]
4167.
4168.     for i in range(len(PME)):
4169.         ws_e.write(4,i+1,PME[i])
4170.
4171.     # ==================================> Cálculo EXACTITUD DE LA PROYECCION
4172.     ws_e.write(5,0,'EP')
4173.     for i in range(len(PEMA)):
4174.         ws_e.write(5,i+1,(100-PEMA[i]))
4175.
4176.
4177.     for i in range(len(titulos)-1):
4178.         ws_e.write(0,i+1,titulos[i+1])
4179.
4180.     wb.save('03_PROYECCION_SVM.xls')
4181.     #=============================================================================
4182.     #=============================================================================
4183.
```

```
4184.     print(' ')
4185.     print(' ')
4186.     print('* La fecha seleccionada es:',int(po_anio_p),'/',(pos_mes_p+1),'/',int(po
      _dia_p))
4187.     print(' ')
4188.     print(' ')
4189.     print('* Porcentaje de Exactitud de la predicción por LR :')
4190.     print(' ')
4191.     print('==> S/E Ambato :',accuracy_amba*100,'%')
4192.     print('==> S/E Totoras:',accuracy_toto*100,'%')
4193.     print('==> S/E Puyo   :',accuracy_puyo*100,'%')
4194.     print('==> S/E Tena   :',accuracy_tena*100,'%')
4195.     print('==> S/E Baños  :',accuracy_bani*100,'%')
4196.     print(' ')
4197.     print(' ')
4198.     print('*** SVM Completed...!!!')
4199.     print(' ')
4200.     print(' ')
4201.
4202. start_time = time()
4203. test()
4204. elapsed_time = time() - start_time
4205. print(' ')
4206. print(' ')
4207. print('////////////////////////////////////////')
4208. print(' ')
4209. print('ALGORITMO UTILIZADO: SVM')
4210. print("Tiempo transcurrido: %.10f segundos." % elapsed_time)
4211. print(' ')
4212. print('////////////////////////////////////////')
```

8. MÓDULO DE PROYECCIÓN CON ALGORITMO MLP

```
4213.
4214. """
4215. Created on Tue Apr  9 12:03:53 2019
4216. ESCUELA POLITÉCNICA NACIONAL
4217. AUTOR: ALEXIS GUAMAN FIGUEROA
4218. 8_MÓDULO DE PROYECCIÓN CON ALGORITMO MLP
4219. """
4220.
4221. from time import time
4222. def test():
4223.     #====================== MODELO DE PERCEPTRÓN MULTICAPA ========================
4224.
4225.     #===> PREDICCIÓN DE CARGA (t+1, dado t), Exactitud del 97%
4226.
4227.     import numpy
4228.     import matplotlib.pyplot as plt, mpld3
4229.     import pandas
4230.     import math
4231.     from keras.models import Sequential
4232.     from keras.layers import Dense
4233.
4234.
4235.     import pandas as pd
4236.     from pandas import ExcelWriter
4237.
4238.     from xlrd import open_workbook
4239.     from xlutils.copy import copy
4240.
4241.     from datetime import date
4242.     from datetime import datetime,timedelta
4243.
```

```
4244.     #============================== VAR GLOBALES ================================

4245.     global pos_Metodologia, pos_Realizado, pos_Comparacion
4246.     global pos_anio_p, pos_mes_p, pos_dia_p
4247.     global pos_anio_c, pos_mes_c, pos_dia_c
4248.     #************************************************************************

4249.     global po_Metodologia, po_Realizado, po_Comparacion
4250.     global po_anio_p, po_mes_p, po_dia_p
4251.     global po_anio_c, po_mes_c, po_dia_c
4252.     #========================================================================

4253.
4254.     doc_Proy       = 'HIS_POT_' + po_anio_p + '.xls'
4255.     doc_Proy_extra = 'HIS_POT_' + str(int(po_anio_p)-1) + '.xls'
4256.     doc_Comp       = 'HIS_POT_' + po_anio_c + '.xls'
4257.     doc_Comp_extra = 'HIS_POT_' + str(int(po_anio_c)-1) + '.xls'
4258.
4259.     # Arreglo de semilla para generar números pseaudo aleatorios para reproducibili
      dad
4260.     #numpy.random.seed(7)
4261.
4262.     # Genera el DataFrame a partir del archivo de datos
4263.     df_Proy       = pd.read_excel(doc_Proy      , sheetname='Hoja1',  header=None)

4264.     df_Proy_extra = pd.read_excel(doc_Proy_extra, sheetname='Hoja1',  header=None)

4265.     df_Comp       = pd.read_excel(doc_Comp      , sheetname='Hoja1',  header=None)

4266.     df_Comp_extra = pd.read_excel(doc_Comp_extra, sheetname='Hoja1',  header=None)

4267.
4268.     # Numero total de datos a proyectar, se toma 24hrs x 7 días
4269.     forecast_out = 24*7
4270.     #************************************************************************

4271.     #************************************************************************

4272.     #************************************************************************

4273.     #************************************************************************

4274.     #************************************************************************

4275.     # Lista de datos de históricos de proyección
4276.     amba_1 = (df_Proy_extra.iloc[:,4].values.tolist())
4277.     toto_1 = (df_Proy_extra.iloc[:,5].values.tolist())
4278.     puyo_1 = (df_Proy_extra.iloc[:,6].values.tolist())
4279.     tena_1 = (df_Proy_extra.iloc[:,7].values.tolist())
4280.     bani_1 = (df_Proy_extra.iloc[:,8].values.tolist())
4281.
4282.     amba_2 = (df_Proy.iloc[:,4].values.tolist())
4283.     toto_2 = (df_Proy.iloc[:,5].values.tolist())
4284.     puyo_2 = (df_Proy.iloc[:,6].values.tolist())
4285.     tena_2 = (df_Proy.iloc[:,7].values.tolist())
4286.     bani_2 = (df_Proy.iloc[:,8].values.tolist())
4287.
4288.     # Elimino la primera fila de cada DF (TEXTOS TÍTULOS)
4289.     amba_1.pop(0)
4290.     toto_1.pop(0)
4291.     puyo_1.pop(0)
4292.     tena_1.pop(0)
4293.     bani_1.pop(0)
4294.
4295.     amba_2.pop(0)
4296.     toto_2.pop(0)
```

```
4297.     puyo_2.pop(0)
4298.     tena_2.pop(0)
4299.     bani_2.pop(0)
4300.
4301.     # Crea nueva lsita con los datos de los dos años el actual y el anterior
4302.     amba_p = amba_1 + amba_2
4303.     toto_p = toto_1 + toto_2
4304.     puyo_p = puyo_1 + puyo_2
4305.     tena_p = tena_1 + tena_2
4306.     bani_p = bani_1 + bani_2
4307.     # Transforma la lista a un DF
4308.     amba_p = pd.DataFrame(amba_p)
4309.     toto_p = pd.DataFrame(toto_p)
4310.     puyo_p = pd.DataFrame(puyo_p)
4311.     tena_p = pd.DataFrame(tena_p)
4312.     bani_p = pd.DataFrame(bani_p)
4313.
4314.     #=============================================================================

4315.     #=============================================================================

4316.     #===>UBICACIONES GENERALES de las posiciones para la proyección
4317.
4318.     #===> Variables internas
4319.     # FECHA SELECCIONADA
4320.     fehca_selec = datetime(int(po_anio_p),(pos_mes_p+1),int(po_dia_p))
4321.     # DÍAS DE PROYECCIÓN
4322.     dias_proy = timedelta(days = 5952/24 )   # 5952 * 24 = 248 días mas un día de s
       alto
4323.
4324.     # FECHA DE INICIO
4325.     fecha_inicio = (fehca_selec + timedelta(days = 7)) - dias_proy
4326.     # FECHA DE FIN
4327.     fecha_fin = fehca_selec + timedelta(days = 7) # 24 horas de un día
4328.     # FECHAS DE INICIO DEl AÑO INICIAL
4329.     enero_1 = datetime(fecha_inicio.year,1,1)
4330.
4331.     # POSICIÓN INICIAL & FINAL
4332.     val_ini = (abs(fecha_inicio - enero_1).days) * 24 # Posición del valor inicial
       5952 DATOS
4333.     val_fin = 3+(abs(fecha_fin    - enero_1).days) * 24 -
       169# Posición del valor final 5952 + 24 = 5976 DATOS
4334.
4335.     # ELIMINA LOS ÚLTIMOS VALORES DEL DF
4336.     amba_p = amba_p[:val_fin]
4337.     toto_p = toto_p[:val_fin]
4338.     puyo_p = puyo_p[:val_fin]
4339.     tena_p = tena_p[:val_fin]
4340.     bani_p = bani_p[:val_fin]
4341.     # ELIMINA LOS PRIMEROS VALORES DEL DF ** Queda un DF con 5952 datos
4342.     amba_p = amba_p[val_ini:]
4343.     toto_p = toto_p[val_ini:]
4344.     puyo_p = puyo_p[val_ini:]
4345.     tena_p = tena_p[val_ini:]
4346.     bani_p = bani_p[val_ini:]
4347.     # REINICIA EL INDEX DE CADA DF   ***** matriz datos de entrenamiento
4348.     amba_p = amba_p.reset_index(drop=True)
4349.     toto_p = toto_p.reset_index(drop=True)
4350.     puyo_p = puyo_p.reset_index(drop=True)
4351.     tena_p = tena_p.reset_index(drop=True)
4352.     bani_p = bani_p.reset_index(drop=True)
4353.
4354.     #*****************************************************************************

4355.     # Hatas aquí, amba_p y las demás son los DF utilizados para la proyección
4356.     # tienen 5952 datos cada DF
```

```
4357.     #*****************************************************************************
4358.     #*****************************************************************************
4359.     #*****************************************************************************
4360.     #*****************************************************************************
4361.
4362.     # Lista de datos de históricos de COMPARACIÓN
4363.     amba_1 = (df_Comp_extra.iloc[:,4].values.tolist())
4364.     toto_1 = (df_Comp_extra.iloc[:,5].values.tolist())
4365.     puyo_1 = (df_Comp_extra.iloc[:,6].values.tolist())
4366.     tena_1 = (df_Comp_extra.iloc[:,7].values.tolist())
4367.     bani_1 = (df_Comp_extra.iloc[:,8].values.tolist())
4368.
4369.     amba_2 = (df_Comp.iloc[:,4].values.tolist())
4370.     toto_2 = (df_Comp.iloc[:,5].values.tolist())
4371.     puyo_2 = (df_Comp.iloc[:,6].values.tolist())
4372.     tena_2 = (df_Comp.iloc[:,7].values.tolist())
4373.     bani_2 = (df_Comp.iloc[:,8].values.tolist())
4374.
4375.     # Elimino la primera fila de cada DF (TEXTOS TÍTULOS)
4376.     amba_1.pop(0)
4377.     toto_1.pop(0)
4378.     puyo_1.pop(0)
4379.     tena_1.pop(0)
4380.     bani_1.pop(0)
4381.
4382.     amba_2.pop(0)
4383.     toto_2.pop(0)
4384.     puyo_2.pop(0)
4385.     tena_2.pop(0)
4386.     bani_2.pop(0)
4387.     # Une las dos listas en una sola
4388.     amba_c = amba_1 + amba_2
4389.     toto_c = toto_1 + toto_2
4390.     puyo_c = puyo_1 + puyo_2
4391.     tena_c = tena_1 + tena_2
4392.     bani_c = bani_1 + bani_2
4393.     # Transforma la lista al tipo DF
4394.     #amba_c = pd.DataFrame(amba_c)
4395.     #toto_c = pd.DataFrame(toto_c)
4396.     #puyo_c = pd.DataFrame(puyo_c)
4397.     #tena_c = pd.DataFrame(tena_c)
4398.     #bani_c = pd.DataFrame(bani_c)
4399.     #=============================================================================
4400.     #=============================================================================
4401.     #===>UBICACIONES GENERALES de las posiciones para la COMPARACIÓN
4402.
4403.
4404.     #===> Variables internas
4405.     # FECHA SELECCIONADA
4406.     fehca_selec = datetime(int(po_anio_c),(pos_mes_c+1),int(po_dia_c))
4407.     # DÍAS DE PROYECCIÓN
4408.     dias_proy_c = timedelta(days = 7)
4409.
4410.     # FECHA DE INICIO
4411.     fecha_inicio = (fehca_selec - dias_proy_c)
4412.     # FECHA DE FIN
4413.     fecha_fin = fehca_selec
4414.     # FECHAS DE INICIO DEl AÑO INICIAL
4415.     enero_1 = datetime((int(po_anio_c)-1),1,1)
4416.
```

```
4417.     # POSICIÓN INICIAL & FINAL
4418.     val_ini = (abs(fecha_inicio - enero_1).days) * 24 # Posición del valor inicial
       5952 DATOS
4419.     val_fin = (abs(fecha_fin    - enero_1).days) * 24 # Posición del valor final 59
       52 + 24 = 5976 DATOS
4420.
4421.     # ELIMINA LOS ÚLTIMOS VALORES DEL DF
4422.     amba_c = amba_c[:val_fin]
4423.     toto_c = toto_c[:val_fin]
4424.     puyo_c = puyo_c[:val_fin]
4425.     tena_c = tena_c[:val_fin]
4426.     bani_c = bani_c[:val_fin]
4427.     # ELIMINA LOS PRIMEROS VALORES DEL DF ** Queda un DF con 5952 datos
4428.     amba_c = amba_c[val_ini:]
4429.     toto_c = toto_c[val_ini:]
4430.     puyo_c = puyo_c[val_ini:]
4431.     tena_c = tena_c[val_ini:]
4432.     bani_c = bani_c[val_ini:]
4433.     # REINICIA EL INDEX DE CADA DF   ***** matriz datos de comparación
4434.     CompAmba = amba_c
4435.     CompToto = toto_c
4436.     CompPuyo = puyo_c
4437.     CompTena = tena_c
4438.     CompBani = bani_c
4439.
4440.
4441.     # Datos de comparación
4442.     #CompAmba = (amba_p[0][-forecast_out:]).reset_index(drop=True)
4443.     #CompToto = (toto_p[0][-forecast_out:]).reset_index(drop=True)
4444.     #CompPuyo = (puyo_p[0][-forecast_out:]).reset_index(drop=True)
4445.     #CompTena = (tena_p[0][-forecast_out:]).reset_index(drop=True)
4446.     #CompBani = (bani_p[0][-forecast_out:]).reset_index(drop=True)
4447.
4448.
4449.     #*****************************************************************************
4450.     #*****************************************************************************
4451.     #*****************************************************************************
4452.     #*****************************************************************************
4453.     #*****************************************************************************
4454.
4455.     #=============================================================================
4456.     #=============================================================================
4457.
4458.     # Obtiene los valores del DF
4459.     dataset_amba = amba_p.values
4460.     dataset_toto = toto_p.values
4461.     dataset_puyo = puyo_p.values
4462.     dataset_tena = tena_p.values
4463.     dataset_bani = bani_p.values
4464.
4465.     # Transforma los valores del DS al tipo float32
4466.     dataset_amba = dataset_amba.astype('float32')
4467.     dataset_toto = dataset_toto.astype('float32')
4468.     dataset_puyo = dataset_puyo.astype('float32')
4469.     dataset_tena = dataset_tena.astype('float32')
4470.     dataset_bani = dataset_bani.astype('float32')
4471.
4472.
4473.     # Obtiene la media del conjunto de datos del DF
```

```
4474.     avgv_amba = amba_p.mean()
4475.     avgv_toto = toto_p.mean()
4476.     avgv_puyo = puyo_p.mean()
4477.     avgv_tena = tena_p.mean()
4478.     avgv_bani = bani_p.mean()
4479.
4480.     # Se dividide en datos de entrenamiento y de prueba:
4481.     # Tamaño del conjunto de datos de entrenamiento
4482.     #train_size = int(len(dataset) * 0.67) # Usa el 67% del total de los datos
4483.     train_size = int(len(dataset_amba) - 179) #168
4484.     # Tamaño del conjunto de datos de prueba
4485.     test_size = len(dataset_amba) - train_size # Resulta el 33% de datos restantes
4486.
4487.     # Se genera las columnas de datos de entrenamiento y de prueba
4488.     train_amba, test_amba = dataset_amba[0:train_size,:], dataset_amba[train_size:len(dataset_amba),:]
4489.     train_toto, test_toto = dataset_toto[0:train_size,:], dataset_toto[train_size:len(dataset_toto),:]
4490.     train_puyo, test_puyo = dataset_puyo[0:train_size,:], dataset_puyo[train_size:len(dataset_puyo),:]
4491.     train_tena, test_tena = dataset_tena[0:train_size,:], dataset_tena[train_size:len(dataset_tena),:]
4492.     train_bani, test_bani = dataset_bani[0:train_size,:], dataset_bani[train_size:len(dataset_bani),:]
4493.     #================================================================================
4494.
4495.     # Convierte una matriz de valores en una matriz de conjunto de datos
4496.     def create_dataset_amba(dataset_amba, look_back=1):
4497.             dataX_amba, dataY_amba = [], []
4498.             for i in range(len(dataset_amba)-look_back-1):
4499.                     a_amba = dataset_amba[i:(i+look_back), 0]
4500.                     dataX_amba.append(a_amba)
4501.                     dataY_amba.append(dataset_amba[i + look_back, 0])
4502.             return numpy.array(dataX_amba), numpy.array(dataY_amba)
4503.
4504.     # reshape into X=t and Y=t+1
4505.     look_back = 10
4506.
4507.     trainX_amba, trainY_amba = create_dataset_amba(train_amba, look_back)
4508.     testX_amba, testY_amba = create_dataset_amba(test_amba, look_back)
4509.     #================================================================================
4510.
4511.     # Convierte una matriz de valores en una matriz de conjunto de datos
4512.     def create_dataset_toto(dataset_toto, look_back=1):
4513.             dataX_toto, dataY_toto = [], []
4514.             for i in range(len(dataset_toto)-look_back-1):
4515.                     a_toto = dataset_toto[i:(i+look_back), 0]
4516.                     dataX_toto.append(a_toto)
4517.                     dataY_toto.append(dataset_toto[i + look_back, 0])
4518.             return numpy.array(dataX_toto), numpy.array(dataY_toto)
4519.
4520.     # reshape into X=t and Y=t+1
4521.     look_back = 10
4522.
4523.     trainX_toto, trainY_toto = create_dataset_toto(train_toto, look_back)
4524.     testX_toto, testY_toto = create_dataset_toto(test_toto, look_back)
4525.     #================================================================================
4526.
4527.     # Convierte una matriz de valores en una matriz de conjunto de datos
4528.     def create_dataset_puyo(dataset_puyo, look_back=1):
4529.             dataX_puyo, dataY_puyo = [], []
4530.             for i in range(len(dataset_puyo)-look_back-1):
```

```
                a_puyo = dataset_puyo[i:(i+look_back), 0]
                dataX_puyo.append(a_puyo)
                dataY_puyo.append(dataset_puyo[i + look_back, 0])
        return numpy.array(dataX_puyo), numpy.array(dataY_puyo)

# reshape into X=t and Y=t+1
look_back = 10

trainX_puyo, trainY_puyo = create_dataset_puyo(train_puyo, look_back)
testX_puyo, testY_puyo = create_dataset_puyo(test_puyo, look_back)
#==============================================================================

# Convierte una matriz de valores en una matriz de conjunto de datos
def create_dataset_tena(dataset_tena, look_back=1):
        dataX_tena, dataY_tena = [], []
        for i in range(len(dataset_tena)-look_back-1):
                a_tena = dataset_tena[i:(i+look_back), 0]
                dataX_tena.append(a_tena)
                dataY_tena.append(dataset_tena[i + look_back, 0])
        return numpy.array(dataX_tena), numpy.array(dataY_tena)

# reshape into X=t and Y=t+1
look_back = 10

trainX_tena, trainY_tena = create_dataset_tena(train_tena, look_back)
testX_tena, testY_tena = create_dataset_tena(test_tena, look_back)
#==============================================================================

# Convierte una matriz de valores en una matriz de conjunto de datos
def create_dataset_bani(dataset_bani, look_back=1):
        dataX_bani, dataY_bani = [], []
        for i in range(len(dataset_bani)-look_back-1):
                a_bani = dataset_bani[i:(i+look_back), 0]
                dataX_bani.append(a_bani)
                dataY_bani.append(dataset_bani[i + look_back, 0])
        return numpy.array(dataX_bani), numpy.array(dataY_bani)

# reshape into X=t and Y=t+1
look_back = 10

trainX_bani, trainY_bani = create_dataset_bani(train_bani, look_back)
testX_bani, testY_bani = create_dataset_bani(test_bani, look_back)
#==============================================================================


# Crea y ajusta el modelo de perceptron multicapa
model = Sequential()
model.add(Dense(8, input_dim=look_back, activation='relu'))
model.add(Dense(1))
model.compile(loss='mean_squared_error', optimizer='adam')

#===> VARIAR EL NÚMERO DE CAPAS PARA MEJORAR PRESICIÓN, LAYER CAPAS
#/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*
#/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*
#/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*
epoca = 10
#/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*
#/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*
```

```
    #/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*

    print('=======>  Iteraciones para la S/E AMBATO: ')
    model.fit(trainX_amba, trainY_amba, epochs=epoca, batch_size=2, verbose=2)
    print('=======>  Iteraciones para la S/E TOTORAS: ')
    model.fit(trainX_toto, trainY_toto, epochs=epoca, batch_size=2, verbose=2)
    print('=======>  Iteraciones para la S/E PUYO: ')
    model.fit(trainX_puyo, trainY_puyo, epochs=epoca, batch_size=2, verbose=2)
    print('=======>  Iteraciones para la S/E TENA: ')
    model.fit(trainX_tena, trainY_tena, epochs=epoca, batch_size=2, verbose=2)
    print('=======>  Iteraciones para la S/E BAÑOS: ')
    model.fit(trainX_bani, trainY_bani, epochs=epoca, batch_size=2, verbose=2)

    # Estimar el rendimiento del modelo
    trainScore_amba = model.evaluate(trainX_amba, trainY_amba, verbose=0)
    trainScore_toto = model.evaluate(trainX_toto, trainY_toto, verbose=0)
    trainScore_puyo = model.evaluate(trainX_puyo, trainY_puyo, verbose=0)
    trainScore_tena = model.evaluate(trainX_tena, trainY_tena, verbose=0)
    trainScore_bani = model.evaluate(trainX_bani, trainY_bani, verbose=0)
    #print(' ')
    #print(' ')
    #print('Train Score: %.2f MSE (%.2f RMSE)' % (trainScore, math.sqrt(trainScore)
  ))
    testScore_amba = model.evaluate(testX_amba, testY_amba, verbose=0)
    testScore_toto = model.evaluate(testX_toto, testY_toto, verbose=0)
    testScore_puyo = model.evaluate(testX_puyo, testY_puyo, verbose=0)
    testScore_tena = model.evaluate(testX_tena, testY_tena, verbose=0)
    testScore_bani = model.evaluate(testX_bani, testY_bani, verbose=0)
    #print('Test Score: %.2f MSE (%.2f RMSE)' % (testScore, math.sqrt(testScore)))

    print(' ')
    print(' ')
    newV_amba = avgv_amba + math.sqrt(trainScore_amba)
    newV_toto = avgv_toto + math.sqrt(trainScore_toto)
    newV_puyo = avgv_puyo + math.sqrt(trainScore_puyo)
    newV_tena = avgv_tena + math.sqrt(trainScore_tena)
    newV_bani = avgv_bani + math.sqrt(trainScore_bani)

    acc_amba = 100-(((newV_amba - avgv_amba)/(avgv_amba))*100)
    acc_toto = 100-(((newV_toto - avgv_toto)/(avgv_toto))*100)
    acc_puyo = 100-(((newV_puyo - avgv_puyo)/(avgv_puyo))*100)
    acc_tena = 100-(((newV_tena - avgv_tena)/(avgv_tena))*100)
    acc_bani = 100-(((newV_bani - avgv_bani)/(avgv_bani))*100)
    #print(avgv)
    #print('Mean Absolute Error of',newV)
    print('* Porcentaje de Exactitud de la predicción por MLP : ')
    print('=> S/E AMBATO  :' ,acc_amba[0])
    print('=> S/E TOTORAS :' ,acc_toto[0])
    print('=> S/E PUYO    :' ,acc_puyo[0])
    print('=> S/E TENA    :' ,acc_tena[0])
    print('=> S/E BAÑOS   :' ,acc_bani[0])

    # Generar predicciones para el entrenamiento.
    #trainPredict_amba = model.predict(trainX_amba)
    ProyAmba  = model.predict(testX_amba)

    #trainPredict_toto = model.predict(trainX_toto)
    ProyToto  = model.predict(testX_toto)

    #trainPredict_puyo = model.predict(trainX_puyo)
    ProyPuyo  = model.predict(testX_puyo)

    #trainPredict_tena = model.predict(trainX_tena)
    ProyTena  = model.predict(testX_tena)

    #trainPredict_bani = model.predict(trainX_bani)
```

```
4652.     ProyBani  = model.predict(testX_bani)
4653.
4654.
4655.     #=============================================================================
4656.     #================================ GRÁFICAS ===================================
4657.     #=============================================================================
4658.     if po_Comparacion == 'SI':
4659.     #===> Creamos una lista tipo entero para relacionar con las etiquetas
4660.         can_datos = []
4661.         for i in range(7*24):
4662.             if i%2!=1:
4663.                 can_datos.append(i)
4664.     #===> Creamos una lista con los números de horas del día seleccionado (etiqueta
        s)
4665.         horas_dia = []
4666.         horas_str = []
4667.         for i in range (7):
4668.             for i in range (1,25):
4669.                 if i%2!=0:
4670.                     horas_dia.append(i)
4671.         for i in range (len(horas_dia)):
4672.             horas_str.append(str(horas_dia[i]))
4673.
4674.     #===> Tamaño de la ventana de la gráfica
4675.         plt.subplots(figsize=(15, 8))
4676.
4677.     #===> Título general superior
4678.         plt.suptitle(u' PROYECCIÓN SEMANAL DE CARGA\n MULTILAYER PERCEPTRON ',fonts
        ize=14, fontweight='bold')
4679.
4680.         plt.subplot(5,1,1)
4681.         plt.plot(CompAmba,'blue', label = 'Comparación')
4682.         plt.plot(ProyAmba,'#DCD037', label = 'Proyección')
4683.         plt.legend(loc='upper left')
4684.         plt.xlabel(u'TIEMPO [ días ]')
4685.         plt.ylabel(u'S/E AMBATO\n\nCARGA  [ kW ]')
4686.         plt.grid(color='#C8C8C8', linestyle='-', linewidth=0.5)
4687.         plt.xticks(can_datos, horas_str, size = 5, color = 'b', rotation = 90)
4688.
4689.         plt.subplot(5,1,2)
4690.         plt.plot(CompToto,'blue', label = 'Comparación')
4691.         plt.plot(ProyToto,'#CD336F', label = 'Proyección')
4692.         plt.legend(loc='upper left')
4693.         plt.xlabel(u'TIEMPO [ días ]')
4694.         plt.ylabel(u'S/E TOTORAS\n\nCARGA  [ kW ]')
4695.         plt.grid(color='#C8C8C8', linestyle='-', linewidth=0.5)
4696.         plt.xticks(can_datos, horas_str, size = 5, color = 'b', rotation = 90)
4697.
4698.         plt.subplot(5,1,3)
4699.         plt.plot(CompPuyo,'blue', label = 'Comparación')
4700.         plt.plot(ProyPuyo,'#349A9D', label = 'Proyección')
4701.         plt.legend(loc='upper left')
4702.         plt.xlabel(u'TIEMPO [ días ]')
4703.         plt.ylabel(u'S/E PUYO\n\nCARGA  [ kW ]')
4704.         plt.grid(color='#C8C8C8', linestyle='-', linewidth=0.5)
4705.         plt.xticks(can_datos, horas_str, size = 5, color = 'b', rotation = 90)
4706.
4707.         plt.subplot(5,1,4)
4708.         plt.plot(CompTena,'blue', label = 'Comparación')
4709.         plt.plot(ProyTena,'#CC8634', label = 'Proyección')
4710.         plt.legend(loc='upper left')
4711.         plt.xlabel(u'TIEMPO [ días ]')
4712.         plt.ylabel(u'S/E TENA\n\nCARGA  [ kW ]')
```

```
4713.         plt.grid(color='#C8C8C8', linestyle='-', linewidth=0.5)
4714.         plt.xticks(can_datos, horas_str, size = 5, color = 'b', rotation = 90)
4715.
4716.         plt.subplot(5,1,5)
4717.         plt.plot(CompBani,'blue', label = 'Comparación')
4718.         plt.plot(ProyBani,'#4ACB71', label = 'Proyección')
4719.         plt.legend(loc='upper left')
4720.         plt.xlabel(u'TIEMPO [ días ]')
4721.         plt.ylabel(u'S/E BAÑOS\n\nCARGA  [ kW ]')
4722.         plt.grid(color='#C8C8C8', linestyle='-', linewidth=0.5)
4723.         plt.xticks(can_datos, horas_str, size = 5, color = 'b', rotation = 90)
4724.
4725.     #=========> Fechas de salida:
4726.     sem_ini = datetime(int(po_anio_p),(pos_mes_p+1),int(po_dia_p))
4727.     sem_fin = sem_ini + timedelta(days = 6)
4728.
4729.     #================================================================================
4730.     #======================== RUTINA GENERA EXCEL ===================================
4731.     #================================================================================
4732.
4733.     ProyAmba = ProyAmba.astype('float64')
4734.     ProyToto = ProyToto.astype('float64')
4735.     ProyPuyo = ProyPuyo.astype('float64')
4736.     ProyTena = ProyTena.astype('float64')
4737.     ProyBani = ProyBani.astype('float64')
4738.
4739.     df = pd.DataFrame([' '])
4740.
4741.     writer = ExcelWriter('04_PROYECCION_MLP.xls')
4742.     df.to_excel(writer, 'Salida_Proyección_CECON', index=False)
4743.     df.to_excel(writer, 'Salida_Comparación_CECON', index=False)
4744.     df.to_excel(writer, 'Salida_ERRORES_CECON', index=False)
4745.     writer.save()
4746.
4747.     #abre el archivo de excel plantilla
4748.     rb = open_workbook('04_PROYECCION_MLP.xls')
4749.     #crea una copia del archivo plantilla
4750.     wb = copy(rb)
4751.     #se ingresa a la hoja 1 de la copia del archivo excel
4752.     ws = wb.get_sheet(0)
4753.     ws_c = wb.get_sheet(1)
4754.     ws_e = wb.get_sheet(2)
4755.     #============================Hoja 1 Proyección ================================
4756.     #ws.write(0,0,'MES')
4757.     #ws.write(0,1,'DÍA')
4758.     #ws.write(0,2,'#')
4759.     ws.write(0,0,'METODOLOGÍA PERCEPTRÓN MULTI - CAPA  (M.L.P.)')
4760.     ws.write(1,0,'PROYECCIÓN SEMANAL DE CARGA')
4761.     ws.write(2,0,'Semana de Proyección: del '+str(sem_ini.day)+'/'
4762.              +str(sem_ini.month) +'/'+str(sem_ini.year)+'\t al '+
4763.              str(sem_fin.day)+'/'+str(sem_fin.month)+'/'+str(sem_fin.year))
4764.
4765.     ws.write(3,0,'Realizado por: '+str(po_Realizado))
4766.
4767.     # Define lista de títulos
4768.     titulos = ['HORA','S/E AMBATO','S/E TOTORAS','S/E PUYO','S/E TENA','S/E BAÑOS']
4769.
4770.     for i in range(len(titulos)):
4771.         ws.write(9,i,titulos[i])
4772.
4773.     Aux = 10
```

```
4774.     Aux2 = 0
4775.     for i in range (7):
4776.         for j in range (24):
4777.             ws.write(Aux,0,j+1)
4778.             ws.write(Aux,1,ProyAmba[j + Aux2][0])
4779.             ws.write(Aux,2,ProyToto[j + Aux2][0])
4780.             ws.write(Aux,3,ProyPuyo[j + Aux2][0])
4781.             ws.write(Aux,4,ProyTena[j + Aux2][0])
4782.             ws.write(Aux,5,ProyBani[j + Aux2][0])
4783.             Aux = Aux + 1
4784.         Aux2 = 24 * (i+1)
4785.         Aux = Aux + 1
4786.     #==========================Hoja 2 Comparación ================================
4787.
4788.     #===> Fecha
4789.     ws_c.write(0,0,'Semana de Proyección: del '+str(sem_ini.day)+'/'
4790.                +str(sem_ini.month) +'/'+str(sem_ini.year)+'\t al '+
4791.                str(sem_fin.day)+'/'+str(sem_fin.month)+'/'+str(sem_fin.year))
4792.     ws_c.write(2,0,'PROYECCIÓN SEMANAL')
4793.
4794.     for i in range(len(titulos)):
4795.         ws_c.write(9,i,titulos[i])
4796.
4797.     Aux = 10
4798.     Aux2 = 0
4799.     for i in range (7):
4800.         for j in range (24):
4801.             ws_c.write(Aux,0,j+1)
4802.             ws_c.write(Aux,1,ProyAmba[j + Aux2][0])
4803.             ws_c.write(Aux,2,ProyToto[j + Aux2][0])
4804.             ws_c.write(Aux,3,ProyPuyo[j + Aux2][0])
4805.             ws_c.write(Aux,4,ProyTena[j + Aux2][0])
4806.             ws_c.write(Aux,5,ProyBani[j + Aux2][0])
4807.             Aux = Aux + 1
4808.         Aux2 = 24 * (i+1)
4809.         Aux = Aux + 1
4810.
4811.     #ws_c.write(0,6,po_dia_c+'/'+po_mes_c+'/'+po_anio_c)
4812.     ws_c.write(2,6,'SEMANA DE COMPARACIÓN')
4813.
4814.     for i in range(len(titulos)):
4815.         ws_c.write(9,i+6,titulos[i])
4816.
4817.     Aux = 10
4818.     Aux2 = 0
4819.     for i in range (7):
4820.         for j in range (24):
4821.             ws_c.write(Aux,6,j+1)
4822.             ws_c.write(Aux,7,CompAmba[j + Aux2])
4823.             ws_c.write(Aux,8,CompToto[j + Aux2])
4824.             ws_c.write(Aux,9,CompPuyo[j + Aux2])
4825.             ws_c.write(Aux,10,CompTena[j + Aux2])
4826.             ws_c.write(Aux,11,CompBani[j + Aux2])
4827.             Aux = Aux + 1
4828.         Aux2 = 24 * (i+1)
4829.         Aux = Aux + 1
4830.
4831.
4832.     # ==========================   Cálculo de errores    ==========================
4833.
4834.     ws_c.write(2,12,'CÁLCULO DE ERRORES')
4835.     ws_c.write(3,12,'PORCENTAJE DE ERROR MEDIO ABSOLUTO (PEMA)')
4836.
4837.     for i in range(len(titulos)):
```

```
4838.         ws_c.write(9,i+12,titulos[i])
4839.
4840.
4841.     errAmba = []
4842.     errToto = []
4843.     errPuyo = []
4844.     errTena = []
4845.     errBani = []
4846.
4847.     for i in range(len(ProyAmba)):
4848.         errAmba.append((abs( ProyAmba[i][0] - CompAmba[i] ) / CompAmba[i] )*100)
4849.         errToto.append((abs( ProyToto[i][0] - CompToto[i] ) / CompToto[i] )*100)
4850.         errPuyo.append((abs( ProyPuyo[i][0] - CompPuyo[i] ) / CompPuyo[i] )*100)
4851.         errTena.append((abs( ProyTena[i][0] - CompTena[i] ) / CompTena[i] )*100)
4852.         errBani.append((abs( ProyBani[i][0] - CompBani[i] ) / CompBani[i] )*100)
4853.
4854.     Aux = 10
4855.     Aux2 = 0
4856.     for i in range (7):
4857.         for j in range (24):
4858.             ws_c.write(Aux,12,j+1)
4859.             ws_c.write(Aux,13,errAmba[j + Aux2])
4860.             ws_c.write(Aux,14,errToto[j + Aux2])
4861.             ws_c.write(Aux,15,errPuyo[j + Aux2])
4862.             ws_c.write(Aux,16,errTena[j + Aux2])
4863.             ws_c.write(Aux,17,errBani[j + Aux2])
4864.             Aux = Aux + 1
4865.         Aux2 = 24 * (i+1)
4866.         Aux = Aux + 1
4867.     # Suma de los valores de las listas
4868.     SumAmba = 0
4869.     SumToto = 0
4870.     SumPuyo = 0
4871.     SumTena = 0
4872.     SumBani = 0
4873.     for i in range (len(ProyAmba)):
4874.         SumAmba = errAmba[i] + SumAmba
4875.         SumToto = errToto[i] + SumToto
4876.         SumPuyo = errPuyo[i] + SumPuyo
4877.         SumTena = errTena[i] + SumTena
4878.         SumBani = errBani[i] + SumBani
4879.     # Almaceno en una lista los resultados de las sumas
4880.     Sumas = [SumAmba, SumToto, SumPuyo, SumTena, SumBani]
4881.
4882.     # Imprime los resultados en la fila correspondiente
4883.     ws_c.write(Aux+1,11,'SUMATORIA TOTAL')
4884.     for i in range (len(Sumas)):
4885.          ws_c.write(Aux+1,i+13,Sumas[i])
4886.
4887.     # Cálculo de los promedios de las sumas
4888.     ws_c.write(Aux+2,11,'ERROR MEDIO ABSOLUTO')
4889.     for i in range(len(Sumas)):
4890.         ws_c.write(Aux+2,i+13,(Sumas[i]/len(errAmba)))
4891.
4892.     # Cálculo de la exactitud de la proyección
4893.     ws_c.write(Aux+3,11,'EXACTITUD DE LA PROYECCIÓN')
4894.     for i in range(len(Sumas)):
4895.         ws_c.write(Aux+3,i+13,(100-(Sumas[i]/len(errAmba))))
4896.
4897.
4898.     for i in range(len(titulos)-1):
4899.         ws_c.write(Aux,i+13,titulos[i+1])
4900.
4901.     # ==========================   SOLO   errores    ==========================
4902.
4903.     # ==================================> cálculo de et = Comp - Proy
```

```
    for i in range(len(titulos)-1):
        ws_e.write(9,i,titulos[i+1])
    ws_e.write(7,0,'et = Comp - Proy')

    et_Amba = []
    et_Toto = []
    et_Puyo = []
    et_Tena = []
    et_Bani = []

    for i in range (len(ProyAmba)):
        et_Amba.append(CompAmba[i] - ProyAmba[i][0])
        et_Toto.append(CompToto[i] - ProyToto[i][0])
        et_Puyo.append(CompPuyo[i] - ProyPuyo[i][0])
        et_Tena.append(CompTena[i] - ProyTena[i][0])
        et_Bani.append(CompBani[i] - ProyBani[i][0])

        ws_e.write(i+10,0, (et_Amba[i]))
        ws_e.write(i+10,1, (et_Toto[i]))
        ws_e.write(i+10,2, (et_Puyo[i]))
        ws_e.write(i+10,3, (et_Tena[i]))
        ws_e.write(i+10,4, (et_Bani[i]))

    # ==================================> cálculo de abs(et) = abs(Comp - Proy)
    for i in range(len(titulos)-1):
        ws_e.write(9,i+6,titulos[i+1])
    ws_e.write(7,6,'abs(et) = abs(Comp - Proy)')

    abs_et_Amba = []
    abs_et_Toto = []
    abs_et_Puyo = []
    abs_et_Tena = []
    abs_et_Bani = []

    for i in range (len(ProyAmba)):
        abs_et_Amba.append(abs(et_Amba[i]))
        abs_et_Toto.append(abs(et_Toto[i]))
        abs_et_Puyo.append(abs(et_Puyo[i]))
        abs_et_Tena.append(abs(et_Tena[i]))
        abs_et_Bani.append(abs(et_Bani[i]))

        ws_e.write(i+10,6,  (abs_et_Amba[i]))
        ws_e.write(i+10,7,  (abs_et_Toto[i]))
        ws_e.write(i+10,8,  (abs_et_Puyo[i]))
        ws_e.write(i+10,9,  (abs_et_Tena[i]))
        ws_e.write(i+10,10, (abs_et_Bani[i]))

    # ==================================> cálculo de et^2
    for i in range(len(titulos)-1):
        ws_e.write(9,i+12,titulos[i+1])
    ws_e.write(7,12,'et^2')

    et_Amba2 = []
    et_Toto2 = []
    et_Puyo2 = []
    et_Tena2 = []
    et_Bani2 = []

    for i in range (len(ProyAmba)):
        et_Amba2.append((et_Amba[i])**2)
        et_Toto2.append((et_Toto[i])**2)
        et_Puyo2.append((et_Puyo[i])**2)
        et_Tena2.append((et_Tena[i])**2)
        et_Bani2.append((et_Bani[i])**2)

        ws_e.write(i+10,12, (et_Amba2[i]))
```

```
4970.          ws_e.write(i+10,13, (et_Toto2[i]))
4971.          ws_e.write(i+10,14, (et_Puyo2[i]))
4972.          ws_e.write(i+10,15, (et_Tena2[i]))
4973.          ws_e.write(i+10,16, (et_Bani2[i]))
4974.
4975.      # ==================================> cálculo de abs(et) / Comp
4976.      for i in range(len(titulos)-1):
4977.          ws_e.write(9,i+18,titulos[i+1])
4978.      ws_e.write(7,18,'abs(et) / Comp')
4979.
4980.      d1_Amba = []
4981.      d1_Toto = []
4982.      d1_Puyo = []
4983.      d1_Tena = []
4984.      d1_Bani = []
4985.
4986.      for i in range (len(ProyAmba)):
4987.          d1_Amba.append(abs_et_Amba[i] / CompAmba[i])
4988.          d1_Toto.append(abs_et_Toto[i] / CompToto[i])
4989.          d1_Puyo.append(abs_et_Puyo[i] / CompPuyo[i])
4990.          d1_Tena.append(abs_et_Tena[i] / CompTena[i])
4991.          d1_Bani.append(abs_et_Bani[i] / CompBani[i])
4992.
4993.          ws_e.write(i+10,18, (d1_Amba[i]))
4994.          ws_e.write(i+10,19, (d1_Toto[i]))
4995.          ws_e.write(i+10,20, (d1_Puyo[i]))
4996.          ws_e.write(i+10,21, (d1_Tena[i]))
4997.          ws_e.write(i+10,22, (d1_Bani[i]))
4998.
4999.      # ==================================> cálculo de et / Comp
5000.      for i in range(len(titulos)-1):
5001.          ws_e.write(9,i+24,titulos[i+1])
5002.      ws_e.write(7,24,'et / Comp')
5003.
5004.      d2_Amba = []
5005.      d2_Toto = []
5006.      d2_Puyo = []
5007.      d2_Tena = []
5008.      d2_Bani = []
5009.
5010.      for i in range (len(ProyAmba)):
5011.          d2_Amba.append(et_Amba[i] / CompAmba[i])
5012.          d2_Toto.append(et_Toto[i] / CompToto[i])
5013.          d2_Puyo.append(et_Puyo[i] / CompPuyo[i])
5014.          d2_Tena.append(et_Tena[i] / CompTena[i])
5015.          d2_Bani.append(et_Bani[i] / CompBani[i])
5016.
5017.          ws_e.write(i+10,24, (d2_Amba[i]))
5018.          ws_e.write(i+10,25, (d2_Toto[i]))
5019.          ws_e.write(i+10,26, (d2_Puyo[i]))
5020.          ws_e.write(i+10,27, (d2_Tena[i]))
5021.          ws_e.write(i+10,28, (d2_Bani[i]))
5022.
5023.      ws_e.write(0,0, 'INDICADORES')
5024.      # ==================================> Cálculo DAM
5025.      DAM_Amba = 0
5026.      DAM_Toto = 0
5027.      DAM_Puyo = 0
5028.      DAM_Tena = 0
5029.      DAM_Bani = 0
5030.
5031.      for i in range (len(ProyAmba)):
5032.          DAM_Amba = abs_et_Amba[i] + DAM_Amba
5033.          DAM_Toto = abs_et_Toto[i] + DAM_Toto
5034.          DAM_Puyo = abs_et_Puyo[i] + DAM_Puyo
5035.          DAM_Tena = abs_et_Tena[i] + DAM_Tena
```

```
5036.         DAM_Bani = abs_et_Bani[i] + DAM_Bani
5037.
5038.     DAM_Amba = DAM_Amba / (len(ProyAmba))
5039.     DAM_Toto = DAM_Toto / (len(ProyAmba))
5040.     DAM_Puyo = DAM_Puyo / (len(ProyAmba))
5041.     DAM_Tena = DAM_Tena / (len(ProyAmba))
5042.     DAM_Bani = DAM_Bani / (len(ProyAmba))
5043.
5044.     ws_e.write(1,0, ('DAM'))
5045.     DAM = [DAM_Amba,DAM_Toto,DAM_Puyo,DAM_Tena,DAM_Bani]
5046.     for i in range(len(DAM)):
5047.         ws_e.write(1,i+1,DAM[i])
5048.
5049.
5050.     # ==================================> Cálculo EMC
5051.     EMC_Amba = 0
5052.     EMC_Toto = 0
5053.     EMC_Puyo = 0
5054.     EMC_Tena = 0
5055.     EMC_Bani = 0
5056.
5057.     for i in range (len(ProyAmba)):
5058.         EMC_Amba = et_Amba2[i] + EMC_Amba
5059.         EMC_Toto = et_Toto2[i] + EMC_Toto
5060.         EMC_Puyo = et_Puyo2[i] + EMC_Puyo
5061.         EMC_Tena = et_Tena2[i] + EMC_Tena
5062.         EMC_Bani = et_Bani2[i] + EMC_Bani
5063.
5064.     EMC_Amba = EMC_Amba / (len(ProyAmba))
5065.     EMC_Toto = EMC_Toto / (len(ProyAmba))
5066.     EMC_Puyo = EMC_Puyo / (len(ProyAmba))
5067.     EMC_Tena = EMC_Tena / (len(ProyAmba))
5068.     EMC_Bani = EMC_Bani / (len(ProyAmba))
5069.
5070.     ws_e.write(2,0, ('EMC'))
5071.     EMC = [EMC_Amba,EMC_Toto,EMC_Puyo,EMC_Tena,EMC_Bani]
5072.
5073.     for i in range(len(EMC)):
5074.         ws_e.write(2,i+1,EMC[i])
5075.
5076.     # ==================================> Cálculo PEMA
5077.     PEMA_Amba = 0
5078.     PEMA_Toto = 0
5079.     PEMA_Puyo = 0
5080.     PEMA_Tena = 0
5081.     PEMA_Bani = 0
5082.
5083.     for i in range (len(ProyAmba)):
5084.         PEMA_Amba = (abs_et_Amba[i] / CompAmba [i]) + PEMA_Amba
5085.         PEMA_Toto = (abs_et_Toto[i] / CompToto [i]) + PEMA_Toto
5086.         PEMA_Puyo = (abs_et_Puyo[i] / CompPuyo [i]) + PEMA_Puyo
5087.         PEMA_Tena = (abs_et_Tena[i] / CompTena [i]) + PEMA_Tena
5088.         PEMA_Bani = (abs_et_Bani[i] / CompBani [i]) + PEMA_Bani
5089.
5090.     PEMA_Amba = (PEMA_Amba / (len(ProyAmba))) *100
5091.     PEMA_Toto = (PEMA_Toto / (len(ProyAmba))) *100
5092.     PEMA_Puyo = (PEMA_Puyo / (len(ProyAmba))) *100
5093.     PEMA_Tena = (PEMA_Tena / (len(ProyAmba))) *100
5094.     PEMA_Bani = (PEMA_Bani / (len(ProyAmba))) *100
5095.
5096.     ws_e.write(3,0, ('PEMA'))
5097.     PEMA = [PEMA_Amba,PEMA_Toto,PEMA_Puyo,PEMA_Tena,PEMA_Bani]
5098.
5099.     for i in range(len(PEMA)):
5100.         ws_e.write(3,i+1,PEMA[i])
5101.
```

```
5102.     # ==================================> Cálculo PME
5103.     PME_Amba = 0
5104.     PME_Toto = 0
5105.     PME_Puyo = 0
5106.     PME_Tena = 0
5107.     PME_Bani = 0
5108.
5109.     for i in range (len(ProyAmba)):
5110.         PME_Amba = (et_Amba[i] / CompAmba [i]) + PME_Amba
5111.         PME_Toto = (et_Toto[i] / CompToto [i]) + PME_Toto
5112.         PME_Puyo = (et_Puyo[i] / CompPuyo [i]) + PME_Puyo
5113.         PME_Tena = (et_Tena[i] / CompTena [i]) + PME_Tena
5114.         PME_Bani = (et_Bani[i] / CompBani [i]) + PME_Bani
5115.
5116.     PME_Amba = (PME_Amba / (len(ProyAmba))) *100
5117.     PME_Toto = (PME_Toto / (len(ProyAmba))) *100
5118.     PME_Puyo = (PME_Puyo / (len(ProyAmba))) *100
5119.     PME_Tena = (PME_Tena / (len(ProyAmba))) *100
5120.     PME_Bani = (PME_Bani / (len(ProyAmba))) *100
5121.
5122.     ws_e.write(4,0, ('PME'))
5123.     PME = [PME_Amba,PME_Toto,PME_Puyo,PME_Tena,PME_Bani]
5124.
5125.     for i in range(len(PME)):
5126.         ws_e.write(4,i+1,PME[i])
5127.
5128.     # ==================================> Cálculo EXACTITUD DE LA PROYECCION
5129.     ws_e.write(5,0,'EP')
5130.     for i in range(len(PEMA)):
5131.         ws_e.write(5,i+1,(100-PEMA[i]))
5132.
5133.
5134.     for i in range(len(titulos)-1):
5135.         ws_e.write(0,i+1,titulos[i+1])
5136.
5137.     wb.save('04_PROYECCION_MLP.xls')
5138.     #=============================================================================

5139.     #=============================================================================


5140.
5141.
5142.     print(' ')
5143.     print(' ')
5144.     print('* La fecha seleccionada es:',int(po_anio_p),(pos_mes_p+1),int(po_dia_p))

5145.     print(' ')
5146.     print(' ')
5147.     print(' ')
5148.     print('*** MLP Completed...!!!')
5149.     print(' ')
5150.     print(' ')
5151.
5152. start_time = time()
5153. test()
5154. elapsed_time = time() - start_time
5155.
5156. if (elapsed_time >= 60):
5157.     minutos = elapsed_time / 60
5158.     segundos = elapsed_time - int (minutos * 60)
5159.     print(' ')
5160.     print(' ')
5161.     print('////////////////////////////////////////')
5162.     print(' ')
5163.     print('ALGORITMO UTILIZADO: MLP')
```

```
5164.     print('Tiempo transcurrido: %.0f minutos '%minutos +'y %.2f segundos' %segundos
      )
5165.     print(' ')
5166.     print('////////////////////////////////////////')
5167. else:
5168.
5169.     print(' ')
5170.     print(' ')
5171.     print('////////////////////////////////////////')
5172.     print(' ')
5173.     print('ALGORITMO UTILIZADO: MLP')
5174.     print("Tiempo transcurrido: %.10f segundos." % elapsed_time)
5175.     print(' ')
5176.     print('////////////////////////////////////////')
5177.
```