División de ciencias exactas

Departamento de física

Curso de física computacional

Reporte

Evaluación 1

Examen

Autor:
Miguel Ernesto Medina León

Profesor:
Carlos Lizárraga Celaya

Año académico 2018-2019

Introducción

En las instrucciones de la evaluación se nos habla acerca de un sistema de puntajes basados en las temperaturas por cada hora para determinar las "horas frío". Este puntaje, dependiendo del rango, asigna ciertos puntos; este es el modelo Utah de Richardson.

Tabla 1: Relación de eficacia para la salida de la dormición, según el «modelo de Utah».

Temperatura (ºC)	UF correspondientes a 1 hora transcurrida a un dado rango térmico
< 1,4	0
1,5 a 2,4	0,5
2,5 a 9,1	1
9,2 a 12,4	0,5
12,5 a 15,9	0
16,0 a 18,0	-0,5
> 18	-1

En base a eso y con los datos proporcionados, se deben conseguir dos gráficas: una que muestre la evolución de las temperaturas máximas y mínimas de cada día en función del tiempo, y otra que nos muestre la acumulación de las horas frío a través del tiempo.
A través de loops, comandos varios, DataFrames, arreglos y demás métodos se irán organizando y consiguiendo los datos necesarios para llevar a cabo la actividad solicitada. A continuación se muestra cómo.

Desarrollo

Para conseguir las gráficas solicitadas, primero se importan las librerías necesarias y se lee el archivo de datos.

```
#Se importan las librerias.
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import calendar
```

```
#Se lee la tabla.
df0 = pd.read_csv('vid18_180219.dat', delimiter = ',')
sns.set(rc={'figure.figsize':(12,8.27)})
```

Tras eso, se hace un DataFrame con los datos que nos interesan (la fecha y la temperatura del aire), y se transforman las fechas a variable temporal, del cuál se sacan una columna de años, meses, días y horas.

```
df1 = pd.DataFrame({"Fecha" : df0.TIMESTAMP, "TemperaturaDelAire" : df0.AirTC_Avg})
```

```
#Se cambia las fechas a variable temporal.
df1['Fecha'] = pd.to_datetime(df1.apply(lambda x: x['Fecha'], 1),dayfirst =True)
```

```
#Se crearán columnas para las horas, meses y años.
df1['Año'] = df1['Fecha'].dt.year
df1['Mes'] = df1['Fecha'].dt.month
df1['Día'] = df1['Fecha'].dt.day
df1['Hora'] = df1['Fecha'].dt.hour
df1['Minuto'] = df1['Fecha'].dt.minute
df1.head()
```

	Fecha	TemperaturaDelAire	Año	Mes	Día	Hora	Minuto
0	2018-05-11 20:10:00	23.50	2018	5	11	20	10
1	2018-05-11 20:20:00	22.96	2018	5	11	20	20
2	2018-05-11 20:30:00	22.73	2018	5	11	20	30
3	2018-05-11 20:40:00	22.40	2018	5	11	20	40
4	2018-05-11 20:50:00	22.46	2018	5	11	20	50

Como se pidieron los datos después del 11 de Noviembre, se toman nomás esos con el comando *loc*, con ayuda de un condicional, reordenando los índices en el proceso con ayuda del comando *index*, y se quita la columna de fecha por medio del comando *drop*.

```
#Nomas me interesan los datos desde el 1ro de Noviembre de 2018
df2 = df1.loc[df1['Fecha']>='2018-11-01 00:00:00']
df2.index = np.arange(0,len(df2))
```

```
df2 = df2.drop({'Fecha'},axis=1)
```

Después de eso, se consigue la temperatura de cada hora haciendo un promedio entre la temperatura de cada 10 minutos por hora, por medio del comando *groupby*, en función de las variables temporales año-mes-día-hora, haciendo el promediaje de la temperatura del aire.

```
#Se convierte y se consigue el promedio de temperatura cada hora.
df2['TempPromH'] = df2.groupby(['Año','Mes','Día','Hora'])['TemperaturaDelAire'].transform('mean')
df2
```

	TemperaturaDelAire	Año	Mes	Día	Hora	Minuto	TempPromH
0	9.130	2018	11	1	0	0	8.708333
1	8.890	2018	11	1	0	10	8.708333
2	8.660	2018	11	1	0	20	8.708333
3	8.520	2018	11	1	0	30	8.708333
4	8.470	2018	11	1	0	40	8.708333
5	8.580	2018	11	1	0	50	8.708333
6	8.560	2018	11	1	1	0	8.493333
7	8.320	2018	11	1	1	10	8.493333
8	8.150	2018	11	1	1	20	8.493333
9	8.250	2018	11	1	1	30	8.493333
10	8.570	2018	11	1	1	40	8.493333
11	9.110	2018	11	1	1	50	8.493333
12	8.830	2018	11	1	2	0	8.690000
13	8.590	2018	11	1	2	10	8.690000
14	8.580	2018	11	1	2	20	8.690000
15	8.710	2018	11	1	2	30	8.690000
16	8.670	2018	11	1	2	40	8.690000
17	8.760	2018	11	1	2	50	8.690000
18	9.130	2018	11	1	3	0	8.846667
19	9.120	2018	11	1	3	10	8.846667
20	9.210	2018	11	1	3	20	8.846667

Como nos interesan los datos por hora, se hace que se filtren haciendo un nuevo DataFrame con los datos que coincidan en el renglón donde los minutos sean cero.

```
#Se elimina la columna minutos.
df3 = pd.DataFrame({"TempAire" : df2[df2.Minuto==0].TemperaturaDelAire, "Año" : df2[df2.Minuto==0].Año, "Mes" : df2[df2.Minuto==0].Mes, "Día" : df2[df2.Minuto==0].Día, "Hora" : df2[df2.Minuto==0].Hora, "TemperaturaPromHora" : df2[df2.Minuto==0].TempPromH})
df3.index = np.arange(0,len(df3))
df3
```

	Año	Día	Hora	Mes	TempAire	TemperaturaPromHora
0	2018	1	0	11	9.130	8.708333
1	2018	1	1	11	8.560	8.493333
2	2018	1	2	11	8.830	8.690000
3	2018	1	3	11	9.130	8.846667
4	2018	1	4	11	7.924	7.397500
5	2018	1	5	11	7.261	7.289833
6	2018	1	6	11	7.723	6.806833
7	2018	1	7	11	6.125	8.110167
8	2018	1	8	11	12.430	14.960000
9	2018	1	9	11	18.080	19.710000
10	2018	1	10	11	21.670	22.396667
11	2018	1	11	11	23.010	23.853333
12	2018	1	12	11	24.360	25.125000
13	2018	1	13	11	26.190	26.520000
14	2018	1	14	11	26.710	27.380000
15	2018	1	15	11	27.760	28.160000
16	2018	1	16	11	29.330	28.621667
17	2018	1	17	11	29.530	27.438333
18	2018	1	18	11	19.870	16.916667

Ahora se usa el modelo Utah de Richardson para conseguir el puntaje.
Con ayuda de un loop y múltiples condicionales *if* que se adaptan al modelo, asignando un valor a la variable en la posición **i** dependiendo de la temperatura, guardando dicho valor en un arreglo **d2**, y haciendo que la variable contadora **n** valga cero de nuevo, para después guardarse en un DataFrame dicho arreglo.

```
#Se hace la condicional y se consigue la puntuación de UF cada hora.
n = 0
d2 = []
for i in range (0,len(df3.TemperaturaPromHora)):
    if(df3["TemperaturaPromHora"][i]<=1.5):
        n = 0
    elif(df3["TemperaturaPromHora"][i]>1.5 and df3["TemperaturaPromHora"][i]<=2.5):
        n = 0.5
    elif(df3["TemperaturaPromHora"][i]>2.5 and df3["TemperaturaPromHora"][i]<=9.2):
        n = 1
    elif(df3["TemperaturaPromHora"][i]>9.2 and df3["TemperaturaPromHora"][i]<=12.5):
        n = 0.5
    elif(df3["TemperaturaPromHora"][i]>12.5 and df3["TemperaturaPromHora"][i]<=16):
        n = 0
    elif(df3["TemperaturaPromHora"][i]>16 and df3["TemperaturaPromHora"][i]<=18):
        n = -0.5
    elif(df3["TemperaturaPromHora"][i]>18):
        n = -1
    d2.append(n)
    n = 0
df3['UFH'] = d2
df3
```

	Año	Día	Hora	Mes	TempAire	TemperaturaPromHora	UFH
0	2018	1	0	11	9.130	8.708333	1.0
1	2018	1	1	11	8.560	8.493333	1.0
2	2018	1	2	11	8.830	8.690000	1.0
3	2018	1	3	11	9.130	8.846667	1.0
4	2018	1	4	11	7.924	7.397500	1.0
5	2018	1	5	11	7.261	7.289833	1.0
6	2018	1	6	11	7.723	6.806833	1.0
7	2018	1	7	11	6.125	8.110167	1.0
8	2018	1	8	11	12.430	14.960000	0.0

Luego, se hace una suma para conseguir las horas frío de cada día, por medio del comando *groupby* en función de la variable temporal año-mes-día, haciendo la suma de la puntuación.

```
#Se convierte y se consigue la suma de cada 24 horas para conseguir el UF de cada día.
df3['UFD'] = df3.groupby(['Año','Mes','Día'])['UFH'].transform('sum')
df3
```

	Año	Día	Hora	Mes	TempAire	TemperaturaPromHora	UFH	UFD
0	2018	1	0	11	9.130	8.708333	1.0	-0.5
1	2018	1	1	11	8.560	8.493333	1.0	-0.5
2	2018	1	2	11	8.830	8.690000	1.0	-0.5
3	2018	1	3	11	9.130	8.846667	1.0	-0.5
4	2018	1	4	11	7.924	7.397500	1.0	-0.5
5	2018	1	5	11	7.261	7.289833	1.0	-0.5
6	2018	1	6	11	7.723	6.806833	1.0	-0.5
7	2018	1	7	11	6.125	8.110167	1.0	-0.5
8	2018	1	8	11	12.430	14.960000	0.0	-0.5
9	2018	1	9	11	18.080	19.710000	-1.0	-0.5
10	2018	1	10	11	21.670	22.396667	-1.0	-0.5
11	2018	1	11	11	23.010	23.853333	-1.0	-0.5
12	2018	1	12	11	24.360	25.125000	-1.0	-0.5
13	2018	1	13	11	26.190	26.520000	-1.0	-0.5
14	2018	1	14	11	26.710	27.380000	-1.0	-0.5
15	2018	1	15	11	27.760	28.160000	-1.0	-0.5
16	2018	1	16	11	29.330	28.621667	-1.0	-0.5
17	2018	1	17	11	29.530	27.438333	-1.0	-0.5
18	2018	1	18	11	19.870	16.916667	-0.5	-0.5
19	2018	1	19	11	14.830	14.708333	0.0	-0.5
20	2018	1	20	11	14.570	14.060000	0.0	-0.5

Se hacen columnas de la temperatura máxima y mínima a través de *groupby*, en función de la variable temporal año-mes-día, tomando el máximo y el mínimo de la temperatura de cada hora.

```
#Se convierte y se consigue la temperatura máxima y mínima de cada día
df3['TemperaturaMáxima'] = df3.groupby(['Año','Mes','Día'])['TemperaturaPromHora'].transform('max')
df3['TemperaturaMínima'] = df3.groupby(['Año','Mes','Día'])['TemperaturaPromHora'].transform('min')
df3
```

	Año	Día	Hora	Mes	TempAire	TemperaturaPromHora	UFH	UFD	TemperaturaMáxima	TemperaturaMínima
0	2018	1	0	11	9.130	8.708333	1.0	-0.5	28.621667	6.806833
1	2018	1	1	11	8.560	8.493333	1.0	-0.5	28.621667	6.806833
2	2018	1	2	11	8.830	8.690000	1.0	-0.5	28.621667	6.806833
3	2018	1	3	11	9.130	8.846667	1.0	-0.5	28.621667	6.806833
4	2018	1	4	11	7.924	7.397500	1.0	-0.5	28.621667	6.806833
5	2018	1	5	11	7.261	7.289833	1.0	-0.5	28.621667	6.806833
6	2018	1	6	11	7.723	6.806833	1.0	-0.5	28.621667	6.806833
7	2018	1	7	11	6.125	8.110167	1.0	-0.5	28.621667	6.806833
8	2018	1	8	11	12.430	14.960000	0.0	-0.5	28.621667	6.806833
9	2018	1	9	11	18.080	19.710000	-1.0	-0.5	28.621667	6.806833
10	2018	1	10	11	21.670	22.396667	-1.0	-0.5	28.621667	6.806833
11	2018	1	11	11	23.010	23.853333	-1.0	-0.5	28.621667	6.806833
12	2018	1	12	11	24.360	25.125000	-1.0	-0.5	28.621667	6.806833
13	2018	1	13	11	26.190	26.520000	-1.0	-0.5	28.621667	6.806833
14	2018	1	14	11	26.710	27.380000	-1.0	-0.5	28.621667	6.806833
15	2018	1	15	11	27.760	28.160000	-1.0	-0.5	28.621667	6.806833
16	2018	1	16	11	29.330	28.621667	-1.0	-0.5	28.621667	6.806833
17	2018	1	17	11	29.530	27.438333	-1.0	-0.5	28.621667	6.806833
18	2018	1	18	11	19.870	16.916667	-0.5	-0.5	28.621667	6.806833
19	2018	1	19	11	14.830	14.708333	0.0	-0.5	28.621667	6.806833

Y se filtran los datos de cada día.

```
#Se hace un dataframe en donde nomás estén las variables temporales que nos interesan y los datos (UF
D y las temperaturas máximas mínimas).
df4 = pd.DataFrame({"Año" : df3[df3.Hora==0].Año, "Mes" : df3[df3.Hora==0].Mes, "Día" : df3[df3.Hora=
=0].Día, "UF24" : df3[df3.Hora==0].UFD, "TemperaturaMáxima" : df3[df3.Hora==0].TemperaturaMáxima, "Te
mperaturaMínima" : df3[df3.Hora==0].TemperaturaMínima})
df4.index = np.arange(0,len(df4))
df4
```

	Año	Día	Mes	TemperaturaMáxima	TemperaturaMínima	UF24
0	2018	1	11	28.621667	6.806833	-0.5
1	2018	2	11	30.960000	10.248333	-9.0
2	2018	3	11	29.998333	10.473333	-9.0
3	2018	4	11	31.243333	11.670000	-11.5
4	2018	5	11	31.093333	11.435000	-8.0
5	2018	6	11	32.391667	12.448333	-11.5
6	2018	7	11	32.208333	11.100000	-8.5
7	2018	8	11	31.191667	11.058333	-9.5
8	2018	9	11	29.526667	11.208333	-8.5
9	2018	10	11	28.881667	8.963333	-5.0
10	2018	11	11	28.591667	9.248333	-8.0
11	2018	12	11	26.171667	6.515167	-2.0
12	2018	13	11	25.026667	6.981167	-1.0
13	2018	14	11	24.758333	4.575000	0.0
14	2018	15	11	26.593333	2.475167	3.5
15	2018	16	11	26.348333	5.308667	3.0
16	2018	17	11	27.460000	7.388500	-1.0
17	2018	18	11	27.530000	7.332833	-2.0

Para graficar las evoluciones, se necesita graficar en función de las fechas, así que necesito una columna con una variable fecha, por lo que convierto esa variable del primer DataFrame hasta que sea por día.

```
#Voy consiguiendo y preservando la columna fecha en función de los días que necesitaré más
adelante.
df1 = df1.loc[df1['Fecha']>='2018-11-01 00:00:00']
df1.index = np.arange(0,len(df1))
df1.head()
```

	Fecha	TemperaturaDelAire	Año	Mes	Día	Hora	Minuto
0	2018-11-01 00:00:00	9.13	2018	11	1	0	0
1	2018-11-01 00:10:00	8.89	2018	11	1	0	10
2	2018-11-01 00:20:00	8.66	2018	11	1	0	20
3	2018-11-01 00:30:00	8.52	2018	11	1	0	30
4	2018-11-01 00:40:00	8.47	2018	11	1	0	40

```
#Adapto la variable fecha a días - 1.
df5 = pd.DataFrame({"Fecha" : df1[df1.Minuto==0].Fecha, "Año" : df1[df1.Minuto==0].Año, "M
es" : df1[df1.Minuto==0].Mes, "Día" : df1[df1.Minuto==0].Día, "Hora" : df1[df1.Minuto==0].
Hora})
df5.index = np.arange(0,len(df5))
df5.head()
```

	Año	Día	Fecha	Hora	Mes
0	2018	1	2018-11-01 00:00:00	0	11
1	2018	1	2018-11-01 01:00:00	1	11
2	2018	1	2018-11-01 02:00:00	2	11
3	2018	1	2018-11-01 03:00:00	3	11
4	2018	1	2018-11-01 04:00:00	4	11

```
#Adapto la variable fecha a días - 2.
df6 = pd.DataFrame({"Fecha" : df5[df5.Hora==0].Fecha, "Año" : df5[df5.Hora==0].Año, "Mes"
: df5[df5.Hora==0].Mes, "Día" : df5[df5.Hora==0].Día})
df6.index = np.arange(0,len(df6))
df6
```

	Año	Día	Fecha	Mes
0	2018	1	2018-11-01	11

Ya que se tiene lo necesario, se crea un DataFrame con los datos importantes.

```
#Creo un DataFrame con las variables que me importan con una variable fecha.
df7 = pd.DataFrame({"Fecha" : df6.Fecha, "TemperaturaMáxima" : df4.TemperaturaMáxima, "TemperaturaMín
ima" : df4.TemperaturaMínima, "UF24" : df4.UF24})
df7
```

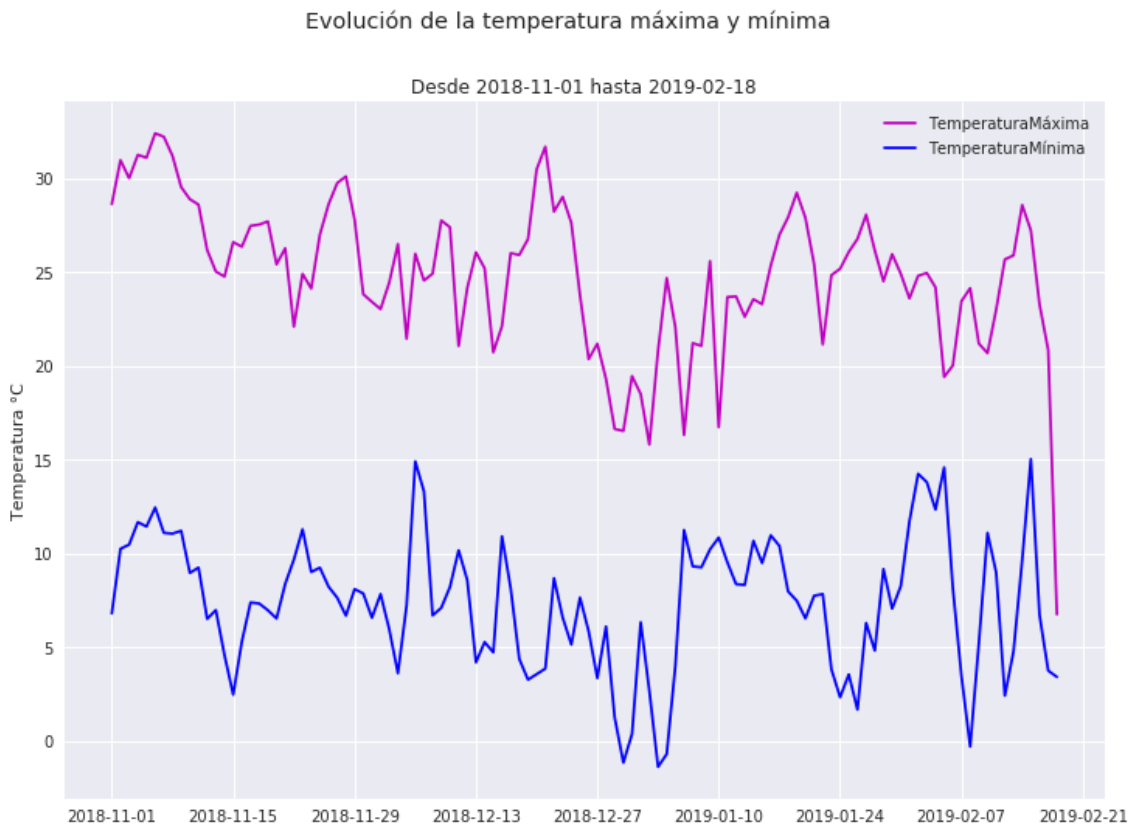
	Fecha	TemperaturaMáxima	TemperaturaMínima	UF24
0	2018-11-01	28.621667	6.806833	-0.5
1	2018-11-02	30.960000	10.248333	-9.0
2	2018-11-03	29.998333	10.473333	-9.0
3	2018-11-04	31.243333	11.670000	-11.5
4	2018-11-05	31.093333	11.435000	-8.0
5	2018-11-06	32.391667	12.448333	-11.5
6	2018-11-07	32.208333	11.100000	-8.5
7	2018-11-08	31.191667	11.058333	-9.5
8	2018-11-09	29.526667	11.208333	-8.5
9	2018-11-10	28.881667	8.963333	-5.0
10	2018-11-11	28.591667	9.248333	-8.0
11	2018-11-12	26.171667	6.515167	-2.0
12	2018-11-13	25.026667	6.981167	-1.0
13	2018-11-14	24.758333	4.575000	0.0
14	2018-11-15	26.593333	2.475167	3.5
15	2018-11-16	26.348333	5.308667	3.0
16	2018-11-17	27.460000	7.388500	-1.0
17	2018-11-18	27.530000	7.332833	-2.0
18	2018-11-19	27.695000	6.973000	0.0
19	2018-11-20	25.410000	6.537167	-0.5

Gráfica de la evolución de la temperatura máxima y mínima

Ahora solo queda graficar los datos por medio de *plt*.

```
#Ahora se proceden a graficar las temperaturas máximas y mínimas a través del tiempo.
plt.plot_date(x=df7.Fecha, y=df7.TemperaturaMáxima, fmt= "m")
plt.plot_date(x=df7.Fecha, y=df7.TemperaturaMínima, fmt= "b-")
plt.legend(loc='best')
plt.suptitle("Evolución de la temperatura máxima y mínima")
plt.title("Desde 2018-11-01 hasta 2019-02-18")
plt.ylabel("Temperatura °C")
plt.grid(True)
plt.savefig('TMaxMinEvo', plt = 2000)
plt.show()
```

Quedando así:



Gráfica de la acumulación de las horas frío

Únicamente queda conseguir la columna de la acumulación de las horas frío, por lo que se usa un loop con un contador **temp** al cuál se le van sumando los puntos de cada día en la posición **i**, registrando la acumulación de cada día en el arreglo **UFC**, y a partir del mismo se crea otro DataFrame con los datos a graficar.

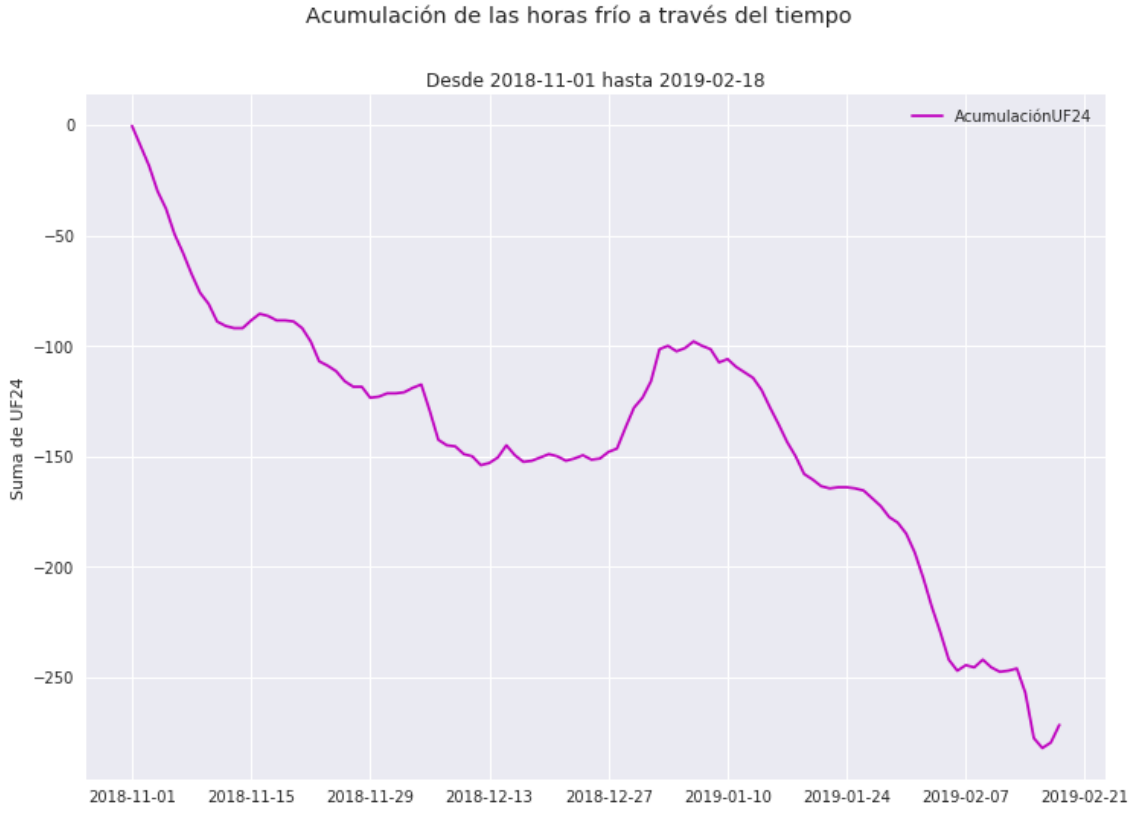
```
#Consigo la acumulación de horas frío desde el primer día.
UFC = []
temp = 0
for i in range (0, len(df7.UF24)):
    temp = df7["UF24"][i] + temp
    UFC.append(temp)
df8 = pd.DataFrame({"Fecha" : df7.Fecha, "AcumulaciónUF24" : UFC})
df8
```

	AcumulaciónUF24	Fecha
0	-0.5	2018-11-01
1	-9.5	2018-11-02
2	-18.5	2018-11-03
3	-30.0	2018-11-04
4	-38.0	2018-11-05
5	-49.5	2018-11-06
6	-58.0	2018-11-07
7	-67.5	2018-11-08
8	-76.0	2018-11-09
9	-81.0	2018-11-10
10	-89.0	2018-11-11
11	-91.0	2018-11-12
12	-92.0	2018-11-13
13	-92.0	2018-11-14
14	-88.5	2018-11-15
15	-85.5	2018-11-16
16	-86.5	2018-11-17
17	-88.5	2018-11-18

Finalmente, se grafica con *mathplotlib*:

```
#Ahora se proceden a graficar la acumulación de horas frío a través del tiempo.
plt.plot_date(x=df7.Fecha, y=df8.AcumulaciónUF24, fmt= "m")
plt.legend(loc='best')
plt.suptitle("Acumulación de las horas frío a través del tiempo")
plt.title("Desde 2018-11-01 hasta 2019-02-18")
plt.ylabel("Suma de UF24")
plt.grid(True)
plt.savefig('AcuUF24', plt = 2000)
plt.show()
```

La acumulación de la estación ubicada en un cultivo de Vid en el kilometro 41 de la carretera Hermosillo a la Bahía Kino tiene esta forma:


Acumulación de las horas frío a través del tiempo
Desde 2018-11-01 hasta 2019-02-18
AcumulaciónUF24
Suma de UF24
0
−50
−100
−150
−200
−250
2018-11-01
2018-11-15
2018-11-29
2018-12-13
2018-12-27
2019-01-10
2019-01-24
2019-02-07
2019-02-21

Conclusión

A lo largo del transcurso del examen aprendí bastante, es decir, algunos conocimientos y comandos no los comprendía del todo; no tenía las bases asentadas, pero a medida que avanzaba en el examen, le pensaba y se me presentaban nuevos comandos, encontré maneras creativas de usarlo, y supe bien cómo usar correctamente las que ya previamente había usado en anteriores actividades, pero que no me habían quedado claras.
Fue una experiencia productiva, a decir verdad.