Reporte de la Actividad 8

Marco Antonio Cabello López
Grupo 1

Domingo 24 de Marzo del 2019

1 Introducción

Primeramente y como introducción, en esta actividad analizamos 2 conjuntos de datos del año 2009 de una estación de meteorología ubicada en un campo de Nogal. Nos interesa crear en los datos del suelo una nueva columna con una variable temporal similar a la que se tiene en los datos meteorológicos, para poder incorporar un conjunto de variables nuevas. Las variables que nos interesan estudiar son los relacionados con las lecturas de temperatura del suelo en 8 profundidades distintas. Tambien se desea incorporar la temperatura del aire del dataframe de datos meteorológicos.

2 Desarrollo de la actividad

2.1 Metodología

Comenzamos a trabajar con el archivo, primero fue necesario descargar librerías para la visualización y el análisis de datos :

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import math
import datetime
```

El primer archivo a leer contiene los datos de una estación de meteorología ubicada en un campo de Nogal:

```
df1 = pd.DataFrame( pd.read_csv("meteo-nogal-09.csv", engine="python" ) )
```

Ya analizamos este archivo anteriormente en la Actividad 7, por lo que realizamos los mismos pasos, pero ahora creanmos un formato de fecha, ya que uniremos los 2 archivos mediante esta variable.

Continuamos con la lectura del segundo archivo que contenía los datos del suelo en el mismo campo de Nogal:

```
df2 = pd.DataFrame( pd.read_csv("soil-nogal-09.csv", engine="python" ) )
```

Filtramos las columnas de interés con datos correspondientes a fechas y temperaturas del suelo a distintas profundidades:

```
df2 = df2.filter(['2 Year_RTM  L','3 Day_RTM  L','4 Hour_Minute_RTM  L',
    'Tsuelo_10cm','Tsuelo_20cm','Tsuelo_30cm','Tsuelo_40cm','Tsuelo_55cm',
    'Tsuelo_70cm','Tsuelo_85cm','Tsuelo_100cm'],axis=1)
```

Ya que en el segundo archivo las horas y los minutos se encuentran mezclados en el formato, se crearon dos arreglos para separarlos. Primero, se convirtió la columna de horas y minutos a tipo string:

```
df2['4 Hour_Minute_RTM  L'] = df2['4 Hour_Minute_RTM  L'].astype(str)
```

Posteriormente recuperamos de dicha columna las horas y los minutos por separado y las estructuramos en los arreglos correspondientes:

```
hora=[]
minuto=[]

for i in range (0, len(df2)):
    #Primero revisaremos si contiene 4 caracteres.
    if (len(str(df2['4 Hour_Minute_RTM  L'][i]))==4):
        #Si resulta ser de 4 caracteres, revisaremos el caso en que la hora sea 2400.
        if (str(df2['4 Hour_Minute_RTM  L'][i])[0:2]=='24'):
            hora.append('00')
            minuto.append('00')
        else:
            hora.append(str(df2['4 Hour_Minute_RTM  L'][i])[0:2])
            minuto.append(str(df2['4 Hour_Minute_RTM  L'][i])[2:4])
    #Ahora revisamos el caso de que contenga 3 caracteres.
    elif (len(str(df2['4 Hour_Minute_RTM  L'][i]))==3):
            hora.append(str(df2['4 Hour_Minute_RTM  L'][i])[0:1])
            minuto.append(str(df2['4 Hour_Minute_RTM  L'][i])[1:3])
    #Por último revisaremos el caso de 2 caracteres.
    elif (len(str(df2['4 Hour_Minute_RTM  L'][i]))==2):
            hora.append('00')
            minuto.append(str(df2['4 Hour_Minute_RTM  L'][i])[0:2])
```

Creamos un arreglo para almacenar los días:

```
dias =[df2['3 Day_RTM  L'][i] for i in range(0,len(df2))]
```

Después creamos un data frame con los días, las horas y los minutos:

```
d = {'dias': dias, 'hora': hora, 'minuto':minuto}
df_fechas = pd.DataFrame(data=d)
```

Ahora con los días corregidos, creamos un arreglo de fechas de tipo string, con el año, los días, horas y minutos separados por espacios:

```
fechas = []
for i in range (0,len(df2)):
    fechas.append('2009 '+str(df_fechas['dia'][i])
    + ' ' + df_fechas['hora'][i]+' '+df_fechas['minuto'][i])
```

Generamos este arreglo y se le dio formato de fecha, consideramos que los días se encuentran escritos en un formato distinto al requerido:

```
FECHA = []
for i in range(0,len(df2)):
    d=datetime.datetime.strptime(fechas[i],'%Y %j %H %M')
    F = d.isoformat(' ')
    FECHA.append(F)
df2['FECHAN']=FECHA
```

Ya que los 2 data frames analizados tienen datos duplicados por fecha, estos fueron desechados de ambos:

```
df1 = df1.drop_duplicates(subset=['FECHA'])
df2 = df2.drop_duplicates(subset=['FECHA'])
```

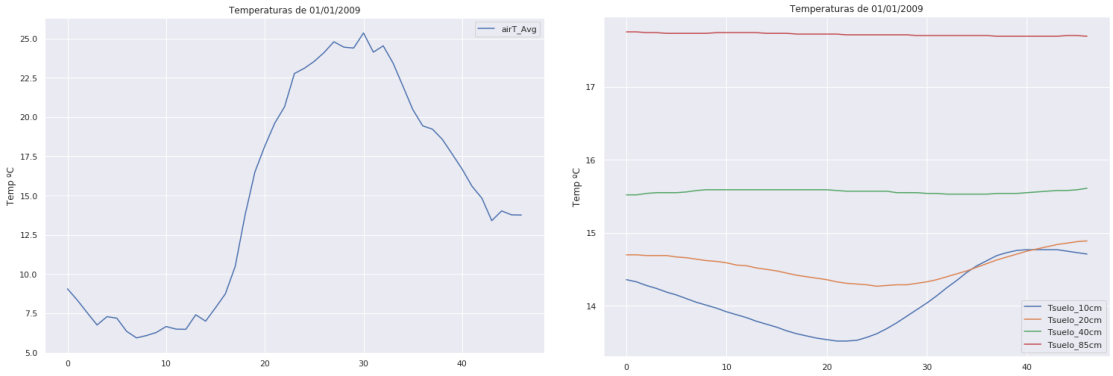
Con los datos filtrados, integramos ambos data frames mediante la función merge, reacomodamos los datos de acuerdo a la columna de fecha creada:

```
df3 = pd.merge(df1, df2, on=['FECHA'])
```

Posteriormente de este nuevo data frame realizamos las gráficas requeridas, consideramos datos para el periodo de tiempo solicitado, y utilizamos las funciones groupby y transform.
Finalmente reproducimos las mismas gráficas, utilizando ahora el concepto de rolling mean como método de suavizar la evolución temporal de una serie de tiempo.

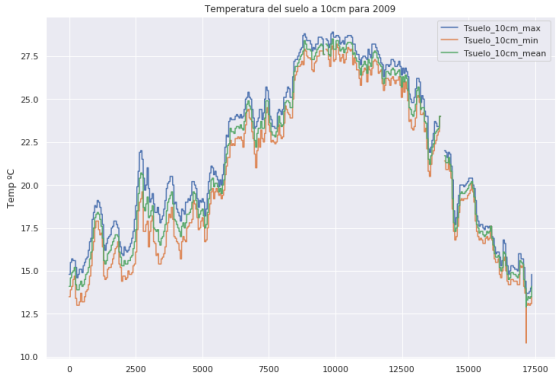
2.2 Resultados

- Seleccionar un día de Enero, y graficar la temperatura del aire, y las 4 temperaturas del subsuelo.

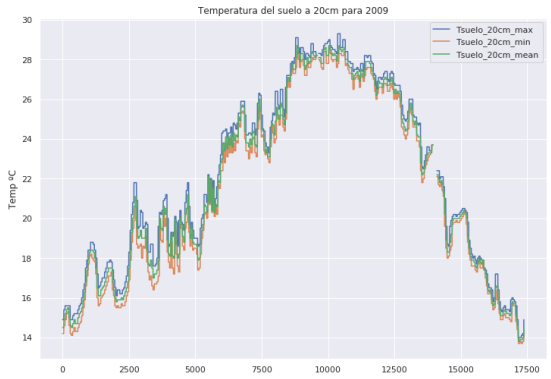


Gráficas de la temperatura del aire y 4 diferentes temperaturas del subsuelo para el 1 de Enero del 2009.

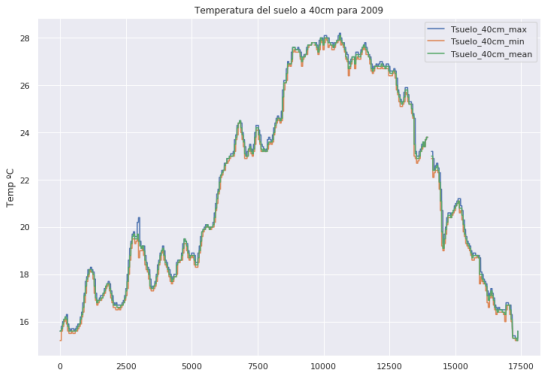
- Realizar una gráfica de temperaturas T_max, T_min y T_promedio diarias para el año completo de datos 2009.



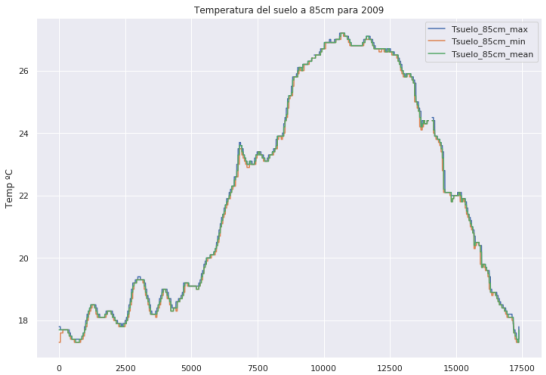
Gráfica de las temperaturas máximas, mínimas y promedio del subsuelo a 10 centímetros de profundidad para cada día de 2009.



Gráfica de las temperaturas máximas, mínimas y promedio del subsuelo a 20 centímetros de profundidad para cada día de 2009.

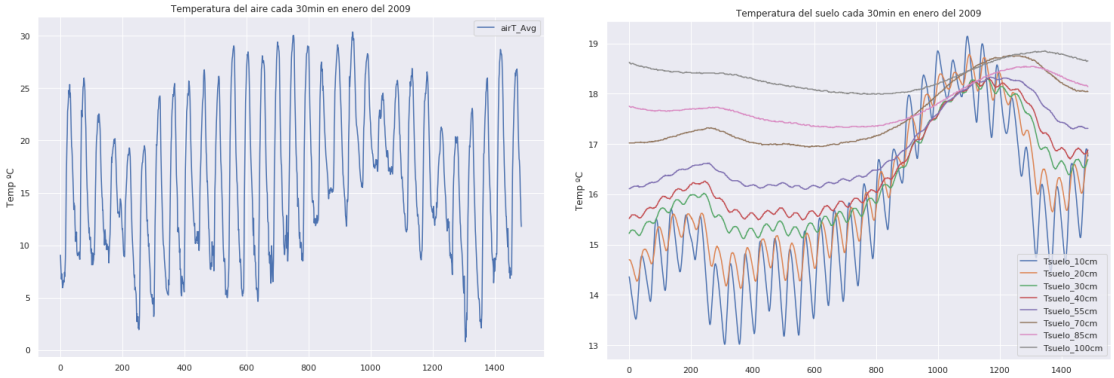


Gráfica de las temperaturas máximas, mínimas y promedio del subsuelo a 40 centímetros de profundidad para cada día de 2009.



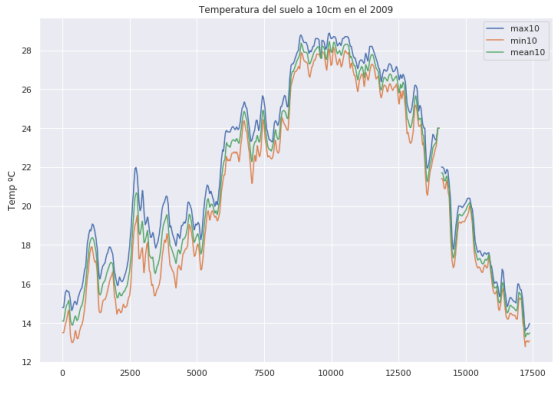
Gráfica de las temperaturas máximas, mínimas y promedio del subsuelo a 85 centímetros de profundidad para cada día de 2009.

- Calcular el promedio cada 30 minutos durante el día para el mes de Enero de la temperatura del aire y las 8 temperaturas promedio de subsuelo, para posteriormente graficar la variación en 24 horas de las temperaturas de interés.

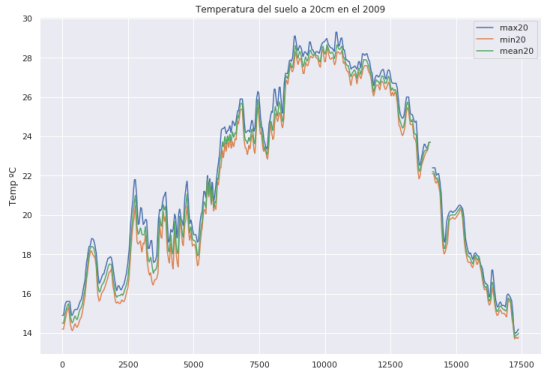


Gráficas de temperatura del aire cada 30 minutos y de temperaturas del subsuelo cada 30 minutos para el mes de Enero del 2009

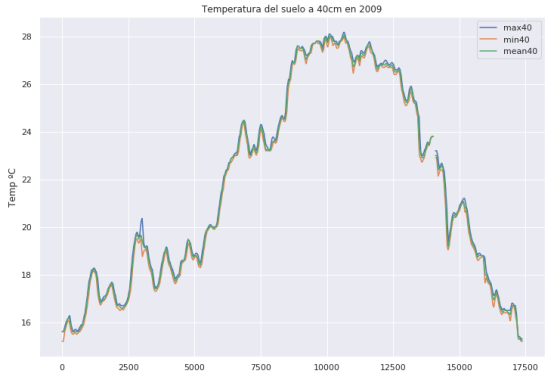
- Introduciremos el concepto de promedio móvil (rolling mean), como método de suavizar la evolución temporal de una serie de tiempo, y se pide reproducir las gráficas suavizadas.



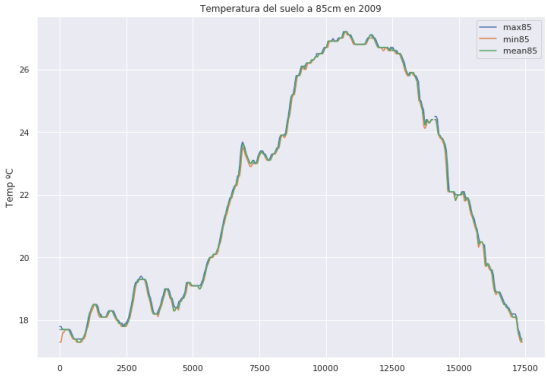
Gráfica suavizada de las temperaturas máximas, mínimas y promedio del subsuelo a 10 centímetros de profundidad para cada día de 2009.



Gráfica suavizada de las temperaturas máximas, mínimas y promedio del subsuelo a 20 centímetros de profundidad para cada día de 2009.



Gráfica suavizada de las temperaturas máximas, mínimas y promedio del subsuelo a 40 centímetros de profundidad para cada día de 2009.



Gráfica suavizada de las temperaturas máximas, mínimas y promedio del subsuelo a 85 centímetros de profundidad para cada día de 2009.

3 Conclusiones

En las gráficas para el 1 de Enero del 2009 podemos observar que la temperatura del subsuelo varía más mientras más cerca de la superficie se tome la medición. A mayor profundidad, la temperatura es mayor y presenta menos variaciones. En contraste, la temperatura del aire tiene marcadas diferencias entre su valor máximo y mínimo a través del día.
En las gráficas de temperatura máxima, mínima y promedio observamos con mayor claridad la conclusión de las gráficas anteriores. Mientras más profundo se tome la medición de temperatura, más parecidas son sus gráficas de valores máximo, mínimo y promedio. En el caso de la gráfica para la temperatura del aire, esta tiene la diferencia de valores más marcada.
Podemos observar en las gráficas para el promedio cada 30 minutos, donde las temperaturas más profundas del subsuelo son las más estables y cálidas.
Las gráficas del rolling mean, nos ayudaron a filtrar la evolucion temporal de los datos.

4 Referencias

- Merge, join, and concatenate. Recuperado de: https://pandas.pydata.org/pandas-docs/stable/user_guide/merging.html
- Python Strings, Functions and Examples. Recuperado de: https://www.techbeamers.com/python-strings-functions-and-examples/
- Seaborn. Recuperado de: https://seaborn.pydata.org/