Práctica 4 Programación Paralela

Laura Herrero y Nefeli Lefa

1 Introducción al problema

Planteamiento, diseño e implementación de una solución a un problema de análisis de datos utilizando Spark. El dataset sobre el que trabajar será el que proporciona el ayuntamiento de Madrid del uso del sistema de bicicletas de préstamo BICIMAD. Nosotros hemos estado utilizando, en los ejercicios de clase, datos relativos a un único mes, pero pueden encontrarse los datos entre desde abril de 2017, por lo que los análisis pueden ser mucho más amplios. Además de ser un dataset conocido, el modelo subyacente en los datos son grafos, que son muy naturales matemática y computacionalmente hablando.

La práctica se realizará en grupo y el resultado del trabajo constará al menos de:

- Definición clara y precisa del problema a resolver,
- el diseño y la implementación en Spark de la solución al problema propuesto,
- un documento que recoja y explique el problema, el proceso de diseño e implementación de la solución y su aplicación a los resultados.
- El documento asociado a la práctica tiene un formato libre, puede ser un documento pdf, html, notebook... El objetivo fundamental del mismo es que explique con claridad tanto el problema como la solución al mismo. Algunos aspectos que puede ser interesante tratar en el documento para contribuir a su claridad son: motivación, ejemplos, detalles importantes de la implementación, evaluación de resultados, etc.

2 Problema propuesto

En esta práctica queremos observar para cada rango de edad la diferencia del uso de las bicicletas entre los días de diario y los fines de semana por franjas horarias. De esta manera, podremos sacar algunas conclusiones (ficticias) basándonos en nuestros conocimientos previos sobre la sociedad española. Antes de ver los resultados podemos imaginar algunas preguntas a resolver: ¿les jóvenes utilizan las bicicletas para volver a casa los fines de semana de fiesta? ¿les adultes las usan para ir al trabajo a diario? Veámoslo.

3 Materiales

Para esta práctica se han utilizado los siguientes materiales:

- Apuntes del profesor sacados de la plataforma *wild.ucm.es*
- Datos obtenidos de la web BiciMAD. Nos hemos centrado en:
  - unplug_hourTime: Franja horaria en la que se realiza el desenganche de la bicicleta.
  - ageRange: Número que indica el rango de edad del usuario que ha realizado el movimiento. Sus posibles valores son:
    - 0: No se ha podido determinar el rango de edad del usuario. Hemos eliminado estos datos de nuestro análisis.
    - 1: El usuario tiene entre 0 y 16 años
    - 2: El usuario tiene entre 17 y 18 años

    - 3: El usuario tiene entre 19 y 26 años
    - 4: El usuario tiene entre 27 y 40 años
    - 5: El usuario tiene entre 41 y 65 años
    - 6: El usuario tiene 66 años o más
- Cluster

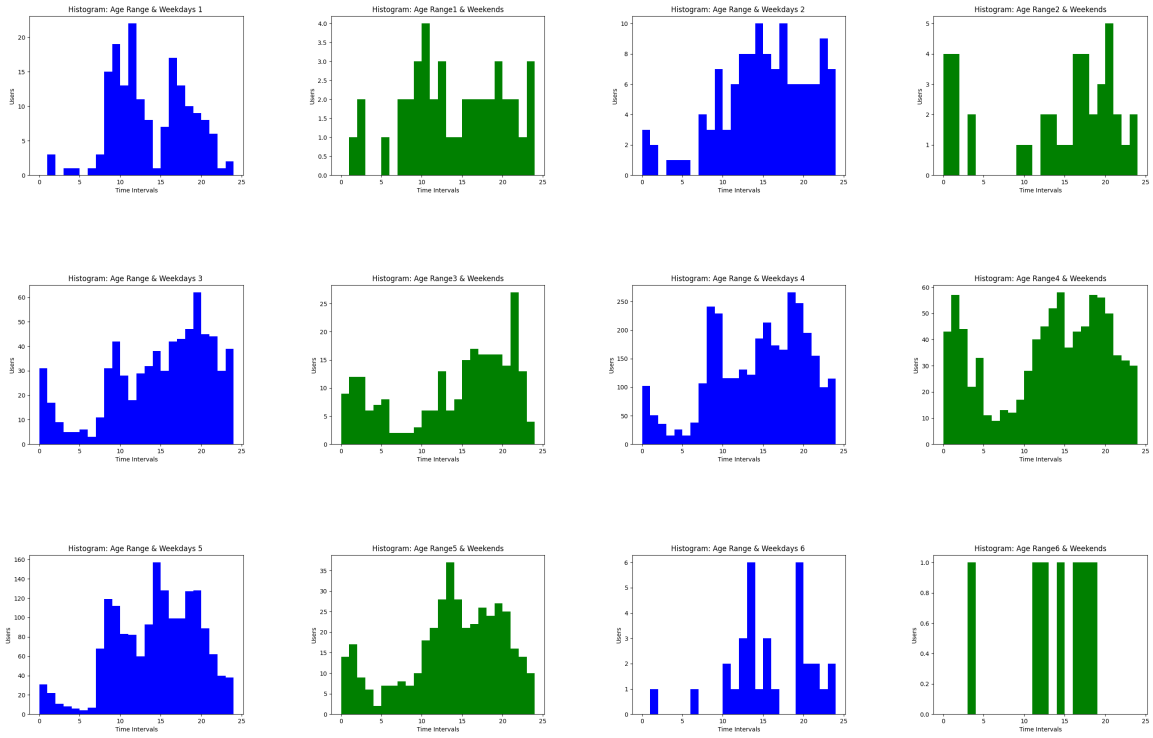
4 Archivos de GitHub

Los archivos que podemos encontrar en el repositorio son:

- *BiciMAD DataFrame.ipynb*, *BiciMAD RDD.ipynb*: apuntes sacados de *wild*.
- *README.md*
- *Servicios-y-estructuras-Bicimad-V1-1.pdf*: modelo de datos de la información de uso del servicio BiciMAD.
- *instrucciones cluster*: comandos para usar el cluster.
- *sample_10e4.json*: muestra pequeña de datos.
- *practica4_simple.py*: código para mostrar histogramas utilizando *sample_10e4.json*.
- *histo_weekdays_1.png*,*histo_weekdays_2.png*, *histo_weekdays_3.png*, *histo_weekdays_4.png*, *histo_weekdays_5.png*, *histo_weekdays_6.png*, *histo_weekends_1.png*, *histo_weekends_2.png*, *histo_weekends_2.png*, *histo_weekends_4.png*, *histo_weekends_3.png*, *histo_weekends_6.png*: histogramas obtenidos con *practica4_simple.py* y *sample_10e4.json*.
- *practica4_spark_plot_all.py*: intento de obtener gráficas ejecutando este código en el cluster. Esto no pudo hacerse. En *Screenshot from 2023-05-28 20-46-50.png* se muestra el error en el cluster al intentar ejecutar este archivo que usa *matplotlib*.
- *funciona.py*: código para extraer del cluster los datos que necesitamos como archivo de texto para resolver nuestro problema.
- *sample_spark*: datos obtenidos del cluster del archivo *201712_movements.json*.
- *obtener_histogramas.py*: código para obtener los histogramas con la información de *sample_spark*.
- *Histograma Weekdays 1:.png*, *Histograma Weekdays 2:.png*, *Histograma Weekdays 3:.png*, *Histograma Weekdays 4:.png*, *Histograma Weekdays 5:.png*, *Histograma Weekdays 6:.png*, *Histograma Weekend 1:.png*, *Histograma Weekend 2:.png*, *Histograma Weekend 3:.png*, *Histograma Weekend 4:.png*, *Histograma Weekend 5:.png*, *Histograma Weekend 6:.png*: histogramas obtenidos con los datos del cluster.

5 Procedimiento y resultados

Para resolver nuestro problema primero realizamos *practica4_simple.py*. Con este código y la muestra *sample_10e4.json* observamos si el problema propuesto tenía algún interés. Nos gustaron los resultados obtenidos que se muestran a continuación:

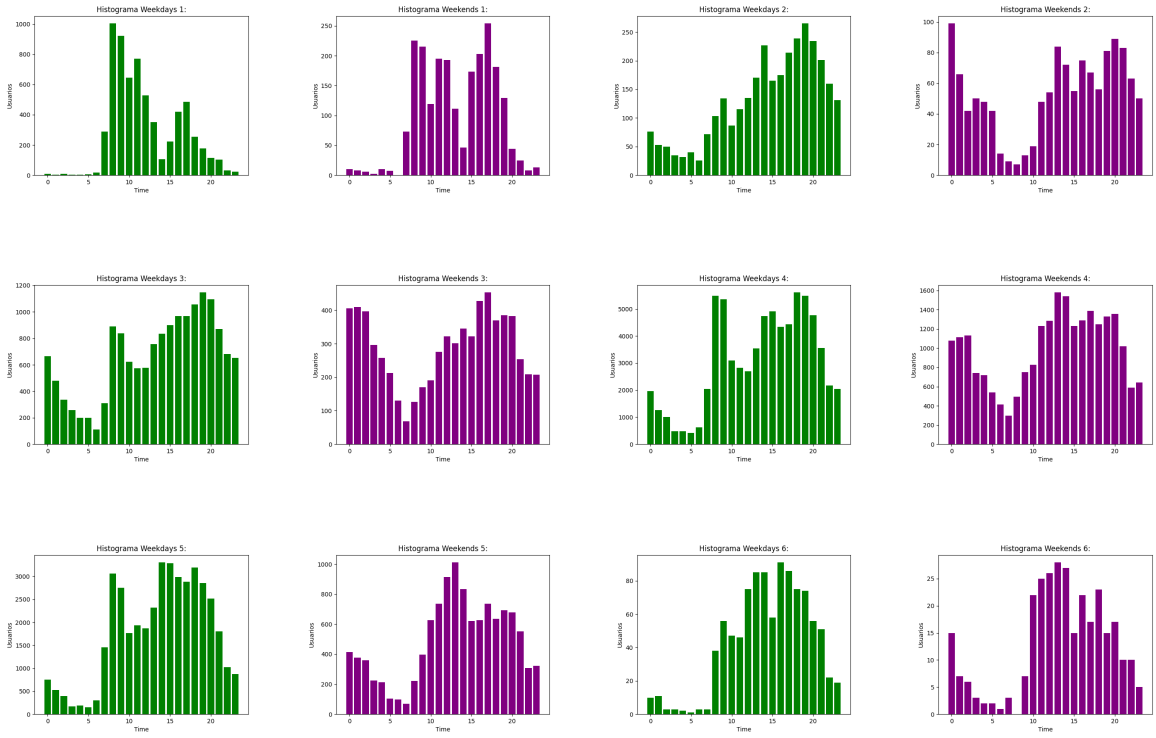


Una vez hecho esto, decidimos realizar lo mismo utilizando *pyspark*. El archivo *practica4_spark_plot_all.py* es la muestra de este intento que no pudimos realizar. Creemos que el problema se debe a utilizar la libreria *matplotlib* de Python en el cluster. En la Figura 1 se muestra el error obtenido.

```
lherre10@wild:~$ spark-submit practica4_spark_plot_all.py
Traceback (most recent call last):
  File "/home/lherre10/practica4_spark_plot_all.py", line 2, in <module>
    import matplotlib.pyplot as plt
  File "/home/lherre10/.local/lib/python3.9/site-packages/matplotlib/pyplot.py", line 61, in <module>
    from matplotlib.figure import Figure, FigureBase, figaspect
  File "/home/lherre10/.local/lib/python3.9/site-packages/matplotlib/figure.py", line 43, in <module>
    from matplotlib import _blocking_input, backend_bases, _docstring, projections
  File "/home/lherre10/.local/lib/python3.9/site-packages/matplotlib/projections/__init__.py", line 55, in <module>
    from .. import axes, _docstring
  File "/home/lherre10/.local/lib/python3.9/site-packages/matplotlib/axes/__init__.py", line 2, in <module>
    from ._axes import *
  File "/home/lherre10/.local/lib/python3.9/site-packages/matplotlib/axes/_axes.py", line 11, in <module>
    import matplotlib.category  # Register category unit converter as side effect.
  File "/home/lherre10/.local/lib/python3.9/site-packages/matplotlib/category.py", line 14, in <module>
    import dateutil.parser
  File "/home/lherre10/.local/lib/python3.9/site-packages/dateutil/parser/__init__.py", line 2, in <module>
    from ._parser import parse, parser, parserinfo, ParserError
  File "/home/lherre10/.local/lib/python3.9/site-packages/dateutil/parser/_parser.py", line 42, in <module>
    import six
ModuleNotFoundError: No module named 'six'
2023-05-28 20:42:41,613 INFO util.ShutdownHookManager: Shutdown hook called
2023-05-28 20:42:41,614 INFO util.ShutdownHookManager: Deleting directory /tmp/spark-d81ec51a-ef43-43e4-8dac-1d2b955074b5
lherre10@wild:~$
```

Figure 1: Screenshot from 2023-05-28 20-46-50.png, muestra el error obtenido en el cluster

Pensamos en solucionar este problema obteniendo del cluster un archivo de texto con los datos que necesitábamos para hacer nuestros histogramas. De esta forma, en nuestro ordenador podíamos leer estos datos y hacer nuestros dibujos. Esto se hizo con *funciona.py* utilizando la muestra del cluster *201712_movements.json*, guardándo los datos en *sample_spark* y leyéndolos con *obtener_histogramas.py*. Los resultados obtenidos fueron:



6 Conclusiones

Basándonos en los últimos resultados obtenidos podemos pensar que:

- Las personas entre 0 y 16 años no utilizan casi las bicicletas de 00h a 06h ni de 21h a 00h. Además, cuando más las utilizan es a diario temprano, posiblemente para ir al colegio. Los fines de semana suelen utilizarlas por las tardes, suponemos que para hacer excursiones en familia.

- En las personas de entre 17 y 18 años vemos en general un decrecimiento de su uso. En la franja anterior llegábamos a les 1000 usuaries,mientras que ahora el máximo se encuentra en torno a los 250. Los fines de semana vemos un aumento en el uso de las bicicletas por la noche, posiblemente para ir de fiesta. Además, hay una importante bajada en su uso por la mañana, lo que nos invita nuevamente a pensar en esta idea.

- En las personas de 19 a 26 años se observa una subida en el uso de las bicicletas (llegamos a cifras que rozan los 1200 a diario y 450 los findes). Entre semana encontramos los picos más altos entre las 17h y las 20h, horas a las que muchas personas salen del trabajo y vuelven a sus casas. Por el histograma de los findes creemos que estamos en una situación similar a la anterior: quizás vuelvan a casa en bicicleta después de una fiesta.

- En la franja de 27 a 40 años se observa el mayor uso de la bicicleta. Estas edades son muy diversas para concluir alguna posible teoría. Viendo la gráfica de entre semana y comparándola con la de la franja 0-16 años, quizás muchas personas las utilizen para ir con sus hijes al colegio, aunque esto también puede deberse al inicio de la jornada laboral.

- Las personas entre 41 y 65 años comienzan a utilizar menos las bicicletas. Si lo comparamos con la franja anterior, las gráficas tienen una forma similar. A diario observamos picos a las horas que coinciden con la entrada de les niñes al colegio o el inicio de la jornada laboral (8-10h) y con el final (14-20h).

- La franja de 66 o más años es la que menos uso presenta. Esto puede ser porque las personas son mayores y tienen mayores dificultades para usar las bicicletas.

7 Anexo código práctica4_simple.py

```
import json
from datetime import datetime
import matplotlib.pyplot as plt

#tiene como argumento una linea del archivo json, y extrae los datos ageRange,
    unplug_hour_time en una lista
def extraer(line):
    data = json.loads(line)
    age_range = data.get('ageRange')
    unplug_hour_time = data.get('unplug_hourTime', {}).get('$date')
    return [age_range, unplug_hour_time]

# cree una lista de listas para cada linea del archivo json
data_list = []
with open('sample_10e4.json', 'r') as file:
    for line in file:
        extracted_data = extraer(line)
        data_list.append(extracted_data)

# filtra listas con ageRange = 0
filtered_list = list(filter(lambda x: x[0] != 0, data_list))

# Ordena según unplug_hourTime
#sorted_list = sorted(filtered_list, key=lambda x: datetime.strptime(x[1], "%Y-%m-%dT%
    H:%M:%S.%f%z"))

# IDEA: Distinguir casos, entresemana y fin de semana

weekdays_list = []
weekends_list = []

#para cada sub-lista en la lista filtrada
for data in filtered_list:
    date_string = data[1][:10]  # Los 10 primeros str representan la fecha, lo que
        quiero extraer
    weekday = datetime.strptime(date_string, "%Y-%m-%d").weekday() #weekday() method
        to get the day of the week as a number, where Monday is 0 and Sunday is 6.

    if weekday < 5:  # Entresemana, lunes a viernes
        weekdays_list.append(data)
    else:  # fin de semana, sabado o domingo
        weekends_list.append(data)

# Ordena según unplug_hourTime
sorted_weekdays = sorted(weekdays_list, key=lambda x: datetime.strptime(x[1], "%Y-%m-%
    dT%H:%M:%S.%f%z").time())
sorted_weekends = sorted(weekends_list, key=lambda x: datetime.strptime(x[1], "%Y-%m-%
    dT%H:%M:%S.%f%z").time())
# Creamos dos plots para cada franja de edad

age_ranges = range(1,7)

for age_range in age_ranges:
    filtered_age_weekdays = list(filter(lambda x: x[0] == age_range, sorted_weekdays))
    filtered_age_weekends = list(filter(lambda x: x[0] == age_range, sorted_weekends))

    # Extraer horas
    hour_data_weekdays = [datetime.strptime(data[1], "%Y-%m-%dT%H:%M:%S.%f%z").hour
        for data in filtered_age_weekdays]
    hour_data_weekends = [datetime.strptime(data[1], "%Y-%m-%dT%H:%M:%S.%f%z").hour
        for data in filtered_age_weekends]

    # Histograma age_Range, weekdays

    plt.hist(hour_data_weekdays,bins=24, range=(0, 24), color='blue')
    plt.xlabel('Time Intervals')
    plt.ylabel('Users')
    plt.title(f'Histogram: Age Range & Weekdays {age_range} ')
    plt.savefig(f'histo_weekdays_{age_range}.png')
```

```
    plt.show()

    # Histigrama age_Range, weekends

    plt.hist(hour_data_weekends,bins=24, range=(0, 24), color='green')
    plt.xlabel('Time Intervals')
    plt.ylabel('Users')
    plt.title(f'Histogram: Age Range{age_range} & Weekends')
    plt.savefig(f"histo_weekends_{age_range}.png")
    plt.show()
'''
# Guardar los resultados en un archivo txt
with open('results1.txt', 'w') as output_file:
    output_file.write("Weekdays:\n")
    for data in sorted_weekdays:
        age_range = data[0]
        unplug_hour_time = data[1]
        output_file.write("Age Range: {}\n".format(age_range))
        output_file.write("Unplug Hour Time: {}\n".format(unplug_hour_time))
        output_file.write("-------------------------\n")

    output_file.write("Weekends:\n")
    for data in sorted_weekends:
        age_range = data[0]
        unplug_hour_time = data[1]
        output_file.write("Age Range: {}\n".format(age_range))
        output_file.write("Unplug Hour Time: {}\n".format(unplug_hour_time))
        output_file.write("-------------------------\n")
'''
'''
#Print para comprobar

print("Weekdays:")
for data in sorted_weekdays:
    age_range = data[0]
    unplug_hour_time = data[1]
    print("Age Range:", age_range)
    print("Unplug Hour Time:", unplug_hour_time)
    print("-------------------------")


print("Weekends:")
for data in sorted_weekends:
    age_range = data[0]
    unplug_hour_time = data[1]
    print("Age Range:", age_range)
    print("Unplug Hour Time:", unplug_hour_time)
    print("-------------------------")
'''
```

8 Anexo código práctica4_spark_plot_all.py

```
from pyspark import SparkContext
import matplotlib.pyplot as plt
import json
from datetime import datetime


def extraer(line):
    data = json.loads(line)
    age_range = data.get('ageRange')
    unplug_hour_time = data.get('unplug_hourTime', {}).get('$date')
    return [age_range, unplug_hour_time]

# IDEA: Distinguir casos, entresemana y fin de semana
# weekday() method to get the day of the week as a number, where Monday is 0 and
    Sunday is 6.

def get_weekday(date_string):
    weekday = datetime.strptime(date_string, "%Y-%m-%d").weekday()
    return weekday

def main(filename):
    with SparkContext() as sc:

        # leer JSON archivo como RDD
        lines_rdd = sc.textFile(filename)

        # extrae los datos ageRange, unplug_hour_time y filtra ageRange = 0
        data_rdd = lines_rdd.map(extraer).filter(lambda x: x[0] != 0) #narrow
            transformation

        # Ordena según unplug_hourTime
        #sorted_rdd = data_rdd.sortBy(lambda x: datetime.strptime(x[1], "%Y-%m-%dT%H:%
            M:%S.%f%z")) #wide transformation

        # IDEA: Distinguir casos, entresemana y fin de semana
        # narrow transformation
        # Los 10 primeros str representan la fecha, lo que quiero extraer
        weekdays_rdd = data_rdd.filter(lambda x: get_weekday(x[1][:10]) < 5)
        weekends_rdd = data_rdd.filter(lambda x: get_weekday(x[1][:10]) >= 5)

        # Ordena según unplug_hourTime
        # wide transformation
        sorted_weekdays_rdd = weekdays_rdd.sortBy(lambda x: datetime.strptime(x[1], "%
            Y-%m-%dT%H:%M:%S.%f%z").time())
        sorted_weekends_rdd = weekends_rdd.sortBy(lambda x: datetime.strptime(x[1], "%
            Y-%m-%dT%H:%M:%S.%f%z").time())


        age_ranges = range(1, 7)
        # Creamos dos plots para cada franja de edad
        for age_range in age_ranges:

            # Filtra age_Range == 1
            # Extrae la componente de la hora
            # countByValue() :Return the count of each unique value in this RDD as a
                dictionary of (value, count) pairs
            age_range_1_weekdays = sorted_weekdays_rdd.filter(lambda x: x[0] ==
                age_range).\
                                map(lambda x: datetime.strptime(x[1], "%Y-%m-%dT%H:%M
                                    :%S.%f%z").hour).\
                                countByValue()

            age_range_1_weekends = sorted_weekends_rdd.filter(lambda x: x[0] ==
                age_range).\
                                map(lambda x: datetime.strptime(x[1], "%Y-%m-%dT%H:%M
                                    :%S.%f%z").hour).\
                                countByValue()
```

```
            # Para cada intervalo, extraemos "count" usando .get(), que devuelva el
                valor para cada "key"
            # Si no existe este "key", devuelve 0

            result_dict_weekdays = {}
            for hour in range(0, 24, 2):
                result_dict_weekdays[hour] = age_range_1_weekdays.get(hour, 0)

            fig, ax = plt.subplots()
            ax.bar(result_dict_weekdays.keys(),result_dict_weekdays.values())
            ax.set_xlabel("Time Intervals")
            ax.set_ylabel("Users")
            ax.set_title(f"Histogram: Weekdays & age Range {age_range}")
            fig.savefig(f"weekdays_{age_range}.png")
            fig.show()

            result_dict_weekends = {}
            for hour in range(0, 24, 2):
                result_dict_weekends[hour] = age_range_1_weekends.get(hour, 0)

            fig, ax = plt.subplots()
            ax.bar(result_dict_weekends.keys(),result_dict_weekends.values(), color='
                orange')
            ax.set_xlabel("Time Intervals")
            ax.set_ylabel("Users")
            ax.set_title(f"Histogram: Weekends & age Range {age_range}")
            fig.savefig(f"weekends_{age_range}.png")
            fig.show()


'''
#Añadir al main

# wide transformation collect()
# action operation that is used to retrieve all the elements of the dataset
# (from all nodes) to the driver node.

# Guardar los resultados en un archivo txt

with open('results.txt', 'w') as output_file:

    output_file.write("Weekdays: \n")

    for data in sorted_weekdays_rdd.collect():
        age_range = data[0]
        unplug_hour_time = data[1]
        print("Age Range:", age_range)
        print("Unplug Hour Time:", unplug_hour_time)
        print("--------------------")

    output_file.write("Weekends:\n")

    for data in sorted_weekends_rdd.collect():
        age_range = data[0]
        unplug_hour_time = data[1]
        print("Age Range:", age_range)
        print("Unplug Hour Time:", unplug_hour_time)
        print("--------------------")
'''
```

9 Anexo código funciona.py

```
from pyspark import SparkContext
import json
from datetime import datetime
import sys


def extraer(line):
    data = json.loads(line)
    age_range = data.get('ageRange')
    unplug_hour_time = data.get('unplug_hourTime', {}).get('$date')
    return [age_range, unplug_hour_time]

# IDEA: Distinguir casos, entresemana y fin de semana
# weekday() method to get the day of the week as a number, where Monday is 0 and
    Sunday is 6.

def get_weekday(date_string):
    weekday = datetime.strptime(date_string, "%Y-%m-%d").weekday()
    return weekday

def main(filename):
    with SparkContext() as sc:

        # leer JSON archivo como RDD
        lines_rdd = sc.textFile(filename)

        # extrae los datos ageRange, unplug_hour_time y filtra ageRange = 0
        data_rdd = lines_rdd.map(extraer).filter(lambda x: x[0] != 0) #narrow
            transformation

        # IDEA: Distinguir casos, entresemana y fin de semana
        # Los 10 primeros str representan la fecha, lo que quiero extraer
        weekdays_rdd = data_rdd.filter(lambda x: get_weekday(x[1][:10]) < 5)
        weekends_rdd = data_rdd.filter(lambda x: get_weekday(x[1][:10]) >= 5)

        age_ranges = range(1, 7)
        with open("results.txt", "w") as output_file:

            for age_range in age_ranges:
                output_file.write(f"Weekdays {age_range}:\n")
                # Filter data for the current age range
                age_range_weekdays_rdd = weekdays_rdd.filter(lambda x: x[0] ==
                    age_range).\
                                        map(lambda x: (x[1][:13], 1)).\
                                        countByValue()

                age_range_weekends_rdd = weekends_rdd.filter(lambda x: x[0] ==
                    age_range).\
                                        map(lambda x: (x[1][:13], 1)).\
                                        countByValue()

                for time_interval, user_count in age_range_weekdays_rdd.items():
                    output_file.write(f" {time_interval} Users: {user_count}\n")

                output_file.write(f"Weekends {age_range}:\n")

                for time_interval, user_count in age_range_weekends_rdd.items():
                    output_file.write(f" {time_interval} - Users: {user_count}\n")

if __name__=="__main__":
    file = ""
    if len(sys.argv)>1:
        file = sys.argv[1]
    main(file)
```

10 Anexo código obtener_histogramas.py

```
from datetime import datetime
import matplotlib.pyplot as plt

'''

    TIPO DE TEXTO:

    Weekdays 1:
    ('2017-12-01T06', 1) Users: 1
    ('2017-12-01T07', 1) Users: 21

'''

data_list_weekdays = []
data_list_weekends = []

histogramas = []
with open("sample_spark.txt", 'r') as file:
    histograma = []
    for line in file:
        line = line.strip() # suprimir \n
        if line.startswith('Week'):
            if histograma != []:
                histogramas.append(histograma)
            histograma = [line,[]]
        #if not (line.startswith('Week')): # pasar los titulos
        else:
            #encuentro los indices de los str para extraer los datos que quiero

            time_start_index = line.find('T') + 1
            time_end_index = line.find("'", time_start_index)

            time = line[time_start_index:time_end_index]

            user_type_start_index = line.find(",␣") + 1
            user_type_end_index = line.find(")", user_type_start_index)

            user_type = line[user_type_start_index:user_type_end_index]

            users_start_index = line.find('U') + 6

            users = line[users_start_index:]
            histograma[1].append([int(time), int(users)])
            #separar entresemana y fin de semana
            #if datetime.strptime(line[2:12], "%Y-%m-%d").weekday() < 5:
            #hago una lista con los datos
            #    data_list_weekdays.append([int(time), int(users)])
            #else:
            #    data_list_weekends.append([int(time), int(users)])


#print(data_list_weekdays)
#print(data_list_weekends)

for histograma in histogramas:
        #Hago lista con sublistas[tiempo, usuarios] para las 24 horas
        suma_week = 24*[0]

        for sublist in histograma[1]:
            index = sublist[0]
            value = sublist[1]
            suma_week[index] += value  #sumo los usuarios en las mismas horas

        suma_list_week = [[i,suma_week[i]] for i in range(24)]
        #print(suma_weekdays)
        #print(f"SUMA {suma_list_weekdays}")

        x = [sublist[0] for sublist in suma_list_week] #TIEMPO  X - AXIS
        y = [sublist[1] for sublist in suma_list_week] #USUARIOS Y - AXIS
```

```
if 'day' in histograma[0]:
    plt.bar(x, y, color = "green")
    plt.xlabel('Time')
    plt.ylabel('Usuarios')
    plt.title(f'Histograma {histograma[0]}')
    plt.savefig(f'Histograma {histograma[0]}')
    plt.show()
else:
    plt.bar(x, y, color = "purple")
    plt.xlabel('Time')
    plt.ylabel('Usuarios')
    plt.title(f'Histograma {histograma[0]}')
    plt.savefig(f'Histograma {histograma[0]}')
    plt.show()
```