UNIVERSIDAD COMPLUTENSE DE MADRID
FACULTAD DE CIENCIAS MATEMÁTICAS

PRÁCTICA OBLIGATORIA PROGRAMACIÓN PARALELA: BiciMAD

Autores:

José Ignacio Alba Rodríguez
Álvaro Ezquerro Pérez
Alejandro Millán Arribas

Programación Paralela

2022-2023

Índice

1. Introducción

Esta práctica consiste en realizar un estudio sobre la base de datos de la información de uso del servicio *BiciMAD* porporcionada por el Ayuntamiento de Madrid. El objetivo principal es aplicar los métodos de *Spark* desarrollados en la asignatura para estudiar una problemática sobre este conjunto de datos y dar la solución correspondiente.

1.1. Problemática planteada

En nuestro caso nos hemos propuesto intentar diseñar un plan de medidas para la mejora del funcionamiento y eficiencia del servicio BiciMAD. Para elaborar esta serie de medidas nos hemos planteado una serie de cuestiones que consideramos puntos clave a mejorar para garantizar un mejor servicio a los clientes. En algunos casos estas cuestiones se han planteado desde el punto de vista de las necesidades de un cliente, y en otros casos está más enfocado hacia la búsqueda de una mayor eficiencia en la gestión por parte del responsable del servicio.

Las cuestiones que nos hemos planteado son las siguientes:

1. Determinar los trayectos más realizados y los menos.
2. Calcular la cantidad de bicicletas rotas.
3. Calcular cuáles son las estaciones más utilizadas y las que menos.
4. Determinar la necesidad de transferencia de cada estación
5. Estimar la hora punta de uso.
6. Calcular los porcentajes de uso en función del rango de edad y el tipo de usuario.
7. Intentar estimar el número de clientes habituales que tiene este servicio.

1.2. Base de datos

Hemos enfocado nuestro estudio sobre los datos comprendidos entre los meses de enero a junio de 2021 publicados por el Ayuntamiento de Madrid. Los datos aportados incluyen los siguientes atributos:

- **Tipo de usuario:**
  - 0: No se ha podido determinar el tipo de usuario.
  - 1: Usuario anual (poseedor de un pase anual).
  - 2: Usuario ocasional.
  - 3: Trabajador de la empresa.
- **Código de usuario.**
- **Número de la estación donde se desengacha la bicicleta.**
- **Número de la estación donde se enchancha la bicicleta.**

- **Número de la base de la que se desengancha la bicicleta.**
- **Número de la base en la que se engancha la bicicleta.**
- **Tiempo transcurrido entre el enganche y el desenganche de la bicicleta.**
- **Hora a la que se realiza el desenganche de la bicicleta:** El fórmato es: "2019-06-01T00:00:00Z".
- **Rango de edad del usuario:**
  - Desconocido.
  - <17.
  - 17-18.
  - 19-26.
  - 27-40.
  - 41-65.
  - >65.

1.3. Tratamiento de los datos

Como ya ha sido mencionado, para el tratamiento de los datos se utilizará la librería *Spark*. Para leer los archivos en formato *.json*, utilizamos el paquete *json* de Python. Estos datos se guardan en una estructura RDD que manejaremos a partir de ahora. El primer paso para acomodar el manejo de los datos será formatearlos de manera que sean más accesibles. Utilizaremos la siguiente función, que nos permite manejar por separado cada uno de los distintos atributos:

```
def mapper(line):
    data = json.loads(line)
    user_type = data['user_type']
    user_age = data['ageRange']
    user_day_code = data['user_day_code']
    start_station = data['idunplug_station']
    end_station = data['idplug_station']
    duration = data['travel_time']
    date = datetime.strptime(data["unplug_hourTime"], '%Y-%m-%dT%H:%M:%SZ')
    return user_type, user_day_code, start_station, end_station, duration, date, user_age
```

Una vez formateados los datos, podemos enfocarnos a responder cada una de las cuestiones planteadas.

2. Resolución de las cuestiones

A continuación veremos como hemos abordado cada una de las cuestiones y que conclusiones hemos obtenido de cada una de ellas. Para cada cuestión adjuntamos el código de la función que hemos empleado, aunque para un mayor detalle sobre el código es mejor mirar el archivo *bicimad.py*.

2.1. Determinar los trayectos más realizados y los que menos

La función que hemos diseñado para resolver esta cuestión es:

```
def rutas_ordenadas(rdd):
    rutas_ordenadas = rdd.map(lambda x: ((x[2], x[3]),1)).\
        reduceByKey(lambda x, y: x+y).\
        sortBy(lambda x: x[1], ascending = False)
    return rutas_ordenadas
```

Los resultados que hemos obtenido en cada uno de los ficheros son:

- **Enero 2021**:

```
Las 5 rutas mas repetidas son:
[(149, 149), (57, 57), (187, 187), (132, 132), (175, 175)]
realizadas cada una este numero de veces:
[154, 144, 111, 108, 95]
```

- **Febrero 2021:**

```
Las 5 rutas mas repetidas son:
[(132, 132), (57, 57), (220, 220), (135, 135), (212, 212)]
realizadas cada una este numero de veces:
[251, 233, 173, 169, 158]
```

- **Marzo 2021:**

```
Las 5 rutas mas repetidas son:
[(132, 132), (57, 57), (90, 90), (135, 135), (56, 56)]
realizadas cada una este numero de veces:
[374, 272, 262, 248, 214]
```

- **Abril 2021:**

```
Las 5 rutas mas repetidas son:
[(132, 132), (57, 57), (135, 135), (56, 56), (175, 175)]
realizadas cada una este numero de veces:
[341, 199, 196, 186, 185]
```

- **Mayo 2021:**

```
Las 5 rutas mas repetidas son:
[(132, 132), (220, 220), (135, 135), (175, 175), (57, 57)]
realizadas cada una este numero de veces:
[313, 296, 265, 255, 250]
```

- **Junio 2021:**

```
Las 5 rutas mas repetidas son:
[(135, 135), (132, 132), (57, 57), (64, 64), (175, 175)]
realizadas cada una este numero de veces:
[258, 249, 212, 193, 188]
```

Llama la atención que los trayectos más realizado son aquellos donde la estación de partida y la de llegada son la misma. Nos hemos planteado intentar descubrir cuál es la razón de esta situación. Una posibilidad podría ser que las bicicletas se utilizasen mayormente para dar paseos por la ciudad y por ello empiezan y terminan en el mismo punto. Sin embargo, observando la duración de muchos de estos a priori paseos, observamos que a menudo no alcanzan el minuto, lo que nos llevó a plantearnos si entonces podía ser el caso de que estos trayectos coincidieran con las veces que un usuario se ha encontrado con que la bicicleta que había escogido estaba averiada, lo cual estudiaremos en la siguiente cuestión.

Hemos repetido el análisis de esta cuestión, pero esta vez descartando los casos en los que el trayecto se considera que es trivial debido a una bicicleta estropeada, para lo cual hemos diseñado la siguiente función:

```
def rutas_ordenadas_no_triviales(rdd):
    rutas_ordenadas = rdd.map(lambda x: ((x[2], x[3]), x[4])).\
        filter(lambda x : (x[0][0] != x[0][1]) or (x[4] > 60) ).
           countByKey().\
        sortBy(lambda x: x[1], ascending = False)
    return rutas_ordenadas
```

Los resultados obtenidos en cada uno de los ficheros son:

- **Enero 2021**:

- **Febrero 2021:**

- **Marzo 2021:**

- **Abril 2021:**

- **Mayo 2021:**

- **Junio 2021:**

Conocer estos trayectos más frecuentes permite tener un mayor conocimiento sobre como estructurar los planes de urbanismo, de manera que estos trayectos puedan ser acompañados de carriles bicis para garantizar la seguridad de los ciclistas.

2.2. Calcular la cantidad de bicicletas rotas

En esta cuestión buscamos calcular el número de veces que se ha escogido una bicicleta y esta resultaba tener una avería, ver cuál es la frecuencia de estos sucesos y compararlos con los resultados anteriores. Para ello, vamos a suponer que existe una bicicleta rota cuando un usuario ha sacado una bicicleta y acto seguido la ha vuelto a dejar en su misma estación. Debemos estudiarlo de esta manera ya que las averías no se encuentran especificadas en la base de datos. Vamos a tomar para ello aquelos trayectos que lleguen a la misma estación de partida en menos de un minuto.

La función que hemos diseñado para resolver esta cuestión es:

```
def bicis_rotas(rdd):
    total = rdd.count()
    datos_rotas = rdd.filter(lambda x: (x[2]==x[3]) and (x[4]<60))
    estaciones_rotas = datos_rotas.map(lambda x: x[2]).countByValue().\
                                        sortBy(lambda x: x[1],ascending = False)
    return datos_rotas.count(), total, round(rotas/total, 4), estaciones_rotas
```

Los resultados obtenidos en cada uno de los ficheros ha sido:

- **Enero 2021:**

```
    El numero de veces que alguien ha cogido una bici rota es:
    2996
    de un total de viajes de:
    125642 (0.0238%)
```

- **Febrero 2021:**

```
    El numero de veces que alguien ha cogido una bici rota es:
    8634
    de un total de viajes de:
    262103 (0.0329%)
```

- **Marzo 2021:**

```
    El numero de veces que alguien ha cogido una bici rota es:
    14732
    de un total de viajes de:
    360684 (0.0408 %)
```

- **Abril 2021:**

```
    El numero de veces que alguien ha cogido una bici rota es:
    11195
    de un total de viajes de:
    341061 (0.0328 %)
```

- **Mayo 2021:**

```
    El numero de veces que alguien ha cogido una bici rota es:
    12688
    de un total de viajes de:
    414249 (0.0306 %)
```

- **Junio 2021:**

```
    El numero de veces que alguien ha cogido una bici rota es:
    11098
    de un total de viajes de:
    413370 (0.0268 %)
```

En efecto, la mayoría de las estaciones que tenían un mayor número de trayecto coinciden con aquellas en las que hay un mayor número de averías. Esto nos enseña que hay que descartar este tipo de outliers a la hora de hacer nuestros análisis de datos. A partir de estos datos podemos estudiar las zonas para las cuales un mayor número de personas se encuentran un problema en alguna bicicleta. Esto permite priorizar las labores de mantenimiento en estos lugares. Cabe destacar que una misma bicicleta estropeada puede ser escogida más de una vez, por lo que estos datos no nos dicen nada sobre el número de bicicletas estropeadas absoluto, pero si sobre el número de personas que se encuentren una bicicleta con alguna avería. Esto nos interesa más, pues las estaciones donde esto ocurra serán las que más tráfico tengan y solucionar este problema conseguirá contentar al mayor número de clientes del servicio.

2.3. Cuáles son las estaciones más utilizadas y las que menos

La función que hemos diseñado para resolver esta cuestión es:

```
def estaciones_ordenadas(rdd):
    Estaciones = rdd.flatMap(lambda x: [(x[2], 1), (x[3], 1)]).\
        reduceByKey(lambda x, y: x+y).\
        sortBy(lambda x: x[1], ascending = False)
```

```
    estacionesS = rdd.flatMap(lambda x: [(x[2], 1), (x[3], 1)]).\
        reduceByKey(lambda x, y: x+y).\
        sortBy(lambda x: x[1], ascending = True)
    return Estaciones, estacionesS
```

El resultado obtenido en cada uno de los ficheros es:

- **Enero 2021:**

```
Estas son las 5 estaciones mas transitadas:
[57, 43, 175, 208, 149]
con este numero de usos cada una:
[3276, 3180, 2352, 2337, 2176]
Ademas, estas son las 5  estaciones menos transitadas:
[257, 265, 266, 261, 267]
con este numero de usos cada una:
[84, 104, 112, 124, 125]
```

- **Febrero 2021:**

```
Estas son las 5 estaciones mas transitadas:
[43, 57, 175, 208, 132]
con este numero de usos cada una:
[6419, 6284, 4852, 4838, 4624]
Ademas, estas son las 5  estaciones menos transitadas:
[209, 266, 257, 258, 260]
con este numero de usos cada una:
[21, 190, 263, 322, 346]
```

- **Marzo 2021:**

```
Estas son las 5 estaciones mas transitadas:
[57, 43, 208, 132, 90]
con este numero de usos cada una:
[8544, 8239, 6622, 6494, 6449]
Ademas, estas son las 5  estaciones menos transitadas:
[209, 124, 257, 266, 28]
con este numero de usos cada una:
[79, 85, 413, 459, 474]
```

- **Abril 2021:**

```
Estas son las 5 estaciones mas transitadas:
[57, 43, 132, 90, 175]
con este numero de usos cada una:
[7747, 7722, 6179, 6085, 6013]
Ademas, estas son las 5  estaciones menos transitadas:
[209, 266, 257, 258, 225]
con este numero de usos cada una:
[48, 389, 409, 537, 624]
```

- **Mayo 2021:**

```
Estas son las 5 estaciones mas transitadas:
[43, 57, 175, 132, 208]
con este numero de usos cada una:
[9676, 9303, 7872, 7504, 7324]
Ademas, estas son las 5  estaciones menos transitadas:
[209, 266, 157, 257, 28]
con este numero de usos cada una:
[258, 264, 397, 508, 617]
```

- **Junio 2021:**

```
Estas son las 5 estaciones mas transitadas:
[43, 57, 175, 208, 132]
con este numero de usos cada una:
[9684, 9019, 7477, 7314, 7102]
Ademas, estas son las 5  estaciones menos transitadas:
[209, 266, 257, 28, 225]
con este numero de usos cada una:
[40, 447, 599, 612, 767]
```

A partir de estos resultados, se observa que la mayoría de estaciones más utilizadas coinciden las estaciones de salida y llegada de los trayectos cíclicos más populares. Sin embargo, hay algunas que no están en este grupo, por lo que podemos concluir que estas sirven como punto de partida o de llegada para multitud de viajes entre distintos puntos de la ciudad, y por ello serán las estaciones de las cuales haya que retirar bicicletas más a menudo o llevar más para cubrir todo el servicio. Conocer cuales son las estaciones más utilizadas relevante pues también serán las que sufran un mayor desgaste, por lo que sería necesario aplicar un mayor mantenimiento sobre ellas que sobre las demás estaciones. Además, cabría considerar la posibilidad de aumentar la capacidad de bicicletas en estas estaciones. Si son las que sufren mayor demanda, debe ser una prioridad que estas aseguren el servicio que los ciudadanos requieren. En cambio, en cuanto a las estaciones menos utilizadas podemos oberservar que son prácticamente las mismas en ambos casos, por lo que sería recomendable llevar a cabo campañas para incentivar el uso en las zonas donde se encuentran estas estaciones.

2.4. Determinar la necesidad de transferencia de cada estación

Los trayectos que empiezan y acaban en estaciones diferentes suponen un movimiento de bicicletas que puede ocasionar que una estación quede vacía mientras que otra estación llene su capacidad de almacenar bicicletas. Para evitar esta situación, se requerirá que los operarios transporten bicicletas desde las estaciones que más bicicletas reciban hacia las que queden más vacías. Por esta razón, nos hemos preguntado cuáles son las estaciones que tienen una mayor diferencia entre las bicicletas que reciben y las que salen.

Para resolver esta pregunta, hemos utilizado el siguiente código:

```
def salidaVsEntrada(rdd):
    dif_por_estacion =  rdd.filter(lambda x: (x[2]!=x[3]) and (x[0]
       != 3)).\
                        flatMap(lambda x: [(x[2],-1),(x[3],1)]).\
                        reduceByKey(lambda x,y : x+y).\
                        sortBy(lambda x: x[1], ascending = False)
    return dif_por_estacion
```

El resultado obtenido en cada uno de los ficheros es:

- **Enero 2021:**

```
Las estaciones con mayor superavit de bicicletas:
[220, 132, 212, 215, 213]
La cantidad de bicicletas que llegan superan a las que salen en:
[98, 85, 80, 72, 70]
Las estaciones con mayor deficit de bicicletas:
[156, 102, 117, 145, 93]
La cantidad de bicicletas que llegan superan a las que salen en:
[91, 55, 49, 49, 48]
```

- **Febrero 2021:**

```
Las estaciones con mayor superavit de bicicletas:
[132, 220, 135, 43, 187]
La cantidad de bicicletas que llegan superan a las que salen en:
[147, 114, 93, 90, 89]
Las estaciones con mayor deficit de bicicletas:
[141, 158, 145, 156, 77]
La cantidad de bicicletas que llegan superan a las que salen en:
[138, 118, 94, 90, 79]
```

- **Marzo 2021:**

```
Las estaciones con mayor superavit de bicicletas:
[220, 69, 132, 224, 177]
La cantidad de bicicletas que llegan superan a las que salen en:
[158, 113, 111, 102, 99]
Las estaciones con mayor deficit de bicicletas:
[200, 141, 156, 77, 154]
La cantidad de bicicletas que llegan superan a las que salen en:
[135, 127, 124, 121, 115]
```

- **Abril 2021:**

```
Las estaciones con mayor superavit de bicicletas:
[43, 220, 132, 135, 64]
La cantidad de bicicletas que llegan superan a las que salen en:
[153, 151, 142, 127, 95]
```

```
Las estaciones con mayor deficit de bicicletas:
[141, 154, 206, 140, 158]
La cantidad de bicicletas que llegan superan a las que salen en:
[263, 176, 123, 122, 107]
```

- **Mayo 2021:**

```
Las estaciones con mayor superavit de bicicletas:
[135, 132, 220, 177, 43]
La cantidad de bicicletas que llegan superan a las que salen en:
[288, 271, 201, 137, 136]
Las estaciones con mayor deficit de bicicletas:
[141, 145, 148, 140, 27]
La cantidad de bicicletas que llegan superan a las que salen en:
[173, 163, 145, 139, 114]
```

- **Junio 2021:**

```
Las estaciones con mayor superavit de bicicletas:
[132, 213, 215, 135, 220]
La cantidad de bicicletas que llegan superan a las que salen en:
[320, 298, 222, 207, 171]
Las estaciones con mayor deficit de bicicletas:
[156, 140, 141, 154, 148]
La cantidad de bicicletas que llegan superan a las que salen en:
[203, 183, 170, 156, 155]
```

Con estos resultados podemos analizar cuáles son las estaciones de las cuáles habrá que retirar bicicletas y cuáles son las estaciones que deben de recibir estas bicicletas por los operarios de *BiciMad*. A partir de estos resultados, podría plantearse un problema de flujo para optimizar los recursos en el transporte de bicicletas de una estación a otra.

Cabe destacar que muchas de las estaciones que aparecen en estos resultados son también las que aparecen en los trayectos más habituales. Si esto ocurre, quiere decir que a menudo se realiza el trayecto solo de ida, pero se opta por otro medio de transporte para la vuelta. Esto puede ocurrir debido a los horarios, ya que es más fácil coger la bicicleta por la tarde, pero al volver por la noche un medio de transporte como autobuses o metro resultarán más accesibles.

2.5. Determinar la hora punta de uso

La función que hemos diseñado para resolver esta cuestión es:

```
def horas_ordenadas(rdd):
    horas = rdd.map(lambda x: (x[5].hour, 1)).\
        reduceByKey(lambda x, y: x+y).\
        sortBy(lambda x: x[1], ascending = False)
    total = horas.map(lambda x: x[1]).sum()
```

```
    horas = horas.map(lambda x: (x[0], x[1]*100/total))
    return horas
```

El resultado obtenido en cada uno de los ficheros es:

- **Enero 2021:**

```
Las horas ordenadas en cuanto a mayor uso son:
[13, 17, 18, 16, 14]
con porcentajes:
[8.1334, 8.0881, 7.5118, 7.2651, 6.8202]
```

- **Febrero 2021:**

```
Las horas ordenadas en cuanto a mayor uso son:
[17, 13, 18, 16, 14]
con porcentajes:
[8.8801, 8.3952, 7.9301, 7.4238, 7.3002]
```

- **Marzo 2021:**

```
Las horas ordenadas en cuanto a mayor uso son:
[17, 18, 13, 16, 14]
con porcentajes:
[8.5659, 8.0358, 7.6347, 7.3155, 6.8783]
```

- **Abril 2021:**

```
Las horas ordenadas en cuanto a mayor uso son:
[17, 16, 12, 18, 15]
con porcentajes:
[8.9743, 8.8333, 7.652, 7.3359, 7.1615]
```

- **Mayo 2021:**

```
Las horas ordenadas en cuanto a mayor uso son:
[17, 16, 12, 18, 15]
con porcentajes:
[8.3155, 8.161, 7.2876, 7.0969, 6.646]
```

- **Junio 2021:**

```
Las horas ordenadas en cuanto a mayor uso son:
[17, 16, 18, 12, 15]
con porcentajes:
[8.055, 7.7642, 7.2579, 6.91, 6.2039]
```

Se puede observar claramente que las horas del día en las que se concentra el mayor uso del servicio son entre las 12:00h y las 18:00h, que coincide con las horas con más luz. Esto nos indica que a estas horas también debe ser cuando se concentre la mayor actividad de los servicios de mantenimiento, reparación, atención al cliente y demás, pues será cuando más averías, dudas e incidencias haya. Durante estas horas se debe garantizar el funcionamiento completo del sistema para mantener el mayor número de clientes satisfechos. Por otro lado, durante la mañana y al anochecer, estos servicios no serán tan necesarios. Esto permite una mayor organización de los horarios de los trabajadores de forma que se puedan optimizar los recursos.

2.6. Calcular los porcentajes de uso dependiendo del rango de edad y tipo de usuario

Las funciones que hemos diseñado para resolver esta cuestión son:

```
def edades_ordenadas(rdd):
    edades = rdd.map(lambda x: (user_ages[x[6]], 1)).\
        reduceByKey(lambda x, y: x+y).\
        sortBy(lambda x: x[1], ascending = False)
    total = edades.map(lambda x: x[1]).sum()
    edades = edades.map(lambda x: (x[0], x[1]*100/total))
    return edades
def tipos_ordenados(rdd):
    tipos = rdd.map(lambda x: (user_types[x[0]], 1)).\
        reduceByKey(lambda x, y: x+y).\
        sortBy(lambda x: x[1], ascending = False)
    total = tipos.map(lambda x: x[1]).sum()
    tipos = tipos.map(lambda x: (x[0], x[1]*100/total))
    return tipos
```

El resultado obtenido en cada uno de los ficheros es:

- **Enero 2021:**

```
Las edades de los usuarios ordenadas en cuanto a mas uso son:
[NaN, 27-40, 41-65, 19-26, <17, >65, 17-18]
con porcentajes:
[53.9963, 21.0145, 19.0422, 3.9079, 1.0928, 0.4974, 0.4489]
Los tipos de usuario ordenadas en cuanto a mas uso son:
[Usuario anual, Trabajador de empresa, NaN, Usuario ocasional]
con porcentajes:
[95.4394, 3.7901, 0.5444, 0.226]
```

- **Febrero 2021:**

```
Las edades de los usuarios ordenadas en cuanto a mas uso son:
[NaN, 27-40, 41-65, 19-26, <17, >65, 17-18]
con porcentajes:
```

```
[54.4523, 21.1341, 18.3466, 3.8328, 1.412, 0.5231, 0.2991]
Los tipos de usuario ordenadas en cuanto a mas uso son:
[Usuario anual, Trabajador de empresa, NaN, Usuario ocasional]
con porcentajes:
[94.7822, 4.2335, 0.7035, 0.2808]
```

- **Marzo 2021:**

```
Las edades de los usuarios ordenadas en cuanto a mas uso son:
[NaN, 27-40, 41-65, 19-26, <17, >65, 17-18]
con porcentajes:
[54.6143, 20.9724, 18.7327, 3.4235, 1.5218, 0.4808, 0.2545]
Los tipos de usuario ordenadas en cuanto a mas uso son:
[Usuario anual, Trabajador de empresa, NaN, Usuario ocasional]
con porcentajes:
[93.26, 5.4785, 0.7658, 0.4957]
```

- **Abril 2021:**

```
Las edades de los usuarios ordenadas en cuanto a mas uso son:
[NaN, 27-40, 41-65, 19-26, <17, >65, 17-18]
con porcentajes:
[55.0183, 20.9787, 18.8116, 3.3707, 1.0083, 0.5152, 0.2973]
Los tipos de usuario ordenadas en cuanto a mas uso son:
[Usuario anual, Trabajador de empresa, NaN, Usuario ocasional]
con porcentajes:
[94.1547, 4.6018, 0.7447, 0.4987]
```

- **Mayo 2021:**

```
Las edades de los usuarios ordenadas en cuanto a mas uso son:
[NaN, 27-40, 41-65, 19-26, <17, >65, 17-18]
con porcentajes:
[55.8058, 20.6822, 18.2624, 3.3106, 1.0991, 0.5115, 0.3283]
Los tipos de usuario ordenadas en cuanto a mas uso son:
[Usuario anual, Trabajador de empresa, NaN, Usuario ocasional]
con porcentajes:
[93.818, 4.6305, 1.0148, 0.5366]
```

- **Junio 2021:**

```
Las edades de los usuarios ordenadas en cuanto a mas uso son:
[NaN, 27-40, 41-65, 19-26, <17, >65, 17-18]
con porcentajes:
[55.5935, 20.6716, 18.6095, 3.1952, 1.1585, 0.4964, 0.2753]
Los tipos de usuario ordenadas en cuanto a mas uso son:
[Usuario anual, Trabajador de empresa, NaN, Usuario ocasional]
con porcentajes:
[94.6854, 3.6694, 1.1733, 0.472]
```

El porcentaje de uso según edades es casi idéntico en los tres casos. Por un lado tenemos que la inmensa mayoría de usuarios son de una edad menor de 26 años, es decir, jóvenes. Esto es algo muy positivo pues significa que el uso de la bici está muy extendido entre los más jóvenes y optan por ello como medio de transporte por encima de otros medios más contaminantes. Sin embargo, en cuanto a la población adulta y más mayor pues prácticamente no usan este servicio. Creemos que es importante intentar fomentar el servicio para las personas de este rango de edad.

2.7. Intentar estimar el número de clientes habituales que tiene este servicio

Las funciones que hemos diseñado para resolver esta cuestión son:

```
def to_listas(a):
    return [(a,1)]
def combinar_si_dentro_del_margen(listas, duracion, margen):
    b = True; i = 0
    while b and i < len(listas):
        media, num = listas[i][0]
        if media - margen <= duracion <= media + margen:
            listas[i] = ( (media*num + duracion)/(num+1) , num+1)
            b = False
        i += 1
    if b: listas.append( (duracion, 1) )
    return listas
def clientes_habituales(rdd, margen):
    clientes_horario = rdd.map(lambda x: ((x[0],x[2],x[3], x[5].hour
       , x[6]) , x[4]) ).\
            aggregateByKey(to_listas, lambda a, b :
               combinar_si_dentro_del_margen(a,b,margen))
    return clientes_horario
```

Los resultado obtenidos en cada uno de los ficheros son:

- **Enero 2021**:

- **Febrero 2021:**

- **Marzo 2021:**

- **Abril 2021:**

- **Mayo 2021:**

- **Junio 2021:**

3. Conclusión del plan de mejora y eficiencia para el servicio BiciMAD

Una vez que hemos estudiado los distintos aspectos a tener en cuenta del servicio de transporte BiciMAD vamos a proponer un plan con medidas a tener en cuenta para mejorar y aumentar la eficiencia de dicho servicio. Nuestro plan consta esencialmente de los siguientes puntos:

1. Potenciar el uso de BiciMAD como medio de transporte para desplazarse en la ciudad, es decir, que su uso principal no sea únicamente el de pasear en bicicleta por la ciudad, si no que sirva para sustituir a autobuses, metro, vehículo particular, etc. Este factor es importante para reducir el uso de medios de transporte más contaminantes y apostar por medidas más ecológicas y saludables, donde el servicio BiciMAD debe jugar un importante papel.

2. Las estaciones más utilizadas y que por tanto sufrirán un mayor desgaste como las estaciones 57, 43 o 175 deben contar con un mantenimiento más intenso y constante que el resto de estaciones. Además, se debe intentar, en la medida de lo posible, aumentar el número de bicicletas disponibles en estas estaciones ya que no debería permitirse que se queden sin servicio en ningún momento.

3. A las horas puntas de uso, es decir, entre las 12:00h y las 18:00h apróximadamente se debe contar con el cien por cien de los servicios de BiciMAD (reparaciones, atención al ciente, etc.), ya que en estos momentos es cuando se darán un mayor número de incidencias a resolver de forma simultánea. En caso de que aún así fuese insuficiente para mantener el funcionamiento del servicio, lo que se debería hacer es aumentar la capacidad de BiciMAD en todos los aspectos.

4. En cuanto a las horas puntas de uso, también se debe intentar garantizar, en la medida de lo posible, que sean accesibles en los meses de más calor, es decir, en verano. Ya que mucha gente que requiera de este servicio en los meses que nosotros hemos estudiado lo seguirá necesitando en los meses más calurosos. Como medidas para ayudar en estos casos proponemos cubrir las estaciones para que estas se encuentren a la sombra y que no sea un problema la alta temperatura de las bicicletas para su uso. Además, esta medida también sería útil en verano pues permitiría que no se mojasen las bicicletas en caso de lluvias. Otra medida sería proporcionar fuentes de agua potable en las estaciones para que los usuarios puedan evitar sufrir por la deshidratación en caso de altas temperaturas.

5. Por último, creemos importante llevar a cabo planes de atracción de usuarios para fomentar el uso de BiciMAD. Sería importante enfocar estos planes tanto en las zonas de las estaciones que sufren un menor, como en el sector de la población adulta que hemos visto que apenas dan uso al servicio BiciMAD.