Proyecto Análisis de Datos

Parte 3: Clustering

José Manuel Bustos Muñoz

Santiago Martínez De La Riva

1.- Seleccionar el número de clusters para K-means.

Lo primero que llevamos a cabo es la carga de los datos con los que vamos a trabajar en el proyecto, correspondientes al dataset de "Bike-Sharing-Dataset". Este dataset tiene como objetivo predecir el uso diario y horario de un sistema de alquiler de bicicletas basándose en datos climatológicos, día de la semana, temporada, y otra serie de variables. El dataset incluye dos años de uso de bicicletas del sistema público de Washington DC.

Cargamos el dataset resultante de prácticas anteriores donde se han eliminado o añadido algunas variables.

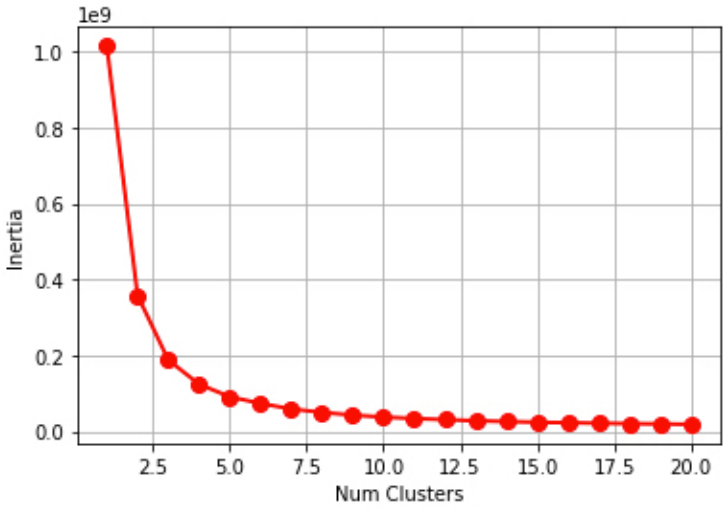
El dataframe resultante es el siguiente:

	season	yr	mnth	hr	holiday	weekday	workingday	weathersit	temp	hum	windspeed	casual	cnt	season_weather	day	registered	cnt_class
0	1	0	1	0	0	6	0	1	0.24	0.81	0.0	3	16	1	1	13.0	0
1	1	0	1	1	0	6	0	1	0.22	0.80	0.0	8	40	1	1	32.0	0
2	1	0	1	2	0	6	0	1	0.22	0.80	0.0	5	32	1	1	27.0	0
3	1	0	1	3	0	6	0	1	0.24	0.75	0.0	3	13	1	1	10.0	0
4	1	0	1	4	0	6	0	1	0.24	0.75	0.0	0	1	1	1	1.0	0

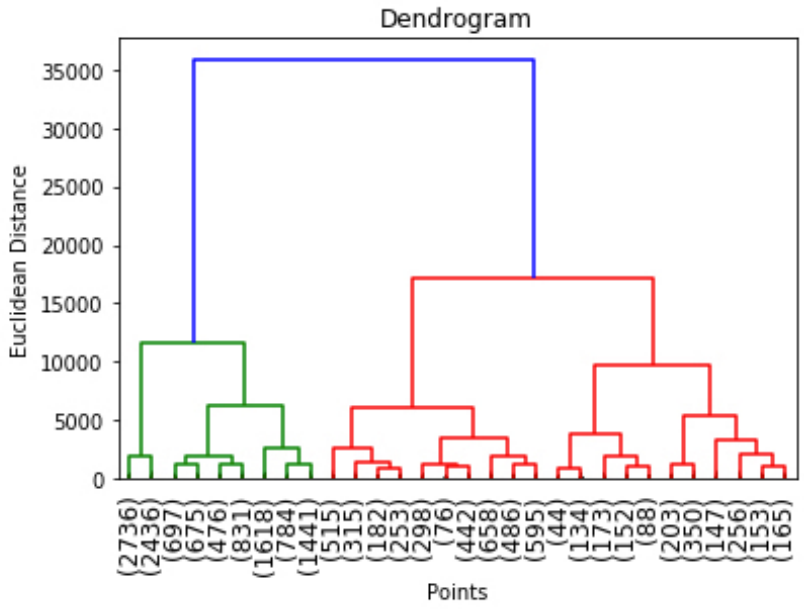
Vamos a aplicar 3 técnicas para seleccionar el número de clusters que posteriormente aplicaremos en el algoritmo k-means.

1. Una primera opción para seleccionar el número de clusters se basa en el **cómputo de la inercia de cada uno de los clusters**.

Se prueba con distintos candidatos de número de cluster, y se calcula la inercia global. Esa inercia no es sino el valor medio de la varianza de cada cluster respecto a su centroide.

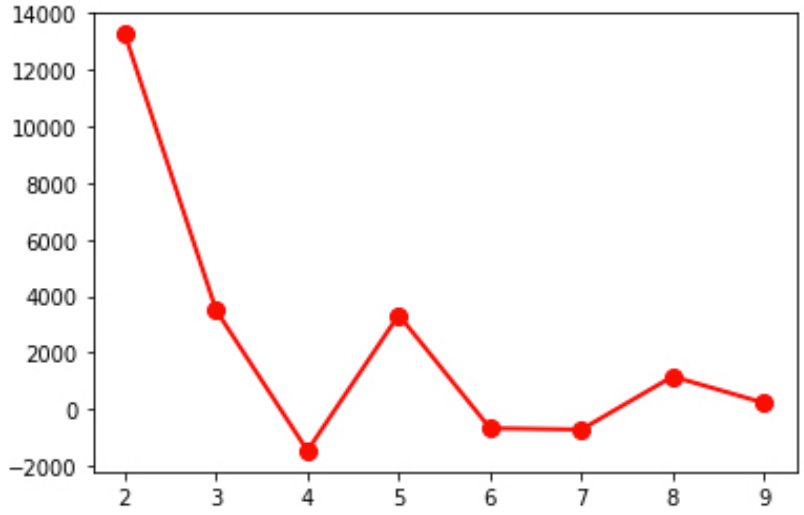


2. Los **dendrogramas** también nos pueden ser de ayuda a la hora de determinar posibles candidatos para el número de clusters:



3. Finalmente, se puede medir la distancia existente entre clusters. Se buscará el valor de cluster que maximiza la **distancia inter-cluster**.

Aquí nos apoyamos de nuevo en el dendrograma, pues nos permite medir en cada nivel la distancia que existen entre las diversas ramas. En cada uno de los niveles nos quedamos con las últimas 10 distancias.



Después de aplicar cada uno de los métodos referidos anteriormente, vamos a proceder a analizar los resultados obtenidos en cada una de ellos:

- **Dendrograma**: A través de esta técnica podemos ver como claramente tenemos dos grupos, los cuales están ramificados a través de la agrupación definida con la línea azul. Si nos fijamos la distancia que hay si creamos una linea horizontal entre el top azul que está en 35.000 y el grupo rojo y verde, que están en torno a 12.000 y 18.000, es muy grande, es por esta razón que consideramos que sería mejor tener dos clusters, pues la distancia entre el cluster rojo y el verde es menor, que entre el azul y las las otras dos ramas.

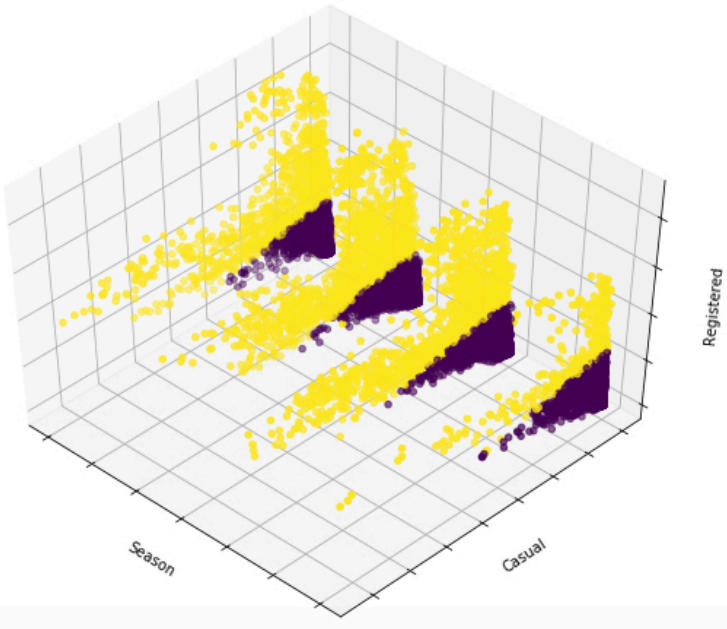
- **Elbow**: Para este caso se nos hace más complicado determinar el número de clusters que deberíamos seleccionar. El problema es que a través del gráfico está claro que la diferencia entre el punto 1 y el 2, es máxima, y es en el punto 2 a partir del cual empieza a formarse la curva, pero si nos fijamos en el siguiente punto que está entorno al 3, parece que la distancia entre cada punto es menor. Por lo que podríamos tener la duda en este caso si el número de cluster a definir sería 2 o 3.

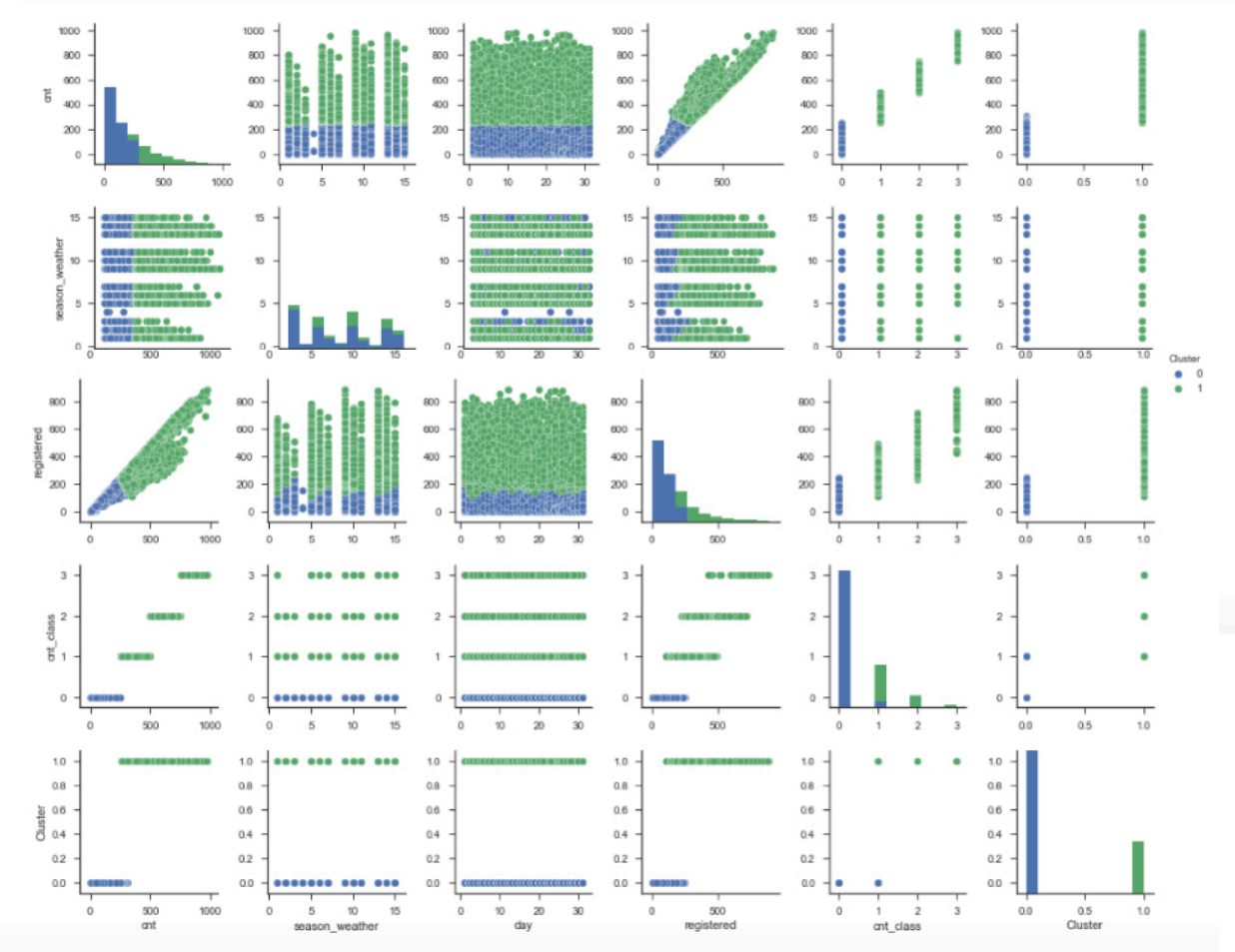
- **Maximizar la distancia entre clusters**: Gracias a esta técnica tenemos mayor seguridad que el número de clusters a definir es 2, pues podemos ver claramente en la gráfica como es en el cluster 2 donde la distancia máxima es mayor, en torno a los 13.800, siendo en consecuencia el número de clusters que deberíamos definir.

Estas tres técnicas nos sirven para orientarnos en el número de clusters que queremos definir, pero de forma individual no son determinantes. Es por esta razón que haciendo un análisis de los resultados obtenidos en el codo de la curva de inercias, el dendrograma y la técnica de maximizar la distancia entre clusters, consideramos que el número de clusters a definir es de 2. El estudio conjunto de todas las técnicas nos permiten llegar a una mejor conclusión.

Una vez seleccionado el número de clusters aplicamos a los datos el algoritmo **k-means**. Cuando se ha aplicado y separa los datos en los 2 clusters, se añade al dataframe una columna con el cluster a cada registro. Además podemos hacer algunas representaciones gráficas para ver la distribución de datos por cluster como podría ser en 3D o en scatter-plot.

Por último, con el uso de describe(), analizamos los datos de cada cluster para ver las distintas medidas estadísticas que nos ayudan a comprender como ha separado los datos el algoritmo aplicado.





Analizando la representación gráfica mediante scatter plot de los datos diferenciados por cluster, y las medidas estadísticas obtenidas con el uso de describe(), con el principal objetivo de sacar conclusiones respecto a cómo **k-means** ha separado los datos en dos clusters, observamos que:

Claramente podemos apreciar que el cluster 0, tiene aproximadamente el triple de registros que el cluster catalogado como 1. También podemos ver, como las variables "*casual*", "*cnt*" y "*registered*" es donde vemos las mayores diferencias a nivel estadístico, pues la medias de estos atributos en el cluster 0 son mucho menores a las obtenidas en el cluster 1.

Por otro lado nos encontramos un cluster con más registros y un uso pequeño y moderado del servicio de bicis, y otro con menor número y un uso muy alto. Es el cluster que tiene mayor uso el que está ligado a mejores condiciones climatológicas, aspecto que se puede apreciar en los datos estadísticos de las variables "*temperatura*" y "*humedad*".

Por todo lo expuesto podemos inferir que lo que más ha influido en la separación de los 2 clusters es el uso del servicio de bicicletas reflejado en las variables analizadas.

Posteriormente se ha vuelto a realizar el cálculo de **k-means**, pero antes **normalizando** el dataframe. Vemos como la separación en 2 clusters arroja unos clusters algo más igualados en número a la anterior prueba realizada. Suponemos que esto es debido a que algunos atributos tienen menos peso que antes e influyen algo menos a la hora de separar los datos.

2.- Comparar los resultados obtenidos mediante K-means con los resultantes de aplicar clustering jerárquico y GMM al conjunto de datos considerado. Analiza el tiempo de ejecución de cada procedimiento.

En el segundo ejercicio vamos a aplicar distintos métodos de Clustering a los datos y también analizaremos el tiempo de cada uno.

Aplicamos el método de clustering **agglomerative**, y el **GMM**, y comparamos los resultados teniendo en cuenta la separación obtenida anteriormente con k-means.

Uso de **agglomerative**:

```
t0 = time()

# Clustering jerárquico aglomerativo
single_link = AgglomerativeClustering(linkage='ward',n_clusters=2)

h_clusters = single_link.fit(X)

print("done in %0.3fs" % (time() - t0))
```

```
done in 15.865s
```

Reparto por cluster obtenido con agglomerative:

```
# recuento por cluster de lo obtenido con aglomerativo
pd.value_counts(df_bike_aglom['Cluster'])
```

```
1    11694
0     5685
Name: Cluster, dtype: int64
```

Uso de **GMM**:

```
df_bike_3 = carga_datos()
X_3 = df_bike_3.values

t0 = time()

gmm = mixture.GaussianMixture(n_components=2).fit(X_3)

print("done in %0.3fs" % (time() - t0))
```

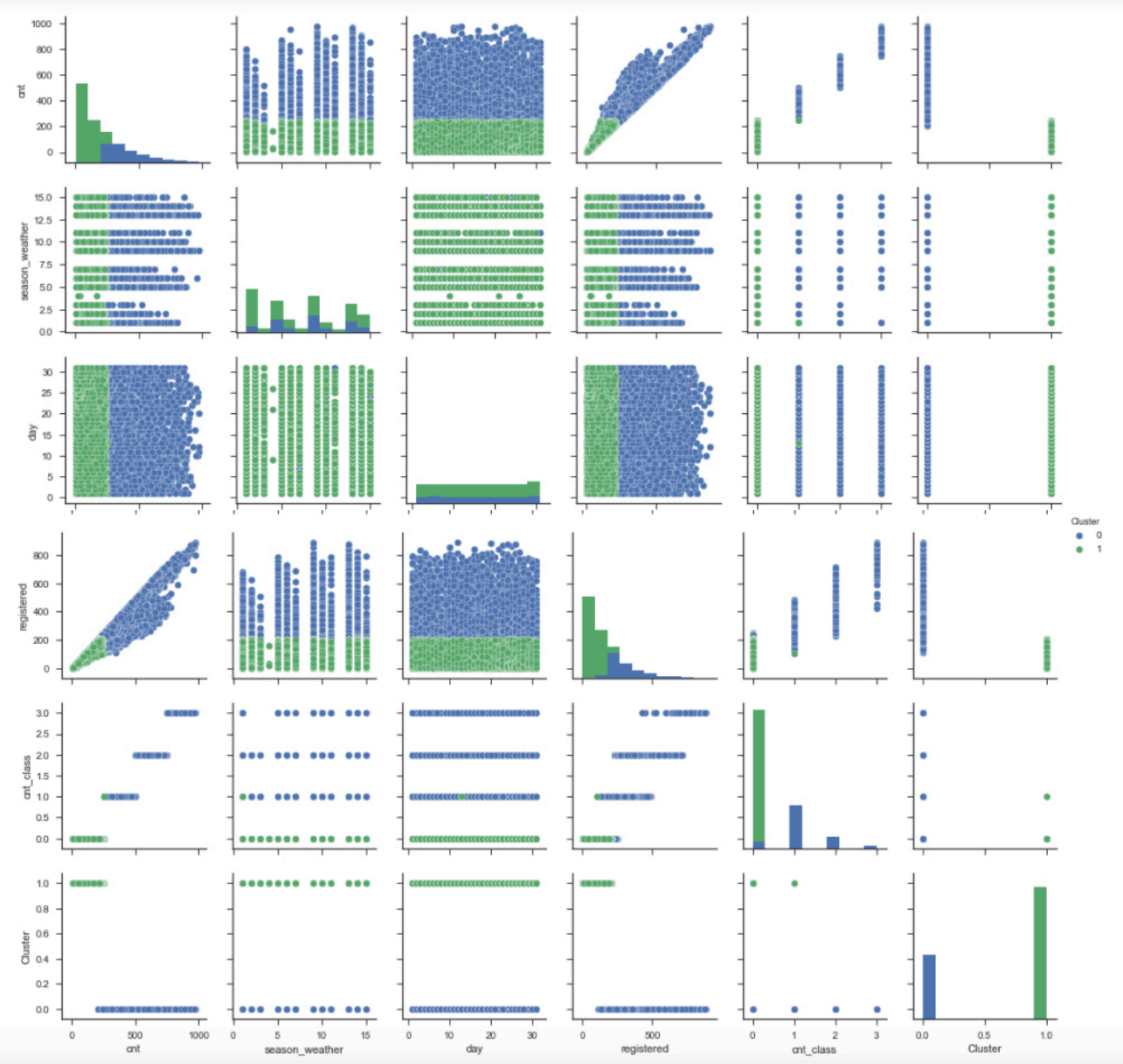
```
done in 0.682s
```

Reparto por cluster obtenido con GMM:

```
# recuento por cluster de lo obtenido con gmm
pd.value_counts(df_bike_gmm['Cluster'])
```

```
0    12237
1     5142
Name: Cluster, dtype: int64
```

Ejemplo de representación en scatter-plot de los datos repartidos por cluster obtenidos con el método agglomerative:



Una vez tenemos los dataframes obtenidos con el reparto por cluster, con el uso de describe() también comparamos los datos de agglomerative y de GMM:

Si nos fijamos comparando las medidas para el cluster 0 de **aglomerativo** y el cluster 1 de **gmm**, ambos son los clusters que tienen menor número de registros, y podemos apreciar como los datos se reparten de forma muy parecida, e igual que con K-means las variables que contiene la información de recuento del uso del servicio son las que influyen para el reparto de los datos por cluster.

También al igual que ocurría al comparar los clusters con menor número de registros, nos damos cuenta que obtenemos las mismas conclusiones si nos fijamos en los clusters de cada método en los cuales tenemos mayor número de registros, ambos también son muy parecidos en cuanto a su distribución.

Por lo que ambos métodos y junto a k-means hacen un reparto similar de los datos.

Comparación de los datos obtenidos con aglomerativo, GMM y lo obtenido anteriormente con K-means:

*Reparto de los **registros*** por cluster:

- **Aglomerativo**:
  - Cluster 0: 5685
  - Cluster 1: 11694
- **GMM**:
  - Cluster 0: 12237
  - Cluster 1: 5142
- **K-means**:
  - Cluster 0: 12776
  - Cluster 1: 4603

Tiempos obtenidos al ejecutar los algoritmos:

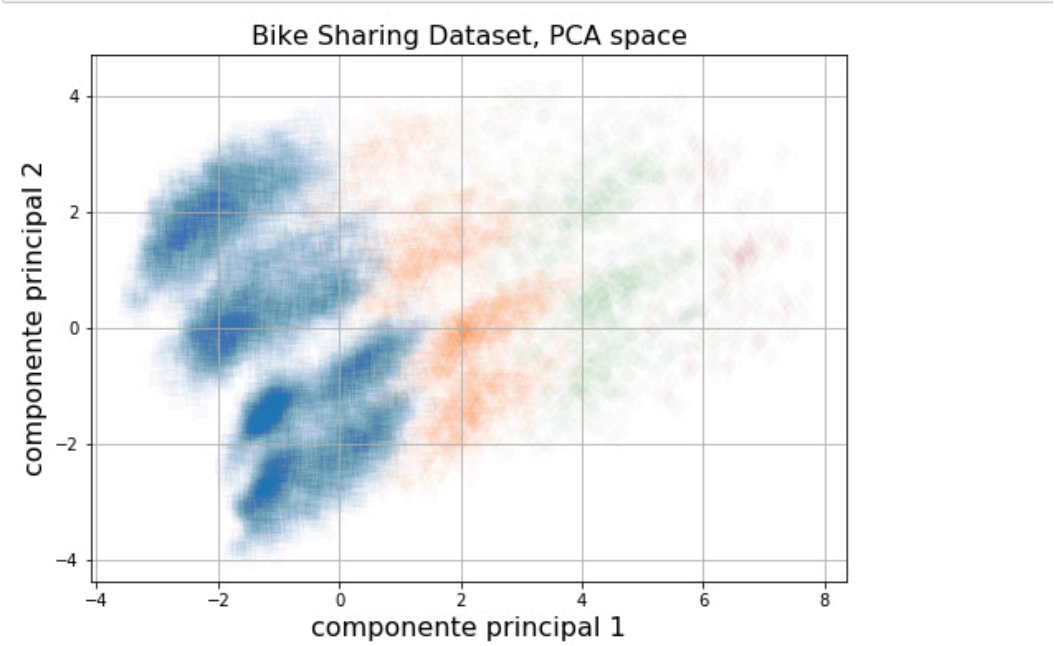
- **Aglomerativo**:
  - Sin usar kneighbors: 15.865s
  - Con kneighbors: 6.059s
- **GMM**: 0.682s

3.-Aplicar K-means al resultado obtenido en el apartado 3 de la práctica 2.

Ahora vamos a aplicar K-means a lo obtenido de aplicar PCA en una práctica anterior.

Aplicamos PCA a nuestros datos, reduciendo la dimensionalidad y características del problema. Obtenemos unos datos de cada componente principal y graficamos las dos primeras y más importantes de estas componentes principales.

```
-- Estadísticas de los datos proyectados en las componentes principales --

* Datos en componente principal 1 :
  - Media    : -0.0
  - Varianza : 4.146
  - Autovalor: 4.146
  - Varianza explicada: 24.386 %

* Datos en componente principal 2 :
  - Media    : -0.0
  - Varianza : 2.869
  - Autovalor: 2.869
  - Varianza explicada: 16.875 %

* Datos en componente principal 3 :
  - Media    : 0.0
  - Varianza : 1.397
  - Autovalor: 1.397
  - Varianza explicada: 8.215 %

* Datos en componente principal 4 :
  - Media    : -0.0
  - Varianza : 1.207
  - Autovalor: 1.207
  - Varianza explicada: 7.101 %

* Datos en componente principal 5 :
  - Media    : 0.0
```



Una vez se ha aplicado PCA como se hizo en la anterior práctica aplicamos K-means, y obtenemos el dataframe repartido en 2 clusters.

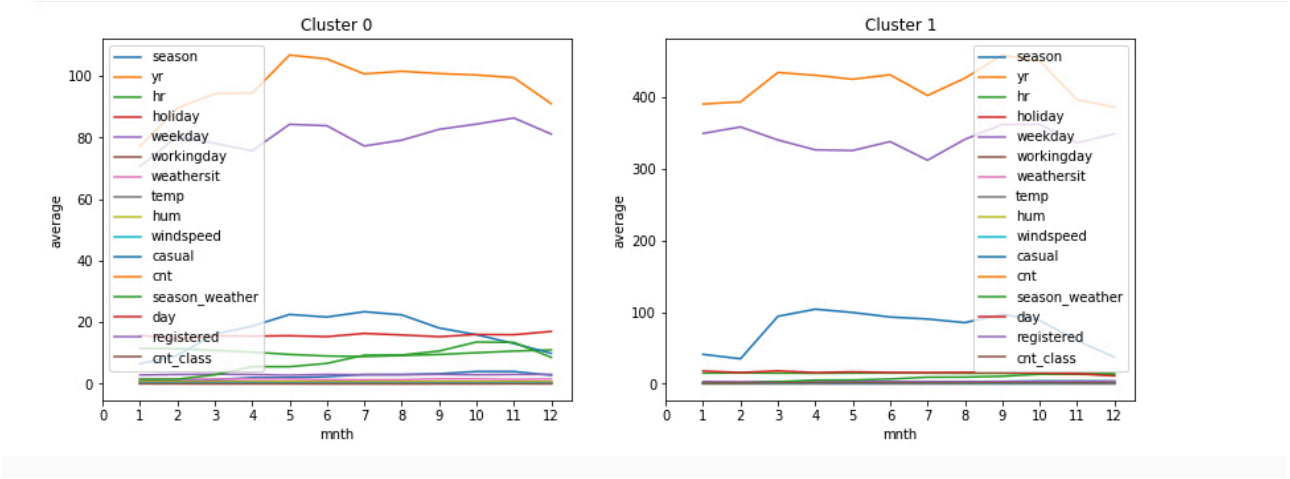
Recuento de registros por cluster:

```
# ver los registros de cada cluster
pd.value_counts(df_bike_kmeans['Cluster'])
```

```
0    12425
1     4954
Name: Cluster, dtype: int64
```

Vemos como el reparto es muy similar a lo obtenido en anteriores ejercicios cuando se uso K-means.

Para dibujar los datos de cada cluster agrupamos mediante “groupBy” por cluster, y la variable mes, y pintamos en un par de gráficas la evolución de la media de cada atributo por mes separada por cluster.



Podemos analizar como las variables "*cnt*" y "*registered*", están muy separadas del resto con medias de valores más dispares, y como en cada uno de los clusters toman valores claramente distintos. Esto de nuevo nos permite corroborar que son las variables más determinantes a través de las cuales los diferentes métodos llevan a cabo la separación por clusters.

Por otro lado, concluimos que no hay gran diferencia al aplicar K-means tras PCA, que a lo obtenido en anteriores ejercicios de la práctica, debido a que PCA tiene como objetivo la compresión de la T de características, mientras que la agrupación tiene como objetivo la compresión de los N datos de puntos.

Creamos un par de samples de los datos para calcular sobre ellos un **score** que nos ayude a calcular el número óptimo de clusters:

```
# hacemos samples del dataset para reducir su tamaño y no tener problemas con el método silhouette_score y poder
# ver según esta medida score el número óptimo de clusters.
sample_1 = X.sample(frac=0.2, replace=True)
sample_1.head()
```

	season	yr	mnth	hr	holiday	weekday	workingday	weathersit	temp	hum	windspeed	casual	cnt	season_weather	day	registered	cnt_class	Clus
4160	3	0	6	6	0	1	1	2	0.62	0.78	0.1642	4	94	10	27	90.0	0	
3443	2	0	5	9	0	6	0	1	0.64	0.73	0.2537	61	177	5	28	116.0	0	
15344	4	1	10	13	0	6	0	2	0.64	0.57	0.5224	310	710	14	6	400.0	2	
11852	2	1	5	1	0	1	1	2	0.60	0.60	0.2836	5	11	6	14	6.0	0	
16782	4	1	12	1	0	5	1	2	0.24	0.70	0.1940	2	28	14	7	26.0	0	

Una vez tenemos los dos samples, calculamos sobre ellos PCA y de nuevo K-means y obtenemos al final los scores de cada uno para un número de clusters desde 2 a 6. El score utilizado es el método "***silhouette_score***".

```
# cálculo de componentes principales con PCA para sample 1
pca = PCA(n_components = 2)
pca.fit(sample_1)

reduced_data = pca.transform(sample_1)

reduced_data = pd.DataFrame(reduced_data, columns = ['Dimension 1', 'Dimension 2'])
```

```
# Calcular scores de distintos números de clusters para el sample 1
for i in range(2, 6):
    KMeans_score(reduced_data, n = i)
```

```
Número de clusters:  2
Score:  0.6041

Número de clusters:  3
Score:  0.5687

Número de clusters:  4
Score:  0.5435

Número de clusters:  5
Score:  0.5366
```

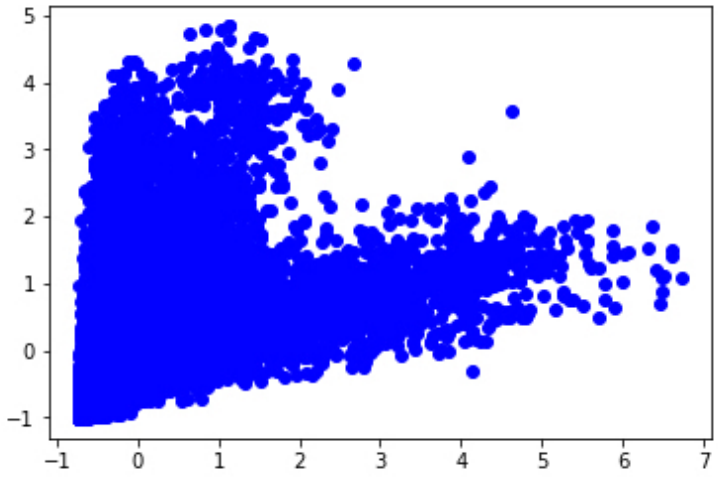
Después de hacer el estudio en los dos samples sobre el dataset de bicis, podemos ver como el máximo score para cada uno de ellos es cuando el número de cluster es **2**. Luego conseguimos verificar los resultados obtenidos en el ejercicio 1.

4.- Comparar la detección de outliers de la práctica 2 con el resultado de aplicar DBSCAN sobre el conjunto de datos considerado.

Vamos a aplicar **DBSCAN** sobre los datos. Para ello antes nos quedamos con las variables continuas, escalamos los datos y para poder dibujarlo cogemos 2 atributos "*casual*" y "*registered*" que indican el uso del servicio. Hemos seleccionado estas variables pues hemos visto que son las variables que mayor importancia tienen a la hora de clasificar los registros del dataset.

	temp	hum	windspeed	casual	cnt	registered
0	0.24	0.81	0.0	3	16	13.0
1	0.22	0.80	0.0	8	40	32.0
2	0.22	0.80	0.0	5	32	27.0
3	0.24	0.75	0.0	3	13	10.0
4	0.24	0.75	0.0	0	1	1.0

Dibujamos los atributos seleccionados:

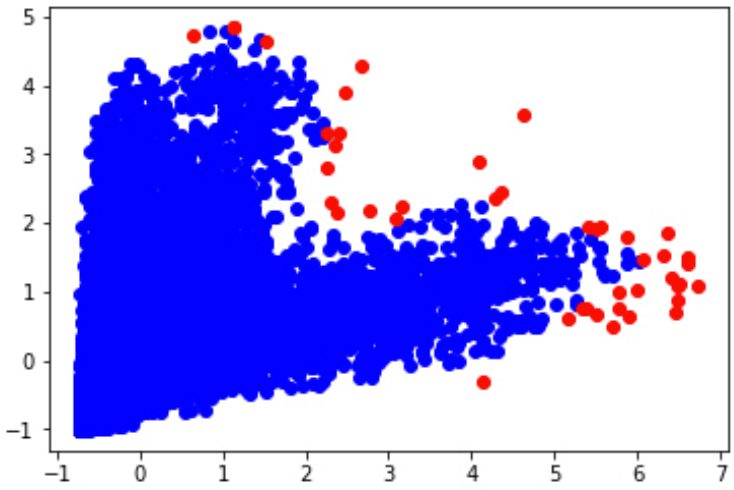


Introducimos y hacemos uso de la función de detección de datos outliers de la práctica anterior. Esta función nos devuelve los registros que pueden ser considerados ***outliers*** para cada atributo según los parámetros de la función.

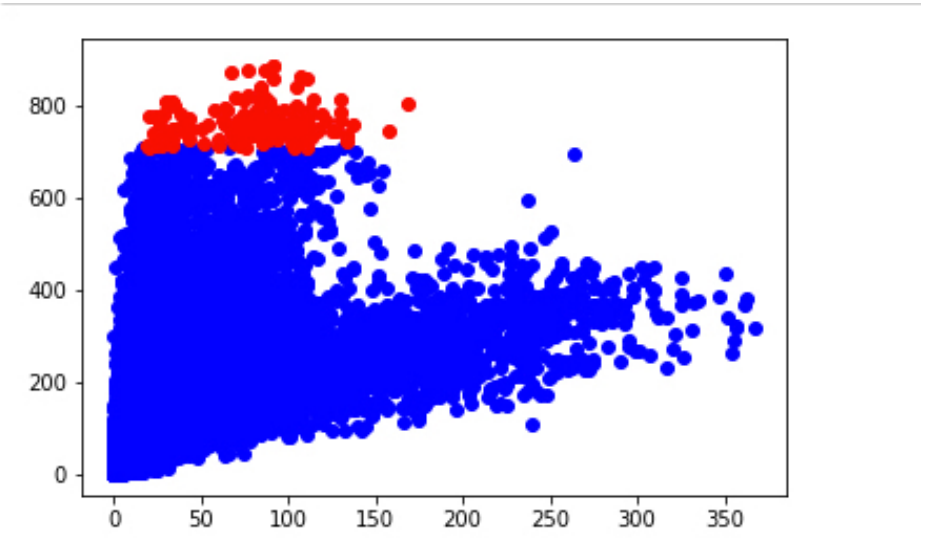
```
Puntos considerados outliers para el atributo 'casual':
```

	casual	registered
2128	219	148.0
2129	240	109.0
2463	181	162.0
2465	179	209.0
2608	182	209.0
2610	180	246.0
2631	205	236.0
2632	197	223.0
2775	185	270.0
2776	184	268.0

Una vez se aplica DBSCAN a los datos, podemos dibujar de nuevo los mismos y ya apreciamos algunos datos con un color diferente:



Así mismo podemos dibujar los datos aplicando en lugar de DBSCAN la función de ***detección de outliers*** comentada anteriormente:



Se puede apreciar como con el uso de DBSCAN se obtienen unos resultados diferentes a la hora de considerar un punto como outlier si lo comparamos con la función de detección de outliers que hemos usado en la práctica anterior. Esto se debe a que DBSCAN es un método basado en la densidad que agrupa en conjuntos de una alta densidad separadas por regiones poco densas.

Sin embargo la función que aplicamos para detectar outliers se basa en calcular para cada atributo a analizar, un percentil que deje % por debajo, basado en el reparto de los valores por atributo para separar los menos habituales de los más habituales.