INFORME DE CAMBIO CLIMÁTICO EN ESPAÑA

Introducción del informe sobre la influencia del cambio climático en la actividad sísmica y volcánica

Análisis sobre el cambio climático en España en base a registros de temperaturas máximas

Visualización de datos registrados
Predicción climática
Ejercicio de clasificación de nuevo registro

cristina vidal
Diciembre 2022

Informe cambio climático a partir de registro de temperaturas máximas

1. **Datos previos y alcance del proyecto.**
2. **Plots y visualización de datos.**
    a. **Visualización básica: Scatterplot (nube de dispersión/puntos)**
    b. **Histograma de frecuencias**
    c. **Boxplot: Diagtrama de cajas y bigotes**
3. **Predicción climática en base a análisis de registros históricos.**
    a. **Regresión lineal**
    b. **Regresión logística**
    c. **Curvas ROC (validación del modelo)**
4. **Clasificación de ubicación (estación meteorológica) de un nuevo registro en base a su analogía con los registros históricos.**
    a. **Clustering**
    b. **Árboles y bosques aleatorios (random forest)**
        i. **Bosque de regresión - Promedio**
        ii. **Bosque de clasificación – Voto por mayoría**
    c. **Máquinas de soporte vectorial (SVM)**
5. **K Nearest Neighbors a partir de datos clusterizados**
6. **Redes neurales y Tensor Flow**

---

1. Datos previos y alcance del proyecto.

El presente informe se ha realizado en base a los datos proporcionados por la Agencia Estatal de Meteorología AEMET, en su página Open Data:

AEMET OpenData - Agencia Estatal de Meteorología - AEMET. Gobierno de España

Se ha recogido información de los datos disponibles que se enumeran a continuación, según el escenario 20C3M y el proyecto AEMET, con salida en formato txt:

Variable	Escenario	Proyecto	Formato	Fichero
Tª máxima	20C3M	AEMET	TXT	Descargar fichero (4 MB)
Tª mínima	20C3M	AEMET	TXT	Descargar fichero (4 MB)
Precipitación total acumulada	20C3M	AEMET	TXT	Descargar fichero (3 MB)
Velocidad del viento a 10m	20C3M	AEMET	TXT	Descargar fichero (3 MB)
Velocidad máxima del viento a 10m	20C3M	AEMET	TXT	Descargar fichero (3 MB)
Humedad relativa	20C3M	AEMET	TXT	Descargar fichero (4 MB)
Percentil 95 de la temperatura máxima diaria	20C3M	AEMET	TXT	Descargar fichero (421 KB)
Percentil 5 de la temperatura mínima diaria	20C3M	AEMET	TXT	Descargar fichero (386 KB)
Percentil 95 de la precipitación diaria	20C3M	AEMET	TXT	Descargar fichero (415 KB)
Precipitación máxima en 24h	20C3M	AEMET	TXT	Descargar fichero (4 MB)
Nº de días con precipitación <1mm	20C3M	AEMET	TXT	Descargar fichero (1 MB)
Nº de días con precipitación >20mm	20C3M	AEMET	TXT	Descargar fichero (612 KB)
Máximo Nº de días consecutivos con precipitación <1mm	20C3M	AEMET	TXT	Descargar fichero (147 KB)

En esta primera fase del estudio, se han descargado los datos de temperatura máxima, que corresponden al valor promedio mensual, y se van a analizar utilizando técnicas de machine learning para analizar la influencia que tiene la temperatura máxima en el cambio climático.

En otras fases del estudio se repetirá este análisis con otros valores registrados como temperatura mínima, precipitación, velocidad del viento o humedad relativa.

El análisis se ha realizado mediante técnicas de machine learning, aplicadas sobre los datos recogidos en cerca de 350 estaciones meteorológicas ubicadas en España desde 1960, y en este documento se presentan los resultados más relevantes.

Ha resultado imposible asociar cada estación meteorológica con una ubicación concreta y sus coordenadas (longitud, latitud, altura) porque el archivo maestro que debe asociar cada ubicación con una estación por medio del código, que no se corresponde con las estaciones que aparecen en la tabla de temperaturas máximas por año/mes.

Alguna estación está repetida, hay estaciones situadas en Canarias cuando la documentación indica que la cuadrícula sólo incluye la península y Baleares, resulta complicado identificar si hay valores repetidos en las estaciones y faltan muchas lecturas que se han corregido a mano de forma que se ha complicado el análisis, pero finalmente se podido realizar el estudio.

La primera visualización correcta de los datos ha sido la que se ve a continuación, se aprecia que falta un encabezado que indique lo que hay en cada una de las columnas, de modo que la primera fila aparece como encabezado.

Antes, se ha comprobado que este dataset, de temperatura máxima, estaba enlazado con el fichero maestro de localizaciones.

```
data.head()
```

	196101	16.08	16.33	16.08.1	16.08.2	16.22	16.12	16.01	16.36	16.50	...	11.79.4	11.79.5	12.54.3	12.54.4	11.36.3	10.84.1	10.26	13.19.2	13.00.1	12.98
0	196102	15.29	15.45	15.56	15.56	15.56	15.31	14.84	16.13	16.09	...	10.76	10.70	11.37	11.37	10.34	9.81	9.34	12.40	12.23	12.13
1	196103	15.88	16.35	16.64	16.64	17.47	17.03	15.89	17.33	18.14	...	11.03	11.03	11.69	11.69	10.63	9.96	9.35	12.55	12.35	12.18
2	196104	16.13	17.15	16.57	16.57	17.96	17.82	16.24	17.27	17.71	...	11.55	11.71	12.47	12.47	11.16	10.46	9.68	12.65	12.36	12.30
3	196105	18.68	20.01	20.31	20.31	23.24	22.28	18.15	22.17	23.90	...	13.10	13.18	14.29	14.29	12.56	11.63	10.63	14.28	14.13	14.06
4	196106	22.32	22.93	24.75	24.75	27.97	25.82	20.41	27.83	30.61	...	15.39	15.38	16.39	16.39	14.53	13.57	12.53	16.33	16.13	15.85

5 rows × 1446 columns

Primera visualización tabla Tª máxima

Identificación de Localidad a partir del Codigo AEMET

```
data2.shape
```

```
(1124, 5)
```

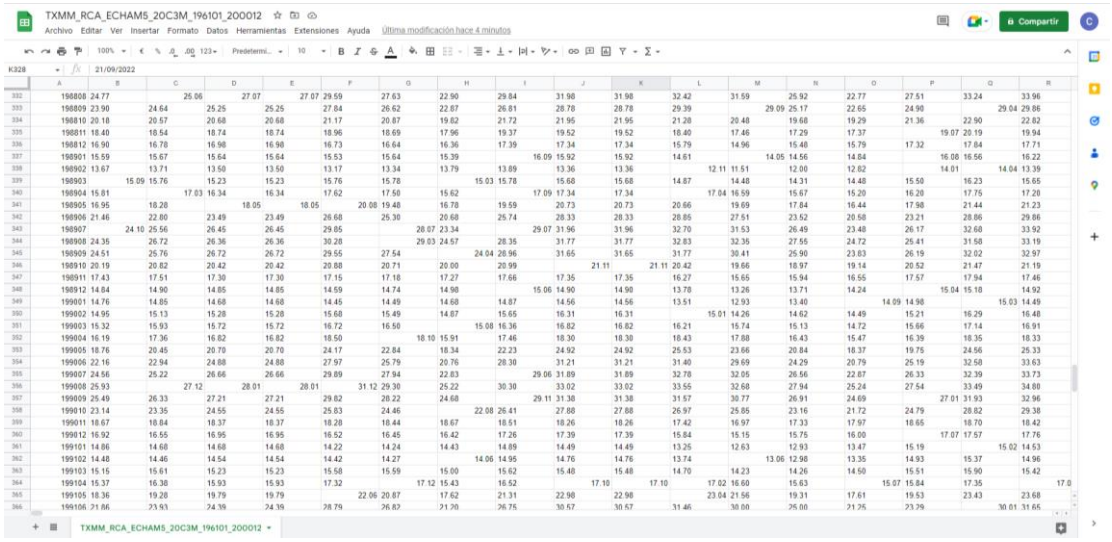
```
data2.tail()
```

	CodigoAEMET	Localidad	Longitud	Latitud	Altura
1119	C449G	ANAGA-SAN ANDRES	-16.1839	28.4967	20
1120	C458A	TACORONTE-A . S.E.A.	-16.4092	28.4864	327
1121	C649I	TELDE/AEROPUERTO DE G.CANARIA (GANDO)	-15.3889	27.9292	24
1122	C659P	LAS PALMAS DE GRAN CANARIA-JUNTA DE OBRAS DEL...	-15.4167	28.15	15
1123	C665I	VALLESECO-CASCO	-15.5736	28.0472	980

Visualización fichero maestro

Se ha añadido a la tabla un encabezado con la fecha y un número de estación meteorológica para poder identificarlas y además se ha dado a todas las celdas formato de número, a excepción de la columna FECHA.

Esta última operación se ha realizado porque se ha visto que muchas celdas tenían formato fecha y aparecían alineadas a la derecha en lugar de alinearse a la izquierda como las celdas con formato número.



Visualización tabla Tª max desde spreadsheet

Una vez realizados estos cambios se ha obtenido la siguiente visualización en la que se aprecia claramente los valores que han sido añadidos.

```
data1.head()
```

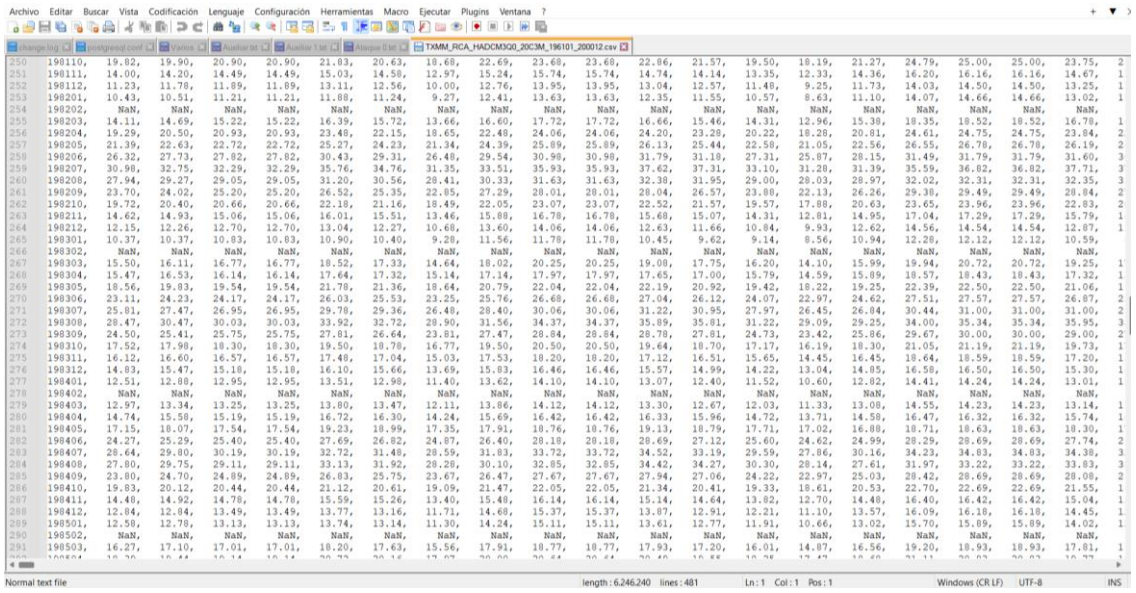
	FECHA	500	501	502	503	504	505	506	507	508	...	1935	1936	1937	1938	1939	1940	1941	1942
0	196101		16.33	44.789,00	44.789,00	16.22	44.911,00	44.577,00	16.36	16.50	...	11.79	11.79	12.54	12.54	11.36	10.84	10.26	13.19
1	196102	15.29	15.45	15.56	15.56	15.56	15.31	14.84	16.13	44.820,00	...	10.76	10.70	11.37	11.37	10.34	9.81	9.34	12.40
2	196103	15.88	16.35	16.64	16.64	17.47	44.637,00	15.89	17.33	18.14	...	44.631,00	44.631,00	11.69	11.69	10.63	9.96	9.35	12.55
3	196104	16.13	17.15	16.57	16.57	17.96	17.82	16.24	17.27	17.71	...	11.55	11.71	12.47	12.47	11.16	10.46	9.68	12.65
4	196105	18.68	44.581,00	20.31	20.31	23.24	22.28	18.15	22.17	23.90	...	44.847,00	13.18	14.29	14.29	12.56	11.63	10.63	14.28

5 rows × 1446 columns

```
data1.shape
```

```
(480, 1446)
```

De hecho, si se abre con un bloc de notas (Notepad++) se ve como esos datos que se han añadidos posteriormente aparecen como datos NaN (Not a Number).



Tiene sentido que falten valores en un dataset porque los datos en crudo se han obtenido de estaciones ubicadas, tanto en lugares institucionales (aeropuertos, colegios, etc.…) como particulares y las lecturas no siempre se han podido hacer en tiempo y forma.

Los productos en localidades puntuales ofrecidos en **Escenarios-PNACC Datos mensuales** se refieren a 374 puntos para los productos relacionados con la temperatura y 2.321 puntos para los relacionados con la precipitación, que se distribuyen por toda la geografía de España como se refleja en los mapas de la *Figura 6*. Estos puntos se corresponden a estaciones climatológicas de AEMET que han sido seleccionadas tras un control de calidad basado en criterios de longitud temporal de la serie, número reducido de lagunas y homogeneidad (Brunet *et al.* 2008, Herrera 2011, Herrera *et al.* 2012).

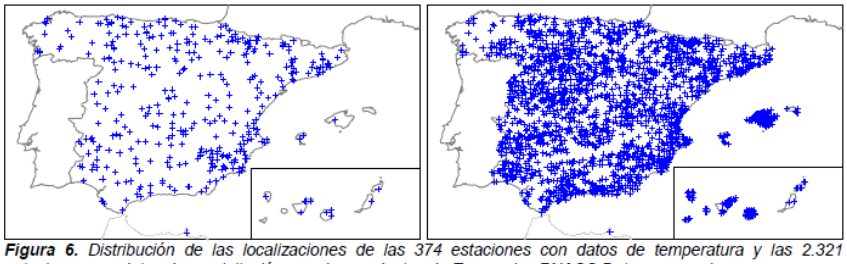


Figura 6. *Distribución de las localizaciones de las 374 estaciones con datos de temperatura y las 2.321 estaciones con datos de precipitación para los productos de Escenarios PNACC Datos mensuales*

Mapa de estaciones climatológicas ubicadas en España

La primera conclusión de este estudio se refiere a la detección de errores provocados por desplazamientos de ubicación, cambios en la instrumentación o el entorno, calibraciones, etc… lo que se refleja en las series como cambios artificiales añadidos con formato diferente.

No obstante, la serie que se ha utilizado para estudiar la influencia de la temperatura en el cambio climático tiene la misma tendencia que la señal climatica real y los datos climáticos son homogéneos, con largo recorrido temporal y probada calidad de modo que aceptan como válidos para esta investigación de carácter general, sin especificar zonas concretas de la geografía española.

A partir de aquí se trabaja con los datos originales y se va a mostrar mediante diferentes visualizaciones la influencia de la temperatura máxima en el cambio climático.

2. Plots y visualización de datos

Se ha cargado el archivo original sobre data1 y se van a mostrar las posibles visualizaciones que se pueden obtener a partir de él.

Plots y visualización de los datos

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

```
data1.head()
```

	FECHA	500	501	502	503	504	505	506	507	508	...	1935	1936	1937	1938	1939	1940	1941	1942	1943	1944
0	196101	16.08	16.33	16.08	16.08	16.22	16.12	16.01	16.36	16.50	...	11.79	11.79	12.54	12.54	11.36	10.84	10.26	13.19	13.00	12.98
1	196102	15.29	15.45	15.56	15.56	15.56	15.31	14.84	16.13	16.09	...	10.76	10.70	11.37	11.37	10.34	9.81	9.34	12.40	12.23	12.13
2	196103	15.88	16.35	16.64	16.64	17.47	17.03	15.89	17.33	18.14	...	11.03	11.03	11.69	11.69	10.63	9.96	9.35	12.55	12.35	12.18
3	196104	16.13	17.15	16.57	16.57	17.96	17.82	16.24	17.27	17.71	...	11.55	11.71	12.47	12.47	11.16	10.46	9.68	12.65	12.36	12.30
4	196105	18.68	20.01	20.31	20.31	23.24	22.28	18.15	22.17	23.90	...	13.10	13.18	14.29	14.29	12.56	11.63	10.63	14.28	14.13	14.06

5 rows × 1446 columns

a. Visualización básica: Scatterplot (nube de dispersión/puntos)

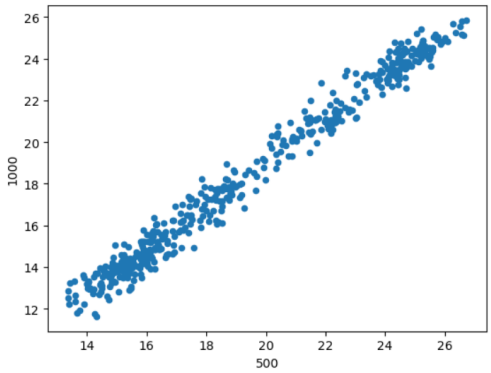
A continuación, se han visualizado los datos recogidos e n el dataset, primero se comparado la estación 500 con la estación 1000, desde enero de 1961 hasta diciembre del año 2000.

Se puede ver que ambas estaciones meteorológicas están situadas en zonas climáticas relativamente parecidas porque la nube de puntos de ambas estaciones tiene una tendencia similar.

```
data1_1961=data1
```

```
data1.plot(kind="scatter", x="500", y="1000")
```

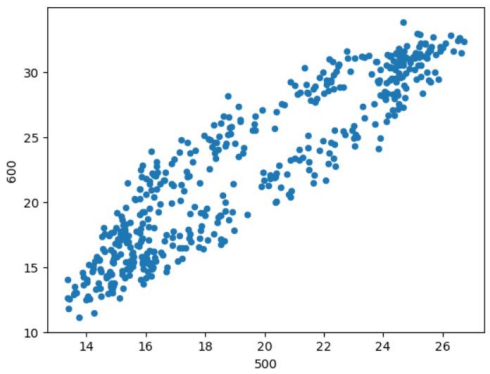
```
<AxesSubplot:xlabel='500', ylabel='1000'>
```



Sin embargo, al comparar la estación 500 con la estación 1940 a lo largo del tiempo desde enero de 1961 hasta diciembre del año 2000 se ve claramente que la separación entre los puntos es mayor, lo cual se debe, seguramente, a que las estaciones están más separadas y tienen climas distintos, aunque no se puede asegurar por el problema mencionado con el fichero maestro.

```
data1.plot(kind="scatter", x="500", y="600")
```

```
<AxesSubplot:xlabel='500', ylabel='600'>
```



De hecho, si se comparan todas las estaciones por pares, se podrían resolver muchas de las incongruencias que contiene este dataset y aproximar con relativa precisión la ubicación de las mismas.

Hace falta un análisis más exhaustivo para verificar la representación gráfica del aumento de temperaturas.

Se necesita ajustar más la visualización para ver más claro el análisis, haciendo uso del Data Wranglin (cirugía de datos).

Se puede extraer un subconjunto de datos, por ejemplo, de una sola estación meteorológica y el objeto que se obtiene es de tipo series o vectores.

Crear un subconjunto de datos

```
Estacion500=data1_1961["500"]
```

```
Estacion500.head()
```

```
0    16.08
1    15.29
2    15.88
3    16.13
4    18.68
Name: 500, dtype: float64
```

```
type(Estacion500)
```

```
pandas.core.series.Series
```

También se pueden extraer varias columnas, varias estaciones meteorológicas, y el objeto obtenido sería un DataFrame.

```
subset=data1_1961[["500","600","700"]]
```

```
subset.head()
```

	500	600	700
0	16.08	18.09	18.86
1	15.29	14.18	16.10
2	15.88	18.35	19.96
3	16.13	21.63	22.58
4	18.68	24.34	24.84

```
type(subset)
```

```
pandas.core.frame.DataFrame
```

Otra opción para esto en especificar las columnas deseadas es utilizar un solo paso mediante la opción desired_columns.

Así, se haría una comparación entre estaciones, en principio, distribuidas por toda la malla.

```
desired_columns=["500","1000","1500","1944"]
subset=data1[desired_columns]
subset.head()
```

	500	1000	1500	1944
0	16.08	14.53	12.42	12.98
1	15.29	13.99	12.15	12.13
2	15.88	14.38	14.66	12.18
3	16.13	15.52	16.75	12.30
4	18.68	17.78	22.48	14.06

Para reducir el peso del archivo se puede escoger una serie de columnas de trabajo y eliminar las demás, lo cual tendría más sentido si se conociera la ubicación exacta de las estaciones, pero aún así se ha hecho escogiendo únicamente las estaciones por centenas.

```
desired_columns=["500","600","700","800","900","1000","1100","1200","1300","1400","1500","1600","1700","1800","1900","2000","210
desired_columns
```

```
['500',
 '600',
 '700',
 '800',
 '900',
 '1000',
 '1100',
 '1200',
 '1300',
 '1400',
 '1500',
 '1600',
 '1700',
 '1800',
 '1900',
 '2000',
 '2100',
 '2200',
 '2300',
 '2400',
 '2500',
 '2600',
 '2700',
 '2800',
 '2900']
```

A continuación, se muestran todos los elementos de la lista del subset:

```
all_columns_list=data1_1961.values.tolist()
```

```
all_columns_list
```

```
[[196101.0,
  16.08,
  16.33,
  16.08,
  16.08,
  16.22,
  16.12,
  16.01,
  16.36,
  16.5,
  16.5,
  15.39,
  14.73,
  15.17,
  15.48,
  16.23,
  16.75,
  16.56,
  16.56,
```

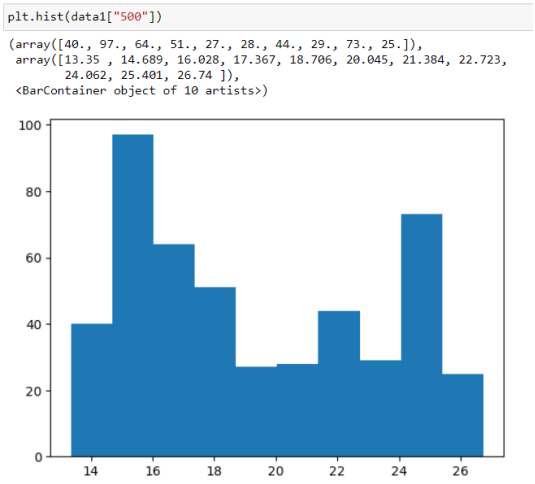
Otra forma de filtrar datos de forma, posiblemente más fácil, es la siguiente:

```
a = set(desired_columns)
b = set(all_columns_list)
sublist = b-a
sublist = list(sublist)
```

b. Histogramas de frecuencias

Un histograma es una forma de ver los rangos en que se agrupan los valores, cómo se distribuyen una variable numérica, si tiene forma normal de campana de gauss o no, …

A continuación se muestran histogramas de 2 estaciones meteorológicas ubicadas en zonas diferentes, en el que se aprecia la diferencia entre sus registros de temperatura.

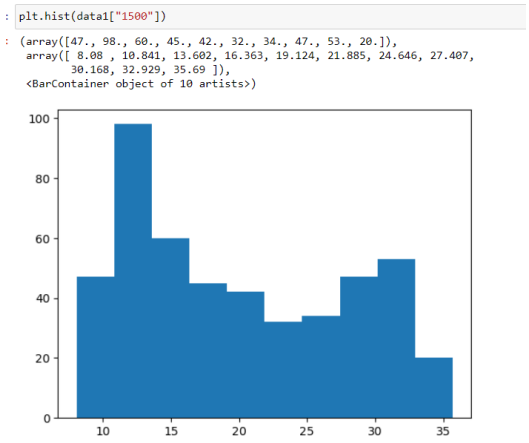


Histograma de frecuencias para la estación 500

Este histograma corresponde a las temperaturas recogidas en la estación 500 desde 1961 hasta el año 2000.

En el eje X aparece el rango de temperaturas máximas que se han recogido y en el eje Y aparece el número de lecturas que corresponden a esas temperaturas.

Se ve rápidamente que la estación 500 está ubicada en una zona templada, probablemente en la costa del norte de España, porque no se han recogido temperaturas entre 14ºC y 26ºC.



Histograma de frecuencias para la estación 1500

Por el contrario, la estación 1500 corresponde a un área con temperaturas más extremas, que bajan de 10º C y suben de 35ºC.

c. <u>Boxplot: diagrama de cajas y bigotes</u>

Es una de las mejores formas que hay para entender la distribución de una variable numérica, aunque antes de visualizarlo se ha investigado más sobre los estadísticos básicos del dataset:

Se hace notar que estos estadísticos tienen sentido en columnas numéricas de tipo float, pero no es columnas str ni en numéricas de tipo entero como FECHA que arroja resultados extraños.

De modo que se ha creado una copia sin la columna FECHA, data10, en la que se vuelven a buscar los estadísticos básicos:

Count: número de elementos no nulos, que en todas las estaciones son 478

Mean: promedio, que es la suma de todos los valores dividido entre el número de valores diferentes

Std: Desviación estándar, cómo se desvían los números respecto de la media

Min: Valor mínimo

Max: Valor máximo

Cuantiles 25% - 50% - 79%: Valor que se sitúa en el % señalado, se hace notar que el valor que se sitúa en el 50% no tiene por que corresponder al valor medio.

```
data1.columns
```

```
Index(['FECHA', '500', '501', '502', '503', '504', '505', '506', '507', '508',
       ...
       '1935', '1936', '1937', '1938', '1939', '1940', '1941', '1942', '1943',
       '1944'],
      dtype='object', length=1446)
```

```
data1.columns.values
```

```
array(['FECHA', '500', '501', ..., '1942', '1943', '1944'], dtype=object)
```

Resumen estadísticos básicos

```
data1.describe()
```

	FECHA	500	501	502	503	504	505	506	507	508	...	1935	1936
count	480.000000	478.000000	478.000000	478.000000	478.000000	478.000000	478.000000	478.000000	478.000000	478.000000	...	478.000000	478.000000
mean	198056.500000	19.310523	19.963473	20.317134	20.317134	21.854017	21.047176	18.801799	21.557385	22.767134	...	13.250188	13.240544
std	1155.549124	3.911835	4.300915	4.726629	4.726629	6.100403	5.366503	3.601067	5.572594	6.810156	...	2.192408	2.219370
min	196101.000000	13.350000	13.280000	13.150000	13.150000	12.700000	12.930000	13.280000	13.140000	12.820000	...	8.800000	8.940000
25%	197078.750000	15.840000	16.027500	15.990000	15.990000	16.067500	15.947500	15.540000	16.415000	16.507500	...	11.372500	11.370000
50%	198056.500000	18.430000	19.055000	19.245000	19.245000	20.430000	20.010000	18.075000	20.190000	20.990000	...	12.820000	12.790000
75%	199034.250000	23.045000	24.045000	25.250000	25.250000	27.970000	26.487500	22.307500	27.180000	29.520000	...	15.280000	15.257500
max	200012.000000	26.740000	28.030000	29.350000	29.350000	33.310000	30.910000	25.890000	31.990000	35.540000	...	17.900000	17.950000

8 rows × 1446 columns

Estadísticos en data1 (incluye columna FECHA)

```
data1.dtypes
```

```
FECHA      int64
500      float64
501      float64
502      float64
503      float64
          ...
1940     float64
1941     float64
1942     float64
1943     float64
1944     float64
Length: 1446, dtype: object
```

Tipo de datos en data1 (incluye columna FECHA)

```
data10.head()
```

	500	501	502	503	504	505	506	507	508	509	...	1935	1936	1937	1938	1939	1940	1941	1942	1943	1944
0	16.08	16.33	16.08	16.08	16.22	16.12	16.01	16.36	16.50	16.50	...	11.79	11.79	12.54	12.54	11.36	10.84	10.26	13.19	13.00	12.98
1	15.29	15.45	15.56	15.56	15.56	15.31	14.84	16.13	16.09	16.09	...	10.76	10.70	11.37	11.37	10.34	9.81	9.34	12.40	12.23	12.13
2	15.88	16.35	16.64	16.64	17.47	17.03	15.89	17.33	18.14	18.14	...	11.03	11.03	11.69	11.69	10.63	9.96	9.35	12.55	12.35	12.18
3	16.13	17.15	16.57	16.57	17.96	17.82	16.24	17.27	17.71	17.71	...	11.55	11.71	12.47	12.47	11.16	10.46	9.68	12.65	12.36	12.30
4	18.68	20.01	20.31	20.31	23.24	22.28	18.15	22.17	23.90	23.90	...	13.10	13.18	14.29	14.29	12.56	11.63	10.63	14.28	14.13	14.06

5 rows × 1445 columns

```
data10.describe()
```

	500	501	502	503	504	505	506	507	508	509	...	1935	1936	
count	478.000000	478.000000	478.000000	478.000000	478.000000	478.000000	478.000000	478.000000	478.000000	478.000000	...	478.000000	478.000000	47
mean	19.310523	19.963473	20.317134	20.317134	21.854017	21.047176	18.801799	21.557385	22.767134	22.767134	...	13.250188	13.240544	1
std	3.911835	4.300915	4.726629	4.726629	6.100403	5.366503	3.601067	5.572594	6.810156	6.810156	...	2.192408	2.219370	
min	13.350000	13.280000	13.150000	13.150000	12.700000	12.930000	13.280000	13.140000	12.820000	12.820000	...	8.800000	8.940000	
25%	15.840000	16.027500	15.990000	15.990000	16.067500	15.947500	15.540000	16.415000	16.507500	16.507500	...	11.372500	11.370000	1
50%	18.430000	19.055000	19.245000	19.245000	20.430000	20.010000	18.075000	20.190000	20.990000	20.990000	...	12.820000	12.790000	1
75%	23.045000	24.045000	25.250000	25.250000	27.970000	26.487500	22.307500	27.180000	29.520000	29.520000	...	15.280000	15.257500	1
max	26.740000	28.030000	29.350000	29.350000	33.310000	30.910000	25.890000	31.990000	35.540000	35.540000	...	17.900000	17.950000	1

8 rows × 1445 columns

Estadísticos en data 10 (NO incluye columna FECHA)

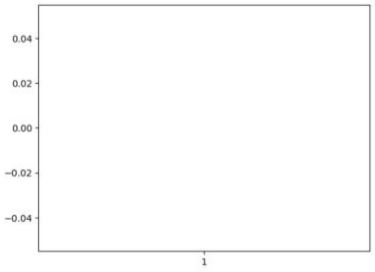
```
14]: data10.dtypes
14]: 500     float64
     501     float64
     502     float64
     503     float64
     504     float64
             ...
     1940    float64
     1941    float64
     1942    float64
     1943    float64
     1944    float64
     Length: 1445, dtype: object

15]: pd.isnull(data10["500"])
15]: 0      False
     1      False
     2      False
     3      False
     4      False
            ...
     475    False
     476    False
     477    False
     478    False
     479    False
     Name: 500, Length: 480, dtype: bool
```

Tipo de dato en data 10 (NO incluye columna FECHA)

El tipo de dato es object (dataframe) con valores numéricos flotantes que no son nulos

```
plt.boxplot(data1["500"])
{'whiskers': [<matplotlib.lines.Line2D at 0x1c5e34ce7c0>,
  <matplotlib.lines.Line2D at 0x1c5e34ce370>],
 'caps': [<matplotlib.lines.Line2D at 0x1c5e34f4fa0>,
  <matplotlib.lines.Line2D at 0x1c5e34f4c70>],
 'boxes': [<matplotlib.lines.Line2D at 0x1c5e34cea00>],
 'medians': [<matplotlib.lines.Line2D at 0x1c5e34f49d0>],
 'fliers': [<matplotlib.lines.Line2D at 0x1c5e34f4460>],
 'means': []}
```



No ha salido porque previamente hay que normalizar los valores.

3. Predicción climática en base a análisis de registros históricos.

Para hacer una predicción climática de evolución de las temperaturas máximas en base únicamente a los registros de temperatura previos a lo largo de 40 años, con carácter geográfico nacional, para facilitar los cálculos se ha querido crear una estación meteorológica ficticia llamada Promedio mediante operaciones con Python.

```
data1["Promedio"]=(data1["500"]+data1["550"]+data1["600"]+data1["650"]+data1["700"]+data1["750"]++data1["800"]+data1["850"]+data
```

```
data1["Promedio"]
```

```
0      13.1600
1      12.3620
2      14.6330
3      15.4610
4      20.7055
        ...
475    30.9920
476    24.5885
477    19.8430
478    14.0285
479    11.6935
Name: Promedio, Length: 480, dtype: float64
```

Finalmente, se ha optado por crear una copia del archivo descargado de AEMET, a la que se le han eliminado columnas y alguna fila con valores NaN, con lo que se han simplificado las operaciones sin perder excesiva validez en el resultado.

```
data_reducido=pd.read_csv("../Dataset/Reducido_promedio.csv")
```

```
data_reducido.head()
```

	Promedio	500	510	520	530	540	550	560	570	580	...	1850	1860	1870	1880	1890	1900	1910	1920	1930	1940
0	18.08	16.08	15.39	13.51	14.14	13.15	16.64	12.94	17.58	16.73	...	12.94	10.37	8.73	11.65	13.04	11.44	11.28	12.09	12.09	10.84
1	14.08	15.29	14.96	12.76	14.02	12.57	16.87	12.56	14.10	17.09	...	12.42	10.18	8.35	10.90	12.38	11.04	10.32	10.92	11.43	9.81
2	26.07	15.88	16.93	14.78	15.16	14.57	19.08	14.68	17.83	19.55	...	12.93	11.39	9.58	11.24	12.69	11.88	10.53	11.13	12.19	9.96
3	6.06	16.13	17.63	15.29	15.50	15.94	18.50	15.07	20.86	19.21	...	12.65	10.00	8.04	11.99	12.58	11.02	10.88	12.41	11.97	10.46
4	20.08	18.68	23.95	20.37	20.99	21.74	25.48	20.63	24.18	26.81	...	14.90	15.67	13.03	16.12	14.45	16.07	13.87	15.35	15.81	11.63

5 rows × 146 columns

La regresión lineal se utiliza cuando se tiene un conjunto de datos históricos, como los registros climáticos que componen este dataset y se busca una variable de salida o predicción y se asume una relación lineal entre las variables de entrada (promedio de registros) y las variables de salida.

Se ha dividido el archivo de trabajo, Reducido_promedio, en conjunto de entrenamiento para crear el modelo y conjunto de testing para comprobar el porcentaje de fiabilidad que tiene el modelo y que no haya problemas de overfitting, que sucede cuando el modelo se ajusta tan bien a los datos suministrados, y para ajustar el modelo.

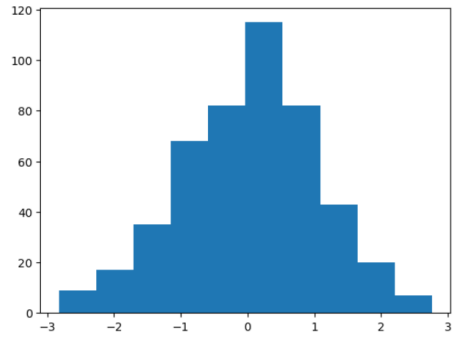
Se ha dividido el dataset escogiendo las filas mediante un método aleatorio, utilizando una distribución normal, creando un vector a la misma longitud del dataset pero con una distribución normal, tal como se ve a continuación:

Dividir utilizando la distribución normal

```
a=np.random.randn(len(data_reducido))
```

```
plt.hist(a)
```

```
(array([  9.,  17.,  35.,  68.,  82., 115.,  82.,  43.,  20.,   7.]),
 array([-2.8245924 , -2.26578182, -1.70697125, -1.14816067, -0.58935009,
        -0.03053952,  0.52827106,  1.08708163,  1.64589221,  2.20470278,
         2.76351336]),
 <BarContainer object of 10 artists>)
```



Los datos generados están entre -3 y 3, aproximadamente; para hacer la división se va a ver cuales de estos valores cumplen una condición, por ejemplo, cuales son inferiores a un valor dado como 0,8 y devuelve un array con valores true or false en función del cumplimiento de la condición.

```
check=(a<0.8)
```

```
check
```

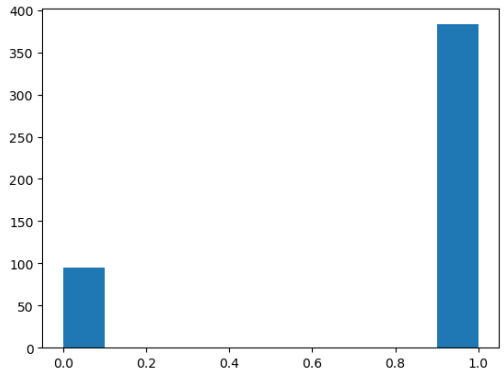
```
array([ True, False,  True,  True,  True, False,  True,  True,  True,
        True, False,  True,  True, False,  True,  True,  True, False,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True, False,  True,  True,  True,  True,  True,  True,
       False, False,  True,  True,  True,  True,  True,  True,  True,
        True,  True, False,  True,  True,  True,  True,  True,  True,
        True,  True,  True, False,  True, False,  True,  True,  True,
        True, False,  True, False,  True,  True, False,  True,  True,
        True,  True, False,  True,  True,  True,  True,  True,  True,
        True,  True,  True, False,  True, False,  True,  True,  True,
        True,  True,  True,  True, False, False,  True, False, False,
        True,  True, False,  True,  True,  True,  True,  True,  True,
        True,  True, False,  True,  True, False,  True,  True,  True,
        True,  True,  True,  True, False,  True,  True,  True, False,
       False,  True, False,  True,  True,  True, False,  True, False,
        True,  True,  True,  True,  True, False,  True,  True,  True,
        True, False,  True,  True,  True,  True,  True,  True,  True,
        True,  True, False,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True, False,  True,  True,  True, False,
        True,  True,  True, False,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True, False,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True, False,  True,  True,  True, False,
        True,  True,  True, False,  True,  True,  True,  True,  True,
       False,  True, False,  True,  True,  True, False,  True,  True,
        True, False, False, False,  True,  True,  True,  True, False,
       False,  True,  True,  True,  True,  True,  True, False, False,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True, False, False, False,  True,  True,  True,  True,
       False,  True, False,  True,  True,  True,  True, False,  True,
        True,  True,  True,  True, False,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True, False,  True,
        True, False,  True,  True,  True, False,  True,  True,  True,
        True,  True,  True, False,  True,  True,  True, False,  True])
```

Y al representar nuevamente mediante un histograma se comprueba que los datos están divididos en conjunto de entrenamiento, por debajo del 20% (False), y conjunto de validación, por encima del 80% (True).

Es como si se hubiera cortado la campana de Gauss en su zona inferior.

```
plt.hist(check.astype(int))
```

```
(array([ 95.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0., 383.]),
 array([0. , 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1. ]),
 <BarContainer object of 10 artists>)
```



A continuación, se ha comprobado la longitud de ambos conjuntos que, efectivamente, se corresponden aproximadamente con el 80% y el 20% de los registros de temperaturas utilizados.

```
training=data_reducido[check]
testing=data_reducido[~check]
```

```
len(training)
```

```
383
```

```
len(testing)
```

```
95
```

Se ha implementado una regresión en Python por medio de la librería scikit-learn, para predecir el valor de la temperatura a partir de la temperatura promedio calculada previamente.

Las temperaturas promedio se ajustan a una función, que puede ser lineal si se ajusta a una línea recta (y=α+βx+è) , puede ser exponencial, potencial, logarítmica, polinómica, etc… y frecuentemente es una combinación de varias.

a. Regresión lineal

La regresión lineal simple pretende encontrar un modelo para predecir la relación lineal entre x e y (y=α+βx+è), estudiando ciertos índices que cuantifican dicha relación:

α: Pendiente de la recta, se calcula despejando una vez conocida β

β: $\beta = \frac{Covarianza\ (x,y)}{Varianza\ (x)}$

- Covarianza (x,y): Valor que refleja la cuantía en que x e y varían de forma conjunta respecto a sus valores medios.
- Varianza (x): Medida de la dispersión de una serie de datos con respecto de su media, que también sería la desviación típica al cuadrado.

Además, la regresión lineal simple calcula e interpreta los coeficientes de la regresión, α y β, y realiza el ajuste utilizando el método de los mínimos cuadrados.

è: Residuo o error que corresponde al desajuste entre el modelo y la recta, que se distribuye según una variable aleatoria normal estándar.

Si el error tiene una distribución en forma de parábola, logarítmica, cuadrática, … significaría que el modelo que se ha tomado como lineal realmente no es lineal porque hay información que no ha sido posible explicar con un modelo lineal.

Otros parámetros estadísticos a tener en cuenta son:

- Coeficiente de Pearson: Medida de la dependencia lineal entre 2 variables aleatorias cuantitativas, que se puede utilizar para medir el grado de relación entre 2 variables, en este caso entre 2 registros térmicos de distintas estaciones meteorológicas.
  Se ha calculado entre los registros de 2 estaciones meteorológicas, concretamente las estaciones 500 y 1000, y cuando se visualizado el archivo, se muestra a la derecha una columna que muestra cuánto se separan cada uno de los números de la media de 500 y 1000.

Correlación

```
import pandas as pd
import numpy as np
```

```
data_reducido=pd.read_csv("../Dataset/Reducido_promedio.csv")
```

```
len(data_reducido)
```

```
478
```

```
_reducido["correlnumber"]=(data_reducido["500"]-np.mean(data_reducido["500"]))-(data_reducido["1000"]-np.mean(data_reducido["1000"
```

```
data_reducido.head()
```

	Promedio	500	510	520	530	540	550	560	570	580	...	1860	1870	1880	1890	1900	1910	1920	1930	1940	correlnumber
0	18.08	16.08	15.39	13.51	14.14	13.15	16.64	12.94	17.58	16.73	...	10.37	8.73	11.65	13.04	11.44	11.28	12.09	12.09	10.84	0.533954
1	14.08	15.29	14.96	12.76	14.02	12.57	16.87	12.56	14.10	17.09	...	10.18	8.35	10.90	12.38	11.04	10.32	10.92	11.43	9.81	0.283954
2	26.07	15.88	16.93	14.78	15.16	14.57	19.08	14.68	17.83	19.55	...	11.39	9.58	11.24	12.69	11.88	10.53	11.13	12.19	9.96	0.483954
3	6.06	16.13	17.63	15.29	15.50	15.94	18.50	15.07	20.86	19.21	...	10.00	8.04	11.99	12.58	11.02	10.88	12.41	11.97	10.46	-0.406046
4	20.08	18.68	23.95	20.37	20.99	21.74	25.48	20.63	24.18	26.81	...	15.67	13.03	16.12	14.45	16.07	13.87	15.35	15.81	11.63	-0.116046

5 rows × 147 columns

Ya calcularé el coeficiente en otro momento, de momento no lo necesito

- P valor: Es el valor de significación mínimo no arbitrario con el que se puede rechazar la hipótesis nula H0, dada una función de distribución y un estadístico de contraste.
  Cuando se ha calculado α y β no son cálculos exactos sino estimaciones y sería interesante hacer un contraste de hipótesis para valorar la significatividad, de hecho lo primero que hay que mirar es si β=0, es decir, si hay suficiente correlación entre x y la variable a predecir.

- Hipótesis nula: Es la suposición que se utiliza para negar o afirmar un suceso en relación a algún o algunosn parámetros de una muestra.
  En todo experimento, se establecen 2 hipótesis: la hipótesis nula, que el investigador pretende rechazar, y la hipótesis alternativa, que es la conclusión a la que el investigador ha llegado a través de su investigación.

Se ha realizado la regresión lineal cargando fichero de temperaturas máximas Reducido_promedio, al que se le han eliminado las filas correspondientes a diciembre de 1972 y a abril de 1992 porque tenían valores NaN e iban a producir errores.

Para la regresión lineal se ha generado dos distribuciones de números:

x: corresponde a una línea recta de 478 valores distribuidos según una normal de media 1.5 y desviación estándar 2.5

y: se crea con la columna Promedio que se ha calculado anteriormente.

z: Es y con el error (residuo) incorporado

Se ha transformado las 3 distribuciones a listas y se ha creado un dataframe al que se incorpora un diccionario con las claves y los valores.

Regresión lineal simple en Python

```
1]: import pandas as pd
    import numpy as np
    import matplotlib.pyplot as plt

2]: data_reducido=pd.read_csv("../Dataset/Reducido_promedio.csv")

3]: data_reducido.head()
```

3]:

	Promedio	500	510	520	530	540	550	560	570	580	...	1850	1860	1870	1880	1890	1900	1910	1920	1930	1940
0	18.08	16.08	15.39	13.51	14.14	13.15	16.64	12.94	17.58	16.73	...	12.94	10.37	8.73	11.65	13.04	11.44	11.28	12.09	12.09	10.84
1	14.08	15.29	14.96	12.76	14.02	12.57	16.87	12.56	14.10	17.09	...	12.42	10.18	8.35	10.90	12.38	11.04	10.32	10.92	11.43	9.81
2	26.07	15.88	16.93	14.78	15.16	14.57	19.08	14.68	17.83	19.55	...	12.93	11.39	9.58	11.24	12.69	11.88	10.53	11.13	12.19	9.96
3	6.06	16.13	17.63	15.29	15.50	15.94	18.50	15.07	20.86	19.21	...	12.65	10.00	8.04	11.99	12.58	11.02	10.88	12.41	11.97	10.46
4	20.08	18.68	23.95	20.37	20.99	21.74	25.48	20.63	24.18	26.81	...	14.90	15.67	13.03	16.12	14.45	16.07	13.87	15.35	15.81	11.63

5 rows × 146 columns

Voy a generar una distribución de 200 numeros, con x según N(1.5,2.5), e y la columna Promedio

El error res estará distribuido según una normal N(0,0.8)

```
4]: x=1.5+2.5*np.random.rand(478)

5]: y=data_reducido["Promedio"]

6]: res=0+0.8*np.random.rand(478)

7]: z=data_reducido["Promedio"]+res

9]: x_list=x.tolist()
    y_list=y.tolist()
    z_list=z.tolist()

0]: data=pd.DataFrame(
        {
            "x":x_list,
            "y":y_list,
            "z":z_list
        }
```

Se ha mostrado una visualización de la tabla con las columnas creadas, así como una representación gráfica del dataframe:

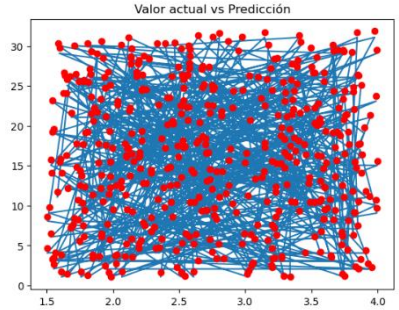
```
: data.head()
```

:

	x	y	z
0	2.677532	18.08	18.725108
1	3.876727	14.08	14.082716
2	2.568410	26.07	26.406130
3	1.911314	6.06	6.492814
4	2.410811	20.08	20.121338

```
: %matplotlib inline
  plt.plot(x,y)
  plt.plot(x,z,"ro")
  plt.title("Valor actual vs Predicción")
```

```
: Text(0.5, 1.0, 'Valor actual vs Predicción')
```



Se observa que el gráfico no tiene una forma coherente, debido a que los datos abarcan un periodo de tiempo excesivo.

En realidad, cada registro de temperaturas de 1 año, debería tener forma de parábola; al representar los 40 años queda este galimatías.

Se ha optado por repetir el proceso con el registro de temperaturas mínimas, y el resultado que se obtiene es tan confuso como el anterior.

```
datamin_reducido=pd.read_csv("../Dataset/Temp-mínimas/TEMPminimasReducido.csv")
```

```
datamin_reducido.head()
```

	Promedio	500	510	520	530	540	550	560	570	580	...	1850	1860	1870	1880	1890	1900	1910	1920	1930	1940
0	5.342	13.55	8.09	6.06	6.90	4.31	10.09	4.72	5.56	9.50	...	11.34	6.37	3.97	6.03	11.28	7.05	6.81	7.73	8.48	8.89
1	6.340	12.97	8.90	6.79	6.97	3.79	8.71	3.43	5.32	7.97	...	10.04	3.60	1.97	4.22	10.21	4.15	4.76	5.99	6.08	7.97
2	6.005	12.04	6.59	4.77	4.85	3.68	8.07	3.26	5.87	7.62	...	10.16	3.84	1.60	3.96	10.46	4.20	4.63	6.01	6.23	7.96
3	4.671	13.04	8.77	6.96	7.13	5.88	10.19	5.32	8.34	9.66	...	10.55	4.66	2.38	5.46	10.57	5.26	5.51	7.30	7.15	8.41
4	4.338	15.17	12.01	9.30	9.72	9.22	13.72	7.95	11.19	13.50	...	12.84	8.20	5.58	8.51	12.87	8.67	8.06	9.64	10.26	9.92

5 rows × 145 columns

```
datamin_reducido.shape
```

```
(478, 145)
```

```
xmin=1.5+0.3*np.random.rand(478)
```

```
ymin=data_reducido["Promedio"]
```

```
resmin=0+0.8*np.random.rand(478)
```

```
zmin=data_reducido["Promedio"]+res
```

```
xmin_list=xmin.tolist()
ymin_list=ymin.tolist()
zmin_list=zmin.tolist()
```

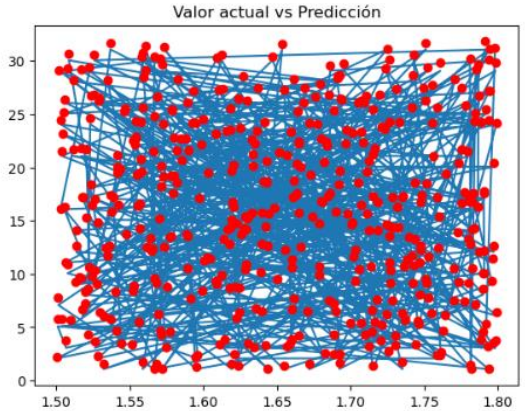
```
datamin=pd.DataFrame(
    {
        "x":xmin_list,
        "y":ymin_list,
        "z":zmin_list
    }
)
```

```
datamin.head()
```

	x	y	z
0	1.551293	18.08	18.380872
1	1.745006	14.08	14.764590
2	1.587851	26.07	26.705176
3	1.701515	6.06	6.243302
4	1.721192	20.08	20.358721

```
%matplotlib inline
plt.plot(x,y)
plt.plot(x,z,"ro")
plt.title("Valor actual vs Predicción")
```

```
Text(0.5, 1.0, 'Valor actual vs Predicción')
```



Finalmente se ha repetido el proceso comparando las temperaturas promedio máximas y mínimas y el resultado sigue siendo confuso pero los puntos rojos temperaturas promedio máximas vs temperaturas máximas promedio con error) aparecen agrupadas en una banda.

Composición de temperaturas máxima (data_reducido) y mínima (datamin_reducido)

```
xfinal=data_reducido["Promedio"]
```

```
yfinal=datamin_reducido["Promedio"]
```

```
resmin=0+0.8*np.random.rand(478)
```

```
zfinal=datamin_reducido["Promedio"]+resmin
```

```
xfinal_list=xfinal.tolist()
yfinal_list=yfinal.tolist()
zfinal_list=zfinal.tolist()
```

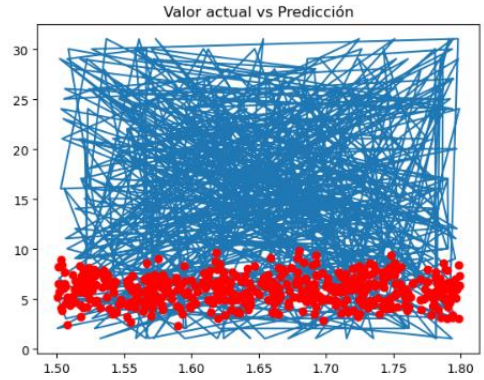
```
datafinal=pd.DataFrame(
    {
        "x":xfinal_list,
        "y":yfinal_list,
        "z":zfinal_list
    }
)
```

```
datafinal.head()
```

	x	y	z
0	18.08	5.342	5.593162
1	14.08	6.340	7.075727
2	26.07	6.005	6.664812
3	6.06	4.671	4.961108
4	20.08	4.338	4.408785

```
7]: %matplotlib inline
    plt.plot(x,y)
    plt.plot(x,z,"ro")
    plt.title("Valor actual vs Predicción")
```

```
7]: Text(0.5, 1.0, 'Valor actual vs Predicción')
```



Para darle sentido a estos gráficos se puede optar por 2 posibilidades:

- Hacer una regresión lineal con una serie que se asemeje a una línea recta, por ejemplo, tomar las temperaturas en un mes concreto, por ejemplo en julio, y estudiar su evolución a lo largo de los años.
- Hacer la regresión con el paquete scikit-learn que tiene implementadas

En este punto del informe se ha querido

Se ha utilizado el paquete scikit-learn para llevar a cabo la regresión porque tiene implementada la detección de las variables más importantes y lo único que hay que hacer es especificar el número de variables con el que debe construir el modelo.

b. Regresión logística
c. Curvas ROC (validación del modelo)

4. Clasificación de ubicación (estación meteorológica) de un nuevo registro en base a su analogía con los registros históricos.

Otro objetivo del estudio es comprobar la capacidad que se tiene de clasificar un nuevo registro de temperaturas como perteneciente a una zona concreta, por su estación meteorológica asociada.

```
]: data1=pd.read_csv("../Dataset/TXMM_RCA_ECHAM5_20C3M_196101_200012 original.csv")
```

```
]: data1
```

]:

	FECHA	500	501	502	503	504	505	506	507	508	...	1935	1936	1937	1938	1939	1940	1941	1942	1943	1944
0	196101	16.08	16.33	16.08	16.08	16.22	16.12	16.01	16.36	16.50	...	11.79	11.79	12.54	12.54	11.36	10.84	10.26	13.19	13.00	12.98
1	196102	15.29	15.45	15.56	15.56	15.56	15.31	14.84	16.13	16.09	...	10.76	10.70	11.37	11.37	10.34	9.81	9.34	12.40	12.23	12.13
2	196103	15.88	16.35	16.64	16.64	17.47	17.03	15.89	17.33	18.14	...	11.03	11.03	11.69	11.69	10.63	9.96	9.35	12.55	12.35	12.18
3	196104	16.13	17.15	16.57	16.57	17.96	17.82	16.24	17.27	17.71	...	11.55	11.71	12.47	12.47	11.16	10.46	9.68	12.65	12.36	12.30
4	196105	18.68	20.01	20.31	20.31	23.24	22.28	18.15	22.17	23.90	...	13.10	13.18	14.29	14.29	12.56	11.63	10.63	14.28	14.13	14.06
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
475	200008	25.85	26.10	27.93	27.93	30.26	28.23	24.23	30.41	32.54	...	17.02	17.13	17.76	17.76	16.27	15.51	14.73	18.02	17.83	17.74
476	200009	23.85	23.96	24.68	24.68	25.41	24.74	23.13	26.15	26.24	...	16.65	16.65	17.66	17.66	16.10	15.27	14.49	18.02	17.90	17.76
477	200010	22.34	22.46	22.61	22.61	23.28	22.86	21.68	23.28	23.46	...	15.29	15.25	15.90	15.90	14.76	14.14	13.57	16.62	16.43	16.40
478	200011	18.06	18.09	17.82	17.82	17.67	17.76	17.96	17.95	17.76	...	13.46	13.45	13.95	13.95	13.12	12.67	12.25	14.85	14.65	14.59
479	200012	16.23	16.12	16.05	16.05	15.72	15.80	16.09	16.11	15.78	...	11.94	11.84	12.25	12.25	11.48	11.06	10.76	13.61	13.36	13.26

480 rows × 1446 columns

Cabe destacar la complejidad de esta clasificación por los motivos indicados en el apartado 1 Datos Previos y alcance del proyecto, ya que ha resultado imposible asociar las estaciones meteorológicas con ubicaciones concretas de la geografía española por desplazamientos en la ubicación de las estaciones, cambios en el entorno, etc.…

No obstante, se ha realizado el estudio tratando de identificar la o las estaciones que más se asemejan al nuevo registro y para ello se han utilizado las entradas del dataset en clusters (grupos) en base a su similitud.

a. Clustering

Se han establecido unos grupos de estaciones meteorológicas que corresponden a las mismas zonas geográficas, calculándose para cada cluster las distancias con los demás clusters.

Estas distancias se pueden calcular en base a diferentes modelos:

- Manhattan distance (distancia del taxi): dd1
- Euclidean distance: dd2
- Chevichev distancey (calculo más complejo)
- Otras como dd10

A partir del dataset con los registros climáticos, data20, se han creado matrices de distancias dd1, dd2 y dd10, no obstante, la formula debe cambiarse para que se pueda utilizar con una matriz rectangular como la del dataset con el que se trabaja.

Para ajustar la fórmula se debe consultar la siguiente documentación:

[Cálculos de distancia (scipy.spatial.distance) — Manual de SciPy v1.9.3]

```
data20=pd.read_csv("../Dataset/TXMM_RCA_ECHAM5_20C3M_196101_200012 original.csv")
```

```
data20
```

	FECHA	500	501	502	503	504	505	506	507	508	...	1935	1936	1937	1938	1939	1940	1941	1942	1943	1944
0	196101	16.08	16.33	16.08	16.08	16.22	16.12	16.01	16.36	16.50	...	11.79	11.79	12.54	12.54	11.36	10.84	10.26	13.19	13.00	12.98
1	196102	15.29	15.45	15.56	15.56	15.56	15.31	14.84	16.13	16.09	...	10.76	10.70	11.37	11.37	10.34	9.81	9.34	12.40	12.23	12.13
2	196103	15.88	16.35	16.64	16.64	17.47	17.03	15.89	17.33	18.14	...	11.03	11.03	11.69	11.69	10.63	9.96	9.35	12.55	12.35	12.18
3	196104	16.13	17.15	16.57	16.57	17.96	17.82	16.24	17.27	17.71	...	11.55	11.71	12.47	12.47	11.16	10.46	9.68	12.65	12.36	12.30
4	196105	18.68	20.01	20.31	20.31	23.24	22.28	18.15	22.17	23.90	...	13.10	13.18	14.29	14.29	12.56	11.63	10.63	14.28	14.13	14.06
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
475	200008	25.85	26.10	27.93	27.93	30.26	28.23	24.23	30.41	32.54	...	17.02	17.13	17.76	17.76	16.27	15.51	14.73	18.02	17.83	17.74
476	200009	23.85	23.96	24.68	24.68	25.41	24.74	23.13	26.15	26.24	...	16.65	16.65	17.66	17.66	16.10	15.27	14.49	18.02	17.90	17.76
477	200010	22.34	22.46	22.61	22.61	23.28	22.86	21.68	23.28	23.46	...	15.29	15.25	15.90	15.90	14.76	14.14	13.57	16.62	16.43	16.40
478	200011	18.06	18.09	17.82	17.82	17.67	17.76	17.96	17.95	17.76	...	13.46	13.45	13.95	13.95	13.12	12.67	12.25	14.85	14.65	14.59
479	200012	16.23	16.12	16.05	16.05	15.72	15.80	16.09	16.11	15.78	...	11.94	11.84	12.25	12.25	11.48	11.06	10.76	13.61	13.36	13.26

480 rows × 1446 columns

Data20: dataset registros climáticos

Como sólo se quiere calcular la distancia entre registros de temperaturas no interesa la columna FECHA de modo que se transforma a lista los valores de las columnas indicadas en el cabecero, desde la columna 1 (500) en adelante.

```
temperaturas=data1.columns.values.tolist()[1:]
temperaturas
```

```
['500',
 '501',
 '502',
 '503',
 '504',
 '505',
 '506',
 '507',
 '508',
 '509',
 '510',
 '511',
 '512',
 '513',
 '514',
 '515',
 '516',
 '517',
 '518',
```

Se han normalizado las temperaturas porque se ha visto que los datos no siguen una distribución gaussiuana, se vio en los histogramas que las distribuciones no tenían forma de campana de Gauss.

En la normalización, se transforman los valores a un rango entre 0 y 1, aplicando la siguiente fórmula:

$$x_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

Esto también se puede implementar en Python, mediante sklearn:

```
sinfecha=pd.read_csv("../Dataset/sinfecha.csv")
```

```
sinfecha
```

	500	501	502	503	504	505	506	507	508	509	...	1935	1936	1937	1938	1939	1940	1941	1942	1943	1944
0	16.08	16.33	16.08	16.08	16.22	16.12	16.01	16.36	16.50	16.50	...	11.79	11.79	12.54	12.54	11.36	10.84	10.26	13.19	13.00	12.98
1	15.29	15.45	15.56	15.56	15.56	15.31	14.84	16.13	16.09	16.09	...	10.76	10.70	11.37	11.37	10.34	9.81	9.34	12.40	12.23	12.13
2	15.88	16.35	16.64	16.64	17.47	17.03	15.89	17.33	18.14	18.14	...	11.03	11.03	11.69	11.69	10.63	9.96	9.35	12.55	12.35	12.18
3	16.13	17.15	16.57	16.57	17.96	17.82	16.24	17.27	17.71	17.71	...	11.55	11.71	12.47	12.47	11.16	10.46	9.68	12.65	12.36	12.30
4	18.68	20.01	20.31	20.31	23.24	22.28	18.15	22.17	23.90	23.90	...	13.10	13.18	14.29	14.29	12.56	11.63	10.63	14.28	14.13	14.06
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
475	25.85	26.10	27.93	27.93	30.26	28.23	24.23	30.41	32.54	32.54	...	17.02	17.13	17.76	17.76	16.27	15.51	14.73	18.02	17.83	17.74
476	23.85	23.96	24.68	24.68	25.41	24.74	23.13	26.15	26.24	26.24	...	16.65	16.65	17.66	17.66	16.10	15.27	14.49	18.02	17.90	17.76
477	22.34	22.46	22.61	22.61	23.28	22.86	21.68	23.28	23.46	23.46	...	15.29	15.25	15.90	15.90	14.76	14.14	13.57	16.62	16.43	16.40
478	18.06	18.09	17.82	17.82	17.67	17.76	17.96	17.95	17.76	17.76	...	13.46	13.45	13.95	13.95	13.12	12.67	12.25	14.85	14.65	14.59
479	16.23	16.12	16.05	16.05	15.72	15.80	16.09	16.11	15.78	15.78	...	11.94	11.84	12.25	12.25	11.48	11.06	10.76	13.61	13.36	13.26

480 rows × 1445 columns

Ya continuación se han calculado las distancias:

```
dd1=distance_matrix(data20[sinfecha],data20[sinfecha],p=1)
dd2=distance_matrix(data20[sinfecha],data20[sinfecha],p=2)
dd10=distance_matrix(data20[sinfecha],data20[sinfecha],p=10)
```

```
dd1
```

```
array([[    0.  ,  1004.42,  2356.08, ...,  8947.19,  1364.48,  1978.3 ],
       [ 1004.42,     0.  ,  3129.68, ...,  9807.51,  2222.94,  1595.06],
       [ 2356.08,  3129.68,     0.  , ...,  6678.83,  1767.42,  4279.  ],
       ...,
       [ 8947.19,  9807.51,  6678.83, ...,     0.  ,  7584.69, 10897.97],
       [ 1364.48,  2222.94,  1767.42, ...,  7584.69,     0.  ,  3313.28],
       [ 1978.3 ,  1595.06,  4279.  , ..., 10897.97,  3313.28,     0.  ]])
```

```
dd1.shape
```

```
(480, 480)
```

Matriz Manhattan distance: dd1

```
dd2
```

```
array([[  0.        ,  38.96161958,  70.27533991, ..., 240.38073238,
         44.4569837 ,  56.5490053 ],
       [ 38.96161958,   0.        ,  88.96746484, ..., 263.34049651,
         73.78115206,  47.27539952],
       [ 70.27533991,  88.96746484,   0.        , ..., 181.73348591,
         54.63110652, 120.2409639 ],
       ...,
       [240.38073238, 263.34049651, 181.73348591, ...,   0.        ,
        206.39178109, 292.49225169],
       [ 44.4569837 ,  73.78115206,  54.63110652, ..., 206.39178109,
         0.        ,  90.35050968],
       [ 56.5490053 ,  47.27539952, 120.2409639 , ..., 292.49225169,
         90.35050968,   0.        ]])
```

```
dd2.shape
```

```
(480, 480)
```

Matriz euclidean distance: dd2

```
dd10
```

```
array([[ 0.        ,  5.39573019,  6.03781562, ..., 14.46059572,
         4.57319196,  3.86360627],
       [ 5.39573019,  0.        ,  6.65502262, ..., 15.826577  ,
         7.60148769,  3.70765741],
       [ 6.03781562,  6.65502262,  0.        , ..., 11.57316017,
         4.66075507,  8.28161328],
       ...,
       [14.46059572, 15.826577  , 11.57316017, ...,  0.        ,
        12.93170454, 17.42523828],
       [ 4.57319196,  7.60148769,  4.66075507, ..., 12.93170454,
         0.        ,  6.2697782 ],
       [ 3.86360627,  3.70765741,  8.28161328, ..., 17.42523828,
         6.2697782 ,  0.        ]])
```

```
dd10.shape
```

```
(480, 480)
```

Matriz: dd10

En este caso se ha utilizado la distancia euclídea, que es el segmento que une los centros (centroides) de los diferentes registros.

```
dd2
```

```
array([[  0.        ,  38.96161958,  70.27533991, ..., 240.38073238,
          44.4569837 ,  56.5490053 ],
       [ 38.96161958,   0.        ,  88.96746484, ..., 263.34049651,
          73.78115206,  47.27539952],
       [ 70.27533991,  88.96746484,   0.        , ..., 181.73348591,
          54.63110652, 120.2409639 ],
       ...,
       [240.38073238, 263.34049651, 181.73348591, ...,   0.        ,
         206.39178109, 292.49225169],
       [ 44.4569837 ,  73.78115206,  54.63110652, ..., 206.39178109,
           0.        ,  90.35050968],
       [ 56.5490053 ,  47.27539952, 120.2409639 , ..., 292.49225169,
          90.35050968,   0.        ]])
```

```
dd2.shape
```

```
(480, 480)
```

En general, cuanto mayor es el índice p aplicado en el cálculo de las matrices de distancias, más pequeña es la distancia y mayor es el número de clusters que se crean, aunque resulta poco práctico.

También se observa que cuanto mayor es el número de clusters, menor es la distancia entre ellos.

aErszzsexdrcytfuvgybihunjimklzsxdrctfvgybhunjikmolñdxfcgvbhjnkml,ñ

Como el tamaño de este dataset es excesivamente grande, 1446 columnas, se han agrupado los datos en torno a algunas columnas, esto tendría más sentido si se conocieran las ubicaciones de las estaciones, no obstante, se ha clasificado revisando los datos de la tabla original descargada de AEMET, concretamente los registros de temperaturas en agosto de un año intermedio, por ejemplo 1973.

Con esta revisión de los datos originales, se han establecido unos baremos de temperatura, escogiendo unas estaciones representativas:

Rango temperatura	**Nº Estación meteorológica**	**Temperatura agosto 1973**
15º	1921	15.67
20º	1772	20.62
25º	500	25.55
30º	522	30.02
35º	529	335.66
40º	582	38.23

Se ha obtenido el resultado que se ve a continuación que no es demasiado bueno porque hace 479 grupos, uno por cada fila y muestra en la zona superior de cada grupo las temperaturas de 6 estaciones meteorológicas seleccionadas previamente.

*

```
grouped_columns=data20.groupby(["1921","1772","500","522","529","582"])
```

```
len(grouped_columns)
```

```
479
```

```
for names, groups in grouped_columns:
    print(names)
    print(groups)
```

```
(7.24, 1.78, 13.91, 13.05, 13.19, 14.58)
      FECHA    500    501    502    503    504    505    506    507    508  \
229  198002  13.91  14.08  13.92  13.92  14.08  14.02  13.85  14.27  14.19

     ...  1935  1936  1937  1938  1939  1940  1941   1942  1943   1944
229  ...  8.97  8.94  9.22  9.22  8.75  8.46  8.24  10.85  10.6  10.39

[1 rows x 1446 columns]
(7.33, -1.22, 13.75, 11.22, 12.3, 12.81)
      FECHA    500    501    502    503   504    505    506    507    508  \
240  198101  13.75  13.58  13.66  13.66  13.0  13.08  13.85  13.81  13.38

     ...  1935  1936  1937  1938  1939  1940  1941   1942   1943   1944
240  ...   9.0  9.03  9.22  9.22  8.95  8.67  8.55  10.78  10.41  10.13

[1 rows x 1446 columns]
(7.71, 3.82, 14.39, 13.88, 14.02, 15.67)
      FECHA    500    501    502    503    504    505    506    507    508  \
434  199703  14.39  14.67  14.53  14.53  14.83  14.78  14.35  14.97  14.96
```

Se ha consultado el método que debe utilizarse para hacer la agrupación óptima utilizando la librería numpy.

Clustering jerárquico (scipy.cluster.hierarchy) — Manual de SciPy v1.9.3

Finalmente se ha optado por hacer un cluster jerárquico en el cual, el algoritmo sea el que calcule el número de clusters óptimo para generar un dendrograma, que es una representación gráfica de los datos en forma de árbol, que organiza los datos en subcategorías que se van dividiendo hasta llegar al nivel de detalle deseado.

a. Clustering jerárquico.

El proceso de agrupación jerárquica implica agrupar subclusters en clusters más grandes de manera ascendente, o bien dividir un cluster más grande en en subgrupos mas pequeños de arriba hacia abajo.

Este proceso se ha conseguido mediante el algoritmo de agrupación linkage, pero al realizarlo sobre el fichero maestro ha aparecido un error que se ha solucionado modificando este fichero maestro, lo cual pone en duda la eficacia del método para ficheros con datos excesivamente desordenados.

```
linked = linkage(data5["CodigoAEMET"], 'ward')
linked
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Input In [124], in <cell line: 1>()
----> 1 linked = linkage(data5["CodigoAEMET"], 'ward')
      2 linked

File ~\anaconda3\envs\Ciberseguridad\lib\site-packages\scipy\cluster\hierarchy.py:1046, in linkage(y, method, metric, optimal_o
rdering)
   1043 if method not in _LINKAGE_METHODS:
   1044     raise ValueError("Invalid method: {0}".format(method))
-> 1046 y = _convert_to_double(np.asarray(y, order='c'))
   1048 if y.ndim == 1:
   1049     distance.is_valid_y(y, throw=True, name='y')

File ~\anaconda3\envs\Ciberseguridad\lib\site-packages\scipy\cluster\hierarchy.py:1572, in _convert_to_double(X)
   1570 def _convert_to_double(X):
   1571     if X.dtype != np.double:
-> 1572         X = X.astype(np.double)
   1573     if not X.flags.contiguous:
   1574         X = X.copy()

ValueError: could not convert string to float: '0016B'
```

Error señalado al no convertir a float un dato string

```
data5=pd.read_csv("../Dataset/Fichero_Maestro_modificado.csv")
```

```
data5
```

	CodigoAEMET	Localidad	Longitud	Latitud	Altura
0	1	EL PERELLO	71.667	408.667	142.0
1	0016B	NaN	NaN	NaN	NaN
2	17	VILASECA DE SOLCINA	1.145	411.117	53.0
3	19	NaN	NaN	NaN	NaN
4	0020O	NaN	NaN	NaN	NaN
...	...	...	...	...	...
369	C449G	ANAGA-SAN ANDRES	-161.839	284.967	20.0
370	C458A	TACORONTE-A . S.E.A.	-164.092	284.864	327.0
371	C649I	TELDE/AEROPUERTO DE G.CANARIA (GANDO)	-153.889	279.292	24.0
372	C659P	LAS PALMAS DE GRAN CANARIA-JUNTA DE OBRAS DEL ...	-154.167	28.150	15.0
373	C665I	VALLESECO-CASCO	-155.736	280.472	980.0

374 rows × 5 columns

Fichero maestro con datos float y string en su columna CodigoAEMET

```
data2=pd.read_csv("../Dataset/Fichero_Maestro_modificado.csv")
```

```
data2
```

	CodigoAEMET	Localidad	Longitud	Latitud	Altura
0	1	EL PERELLO	71.667	408.667	142.0
1	160000	NaN	NaN	NaN	NaN
2	17	VILASECA DE SOLCINA	1.145	411.117	53.0
3	19	NaN	NaN	NaN	NaN
4	200000	NaN	NaN	NaN	NaN
...	...	...	...	...	...
369	449000	ANAGA-SAN ANDRES	-161.839	284.967	20.0
370	458000	TACORONTE-A . S.E.A.	-164.092	284.864	327.0
371	649000	TELDE/AEROPUERTO DE G.CANARIA (GANDO)	-153.889	279.292	24.0
372	659000	LAS PALMAS DE GRAN CANARIA-JUNTA DE OBRAS DEL ...	-154.167	28.150	15.0
373	665000	VALLESECO-CASCO	-155.736	280.472	980.0

374 rows × 5 columns

Fichero maestro modificado

En la modificación se ha visto que ciertos valores son NaN (Not a Number), valores que provocan errores en la agrupación de los clusters.

```
data2.dropna(axis=0, how="any")
```

	CodigoAEMET	Localidad	Longitud	Latitud	Altura
0	1	EL PERELLO	71.6670	408.667	142.0
2	17	VILASECA DE SOLCINA	1.1450	411.117	53.0
5	25	SARRAL	1.2500	41.450	400.0
8	111	CABRIANES	1.8953	417.986	246.0
9	158000	MONTSERRAT	1.8403	415.956	735.0
...	...	...	...	...	...
369	449000	ANAGA-SAN ANDRES	-161.8390	284.967	20.0
370	458000	TACORONTE-A . S.E.A.	-164.0920	284.864	327.0
371	649000	TELDE/AEROPUERTO DE G.CANARIA (GANDO)	-153.8890	279.292	24.0
372	659000	LAS PALMAS DE GRAN CANARIA-JUNTA DE OBRAS DEL ...	-154.1670	28.150	15.0
373	665000	VALLESECO-CASCO	-155.7360	280.472	980.0

350 rows × 5 columns

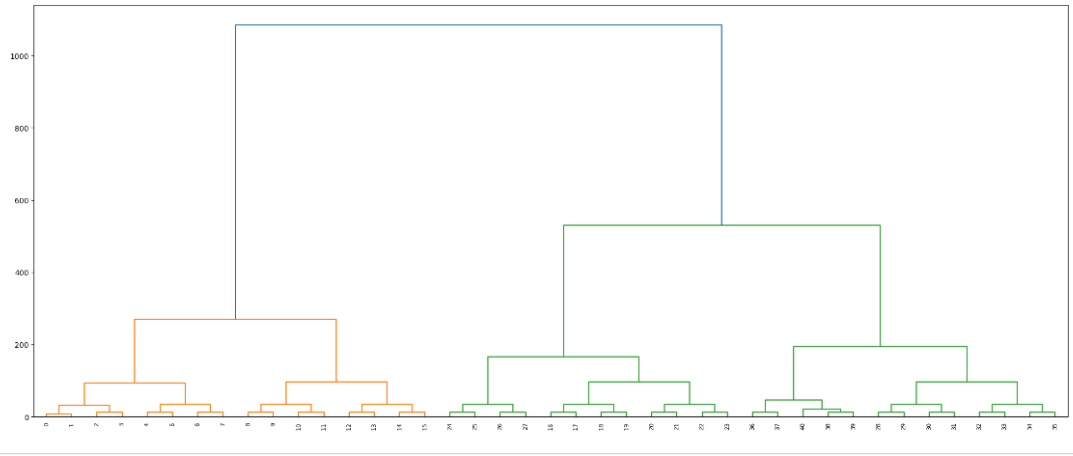
Para implementar el clustering jararquico en Python se ha utilizado la muestra de temperaturas de agosto desde 1961 hasta 2000 y se ha representado en un dendrograma en el cual se aprecian las distancias (vertical) que permitirían unir a un cluster con los demás.

Clustering jerárquico

```
import matplotlib.pyplot as plt
from scipy.cluster.hierarchy import dendrogram, linkage
import numpy as np
import pandas as pd
```

```
X = [[i] for i in [0, 7, 19, 31, 43, 55, 67, 79, 91, 103, 115, 127, 139, 151, 163, 175, 187, 199, 211, 223, 235, 247, 259, 271,
```

```
Z = linkage(X, 'ward')
fig = plt.figure(figsize=(25,10))
dn = dendrogram(Z)
```

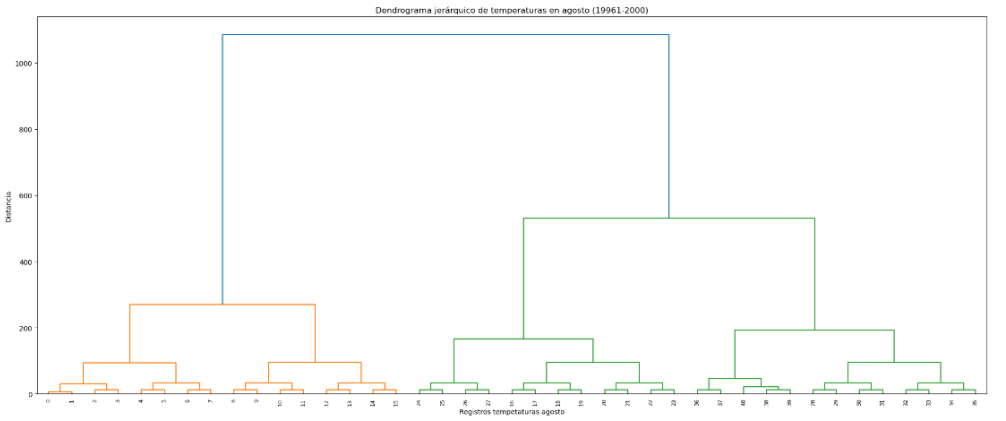


Se le ha dado formato al dendrograma, colocando unos títulos y también se podrían hacer otras modificaciones como girarlo 90º, hacer un dendrograma con la distancia promedio, que se muestra debajo.

.

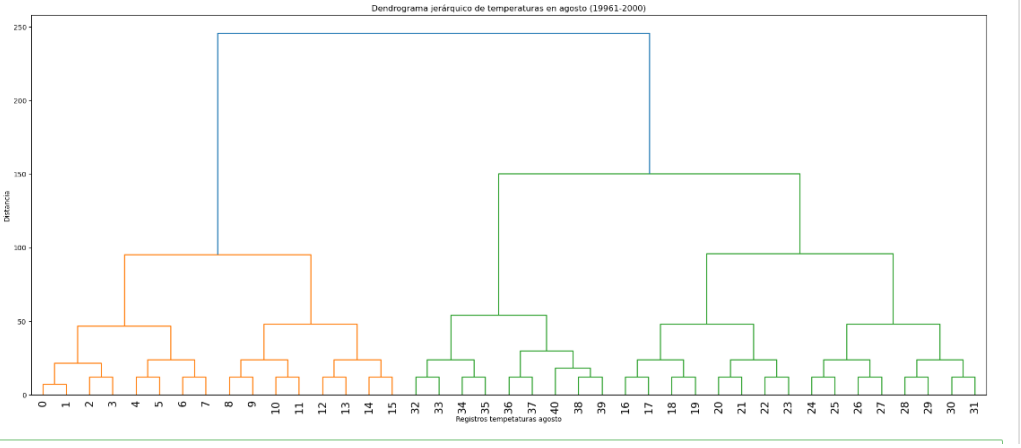
```
plt.figure(figsize=(25,10))
plt.title("Dendrograma jerárquico de temperaturas en agosto (19961-2000)")
plt.xlabel("Registros tempetaturas agosto")
plt.ylabel("Distancia")
dendrogram(Z, leaf_rotation=90.)
plt.show
```

```
<function matplotlib.pyplot.show(close=None, block=None)>
```




```
Z = linkage(X, 'average')
plt.figure(figsize=(25,10))
plt.title("Dendrograma jerárquico de temperaturas en agosto (19961-2000)")
plt.xlabel("Registros tempetaturas agosto")
plt.ylabel("Distancia")
dendrogram(Z, leaf_rotation=90., leaf_font_size=15.5)
plt.show
```

```
<function matplotlib.pyplot.show(close=None, block=None)>
```



Se observa que los primeros 15 meses de agosto se separan bastante de los 15 últimos porque para que junten ambas ramas del dendrograma necesitan mucha distancia vertical.

Los clusters también se pueden representar mediante otro tipo de diagramas mediante random y se han representado en el eje x las estaciones meteorológicas (desde la 0 hasta la 480 tomadas cada 7 unidades) y en el eje y las temperaturas promedio recogidas en estas estaciones.

Se ha encontrado un obstáculo y es que las temperaturas recogidas vienen en formato de número pero con punto y no con coma, lo cual ha supuesto una dificultad añadida a la hora de trabajar con este dataset, pero se ha resuelto visualizándolas estadísticas de columna y colocando en el eje y la media de los valores más frecuentes de las columnas escogidas.

Con las estaciones meteorológicas y las medias de los valores más frecuentes (valores enteros) se han creado unas matrices a partir de las cuales se van a obtener unas visualizaciones

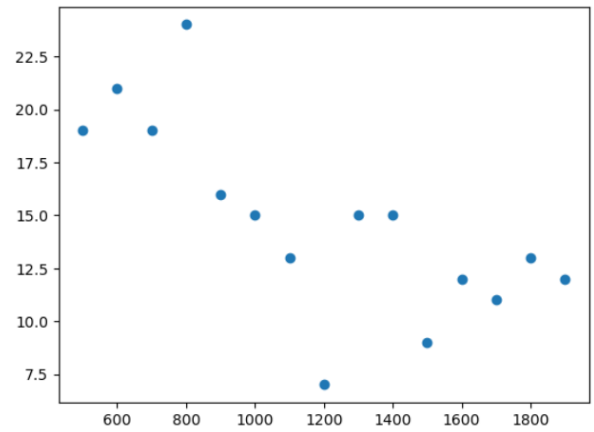
```
x = [500,600,700,800,900,1000,1100,1200,1300,1400,1500,1600,1700,1800,1900]
```

```
y = [19.54,21.03,18.74,24.24,16.16,14.71,13.53,6.59,15.52,14.71,9.33,11.71,11.03,12.59,11.58]
```

```
yentero=[19,21,19,24,16,15,13,7,15,15,9,12,11,13,12]
```

```
plt.scatter(x,yentero)
```

```
<matplotlib.collections.PathCollection at 0x2df88b055e0>
```



Lo cierto es que visualización no queda muy vistosa por la muestra reducida de datos con los que se han trabajado, pero se va ampliar la muestra, escogiendo todas las estaciones (500-1944) y todos los registros de agosto de 1973.

b. Arboles de clasificación

Para escoger una temperatura al azar y clasificarla en la estación meteorológica más probable.