Cajamar UniversityHack 2022

Cajamar Water Footprint

Equipo DPNA:

Pablo Torrijos Arenas
Mykola Mandzyak Melnyk
Alejandro Gómez Escribano

Escuela superior de Ingeniería Informática de Albacete
Universidad de Castilla la Mancha
21 de marzo de 2022

1. Introducción

El objetivo del reto Cajamar Water Footprint, dentro del Cajamar UniversityHack 2022, es predecir correctamente la demanda de agua potable durante dos semanas para 2747 contadores ubicados en el litoral de la Comunidad Valenciana, que pueden ser tanto de viviendas como de locales comerciales o industrias.

Para ello, se nos proporciona un dataset con las variables `ID`, `SAMPLETIME`, `READINGINTEGER`, `READINGTHOUSANDTH`, `DELTAINTEGER` y `DELTATHOUSANDTH`, con las que deberíamos tener el consumo con una frecuencia horaria desde el 01/02/2019 hasta el 31/01/2020.

2. Análisis de los datos

Lo primero que vemos si exploramos los datos es que contamos con contadores a los cuales les faltan muchas entradas horarias. Si cada contador debería contar con 8759 filas, llegamos a tener algunos para los que solo contamos con 4 horas. Esto es un gran problema, ya que realizar predicciones sobre estos contadores será una lotería, al no tener suficientes datos para realizar una predicción correcta. Si embargo, la mayoría de los contadores sí que tienen todos los datos que deberían. Además, hay datos negativos a los que tenemos que cambiar el signo.

Si buscamos valores perdidos, vemos cómo estos solo se encuentran en las columnas `READINGTHOUSANDTH` y `DELTATHOUSANDTH`, correspondientes a la parte decimal de los consumos. Al final esto no nos va a ser relevante, ya que directamente vamos a eliminar estas columnas. Esto se debe a que su representación de los valores decimales no es constante, y cuando aparece por ejemplo un valor de "2" en algunos casos representa un consumo real de "0.2" litros y en otros de "0.02", por lo que es prácticamente imposible entrenar un algoritmo que sepa diferenciar en qué casos debería dividir el número entre 10 y en cuáles entre 100.

Así, como obtendremos unos nuevos "deltas" utilizando la columna `READINGINTEGER`, un consumo constante de, por ejemplo, 7.2 litros durante 5 horas lo traduciremos en cuatro consumos de 7 litros y un consumo de 8. Este procedimiento no es óptimo, pero no debería afectar apenas al resultado final ya que además en este nos piden predecir por día o por semana, no por hora.

También fácil observar cómo cada fila tiene una hora con minutos y segundos distintos (por ejemplo, 16:04:57 y 16:37:12). Hemos supuesto que dichos valores de horas y segundos se han introducido aleatoriamente, ya que si los eliminamos se nos quedan perfectamente los 8759 valores para cada `ID` sin que se repita ninguna hora. Esta es la norma general, ya que hay unos pocos contadores que leen cada quince minutos (con algunos fallos en los que leen solo en 3 ocasiones en una hora, o en 5). En estos contadores hemos agrupado todos esos consumos en una única entrada horaria.

3. Contadores decrecientes y saltos en los valores

Uno de los dos tipos de preprocesamiento de los datos más delicados e importantes que hemos realizado para los casos en los que la variable `READINGINTEGER` decrece (ya que siempre debería de crecer). Esto es muy importante porque con esta variable obtendremos posteriormente nuestra variable `DELTA` final, que será la que usaremos para predecir ya que `DELTAINTEGER` no es confiable.

En la Figura 1 se pueden observar dos ejemplos claros de estos casos, aunque hay muchos más de todo tipo de estilos (se pueden observar algunos de ellos en el script de exploración). Así, parece a priori sencillo arreglar el caso de la primera gráfica de dicha figura (`ID` 1506), ya que cuando en una hora el valor haya decrecido podríamos sumarle a ella y a las posteriores la diferencia con la anterior, eliminando así todas las bajadas, ya que la tendencia creciente en ese contador sí que parece correcta. Sin embargo, ese proceso no funcionaría en el (`ID` 2140) ya que como en ese caso la tendencia correcta es la decreciente, estaríamos obteniendo una recta plana con el método explicado.

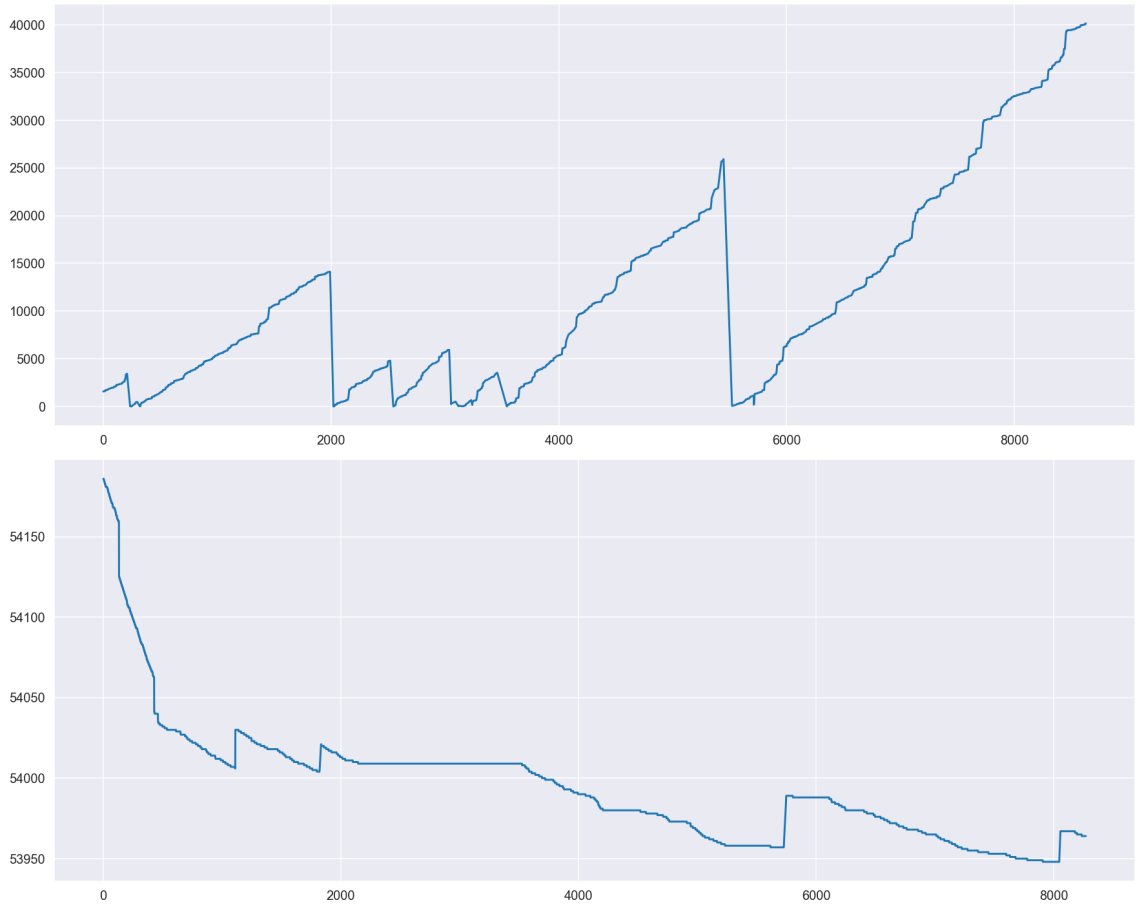

Figura 1: `READINGINTEGER` para los `ID` 1506 y 2140

Así, el método definitivo que hemos usado consiste en que, cuando en una hora el valor del `READINGINTEGER` se reduce, sumamos a esa hora y a todas las posteriores el doble de la diferencia con la hora anterior. Así, básicamente lo que hacemos es invertir la curva, convirtiendo las bajadas en subidas. Este método funcionaría bien en el segundo caso de la Figura 1, pero en el primero convierte los saltos hacia abajo en saltos hacia arriba que deberemos eliminar.

Para eliminarlos, hemos implementado una función propia de detección de anomalías que detecte cuándo un consumo en una hora determinada es muy grande en comparación con los consumos normales de ese `ID` y, cuando eso ocurra, reste a esa hora y a las posteriores el consumo que se había realizado. Sin conocer los datos reales, parece que las gráficas que hemos obtenido al realizar este proceso son bastante más acordes a lo que esperamos, pero seguramente habrá algunos consumos que realmente eran muy grandes en una hora específica y hemos eliminado. En la Figura 2 se puede ver el resultado de este proceso en las dos líneas azules, pasando de la superior a la inferior.

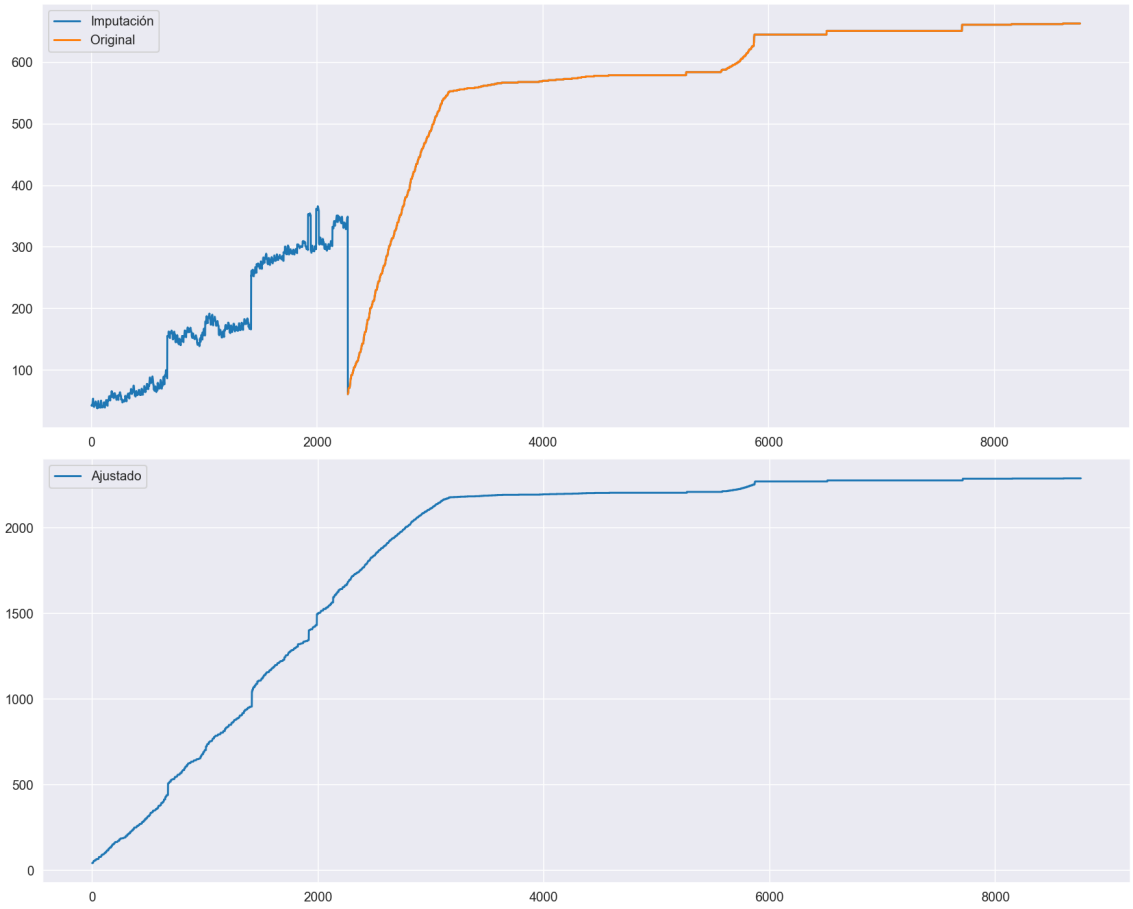

Figura 2: Imputación de valores de `READINGINTEGER` para el `ID` 72

4. Imputación de datos

El otro gran preprocesamiento que hemos realizado es el de la imputación de todos los datos para las filas que nos faltan. Para ello, hemos utilizado dos métodos:

- **df.interpolate():** Cuando hay una serie de horas que nos faltan pero contamos con algunas anteriores y posteriores, directamente hemos utilizado esta función de Pandas que nos traza una línea entre ambos puntos. No es óptima, pero parece que funciona mejor que si utilizamos otro tipo de imputadores o modelos para esta tarea.
- **IterativeImputer:** En los casos en los que tenemos que predecir horas al principio o al final de la serie, utilizaremos este imputador de Scikit-learn. El problema es que en algunos casos (como en el que se puede ver en la gráfica superior de la Figura 2) produce saltos y valores inestables. Esto lo podemos solucionar aplicando otra vez el mismo proceso que para los contadores decrecientes y los saltos en los valores, obteniendo en este ejemplo del `ID` 72 el resultado que podemos ver en la gráfica inferior de la Figura 2, el cual a priori parece bastante coherente.

5. Estacionalidad

Ya que tenemos una serie temporal, es importante comprobar su estacionalidad. Podemos ver en la Figura 3 cómo el valor de DELTA (la variable a predecir que hemos creado) parece ser estacional con respecto al día de la semana, y sigue una tendencia descendente hacia un menor consumo a lo largo de los meses, habiendo un gran cambio sobre el mes de octubre. Además, con el paso de los meses también se reduce el ruido en la serie.

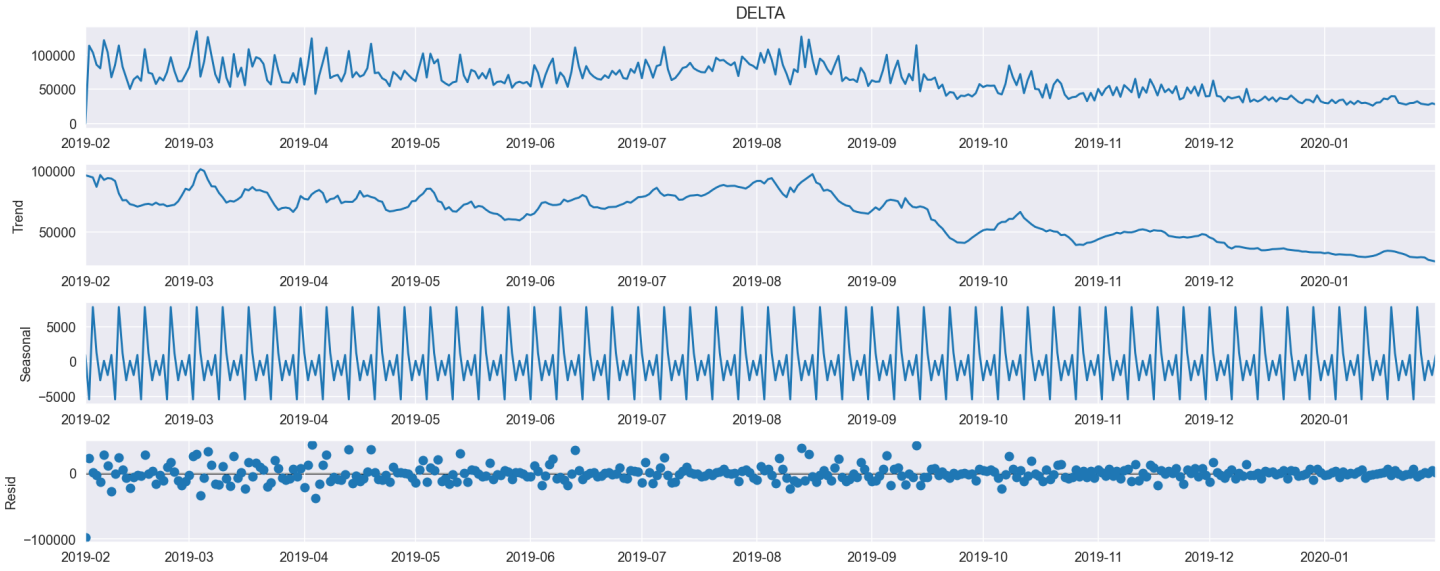

Figura 3: Estacionalidad de DELTA

6. Enriquecimiento de datos

Enriquecer los datos es un proceso muy importante. Para ello, en primer lugar hemos considerado que podría ser muy relevante tener datos climatológicos, y hemos encontrado unos datos horarios muy valiosos de temperatura, precipitaciones, humedad y cantidad de sol en la ciudad de Valencia que nos pueden ser muy útiles [1]. Se supone que los datos en los que estamos prediciendo son de toda la Comunidad Valenciana, pero al final deberían de ser parecidos por lo que los vamos a usar.

Por otro lado, también hemos pensado que el consumo de agua podría aumentar dependiendo de la cantidad de turistas que haya en la Comunidad Valenciana. En este caso, hemos utilizado los datos de [2], de los que hemos obtenido distintos datos de ocupación hostelera en la ciudad de Valencia (tanto diarios como mensuales). Hemos dejado varios de ellos ya que varían dependiendo de la fuente.

Además, hemos eliminado la columna `SAMPLETIME` y añadido una nueva para el año, otra para el mes, el día y la hora. Además, también hemos incluido una nueva columna en la que se refleja el día de la semana, que seguramente será relevante para el consumo.

7. Modelos predictivos

Por último, vamos a explicar los modelos predictivos con los que realizaremos nuestra predicción final, que será un "ensemble" de los mismos haciendo la media de sus predicciones, ya que de esta forma reducimos la varianza y conseguiremos un mejor resultado que utilizando únicamente uno de los modelos (claro está, siempre y cuando no añadamos un modelo con predicciones totalmente erróneas). Estos modelos los podemos entrenar tanto con el conjunto de datos enteros, como entrenando un modelo distinto para cada `ID`, y serán los siguientes:

- **XGBRegressor:** Este método de eXtreme Gradient Boosting parece obtener buenos resultados entrenando un modelo para cada `ID`, por lo que lo añadiremos al "ensemble".
- **GradientBoostingRegressor:** En este caso, vamos a añadir dos tipos de Gradient Boosting: uno entrenándolo con todos los datos, y otro creando un modelo por `ID`, ya que ambos consiguen buenos RMSE.
- **IterativeImputer:** Por último, como hemos comprobado que este imputador funciona bastante bien siguiendo la tendencia de la serie, vamos a incluir la predicción que haría para los siguientes 14 días como si dichos valores fuesen valores perdidos en nuestros datos.

Así, con este "ensemble" obtenemos resultados que a priori deberían ser bastante mejores que los obtenidos en la entrega intermedia.

[1] https://datos.gob.es/es/catalogo/l01462508-datos-horarios-de-calidad-del-aire-desde-2016
[2] https://fundacion.visitvalencia.com/sit