

Universidad Europea de Madrid

Escuela de Arquitectura, Ingeniería y Diseño

SMART ENERGY

Aplicaciones y tendencias de análisis de datos

Profesor: Álvaro Sánchez Pérez

**Presentado por: Sofía Corral, Elena Delgado, María
Fernández y Nicola Fontaine**

26 de octubre 2021

ÍNDICE

INTRODUCCIÓN	2
PREPARACIÓN DE DATOS	2
MODELO	5
VISUALIZACIÓN	9
TEMPERATURAS	10
HUMEDAD	11
VISIBILIDAD	12
VIENTO	12
NUBOSIDAD	13
OTROS	13
CONCLUSIONES Y VALOR AÑADIDO	15

INTRODUCCIÓN

El proyecto Smart Energy tiene como objetivo obtener información del consumo eléctrico en diferentes hogares de la ciudad de Londres, determinando qué variables meteorológicas o relativas al calendario laboral son las que pueden influir en él. Todo el desarrollo del trabajo es posible gracias al IoT (Internet de las cosas), ya que cada medida y registro de energía se obtiene a través de una serie de medidores inteligentes ubicados en diversos hogares londinenses.

En añadido, para la realización del trabajo, donde estudiaremos cómo afecta el tiempo al gasto de energía, generaremos un modelo con el fin de predecir a futuro el consumo, de manera que sirva de ayuda y/o soporte a compañías energéticas para optimizar el abastecimiento en la ciudad.

Para llevarlo a cabo contamos con un conjunto de datos sobre el consumo de energía en más de 5000 hogares a lo largo de un periodo de tiempo de varios años. Estos registros se dividen en varios datasets que contienen información necesaria para la realización del proyecto.

Como empresa, no solo queremos hacer un análisis propio de los datos para saber cuándo se suceden los puntos altos de gasto de energía y poder abastecer a las casas, sino que pretendemos a su vez optimizar la gestión de los recursos energéticos de las compañías al cargo.

Para ello estudiaremos cómo se comporta el tiempo visualizando las épocas más frías y las épocas más calurosas. No solo queremos estudiar la temperatura, asimismo analizaremos más factores que puedan afectar a esa temperatura, como puede ser la visibilidad, el viento, lluvia, etc...

Como primeros pasos, se hará una visualización de los datos temporales para poder ver qué variables afectan más a la temperatura y de esa manera saber con qué valores trabajaremos en nuestro modelo.

PREPARACIÓN DE DATOS

Para enfrentarnos a este proyecto teníamos un total de dos csv: `uk_bank_holidays` y `weather_daily_darksky`; y dos zips con más de 100 csv en su interior: `daily_dataset` y `halfhourly_dataset`.

El dataset de `uk_bank_holidays`, nos indica las fechas de las vacaciones y el nombre de cada festivo. Esto nos puede servir para saber si el calendario laboral de la ciudad afecta al consumo de luz y en qué medida.

Weather_daily_darksky indica los valores sobre diferentes variables climáticas como pueden ser la temperatura, la sensación térmica, el viento, humedad, etc... Datos que se habían obtenido en un periodo de cuatro años, del 2011 al 2014.

- **daily_dataset:** en él encontramos los registros obtenidos a partir de los dispositivos IoT ubicados en cada hogar, es decir, cuenta con todas las medidas de energía en kilovatios hora por día en cada una de las casas.
- **halfhourly_dataset:** en este caso encontramos la misma serie de atributos que en el anterior con la diferencia de que el registro aquí está recogido por horas en lugar de ser diario.

Para entender mejor las variables que contenían cada csv y conseguir un mejor resultado del modelo, procedimos a realizar los siguientes pasos:

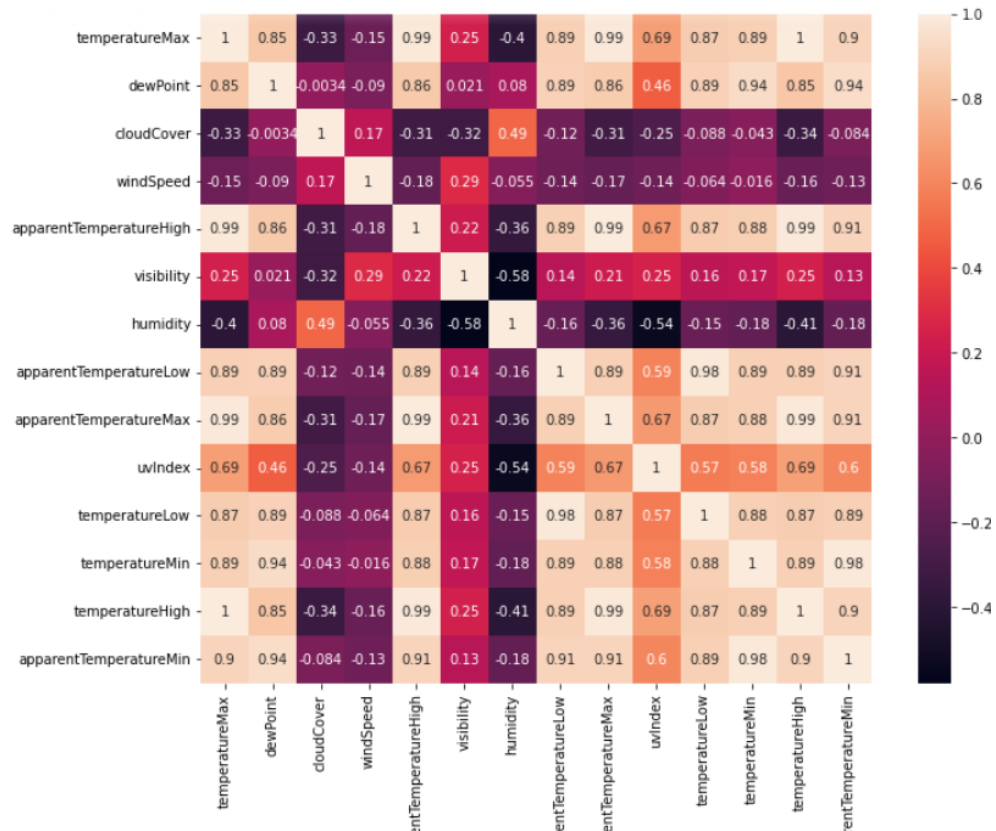
- **Cargar los diferentes csv:** En el caso de los zip, fue necesario realizar un proceso más complejo para conseguir ver las variables y los datos de su interior. Fue necesario crear un array con todas las direcciones o nombres de los csv que se encontraban en el interior del fichero zip. Posteriormente, procedimos a leer todas aquellas direcciones con un bucle 'for' y seguidamente vamos uniéndonos en un mismo csv. Por lo tanto, el resultado final es un csv con la información del zip unida.

```
dfbank = pd.read_csv('uk_bank_holidays.csv')
dfbank.head()
```

```
#Hacer lo de half con daily
rutadaily = 'daily_dataset'
csv_daily = os.listdir(rutadaily)
csv_daily
archivos1 = []
for i in csv_daily:
    direccion = "daily_dataset/" + i
    archivos1.append(direccion)
archivos1
```

- **Creamos un mapa de calor:** Gracias a él podemos detectar que existe multicolinealidad entre algunas variables y por ello, decidimos que la mejor solución para obtener un buen resultado del modelo sería borrar dichas variables. Observamos que el índice de rayos ultravioleta, junto con la visibilidad, se encontraban altamente correladas con la temperatura, por lo

que decidimos no contar con ellas. De la misma manera sucede con el punto de condensación (dew point) y la nubosidad, ya que su correlación con la humedad resultaba considerablemente alta.



- Comprobamos si hay valores nulos: En este aspecto descubrimos que existían un número mínimo de filas con valores nulos por lo que decidimos eliminarlas pese a que con la gran cantidad de filas que teníamos, apenas afectaba al modelo.

```
dfwea.isnull().sum()
dfwea = dfwea.dropna()
len(dfwea)
```

- Comprobar si hay valores duplicados: Para confirmar que el proceso realizado en la carga del fichero zip había sido correcto y no se había duplicado ningún dato, procedimos a ejecutar un comando que nos elimina esas filas duplicadas. No obstante, posteriormente comprobamos que no se había borrado ninguna fila puesto que la longitud del csv era la misma.

```
dfwea_duplicates = dfwea.drop_duplicates()
len(dfwea_duplicates)
```

- Unión de diferentes CSVs: Para finalizar con el proceso de preparación, procedimos a unificar los csv con los que íbamos a trabajar y a aplicar el modelo de regresión. Para ello, usamos el método 'merge' el cual en función a una variable común como es la fecha en nuestro caso, procedemos a unificar ambos csv. Previamente, fue necesario convertir ambas variables de fecha en el mismo formato.

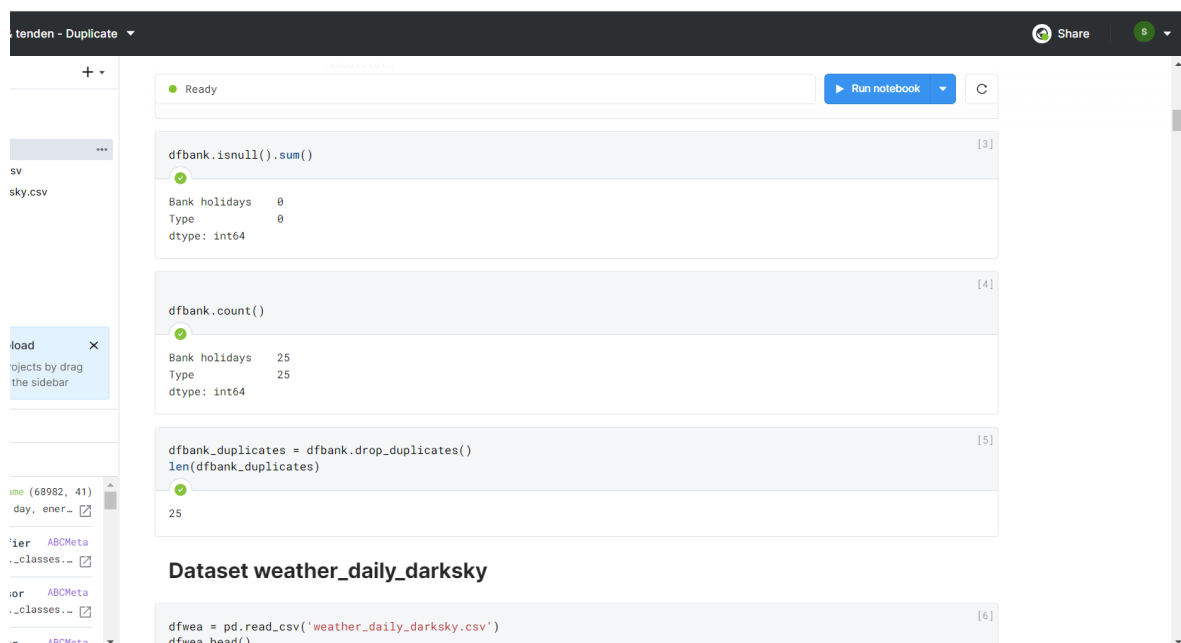
```
dfwea['time'] = pd.to_datetime(dfwea['time'])
df_daily['day'] = pd.to_datetime(df_daily['day'])
```

```
DFOBJETIVO = pd.merge(df_daily, dfwea, left_on='day', right_on='time')
DFOBJETIVO
```

MODELO

La secuencia de pasos que hemos seguido para la realización del modelo se muestra debidamente ejecutada en el notebook de python adjunto a la documentación del proyecto

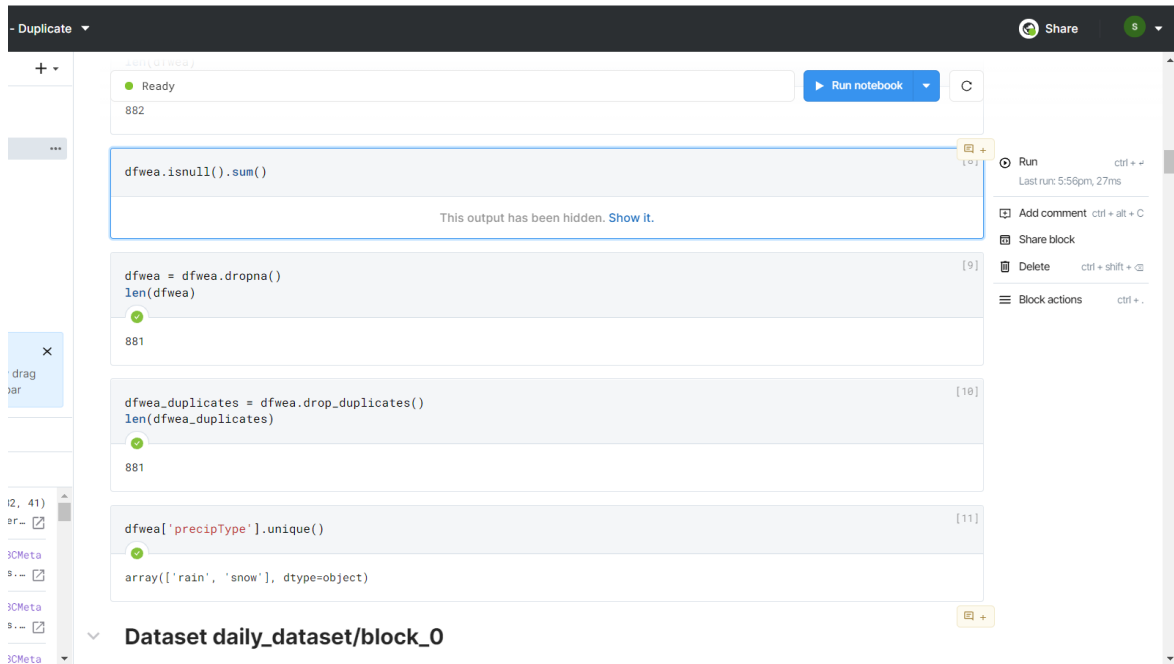
En primer lugar, una vez cargamos en python los diferentes datasets, hemos realizado un proceso de limpieza y tratamiento de los datos, en el cual estudiamos si los registros nulos son necesarios, el formato de fechas, analizamos las variables, etc.



The screenshot shows a Jupyter Notebook with the following code cells and outputs:

- Cell [3]: `dfbank.isnull().sum()`
Output: Bank holidays: 0, Type: 0, dtype: int64
- Cell [4]: `dfbank.count()`
Output: Bank holidays: 25, Type: 25, dtype: int64
- Cell [5]: `dfbank_duplicates = dfbank.drop_duplicates()`
`len(dfbank_duplicates)`
Output: 25
- Cell [6]: `dfwea = pd.read_csv('weather_daily_darksky.csv')`
`dfwea.head()`

The notebook interface includes a sidebar with file explorer and variable inspector, and a top bar with 'tenden - Duplicate' and 'Share' options.



The screenshot shows a Jupyter Notebook with the following code blocks and outputs:

- Block 1: `dfwea.isnull().sum()` (Output: 882)
- Block 2: `dfwea = dfwea.dropna()` and `len(dfwea)` (Output: 881)
- Block 3: `dfwea_duplicates = dfwea.drop_duplicates()` and `len(dfwea_duplicates)` (Output: 881)
- Block 4: `dfwea['precipType'].unique()` (Output: `array(['rain', 'snow'], dtype=object)`)

The notebook interface includes a 'Run notebook' button, a 'Share' button, and a sidebar with a 'Dataset daily_dataset/block_0' entry.

Tras ello, hemos cargado en bucle el zip con la información del consumo de energía. En este bucle, insertamos los registros en una única lista para posteriormente convertirlos en csv.

```
#Hacer lo de half con daily
rutadaily = 'daily_dataset'
csv_daily = os.listdir(rutadaily)
csv_daily
archivos1 = []
for i in csv_daily:
    direccion = "daily_dataset/" + i
    archivos1.append(direccion)
archivos1
```

daily_dataset/block_00-20211007-165036.csv ,
'daily_dataset/block_24-20211007-165037.csv',
'daily_dataset/block_64.csv',
'daily_dataset/block_18.csv',
'daily_dataset/block_74-20211007-165041.csv',
'daily_dataset/block_80.csv',
'daily_dataset/block_51.csv',

```
[16]

list_data_daily = []

# Escribimos un loop que irá a través de cada uno de los nombres de archivo a través de globbing y el resultado final :
#Solo ejecuto los archivos 1-6 porque todo peta
for filename in archivos[1:6]:
    data = pd.read_csv(filename)
    #creo que estan duplicados los csv en la carpeta de daily, así que no añado repetidos, pero no va así jej
    #if data not in list_data_daily:
    list_data_daily.append(data)
#Para chequear que todo está bien, mostramos la list_data por consola
list_data_daily

pd.concat(list_data_daily, ignore_index=True)
```

	LCLid object	day object	energy_median float64	energy_mean float64	energy_max float64	energy_count int64	energy_std float64
0	MAC000095	2011-12-12	0.2365	0.2453846153846154	0.552	26	0.1264851222628
1	MAC000095	2011-12-13	0.245	0.2289574468085106	0.502	47	0.1068188139768
2	MAC000095	2011-12-14	0.08049999999999999	0.09566666666666666	0.481	48	0.0705912216028

```
[26]

DFOBJETIVO = pd.merge(df_daily, dfwea, left_on='day', right_on='time')
DFOBJETIVO
```

	LCLid object	day datetime64[ns]	energy_median float64	energy_mean float64	energy_max float64	energy_count int64	energy_std float64
0	MAC000095	2011-12-12T00:00:00.000000	0.2365	0.2453846153846154	0.552	26	0.1264851222628
1	MAC000155	2011-12-12T00:00:00.000000	0.202	0.32129166666666667	1.044	48	0.257
2	MAC000054	2011-12-12T00:00:00.000000	0.09	0.16685416666666667	0.753	48	0.145
3	MAC000056	2011-12-12T00:00:00.000000	0.1485	0.18610416666666666	1.145	48	0.233

Llegadas a este punto, contamos con un csv con los datos de tiempo y otro con el consumo de energía. Nuestro objetivo aquí es conseguir uno único con todos los registros según la fecha (para lo cual tuvimos que unificar el tipo de dato que es

```
dfwea['time'] = pd.to_datetime(dfwea['time'])
df_daily['day'] = pd.to_datetime(df_daily['day'])
```

fecha en ambos:), por lo que realizamos un merge de ambos y obtenemos un dataframe final con el registro del tiempo y de la energía por días.

Una vez ya tenemos un único dataframe, el siguiente paso será por fin crear entrenar un modelo de regresión.

Los algoritmos utilizados han sido Regresión Lineal y Random Forest. Para la aplicación de ambos sobre nuestros datos resultó necesario categorizar aquellos atributos no numéricos, de modo que posteriormente nos fuera posible convertir cada categoría en un número a través de una función en python.

En el caso de la regresión lineal, al entrenar con un 80% de los datos, obtenemos los siguientes resultados:

```
In [86]: print('Coefficients:', model.coef_)
print('Intercept:', model.intercept_)
print('Mean squared error (MSE): %.2f'
      % mean_squared_error(Y_train, Y_pred_train))
print('Coefficient of determination (R^2): %.2f'
      % r2_score(Y_train, Y_pred_train))
```

Coefficients: [-8.42567477e-08 1.81347406e-01 1.09054646e+00 -3.80199656e-04
-8.41306136e-05 -7.56036967e-04 -3.23984928e-04 1.51451317e-03
3.71305095e-04 -5.49033824e-04 2.72291665e-04 4.28703872e-02
-2.49121080e-04 2.80353683e-04 -2.05827245e-03 5.01262492e-04
-1.13290935e-04 -1.45940299e-03 2.00538312e-05 -9.98224756e-04
2.91965869e-04 -3.71539111e-05]
Intercept: 1.9847832617523475
Mean squared error (MSE): 0.01
Coefficient of determination (R^2): 0.83

En cambio, en el caso del random forest, al utilizar una división 75-25 obtenemos:

```
In [78]: # Sobre el test
predictions = rf.predict(test_features)
# calcular errores absolutos
errors = abs(predictions - test_labels)
# imprimir el error
print('Mean Absolute Error:', round(np.mean(errors), 2), 'degrees.')
```

Mean Absolute Error: 0.03 degrees.

En ambos obtenemos errores muy bajos debido a que el rango de valores de energía media en el que nos encontramos es [0, 7] kilovatios hora por hogar.

En cuanto a error absoluto, observamos que es menor en el caso de random forest, ya que en la regresión lineal obtenemos un MSE de 0.01, lo cual se convierte en 0.1 kilovatios hora de error. No obstante, debido a que la ejecución de random forest en nuestro conjunto resulta mucho más compleja y un error tan pequeño puede provocar sobreajuste, decidimos realizar las predicciones con regresión lineal.

Aplicamos el modelo de manera que solicitamos al usuario una serie de valores que nos permitan obtener una cantidad de energía como resultado:

```

year = input("Año: ")
month = input("Mes: ")
day = input("Día: ")

energia_media = (model.intercept_ + (model.coef_[0]*int(LCLid)) + (model.coef_[1]*energy_max) + (model.coef_[2]*ener
                + (model.coef_[3]*temperatureMax) + (model.coef_[4]*int(icon)) + (model.coef_[5]*dewPoint) + (model.
print ("La energía media consumida en el día que has señalado es: ", '%.4f' % energia_media)

```

Indica código de hogar (1-5555): 218
 Indica si 0-Despejado, 1-Nublado, 2-Niebla, 3-Parcialmente nublado por el día, 4-Parcialmente nublado por la noche,
 0
 Escribe el nivel de nubosidad [0-1]: 0.3
 ¿Y la velocidad del viento? [0-10] 3
 Tipo de precipitación (1 si llueve y 0 si nieva): 1
 Visibilidad del ambiente [1-15]: 13
 Qué humedad [0-1]: 0.5
 Índice de rayos ultravioleta (0,1,2,3): 3
 Año: 2011
 Mes: 7
 Día: 25
 La energía media consumida en el día que has señalado es: 0.2372

De esta forma conseguimos un resultado válido para un posible cliente.

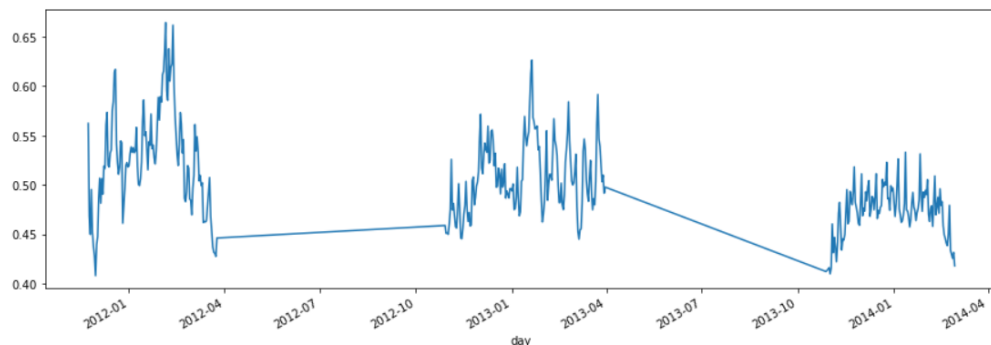
Por último, gracias a una serie de visualizaciones ejecutadas también en python observamos que el consumo energético es cíclico a medida que se suceden los años, los picos se alcanzan siempre en los mismos periodos. Sin embargo, cierto es que encontramos una ligera disminución del consumo de energía, lo cual no podemos determinar a qué se debe, pero quedamos pendientes de un futuro estudio.

```

In [7]: num_households_df = dataviz.reset_index().groupby("day").nunique()["LCLid"] # numero de hogares cada día
energy_df = dataviz.reset_index().groupby("day").sum()["energy_mean"] # suma de las medias por día

energy_per_household_df = pd.concat([num_households_df, energy_df], axis=1)
energy_per_household_df["normalised"] = energy_per_household_df["energy_mean"] / energy_per_household_df["LCLid"]
energy_per_household_df["normalised"].plot(figsize=(15,5), label = "Energía por hogar");

```

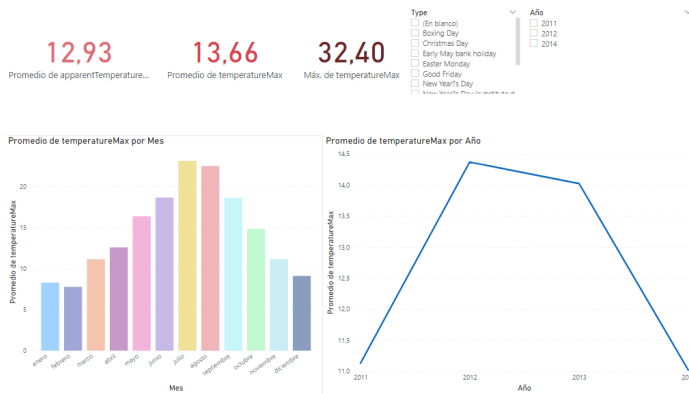


VISUALIZACIÓN

Finalmente, visualizamos los datos aportados para obtener una conclusión y poder aportar algo más visible a nuestro cliente. Para ello se ha hecho uso de Power BI, donde mediante unas gráficas y KIPs se pueden concluir las siguientes conclusiones.

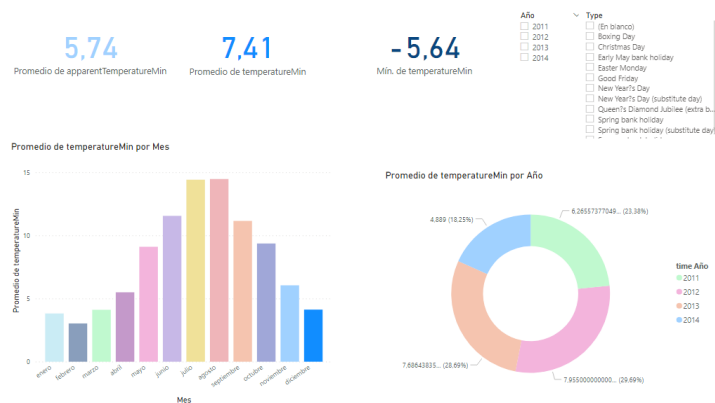
TEMPERATURAS

En cuanto a las temperaturas, se visualizan primero por separado, para poder observar según el año y la época del año ambas temperaturas. Para ello están las siguientes capturas:

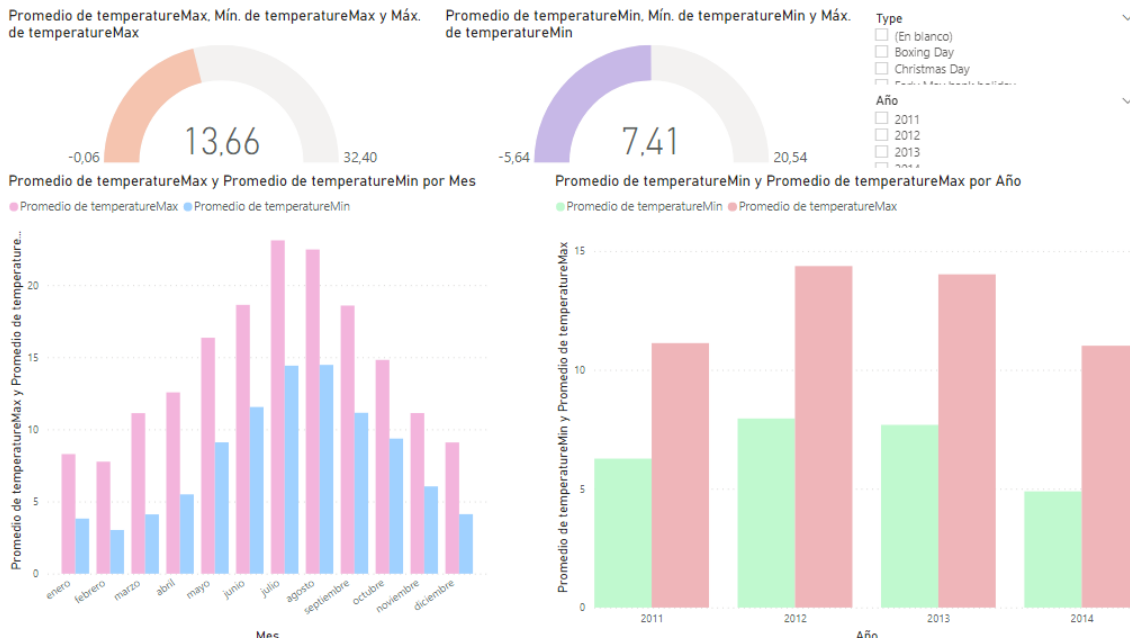


Mediante la captura previa podemos observar que los meses con mayor temperatura son los meses de verano, sobre todo en julio y agosto vemos una subida de temperatura. En cuanto a los años, en el año 2012 hubo una temperatura más alta en comparación con el 2013.

También, visualizamos que los meses con menos temperatura son los meses de invierno, en especial enero y febrero. En estos casos queríamos ver la temperatura en la época de navidad, donde visualizamos que en "CHRISTMAS BREAK" no son nuestras temperaturas más frías.

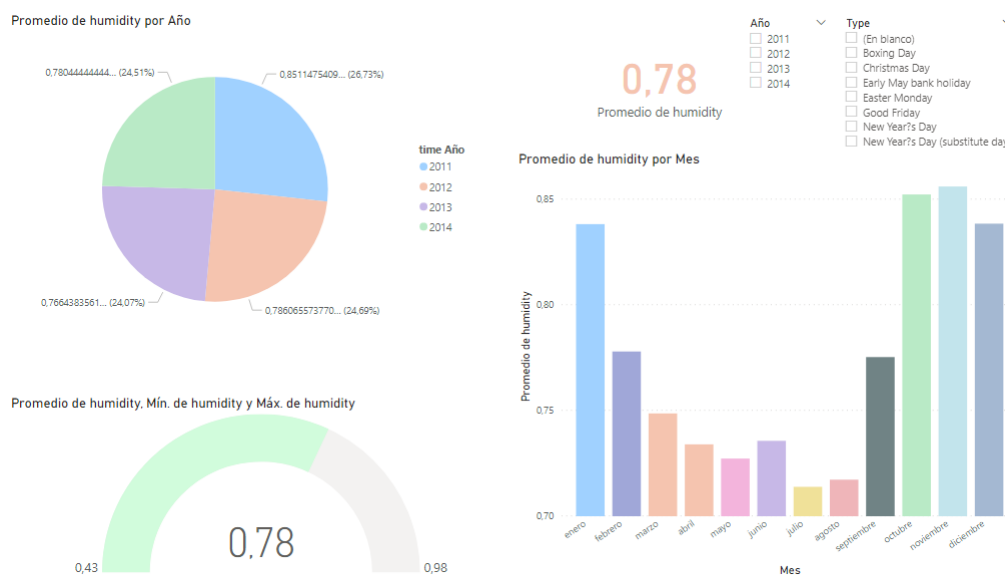


Para visualizar ambas variables a la vez, se realiza la siguiente visualización donde comparamos la temperatura mínima con la temperatura máxima.



HUMEDAD

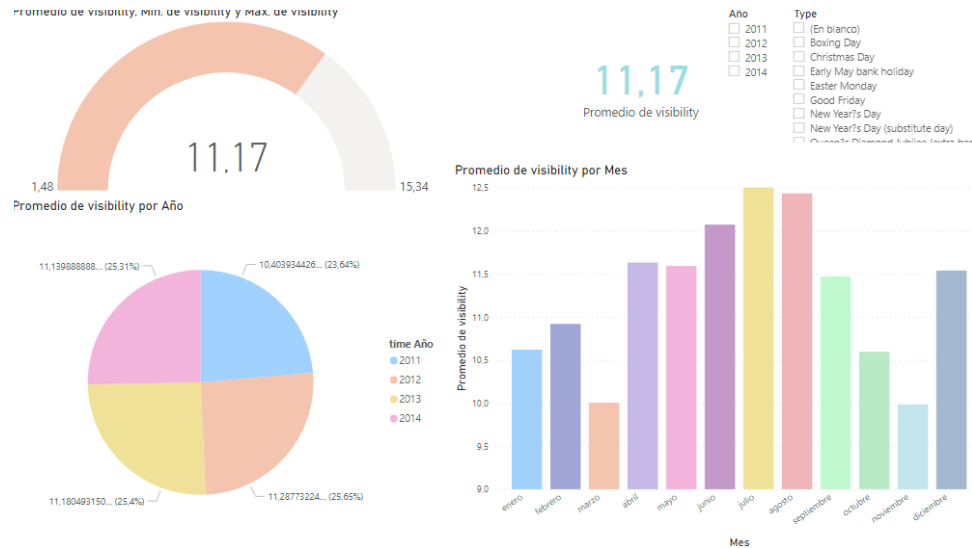
En cuanto a la humedad, se visualiza la siguiente captura, para poder observar según el año y el mes el cambio en humedad, tenemos en cuenta que lo más probable es que la temperatura vaya mano a mano con la humedad. Pero, en nuestros datos no tenemos información sobre la precipitación entonces no podemos hacer un estudio de eso.



En cuanto a la humedad se puede ver que los meses más fríos son en los que más humedad hay, aunque, cabe mencionar que, los meses más fríos son enero y febrero y en cambio, los más húmedos con octubre y noviembre, meses que tienden a ser más lluviosos.

VISIBILIDAD

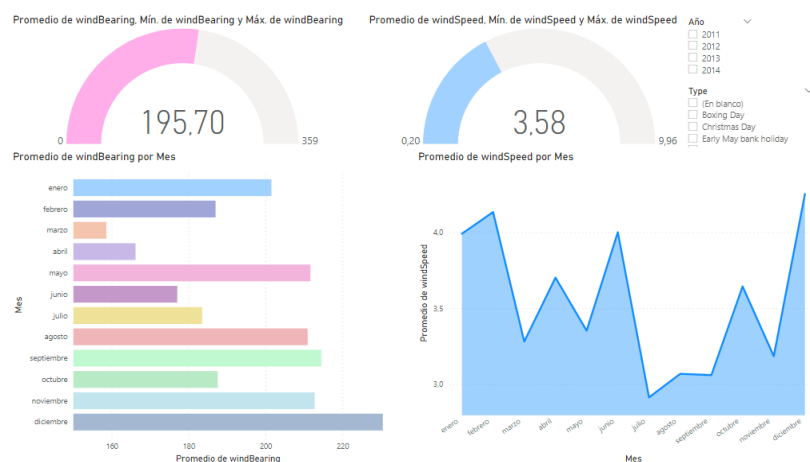
A continuación, teniendo en cuenta la humedad vamos a visualizar la visibilidad que hay según el mes a ver si la humedad y la temperatura afecta la visibilidad.



La visibilidad no cambia mucho durante el año, podemos observar que durante los meses más calurosos la visibilidad es mejor, sobre todo en julio y agosto. Al contrario que en los meses con más humedad, donde la visibilidad baja, estos meses son octubre y noviembre. Podemos observar que marzo también es un mes con poca visibilidad, pero, no es un mes con gran humedad.

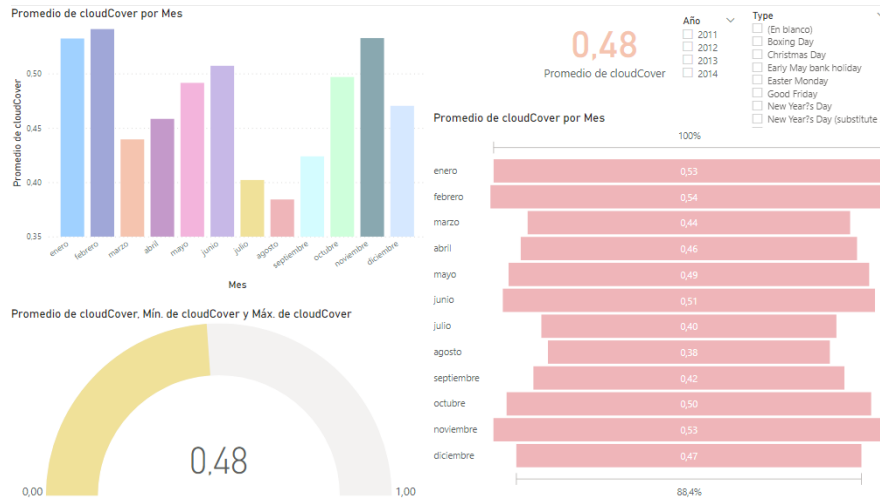
VIENTO

A continuación, hacemos un estudio del viento teniendo en cuenta dos variables "WIND BEARING" y "WIND SPEED". Algo que nos causa curiosidad es que el mes de marzo pese a tener poca visibilidad, es el mes con menos aire. También se observa que el viento no varía tanto por épocas como la temperatura, si no que, fluctúa más según el mes.



NUBOSIDAD

Otra variable que hemos visualizado es la nubosidad, donde visualizamos que en los meses más calurosos es cuando menos nubes hay, como se ve en julio y agosto. En cambio en los meses más fríos, enero y febrero, es en los meses con más nubes.

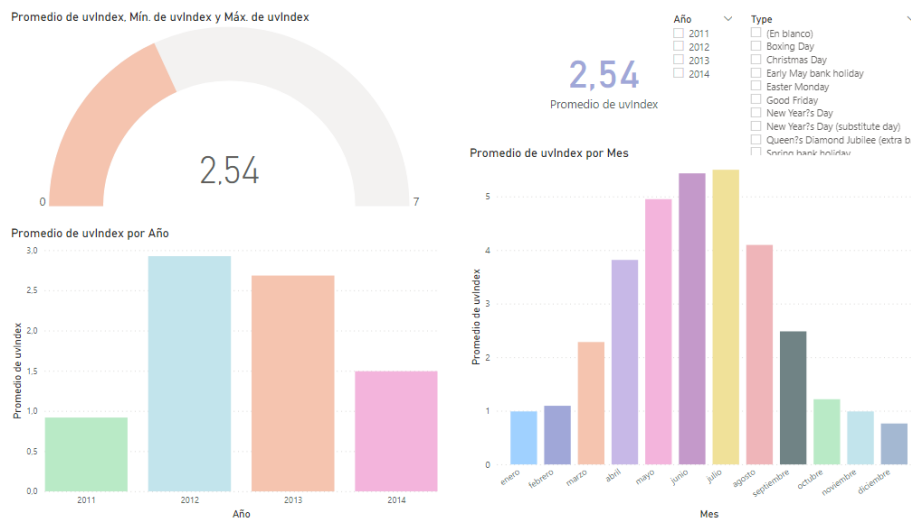


OTROS

Finalmente, faltan algunas variables que sí hemos visualizado pero que analizamos en conjunto, puesto que hemos considerado que no tienen tanta importancia como las mencionadas previamente.

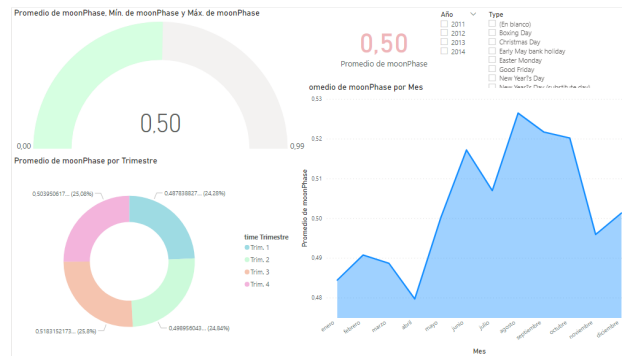
DARK SKY

La variable de “DARK SKY” nos muestra cuando el cielo de noche tiene menos contaminación lumínica. Donde a mayor la temperatura, menos contaminación lumínica nos encontramos.

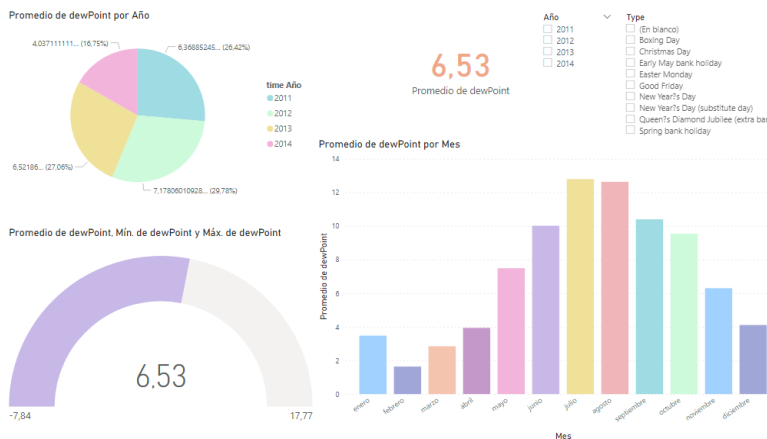


MOON PHASE

También hemos visualizado la luminosidad de la luna según la etapa, aunque esta variable no nos aporta mucho porque la luna va cambiando mensualmente.



DEW POINT

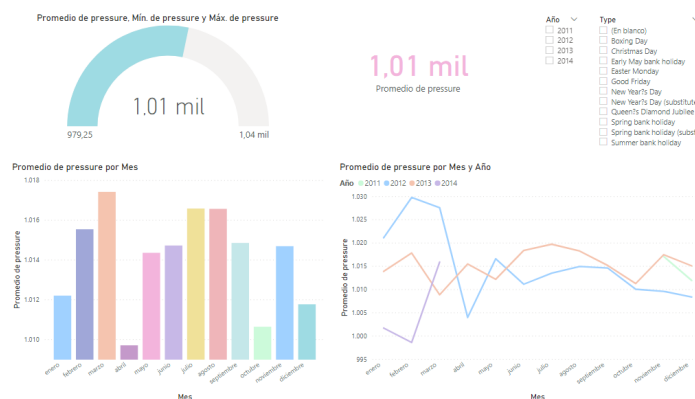


La condensación también es una variable visualizada, donde efectivamente en los meses más fríos la temperatura atmosférica, que varía según la presión (variable que veremos a continuación) y la humedad, por la cual debajo de esa temperatura las gotas de agua se

condensan y crean rocío. Podemos ver que a menor humedad, mayor temperatura hay de "DEW POINT".

PRESSURE

Finalmente tenemos la presión atmosférica, donde vemos que no hay una tendencia por etapas, si no que va cambiando un mensualmente. Aunque se puede visualizar que en abril y octubre es cuando menos presión hay.



CONCLUSIONES Y VALOR AÑADIDO

En conclusión y gracias al estudio de los datos, tanto en Python como en la herramienta de Power BI, podemos enunciar con seguridad que las variables que más afectan al consumo de energía son la temperatura máxima (lo cual ya podíamos prever), la humedad y la velocidad del viento, lo cual, conociendo el pronóstico meteorológico, puede ayudar a las energéticas a modificar su abastecimiento, mas siempre teniendo en cuenta que el hecho de que los festivos del calendario laboral actúan como valores atípicos.

Además, según las visualizaciones obtenidas en python, visibles en los archivos adjuntos, resulta llamativo que la disminución en el consumo puede significar un cambio en el comportamiento del consumidor, por lo que a futuro estaría bien estudiar y analizar qué es lo que lo está provocando.

A modo de valor añadido al cliente, a partir de la información obtenida por el trabajo realizado podemos proporcionar una minimización de las pérdidas energéticas de las compañías, ya que toda aquella energía generada y/o comprada que no puede ser almacenada, se pierde, por lo que a través de las predicciones obtenidas por los algoritmos conseguimos una racionalización del consumo.

Por el momento estamos muy contentos de haber podido obtener toda esta información gracias a los medidores inteligentes ubicados en cada uno de los más de 5000 hogares londinenses, son la clara representación real de la utilidad del IoT en los tiempos que corren, ya que gracias a ellos se ha podido analizar el comportamiento del consumo, lo puede suponer un gran avance en materia de ahorro energético y sostenibilidad.