



UNIVERSIDAD
NEBRIJA

Visualización y predicción de series temporales con Python.

**UNIVERSIDAD NEBRIJA GRADO EN
INGENIERÍA INFORMÁTICA**

Nombre Alumno: Juan Manuel Rodríguez Etchepare

Fecha: Junio, 2024



UNIVERSIDAD
NEBRIJA

Visualización y predicción de series temporales con Python.

UNIVERSIDAD NEBRIJA GRADO EN
INGENIERÍA INFORMÁTICA

Nombre Alumno: Juan Manuel Rodríguez Etchepare

Fecha: Junio, 2024

Nombre Tutor: Adrian Pradilla Portoles

Dedicatoria (opcional).

Índice

Índice.....	5
Índice de ilustraciones.....	6
Índice de tablas.....	9
Resumen.....	11
1. DESARROLLO / INVESTIGACIÓN.....	12
1.1. Descripción de la memoria.....	12
1.2. Desarrollo / Investigación.....	12
1.3. Metodología.....	16
1.4. Resultados.....	30
1.5. Conclusión.....	39
1.6. Lineas futuras.....	39
1.7. Aplicabilidad.....	40
2. BIBLIOGRAFÍA.....	40
3. ANEXOS.....	42

Índice de ilustraciones

- *Figura 2.1: Creación de árboles XGBRegressor sobre dataset de meteorología*
- *Figura 2.2: Creación de árboles XGBRegressor sobre dataset de energía*
- *Figura 2.3: Tipos de datos en dataset de energía*
- *Figura 2.4: Tabla dataset energía con valores iniciales*
- *Figura 2.5: Limpieza de columnas vacías en dataset de energía*
- *Figura 2.6: Dataset de meteorología completo*
- *Figura 2.7: Representación gráfica de datos por ciudad*
- *Figura 2.8: Generación de columnas en dataset de energía*
- *Figura 2.9: Creación de conjuntos de entrenamiento y prueba*
- *Figura 2.10: Dataframe de meteorología sin lags*
- *Figura 2.11: Delimitación temporal entre conjuntos de entrenamiento y prueba*
- *Figura 2.12: Representación gráfica de los conjuntos de entrenamiento y prueba para meteorología*
- *Figura 2.13: Representación gráfica de los conjuntos de entrenamiento y prueba para meteorología*
- *Figura 2.14: Inicialización de modelo XGBRegressor*
- *Figura 2.15: Dataframe de energía simple*
- *Figura 2.16: Dataframe de energía completo*
- *Figura 2.17: Dataframe de meteorología simple*
- *Figura 2.18: Dataframe de meteorología completo*
- *Figura 2.19: Código de factorización de valores de tipo texto*
- *Figura 2.20: Función de añadido de lags en dataframe de energía*
- *Figura 2.21: Función de añadido de lags en dataframe de meteorología*
- *Figura 2.22: Dataframe de meteorología con lags*
- *Figura 2.23: Features finales dataframe de meteorología*
- *Figura 2.24: Código de validación cruzada de conjuntos*

- *Figura 2.25: Contraste entre valores reales y predicciones obtenidas en energía sin mejoras*
- *Figura 2.26: Peso de características de XGBRegressor aplicado a energía sin mejoras*
- *Figura 2.27: Contraste entre valores reales y predicciones obtenidas en meteorología con feature adding*
- *Figura 2.28: Peso de características de XGBRegressor aplicado a meteorología sin mejoras*
- *Figura 2.29: Contraste entre valores reales y predicciones obtenidas en meteorología con feature adding*
- *Figura 2.30: Peso de características de XGBRegressor aplicado a energía con feature adding*
- *Figura 2.31: Peso de características de XGBRegressor aplicado a meteorología con feature adding*
- *Figura 2.32: Contraste entre valores reales y predicciones obtenidas en meteorología con feature adding*
- *Figura 2.33: Contraste entre valores reales y predicciones obtenidas en energía con lag adding*
- *Figura 2.34: Peso de características de XGBRegressor aplicado a energía con lag adding*
- *Figura 2.35: Peso de características de XGBRegressor aplicado a meteorología con lag adding*

Índice de tablas

- *Cuadro 2.1: Métricas XGBRegressor sin mejoras sobre dataset de energía*
- *Cuadro 2.2: Métricas XGBRegressor sin mejoras sobre dataset de meteorología*
- *Cuadro 2.3: Métricas XGBRegressor con feature adding sobre dataset de energía*
- *Cuadro 2.4: Métricas XGBRegressor con feature adding sobre dataset de meteorología*
- *Cuadro 2.5: Métricas XGBRegressor con lag adding sobre dataset de energía*
- *Cuadro 2.6: Métricas XGBRegressor con lag adding sobre dataset de meteorología*
- *Cuadro 2.7: Métricas XGBRegressor con cross validation sobre dataset de energía*
- *Cuadro 2.8: Métricas XGBRegressor con cross validation sobre dataset de meteorología*
- *Cuadro 2.9: Presupuesto del proyecto*

Resumen

En la segunda parte, se presenta un trabajo de desarrollo sobre ciencia de datos. El objetivo es realizar predicciones sobre datos de meteorología y producción energética de España mediante el uso de modelos de Forecasting y técnicas de aumento de precisión.

1. DESARROLLO / INVESTIGACIÓN

1.1. Descripción de la memoria

El objetivo es generar 3 modelos que analicen diferentes series temporales. El análisis de series temporales consiste en localizar patrones de fluctuación de un valor de una variable en función del tiempo. Se toman intervalos temporales definidos con el objetivo de predecir los posibles valores que tendrá en un intervalo de tiempo futuro. Esta técnica de predicción también es conocida como **forecasting**.

Considerando que algunos de los campos de aplicación más relevantes que tienen este tipo de trabajos de ciencia de datos, son la meteorología, las finanzas y la medicina, se ha decidido analizar la evolución en el tiempo de datos de algunas de estas categorías.

1.2. Desarrollo / Investigación

El análisis de series temporales consiste en localizar patrones de fluctuación de un valor de una variable en función del tiempo. Se toman intervalos temporales definidos con el objetivo de predecir los posibles valores que tendrá en un intervalo de tiempo futuro. Esta técnica de predicción también es conocida como **forecasting**.

En cuanto a la relevancia del estudio de series temporales, podemos decir que es uno de los métodos más eficaces de generar predicciones en campos como la climatología, las finanzas o la medicina. En cada uno de los casos, dependiendo del conjunto de datos del que se disponga, se realizará empleará una estrategia diferente para entrenar los modelos. En todos se utilizarán 4 modelos de entrenamiento.

Finanzas:

Análisis del valor del precio de la energía eléctrica en función del tiempo en España.

Podemos localizar una serie de valores asociados a los MW (mega watts) demandados y generados por épocas. Este dataset se ubica dentro de la categoría de **finanzas**, ya que contiene datos sobre producción, demanda y precios de la energía en España.

Meteorología:

Una de las predicciones a largo plazo más difíciles de lograr están siendo las **meteorológicas**. El dataset que se analiza consta de un gran número de características del tiempo en distintos horarios del día durante cuatro años.

En ambos casos, se utilizarán técnicas de preprocesamiento, añadido de características, lags temporales y validación cruzada.

Modelo a utilizar

En el presente estudio, se realizará una predicción de valores donde se conoce una parte del contexto futuro que se intenta predecir. Es decir, se tendrán datos cuantitativos de factores que ocurrirán en el futuro asociados a la variable que se busca predecir. Esto nos permite utilizar un modelo iterativo que emplea el conocimiento sobre este contexto para generar una decisión fiable.

Mientras tanto, otros modelos comúnmente utilizados en forecasting cuando se dispone de poca o ninguna información sobre el contexto, tales como ARIMA o SARIMA nos conducirán a peores resultados. Estos modelos emplean aproximaciones estadísticas que no asignan la misma importancia a los datos sobre el entorno que se predice. Finalmente, **XGBRegressor** es uno de los modelos más adecuado para forecasting bajo las condiciones mencionadas.

XGBRegressor:

Los bloques fundamentales de este algoritmo son los árboles de decisión. Cada árbol se construye seleccionando un subconjunto aleatorio de características y otro subconjunto aleatorio de datos de entrenamiento. Esto genera un conjunto de entrenamiento rico en variedad.

- **Creación de árboles de decisión:** Se crean B árboles de decisión, cada uno entrenado con un subconjunto aleatorio de datos.
- **Selección de características:** En cada nodo de cada árbol, se selecciona aleatoriamente un subconjunto de características, por ejemplo una temporal (año, mes, día de la semana) para determinar el mejor punto de división.

Dataset meteorología

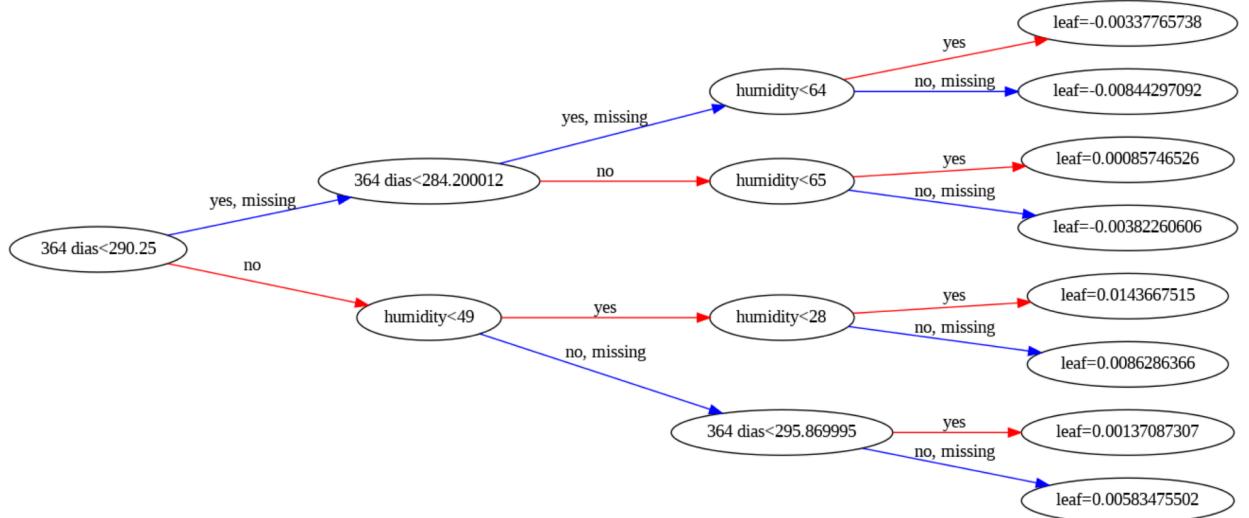


Figura 2.1: Elaboración propia

En este árbol, se pueden observar nodos que toman decisiones en base a características del conjunto inicial como `humidity` y `364 días`. De cada nodo salen 2 flechas. Por un lado, la flecha izquierda (si) determina si la condición analizada en el nodo se cumple. Si la condición se cumple, el siguiente nodo puede seguir ajustando el número asociado a esa condición o puede insertar otra característica nueva. Si la condición no se cumple, el siguiente nodo evalúa una característica diferente.

En este caso el nodo evalúa si el valor de la columna `364 días`, que equivale al valor que se busca predecir menos un año, y `humidity` (humedad) son mayores o menores a un número. Finalmente, en las hojas de la derecha vemos las decisiones finales, es decir, los valores predecidos generados por cada nodo. Estos últimos, determinarán cuál será el árbol

Dataset energía

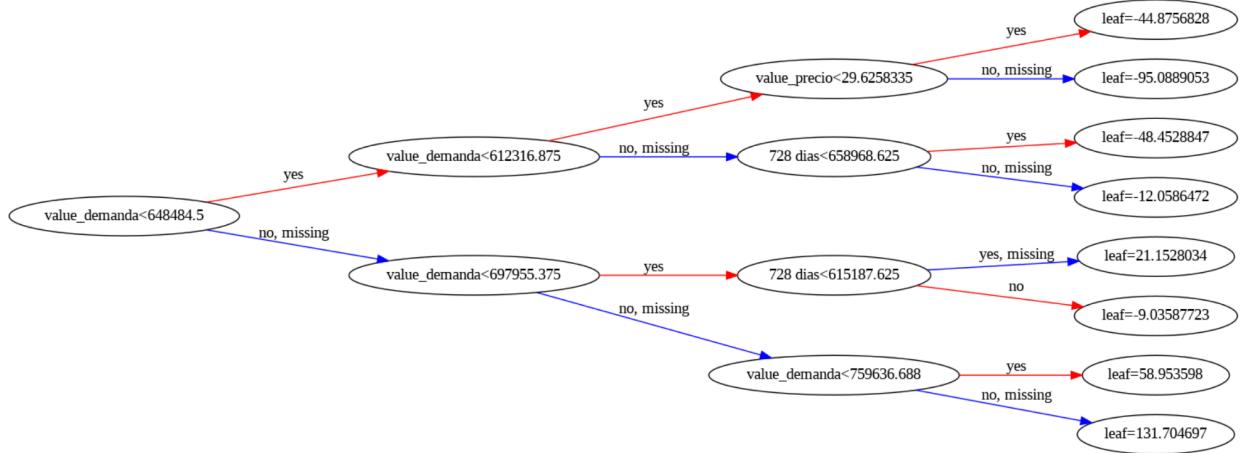


Figura 2.2: Elaboración propia

En este árbol, se pueden observar nodos que toman decisiones en base a características del conjunto inicial como `value_demand` (valor de la demanda de energía) y 728 días.

- **Corrección de errores:** Cada vez que se construye un nuevo árbol, este intenta corregir los errores de los árboles anteriores. Esta corrección se realiza mediante un proceso llamado **Gradient Boosting**. Esto, se traduce en que las características ganan o pierden peso dependiendo de si el árbol que las utiliza consigue un mayor o menor gradiente.
- **Predicción:** Para una nueva entrada x , cada árbol T_b emite una predicción $h_b(x)$.
- **Decisión final:** El valor final se decide por un proceso de agregación, como el promedio ponderado de las predicciones de todos los árboles.

$$H(x) = \text{modo}\{h_1(x), h_2(x), \dots, h_B(x)\}$$

Esto se lee como $H(x)$, la predicción, es igual al promedio entre las predicciones de cada árbol.

1.3. Metodología

El desarrollo de modelos de forecasting consta las secciones a continuación. Los casos en los que el procedimiento y las variables utilizadas coincide para los 2 datasets que se analizan, no son diferenciadas mediante un subtítulo.

Preprocesamiento y visualización de dataset:

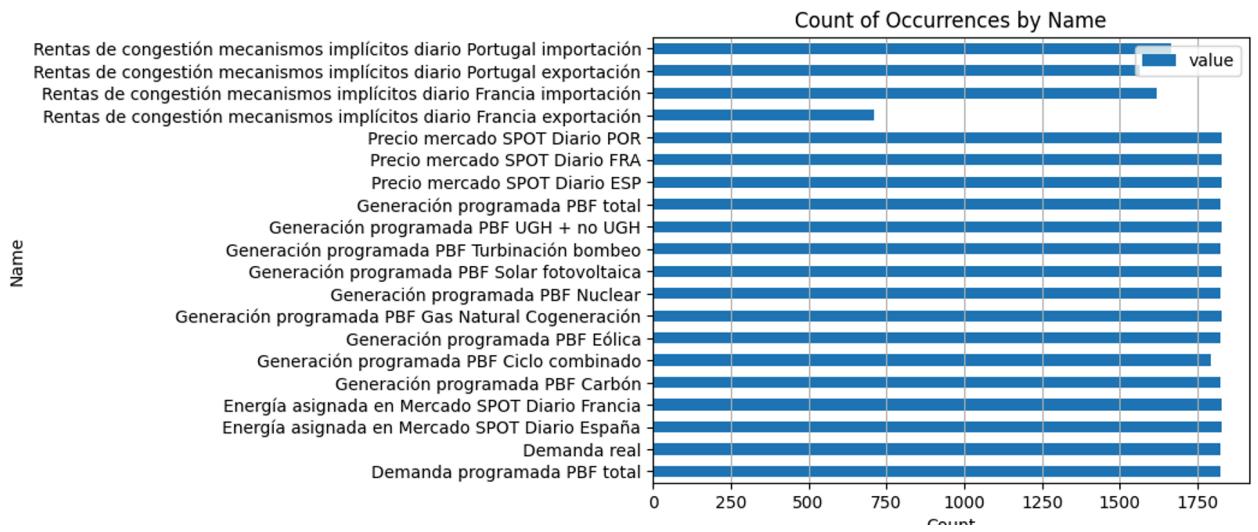
Dataset de energía:

El primer dataset dispone de 5 años de **información diaria** de diferentes aspectos relacionados con la energía. La siguiente lista muestra los diferentes tipos de registros y sus cantidades:

name	value
Demandada programada PBF total	1825
Demandada real	1825
Energía asignada en Mercado SPOT Diario España	1826
Energía asignada en Mercado SPOT Diario Francia	1826
Generación programada PBF Carbón	1823
Generación programada PBF Ciclo combinado	1793
Generación programada PBF Eólica	1825
Generación programada PBF Gas Natural Cogeneración	1826
Generación programada PBF Nuclear	1825
Generación programada PBF Solar fotovoltaica	1826
Generación programada PBF Turbinación bombeo	1824
Generación programada PBF UGH + no UGH	1826
Generación programada PBF total	1825
Precio mercado SPOT Diario ESP	1826
Precio mercado SPOT Diario FRA	1826
Precio mercado SPOT Diario POR	1826
Rentas de congestión mecanismos implícitos diari...	712
Rentas de congestión mecanismos implícitos diari...	1619
Rentas de congestión mecanismos implícitos diari...	1566
Rentas de congestión mecanismos implícitos diari...	1664
suma de filas entre todos los tipos:	34734

Figura 2.3: Elaboración propia

Se puede observar el número de ocurrencias que tiene cada tipo de dato, es decir, agrupadas por nombre:



La mayoría de los tipos de datos, contienen una fila para cada día que existe en el dataset, $365 \times 5 = 1825$.

Para el presente estudio, utilizaremos diferentes datos relacionados con la oferta y demanda de energía para afinar el modelo. Se buscan predecir los valores de **generación total de energía**.

	datetime	id		name	geoid	geoname	value
	datetime						
2014-01-01 23:00:00	2014-01-01 23:00:00	10258	Generación programada PBF total	NaN	NaN	NaN	642771.8
2014-01-02 23:00:00	2014-01-02 23:00:00	10258	Generación programada PBF total	NaN	NaN	NaN	658078.5
2014-01-03 23:00:00	2014-01-03 23:00:00	10258	Generación programada PBF total	NaN	NaN	NaN	680564.6
2014-01-04 23:00:00	2014-01-04 23:00:00	10258	Generación programada PBF total	NaN	NaN	NaN	644494.7
2014-01-05 23:00:00	2014-01-05 23:00:00	10258	Generación programada PBF total	NaN	NaN	NaN	598661.4
...
2018-12-26 23:00:00	2018-12-26 23:00:00	10258	Generación programada PBF total	NaN	NaN	NaN	611465.3
2018-12-27 23:00:00	2018-12-27 23:00:00	10258	Generación programada PBF total	NaN	NaN	NaN	589771.5
2018-12-28 23:00:00	2018-12-28 23:00:00	10258	Generación programada PBF total	NaN	NaN	NaN	543353.5
2018-12-29 23:00:00	2018-12-29 23:00:00	10258	Generación programada PBF total	NaN	NaN	NaN	510049.1
2018-12-30 23:00:00	2018-12-30 23:00:00	10258	Generación programada PBF total	NaN	NaN	NaN	526930.6

Figura 2.4: Elaboración propia

Existen algunas features que no aportan ningún dato, así que procedemos a eliminarlas:

```

▶ print('unique geoids: ', df_generacion_total['geoid'].unique())
print('unique ids: ', df_generacion_total['id'].unique())
print('unique geonames: ', df_generacion_total['geoname'].unique())

```

```

→ unique geoids: [nan]
unique ids: [10258]
unique geonames: [nan]

```

Figura 2.5: Elaboración propia

Dataset de meteorología:

El segundo dataset dispone de 4 años de **información por hora** de diferentes aspectos relacionados con la meteorología de 5 ciudades de España (Valencia, Madrid, Bilbao y Sevilla). En el presente estudio, el foco de atención será la variable temperatura.

dt_iso	city_name	temp	temp_min	temp_max	pressure	humidity	wind_speed	wind_deg	rain_1h	rain_3h	snow_3h	clouds_all	weather_id	weather_main	
2014-12-31 23:00:00+00:00	2014-12-31 23:00:00+00:00	Valencia	270.475	270.475	270.475	1001	77	1	62	0.0	0.0	0.0	0	800	clear
2015-01-01 00:00:00+00:00	2015-01-01 00:00:00+00:00	Valencia	270.475	270.475	270.475	1001	77	1	62	0.0	0.0	0.0	0	800	clear
2015-01-01 01:00:00+00:00	2015-01-01 01:00:00+00:00	Valencia	269.686	269.686	269.686	1002	78	0	23	0.0	0.0	0.0	0	800	clear
2015-01-01 02:00:00+00:00	2015-01-01 02:00:00+00:00	Valencia	269.686	269.686	269.686	1002	78	0	23	0.0	0.0	0.0	0	800	clear
2015-01-01 03:00:00+00:00	2015-01-01 03:00:00+00:00	Valencia	269.686	269.686	269.686	1002	78	0	23	0.0	0.0	0.0	0	800	clear

Figura 2.6: Elaboración propia

A continuación, se presenta un gráfico que muestra la temperatura por cada ciudad:

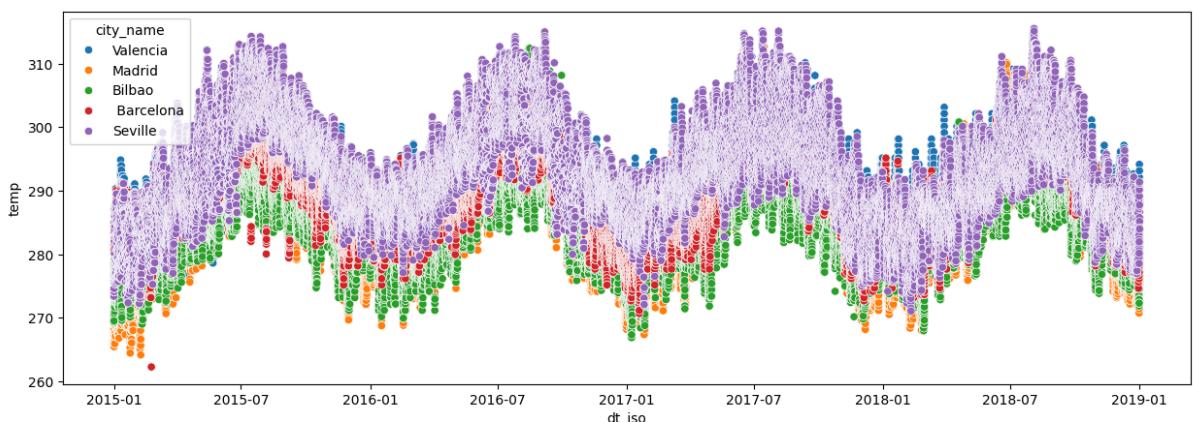


Figura 2.7: Elaboración propia

Factorización de fechas

Para que los modelos interpreten las fechas (en principio en formato datetime), es necesario convertirlas a números. Además, agregar características como mes, DíaDesemana, DíaDeAño, etc., establece diferentes referencias temporales que nuestro modelo puede comparar para hacer predicciones. Siguiendo este ejemplo, el modelo puede detectar que aquellos valores donde la columna mes es 1, es decir, enero, tienen una alta probabilidad de contener un valor elevado. Por lo tanto, si obtiene mejores resultados siguiendo esa característica para ciertos valores, asignará una mayor importancia a ella. De esta manera, también podemos medir la importancia de cada característica.

Para generar features alusivas a la fecha utilizamos la siguiente función:

```
def make_features(df):
    df['Ano'] = df.index.year
    df['Mes'] = df.index.month
    df['Cuarto'] = df.index.quarter
    df['DiaDeSemana'] = df.index.day_of_week
    df['DiaDeAño'] = df.index.day_of_year
    return df

FEATURES = ['Ano', 'Mes', 'Cuarto', 'DiaDeSemana',
           'DiaDeAño']
```

Figura 2.8: Elaboración propia

Para un dataset con valores por hora, como el de meteorología, se agrega el atributo hora y se extrae de index.hour.

Se agregan las características temporales a los conjuntos de prueba y entrenamiento:

```
train = make_features(train)
test = make_features(test)

X_train = train[FEATURES]
y_train = train['temp']

X_test = test[FEATURES]
y_test = test['temp']
```

Figura 2.9: Elaboración propia

Se definen los valores de X con las características de la fecha y las de Y con la variable a predecir, en este caso, la temperatura.

Generación de conjuntos de entrenamiento y prueba:

En los análisis de series temporales, se divide el conjunto total en 2 partes sucesivas en el tiempo y continuas, es decir, se evita componer uno de los conjuntos de segmentos temporales que no se suceden.

Dataset de meteorología:

```
| df_def = df[['temp', 'pressure', 'humidity', 'wind_speed', 'wind_deg', 'weather_main']]  
| df_def
```

		temp	pressure	humidity	wind_speed	wind_deg	weather_main
	dt_iso						
1	2014-12-31 23:00:00+00:00	270.475	1001	77	1	62	clear
2	2015-01-01 00:00:00+00:00	270.475	1001	77	1	62	clear
3	2015-01-01 01:00:00+00:00	269.686	1002	78	0	23	clear
4	2015-01-01 02:00:00+00:00	269.686	1002	78	0	23	clear
5	2015-01-01 03:00:00+00:00	269.686	1002	78	0	23	clear
...
178396	2018-12-31 18:00:00+00:00	287.760	1028	54	3	30	clear
178397	2018-12-31 19:00:00+00:00	285.760	1029	62	3	30	clear
178398	2018-12-31 20:00:00+00:00	285.150	1028	58	4	50	clear
178399	2018-12-31 21:00:00+00:00	284.150	1029	57	4	60	clear
178400	2018-12-31 22:00:00+00:00	283.970	1029	70	3	50	clear

178396 rows × 6 columns

Figura 2.10: Elaboración propia

En este caso concreto del dataset de meteorología, al disponer de datos entre el 2015 y el 2018, 4 años en total, utilizaremos el 75% para el conjunto de entrenamiento y el 25% para el de prueba.

```
] train = df_def.loc[df_def.index < '2018-01-01']  
test = df_def.loc[df_def.index >= '2018-01-01']
```

Figura 2.11: Elaboración propia

Representación visual de los conjuntos:

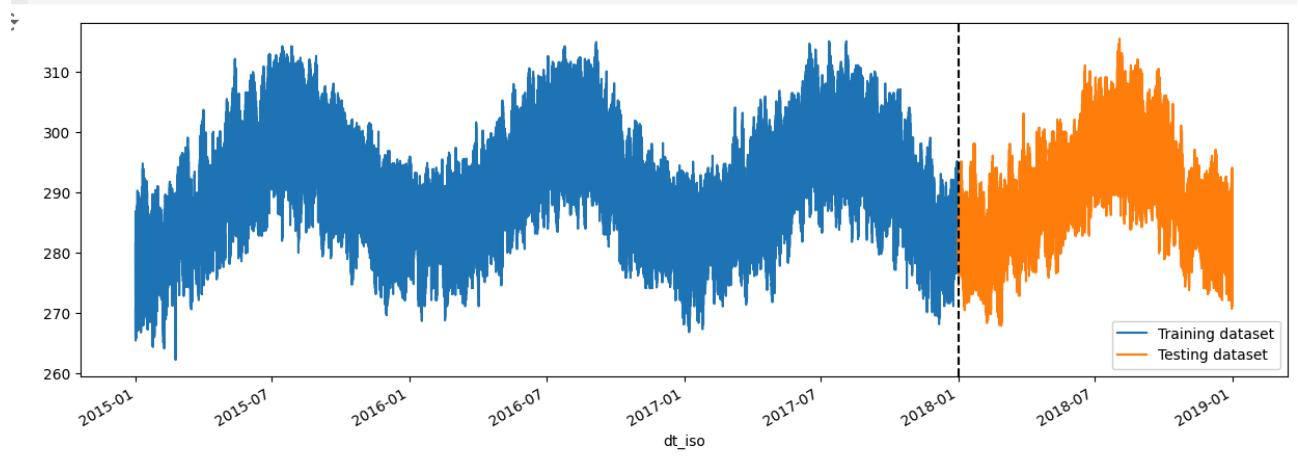


Figura 2.12: Elaboración propia

Dataset de energía:

Por otro lado, el dataset de generación total de energía eléctrica en España, también fue dividido tomando el 01-01-2018. En esta división, el conjunto de entrenamiento representa el 80% y el de prueba el 20%.

```
fig, ax = plt.subplots(figsize=(15,5))
train.plot(ax=ax, y='value', label='Training dataset')
test.plot(ax=ax, y='value',label='Testing dataset')
plt.axvline('2018-01-01', color='black', ls='--')
plt.show()
```

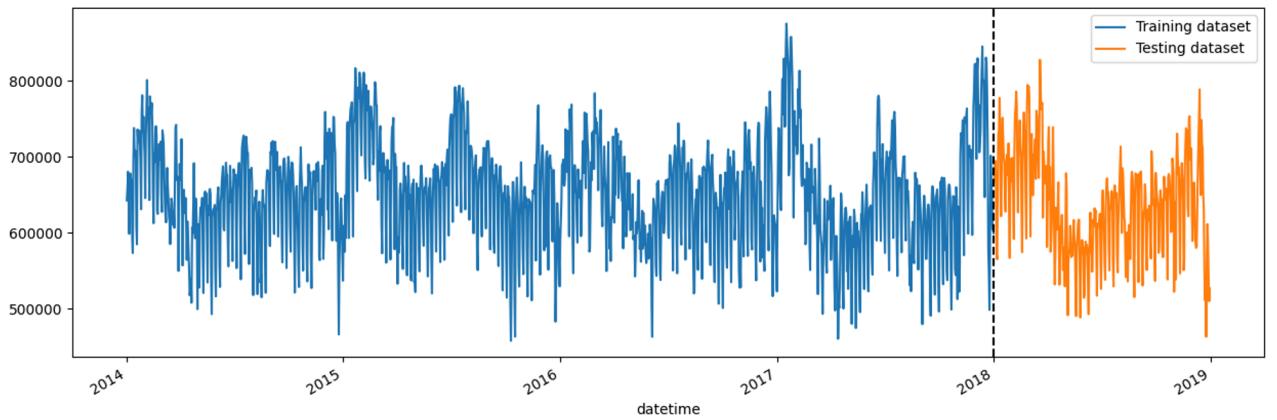


Figura 2.13: Elaboración propia

Modelos de predicción:

XGBRegressor

```
import xgboost as xgb
reg = xgb.XGBRegressor(n_estimators=2000,early_stopping_rounds=50, learning_rate=0.001)
reg.fit(X_train, y_train, eval_set=[(X_train, y_train),(X_test, y_test)], verbose=True)
```

Figura 2.14: Elaboración propia

El modelo de regresión está configurado para realizar 1000 estimaciones con una tasa de aprendizaje del 0.001. La tasa de aprendizaje es la cantidad de información que cada árbol puede incorporar al modelo por iteración. Mantener esa tasa baja, le permite al modelo no sobresaturarse rápidamente y completar el número de iteraciones que le solicitamos. Además, teniendo early_stopping_rounds = 50, el modelo se detendrá si después de 50 rondas su precisión no mejora.

RMSE significa "Root Mean Squared Error" (Raíz del Error Cuadrático Medio) y es una métrica comúnmente utilizada para evaluar la precisión de un modelo de regresión, comparando las predicciones del modelo con los valores reales del conjunto de datos de prueba

Técnicas para aumentar la precisión de los modelos:

1) Adición de Características

Agregar características a un modelo de pronóstico mejora su capacidad para analizar múltiples atributos e identificar relaciones entre los valores. En un conjunto de datos con solo una característica por fila, la capacidad del modelo para construir patrones es limitada. Sin embargo, al incorporar múltiples características, aumentamos el potencial de descubrir relaciones más complejas dentro de los datos.

Cuando los valores futuros que nuestro modelo va a predecir contienen el mismo conjunto de características que los datos de entrenamiento, podemos usar estas características para conectar mejor los eventos y determinar el resultado óptimo para la variable objetivo.

Dataset energía

Inicialmente, el modelo se puede entrenar únicamente con los campos fecha y valor (generación total de energía).

datetime	name	value
datetime		
2014-01-01 23:00:00	2014-01-01 23:00:00	Generación programada PBF total 642771.8
2014-01-02 23:00:00	2014-01-02 23:00:00	Generación programada PBF total 658078.5
2014-01-03 23:00:00	2014-01-03 23:00:00	Generación programada PBF total 680564.6
2014-01-04 23:00:00	2014-01-04 23:00:00	Generación programada PBF total 644494.7
2014-01-05 23:00:00	2014-01-05 23:00:00	Generación programada PBF total 598661.4
...
2018-12-26 23:00:00	2018-12-26 23:00:00	Generación programada PBF total 611465.3
2018-12-27 23:00:00	2018-12-27 23:00:00	Generación programada PBF total 589771.5
2018-12-28 23:00:00	2018-12-28 23:00:00	Generación programada PBF total 543353.5
2018-12-29 23:00:00	2018-12-29 23:00:00	Generación programada PBF total 510049.1
2018-12-30 23:00:00	2018-12-30 23:00:00	Generación programada PBF total 526930.6

Figura 2.15: Elaboración propia

Más tarde, dado que el dataset dispone de filas que pueden tener relación con la generación de energía como las relativas a la oferta y la demanda, añadimos estas características a los conjuntos de entrenamiento y prueba. Para esto, es necesario extraer estos datos y concatenarlos a las filas utilizando el índice como dato en común.

datetime		name	value	value_demanda
datetime				
2014-01-01 23:00:00	2014-01-01 23:00:00	Generación programada PBF total	642771.8	620107.7
2014-01-02 23:00:00	2014-01-02 23:00:00	Generación programada PBF total	658078.5	659865.2
2014-01-03 23:00:00	2014-01-03 23:00:00	Generación programada PBF total	680564.6	632536.8
2014-01-04 23:00:00	2014-01-04 23:00:00	Generación programada PBF total	644494.7	610251.7
2014-01-05 23:00:00	2014-01-05 23:00:00	Generación programada PBF total	598661.4	572534.1
...
2018-12-26 23:00:00	2018-12-26 23:00:00	Generación programada PBF total	611465.3	649103.9
2018-12-27 23:00:00	2018-12-27 23:00:00	Generación programada PBF total	589771.5	639571.2
2018-12-28 23:00:00	2018-12-28 23:00:00	Generación programada PBF total	543353.5	602541.7
2018-12-29 23:00:00	2018-12-29 23:00:00	Generación programada PBF total	510049.1	574254.2
2018-12-30 23:00:00	2018-12-30 23:00:00	Generación programada PBF total	526930.6	597701.2

Figura 2.16: Elaboración propia

Dataset meteorología:

Si nuestro conjunto de datos solo contara con 2 variables, la temporal y la del valor a predecir, podríamos empezar entrenando el modelo únicamente con la característica de temperatura y el índice:

<code>df_def = df[['temp']]</code>	
<code>df_def</code>	
	temp
	dt_iso
2014-12-31 23:00:00+00:00	270.475
2015-01-01 00:00:00+00:00	270.475
2015-01-01 01:00:00+00:00	269.686
2015-01-01 02:00:00+00:00	269.686
2015-01-01 03:00:00+00:00	269.686
...	...
2018-12-31 18:00:00+00:00	287.760
2018-12-31 19:00:00+00:00	285.760
2018-12-31 20:00:00+00:00	285.150
2018-12-31 21:00:00+00:00	284.150
2018-12-31 22:00:00+00:00	283.970

Figura 2.17: Elaboración propia

Luego, como disponemos de más datos podemos iterar las distintas características de las que disponemos en el dataset para transmitir al modelo los mejores datos para predecir la temperatura. Los datos elegidos fueron los siguientes:

		temp	pressure	humidity	wind_speed	wind_deg	weather_main
	dt_iso						
2014-12-31 23:00:00+00:00		270.475	1001	77	1	62	clear
2015-01-01 00:00:00+00:00		270.475	1001	77	1	62	clear
2015-01-01 01:00:00+00:00		269.686	1002	78	0	23	clear
2015-01-01 02:00:00+00:00		269.686	1002	78	0	23	clear
2015-01-01 03:00:00+00:00		269.686	1002	78	0	23	clear
...
2018-12-31 18:00:00+00:00		287.760	1028	54	3	30	clear
2018-12-31 19:00:00+00:00		285.760	1029	62	3	30	clear
2018-12-31 20:00:00+00:00		285.150	1028	58	4	50	clear
2018-12-31 21:00:00+00:00		284.150	1029	57	4	60	clear
2018-12-31 22:00:00+00:00		283.970	1029	70	3	50	clear

Figura 2.18: Elaboración propia

Dado que la característica weather_main está en formato texto, lo conviene es realizar una factorización para pasarlo a número y que no genere dificultades en el entrenamiento:

```
print("Before conversion:", df['weather_main'].unique())
df['weather_main'], _ = pd.factorize(df['weather_main'])
print("After conversion:", df['weather_main'].unique())
Before conversion: ['clear' 'clouds' 'rain' 'mist' 'thunderstorm' 'drizzle' 'fog' 'smoke'
 'haze' 'snow' 'dust' 'squall']
After conversion: [ 0  1  2  3  4  5  6  7  8  9 10 11]
```

Figura 2.19: Elaboración propia

2) Lag features (características de desfase)

Continuando con el añadido de características, podemos generar algunas basadas en lags o desfases de tiempo. Estas columnas, se crean extrayendo de la columna a evaluar

(value) los resultados un ese día menos una constante de tiempo y añadiéndolos a la nuevas columnas (features). Por ejemplo, si queremos crear un lag feature de 1 año, este consistirá en añadir a cada fila el valor de la columna a predecir de esa fecha menos 1 año. Para el 01-01-2019, en la columna de lag 1 año, tendremos el valor de la columna value del día 02-01-2018.

Los distintos lags van a referirse a la cantidad de tiempo que se le quita al índice de las filas del dataset. Nuestro Lag 1, por ejemplo, va a remontarse 1 año atrás, nuestro segundo lag, 2 años, y así con la cantidad de lags que busquemos añadir. El objetivo es agregar referencias que el modelo pueda comparar y que mejoren su aprendizaje.

Dataset de energía

```
def add_lags(df):
    df['364 días'] = df['value'].shift(364)
    df['728 días'] = df['value'].shift(728)
    df['1092 días'] = df['value'].shift(1092)
    return df
```

Figura 2.20: Elaboración propia

Para crear los lags, utilizamos la función shift que se desplaza sobre el índice, en este caso diario, el número de valores que se indiquen.

Dataset de meteorología

```
def add_lags(df):
    df['364 días'] = df['temp'].shift(364)
    df['728 días'] = df['temp'].shift(728)
    df['1092 días'] = df['temp'].shift(1092)
    return df
```

Figura 2.21: Elaboración propia

Las filas correspondientes a las primeras fechas del conjunto, no tendrán valores en las columnas de los lags, ya que no poseen valores previos, mientras que las últimas tendrán las 3 columnas completas, dado que existen más de 3 años de registros previos.

		temp	pressure	humidity	wind_speed	wind_deg	weather_main	364 dias	728 dias	1092 dias
	dt_iso									
2014-12-31 23:00:00+00:00		270.475	1001	77	1	62	clear	NaN	NaN	NaN
2015-01-01 00:00:00+00:00		270.475	1001	77	1	62	clear	NaN	NaN	NaN
2015-01-01 01:00:00+00:00		269.686	1002	78	0	23	clear	NaN	NaN	NaN
2015-01-01 02:00:00+00:00		269.686	1002	78	0	23	clear	NaN	NaN	NaN
2015-01-01 03:00:00+00:00		269.686	1002	78	0	23	clear	NaN	NaN	NaN
...
2018-12-31 18:00:00+00:00		287.760	1028	54	3	30	clear	284.66	285.94	289.54
2018-12-31 19:00:00+00:00		285.760	1029	62	3	30	clear	284.76	287.94	289.15
2018-12-31 20:00:00+00:00		285.150	1028	58	4	50	clear	284.15	289.54	288.76
2018-12-31 21:00:00+00:00		284.150	1029	57	4	60	clear	283.76	290.54	288.36
2018-12-31 22:00:00+00:00		283.970	1029	70	3	50	clear	283.76	291.15	288.58

Figura 2.22: Elaboración propia

Una vez concatenados los lags con las características añadidas en el paso anterior, tenemos estas features:

```
FEATURES = ['Ano', 'Mes', 'Cuarto', 'DiaDeSemana',
           'DiaDeAño', 'Hour', 'pressure', 'humidity', 'wind_speed', 'wind_deg', 'weather_main','364 días','728 días','1092 días']
```

Figura 2.23: Elaboración propia

Entrenamiento con validación cruzada:

Podemos entrenar modelos de forecasting con subconjuntos de los dataset. Para esto, utilizamos la estructura TimeSeriesSplit de la librería model_selection. Este objeto, permite dividir un dataframe en distintos subconjuntos de entrenamiento y de prueba. El siguiente conjunto de prueba siempre tiene que tener un intervalo de tiempo posterior al del subconjunto anterior.

La validación cruzada es una técnica utilizada para evaluar el rendimiento de los modelos de aprendizaje automático mediante diferentes subconjuntos de los datos disponibles. Este proceso implica dividir el conjunto de datos en varios intervalos (o "folds"). De este modo, es posible validar el comportamiento que mostrará el modelo, ejecutándolo sobre distintos segmentos del conjunto. Esto, nos da una pista interesante de cómo mejora el modelo para series desconocidas a medida que agregamos parámetros y características

Este método, permite obtener una estimación más precisa del rendimiento del modelo en datos no vistos, lo que mejora la robustez y generalización de los modelos (Kohavi, 1995).

Una vez aplicada esta técnica, se calculan las métricas de rendimiento (como la precisión, la sensibilidad, etc.) para obtener una estimación más fiable de cómo se comportará el modelo en la práctica. A continuación, se presenta el código que obtiene la puntuación para cada intervalo y las guarda en una lista:

```
preds = []
scores = []
for train_idx, val_idx in tss.split(df):
    train = df.iloc[train_idx]
    test = df.iloc[val_idx]

    train = make_features(train)
    test = make_features(test)

    X_train = train[FEATURES]
    y_train = train[TARGET]

    X_test = test[FEATURES]
    y_test = test[TARGET]
    reg = xgb.XGBRegressor(n_estimators=1000,
                           early_stopping_rounds=50,
                           learning_rate=0.001)

    reg.fit(X_train, y_train, eval_set=[(X_train, y_train),
                                        (X_test, y_test)], verbose=True)

    y_pred = reg.predict(X_test)

    preds.append(y_pred)

    score = np.sqrt(mean_squared_error(y_test, y_pred))

    scores.append(f'{score:.2f}')

    fold+=1
```

Figura 2.24: Elaboración propia

Este código está basado en

https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.TimeSeriesSplit

El código itera el objeto tss que tendrá tantos subconjuntos como hayamos definido en su creación, y genera un conjunto de entrenamiento y otro de prueba en cada una. En este caso, el modelo empleado en el ejemplo de la validación cruzada es XGBRegressor pero es permutable por otro modelos de forecasting.

1.4. Resultados

Modelo 1: XGBRegressor

En las tablas correspondientes al dataset de energía, los datos no porcentuales están expresados en MW (MegaWatts).

En las tablas correspondientes al dataset de meteorología, los datos no porcentuales están expresados en grados kelvin.

Dataset finanzas

Iteración 1: modelos sin añadido de características (feature adding)

Dataset energía

	XGBRegressor
MAE	45056.286
MSE	3183764575.387
RMSE	56424.858
MAPE	7.199%

Cuadro 2.1: Elaboración propia

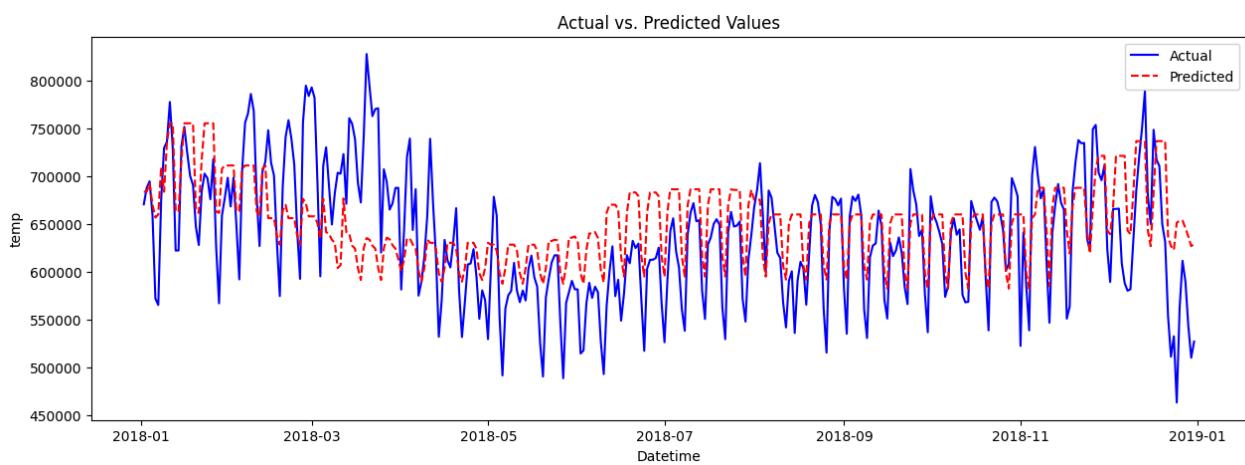


Figura 2.25: Elaboración propia

Peso de características XGBRegressor:

	Importance
DiaDeSemana	0.732190
DiaDeAño	0.180661
Año	0.056457
Mes	0.030691
Cuarto	0.000000

Figura 2.26: Elaboración propia

Dataset meteorología

	XGBRegressor
MAE	3.870
MSE	23.482
RMSE	4.846
MAPE	1.343%

Cuadro 2.2: Elaboración propia

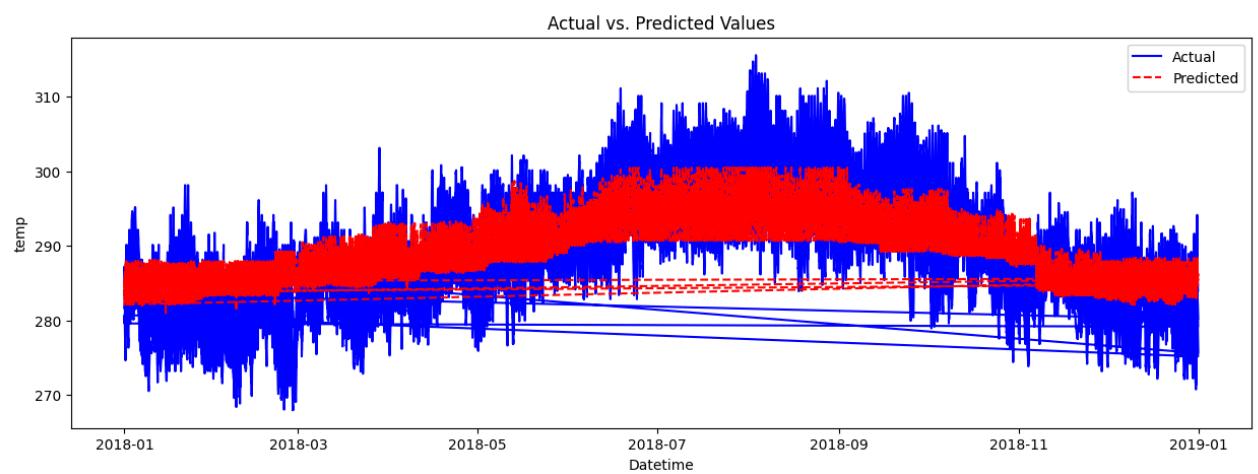


Figura 2.27: Elaboración propia

Peso de características XGBRegressor:

	Importance
DiaDeAno	0.446611
Mes	0.369406
Hour	0.160407
Ano	0.018029
DiaDeSemana	0.005548
Cuarto	0.000000

Figura 2.28: Elaboración propia

Iteración 2: modelos con añadido de características (feature adding)

Dataset energía

	Modelo 1
MAE	36784.240
MSE	2021628498.937
RMSE	44962.523
MAPE	5.920%

Cuadro 2.3: Elaboración propia

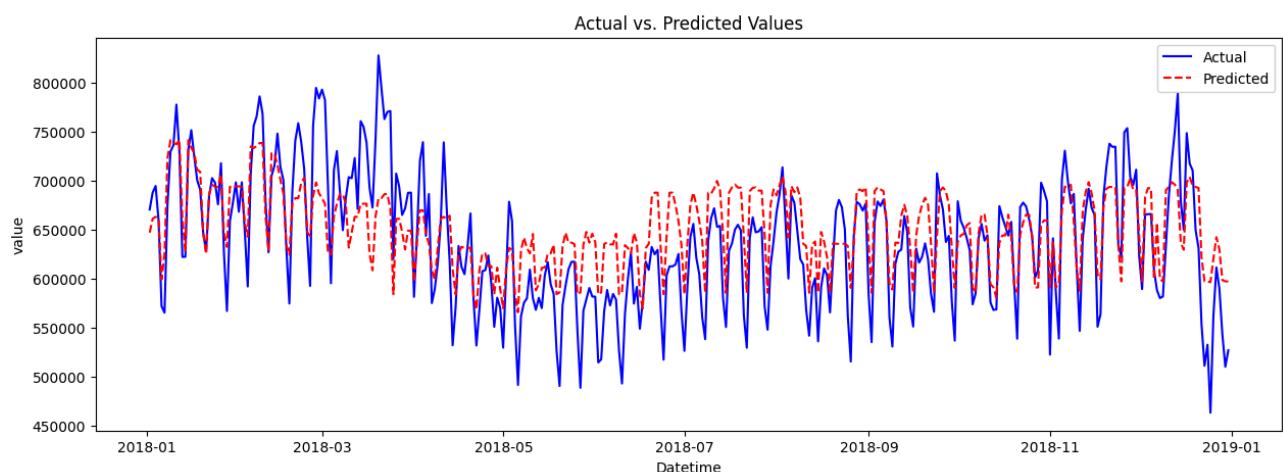


Figura 2.29: Elaboración propia

Peso de características XGBRegressor:

	Importance
value_demanda	0.778433
Ano	0.094305
value_precio	0.059149
DiaDeAño	0.040943
Mes	0.017555
DiaDeSemana	0.009615
Cuarto	0.000000

Figura 2.30: Elaboración propia

Peso de características de XGBRegressor aplicado a energía con feature adding

Dataset meteorología

	XGBRegressor
MAE	3.621
MSE	20.219
RMSE	4.497
MAPE	1.257%

Cuadro 2.4: Elaboración propia

Al agregar nuevas características, su peso en el modelo entrenado se verá afectado:

	Importance
humidity	0.470506
DiaDeAño	0.222203
Mes	0.205394
Hour	0.034203
wind_speed	0.020983
Ano	0.012905
pressure	0.011850
weather_main	0.009045
wind_deg	0.009040
DiaDeSemana	0.003872
Cuarto	0.000000

Figura 2.31: Elaboración propia

Visualización de contraste entre valores reales y predicciones obtenidas luego de añadido de características con XGBRegressor:

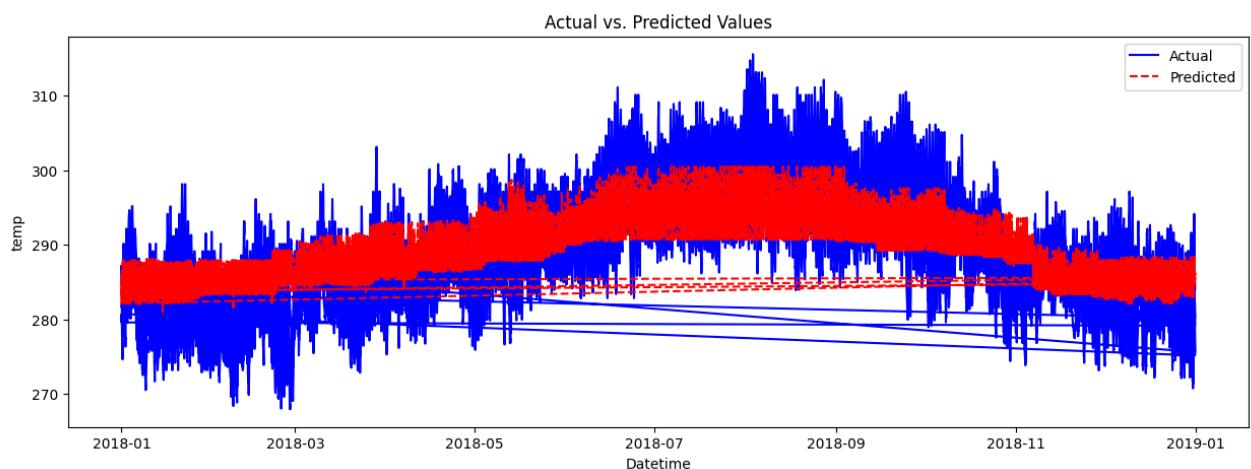


Figura 2.32: Elaboración propia

Contraste entre valores reales y predicciones obtenidas en meteorología con feature adding

Iteración 3: modelo con lags temporales

Dataset energía

	XGBRegressor
MAE	35612.124
MSE	1893386358.428
RMSE	43513.060
MAPE	5.727%

Cuadro 2.5: Elaboración propia

Visualización de contraste entre valores reales y predicciones obtenidas luego de añadido de lags con XGBRegressor:

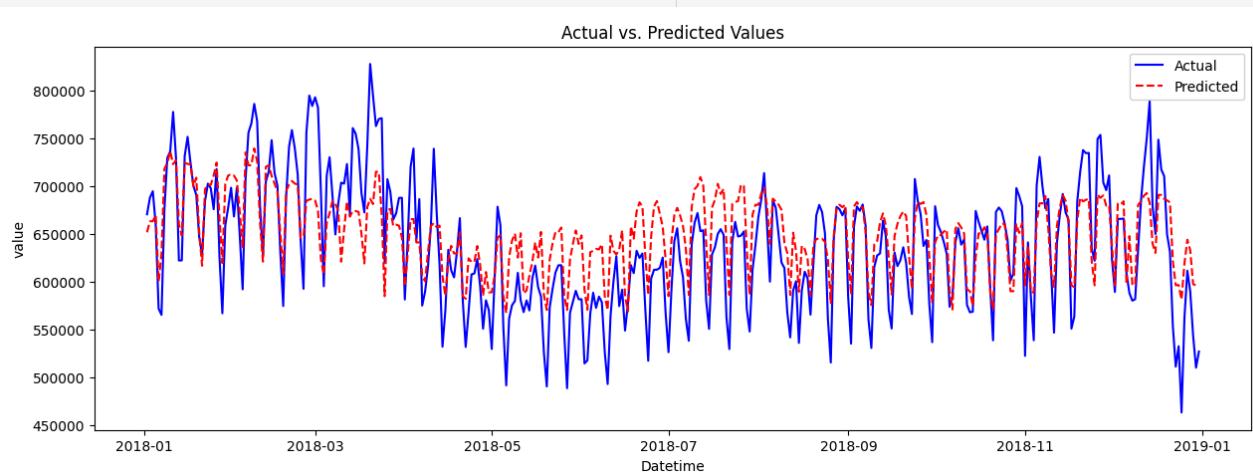


Figura 2.33: Elaboración propia

Peso de características XGBRegressor

	Importance
value_demanda	0.754424
728 dias	0.086151
value_precio	0.046580
1092 dias	0.036506
DiaDeAño	0.033588
364 dias	0.013018
DiaDeSemana	0.011933
Mes	0.009417
Año	0.008383
Cuarto	0.000000

Figura 2.34: Elaboración propia

Dataset meteorología

	XGBRegressor
MAE	3.649
MSE	20.424
RMSE	4.519
MAPE	1.266%

Cuadro 2.6: Elaboración propia

Peso de características de XGBRegressor aplicado a meteorología con lag adding

	Importance
364 dias	0.704357
humidity	0.158273
Mes	0.030363
DiaDeAño	0.028653
Hour	0.014785
weather_main	0.011828
728 días	0.011453
wind_speed	0.010131
pressure	0.009414
1092 días	0.008677
wind_deg	0.005701
Ano	0.005566
DiaDeSemana	0.000799
Cuarto	0.000000

Figura 2.35: Elaboración propia

Iteración 4: modelo con validación cruzada

Dataset energía

	XGBRegressor
MSE 1	47034.06
MSE 2	55087.52
MSE 3	46499.16
MSE 4	43317.21

Cuadro 2.7: Elaboración propia

Dataset meteorología

	XGBRegressor
MSE 1	3.54
MSE 2	4.19
MSE 3	4.64
MSE 4	5.38
MSE 5	6.55

Cuadro 2.8: Elaboración propia

1.5. Conclusión

Los modelos que integran adecuadamente el número de lags y tipos de datos locales muestran un desempeño superior en las predicciones. Estos modelos capturan mejor las tendencias y patrones intrínsecos de las series temporales, lo que resulta en una mayor precisión.

Además, los modelos que incorporan datos cuantitativos sobre el contexto futuro que se intenta predecir logran resultados significativamente mejores. Esto sugiere que la inclusión de variables contextuales relevantes mejora la capacidad del modelo para anticipar cambios y variaciones en los datos.

Sin embargo, es importante destacar que incrementar el número de características (features) del modelo es beneficioso solo hasta cierto punto. Si el número de iteraciones con el que se entrena el modelo no es suficiente para abordar todas las características añadidas, el rendimiento del modelo puede verse comprometido. Por lo tanto, es crucial encontrar un equilibrio adecuado entre la complejidad del modelo y la capacidad de entrenamiento disponible para asegurar predicciones precisas y fiables.

1.6. Lineas futuras

Aplicación en Nuevos Casos de Estudio: Ampliar el uso de las técnicas implementadas a nuevos casos de estudio, como el ámbito de la medicina. Esto permitirá explorar y validar la efectividad de los modelos en diferentes contextos y con diferentes tipos de datos, potenciando su aplicabilidad y relevancia en diversas disciplinas.

Desarrollo de una Interfaz Gráfica: Crear una interfaz gráfica de usuario (GUI) que permita interactuar con los modelos de forecasting. Esta herramienta debería permitir a los usuarios configurar parámetros y visualizar los resultados de manera intuitiva, facilitando el uso y la interpretación de los modelos incluso para aquellos con menor experiencia técnica.

Integración con Fuentes de Datos Externas: Cruzar los datos de las series temporales con fuentes externas de datos para validar los resultados obtenidos. Este enfoque puede mejorar la precisión y la robustez de las predicciones, además de proporcionar una validación externa que refuerce la confiabilidad de los modelos desarrollados.

1.7. Aplicabilidad

Análisis de Acciones en el Mercado de Valores: Los métodos implementados pueden aplicarse eficazmente en el análisis de acciones en el mercado de valores. Esto incluye la predicción de precios futuros, la identificación de tendencias y la optimización de estrategias de inversión.

Pronóstico Meteorológico: Los modelos predictivos pueden emplearse para anticipar condiciones meteorológicas en múltiples ciudades. Esto es crucial para la planificación y gestión de actividades dependientes del clima, como la agricultura, la aviación y eventos al aire libre.

Sistemas de Predicción Financiera Personalizada: Se pueden desarrollar sistemas basados en datos individuales relacionados con los movimientos financieros de una cuenta. Para individuos que realizan compras y ventas diarias de activos, estos sistemas pueden generar previsiones sobre los tipos de movimientos que resultan en mayores ganancias o pérdidas por período. Además, estos sistemas pueden identificar los momentos óptimos del año para realizar movimientos financieros, proporcionando así una herramienta valiosa para la toma de decisiones estratégicas.

2. BIBLIOGRAFÍA

- Kaggle. (2023). Spanish electricity market: Demand, generation and price. Kaggle. <https://www.kaggle.com/datasets/manualrg/spanish-electricity-market-demand-gen-price>

- Kaggle. (2023). Starter hourly energy consumption. Kaggle.
<https://www.kaggle.com/code/robikscube/starter-hourly-energy-consumption>
- Kaggle. (2023). Weather in Spain: Preprocessing timeseries. Kaggle.
https://www.kaggle.com/code/ahmedwaelz/weather-in-spain-preprocessing-timeseries/input?select=weather_features.csv
- Koseoglu, B. (2020, September 24). Guide to time series analysis with Python (2): Moving average process. Medium.
[https://buse-koseoglu13.medium.com/guide-to-time-series-analysis-with-python-2-moving-a verage-process-784328325e5f](https://buse-koseoglu13.medium.com/guide-to-time-series-analysis-with-python-2-moving-average-process-784328325e5f)
- Luna, S. (2022, August 2). Tutorial: Feature engineering for weekly time series forecasting in PySpark. Medium.
<https://medium.com/@soyoungluna/tutorial-feature-engineering-for-weekly-time-series-fore casting-in-pyspark-b207c41869f4>
- Scikit-learn. (n.d.). TimeSeriesSplit. Scikit-learn.
[https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.TimeSeriesSplit.h tml](https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.TimeSeriesSplit.html)
- Searle, J. (2022, July 14). Analyzing the impact of lagged features in time series forecasting: A linear regression approach. Medium.
[https://medium.com/@evertongomede/analyzing-the-impact-of-lagged-features-in-time-seri es-forecasting-a-linear-regression-approach-730aaa99dfd6](https://medium.com/@evertongomede/analyzing-the-impact-of-lagged-features-in-time-serie s-forecasting-a-linear-regression-approach-730aaa99dfd6)
- Tableau. (2023). Time series analysis. Tableau.
<https://www.tableau.com/learn/articles/time-series-analysis>
- Universidad Complutense de Madrid. (n.d.). Series temporales.
https://www.ucm.es/data/cont/media/www/pag_41459/Series%20temporales.pdf
- Upadhyay, V. (2023, March 10). How Lag-Llama transforms time series forecasting with a pretrained transformer model. Medium.
<https://vivekupadhyay1.medium.com/how-lag-llama-transforms-time-series-forecasting-with -a-pretrained-transformer-model-28089fe6a814>
- Visualizing cross-validation behavior in scikit-learn. (2020). Scikit-learn.
https://scikit-learn.org/stable/auto_examples/model_selection/plot_cv_indices.html

- dotData. (2023). Practical guide for feature engineering of time series data. dotData.
<https://dotdata.com/blog/practical-guide-for-feature-engineering-of-time-series-data/>

3. ANEXOS

- **Anexo 1: ARIMA for Time Series Forecasting with Python**

Brownlee, J. (2017, March 22). ARIMA for Time Series Forecasting with Python. Machine Learning Mastery.
<https://machinelearningmastery.com/arima-for-time-series-forecasting-with-python/>

- **Anexo 2: How to set p, d, q and P, D, Q for SARIMA Time Series Model**

Stats Stack Exchange. (2019). How to set p, d, q and P, D, Q for SARIMA Time Series Model.

<https://stats.stackexchange.com/questions/445014/how-to-set-p-d-q-and-p-d-q-for-sarima-time-series-model>

- **Anexo 3: Series Temporales**

Universidad Complutense de Madrid. (n.d.). Series temporales.

<https://www.ucm.es/data/cont/media/www/pag-41459/Series%20temporales.pdf>

- **Anexo 4: Colab Notebook - Time Series Forecasting 1**

Google Colab. (2023). Time Series Forecasting 1.

<https://colab.research.google.com/drive/1uSTsaxfqQ98iqCWPh1mEN8AV6nzyPkSH?usp=sharing>

- **Anexo 5: Colab Notebook - Time Series Forecasting 2**

Google Colab. (2023). Time Series Forecasting 2.

https://colab.research.google.com/drive/1kZBdkBnkJh-_Vnij-kYsRwyVD5YwzpGe?usp=sharing