Deep Price Forecast:

Predicción de precio eléctrico usando Redes Recurrentes

Clara Godoy Morales

Máster en Deep Learning e Inteligencia Artificial

(2019-2020)

1 Introducción

El trabajo que se describe a continuación es el Trabajo de Fin de Máster del Máster en Deep Learning e Inteligencia Artificial de Kschool. En él se realiza una predicción del precio eléctrico de España con Redes Recurrentes.

Las predicciones de precio, se usan de manera directa o indirecta en la gran mayoría de sectores del país. Para una toma de decisiones óptima en muchos aspectos, la energía es clave. Pero la predicción del precio eléctrico es sobre todo útil para Generadoras de energía, Gestoras energéticas, traders en general y Comercializadoras. Las variaciones del precio eléctrico real respecto al esperado marcan diferencias de millones de euros cada día para todos ellos.

Dadas las características que tiene el mercado eléctrico en nuestro podemos hablar de tres tipos de predicciones:

- **Predicción a corto plazo**: se caracteriza por tener horizontes de 1 a 7 días, con previsiones horarias y diarias son usadas en general por comercializadoras y generadores de electricidad y van a mercado diario.
- **Predicciones a medio plazo**: caracterizadas por tener un horizonte de más o menos 12 días, se usan para tomar decisiones operativas y de gestión de riesgo, son usadas por gestoras y comercializadoras para optimizar carteras.
- **Predicciones a largo plazo**: estas tienen de 5 a 20 años, son usadas para toma de decisiones de inversión. Obviamente dependen de cómo progresa la regulación, la actividad económica y los mercados internacionales. Normalmente es usada como input de otras variables como valoración de inversiones y medición de riesgo de mercado.

En este trabajo vamos a tratar las predicciones a corto plazo, ya que se realizan predicciones de 12 h.

La predicción de precio eléctrico está basada en el uso de series temporales como predictores, el análisis de series temporales suele hacerse a través de métodos estadísticos, pero estos tienen una serie de limitaciones:

- Necesitan series de datos completas para entrenar, si se pierde algún valor se pueden generar malos resultados en el modelo, A pesar de que hay formas de solucionarlo, son complejas
- Normalmente se trabaja con series de datos univariables y es bastante complejo trabajar con series multivariables.
- Es extremadamente sensible a valores perdidos

Los métodos de deep learning son capaces de hacer frente a los desafíos mencionados:

- No son sensibles a la falta de valor

- Facilidad para incorporar variables exógenas (se aplica tanto al conjunto de datos univariados como al conjunto de datos multivariados)
- Captura las interacciones de características no lineales
- Extracción automática de características

En este trabajo se explican tanto el proceso de obtención de datos y su posterior procesamiento para poder dar los datos de entrada correctamente a los modelos, como el proceso de estudio de diferentes modelos y métodos para llegar al modelo que mejor se ajusta, finalizando con cómo reproducir lo realizado y cómo ejecutar la APP de predicción.

2 Descripción de los datos

Cuando se fija el precio del mercado diario, algunos de los factores que más afectan son:

- **La demanda de electricidad:** El consumo total de energía para todos los fines. Cuanto mayor sea la demanda de electricidad, mayor será el Precio de Mercado
- **Generación de energía renovable**: La energía renovable tiene prioridad en la red eléctrica y por lo tanto siempre está en primer lugar en el orden de mérito. Cuanto mayor sea la generación de energía renovable, menor será el Precio de Mercado
- **Fecha y hora**: El Precio de Mercado se incrementará en verano e invierno, además los valores máximos suelen tener lugar a primera hora de la mañana y a última hora de la tarde.

Una vez conocidos los datos condicionantes, estos se han extraído de REE (Red Eléctrica Española), un grupo empresarial multinacional de origen español que opera en el mercado eléctrico internacional como operador del sistema eléctrico, los cuales proporcionan un servicio API REST a ESIOS (Sistema de Información del Operador del Sistema Español), mediante el cual se puede descargar toda la información del sistema. La documentación oficial del API REST de ESIOS está disponible aquí.

Para acceder al sistema, es necesario solicitar un token por correo electrónico. La información sobre cómo solicitar el token está disponible aquí.

Para obtener la información se ha desarrollado el fichero `esios.py`. En él se puede utilizar la función `save_indicators_table` para comprobar la lista de todos los indicadores. Los indicadores usados en este caso son:

Nombre del indicador	Identificador	Alias
Previsión diaria de la demanda eléctrica peninsular	460	demand
Previsión de la producción eólica nacional peninsular	541	wind

Generación prevista Solar	10034	solar
Precio medio horario componente mercado diario	805	price

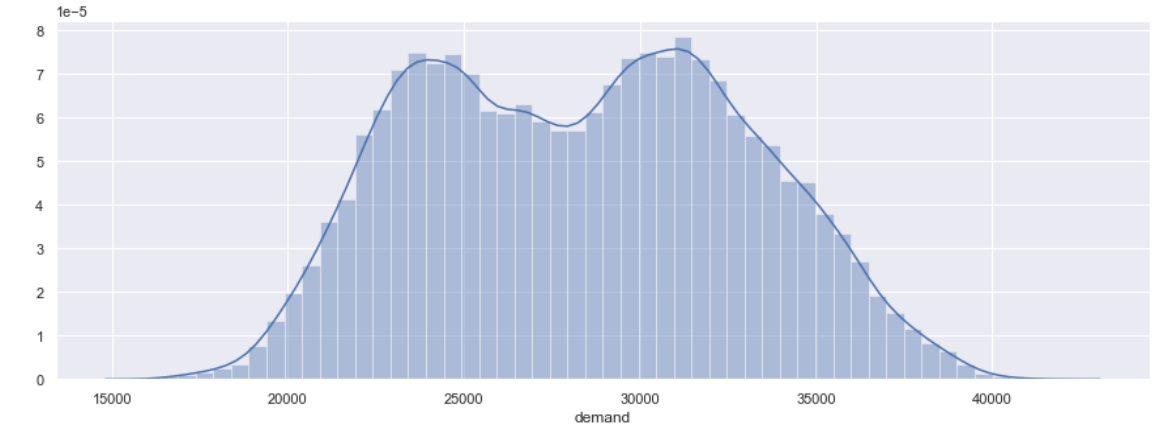
Tabla 1 indicadores API ESIOS

Los datos disponibles son del `01-01-2014` hasta el presente. Los datos históricos utilizados son del `01-01-2014` al `25-08-2020`

De esta manera, la información puede ser descargada a través del servicio REST API. Dado que el Precio del Mercado Eléctrico es un valor que se actualiza cada hora, para desarrollar el modelo es necesario que el resto de los parámetros mantengan la misma escala, por lo que el script está configurado para realizar la descarga de datos de los indicadores anteriormente mencionados con frecuencia horaria.

Para obtener la información de ESIOS es necesario ejecutar el archivo `Get_Data.ipynb` de Jupyter Notebook. Una vez ejecutado el archivo se generará un archivo `Data.csv`.

En ese archivo se realiza un primer procesado de los datos, quitando duplicados y NaNs y haciendo un estudio de outliers. En ese estudio se han revisado variable a variable los outliers que pudieran tener y si tenían decidir si eran suficientemente significativos como para tener que tratarlos o no:



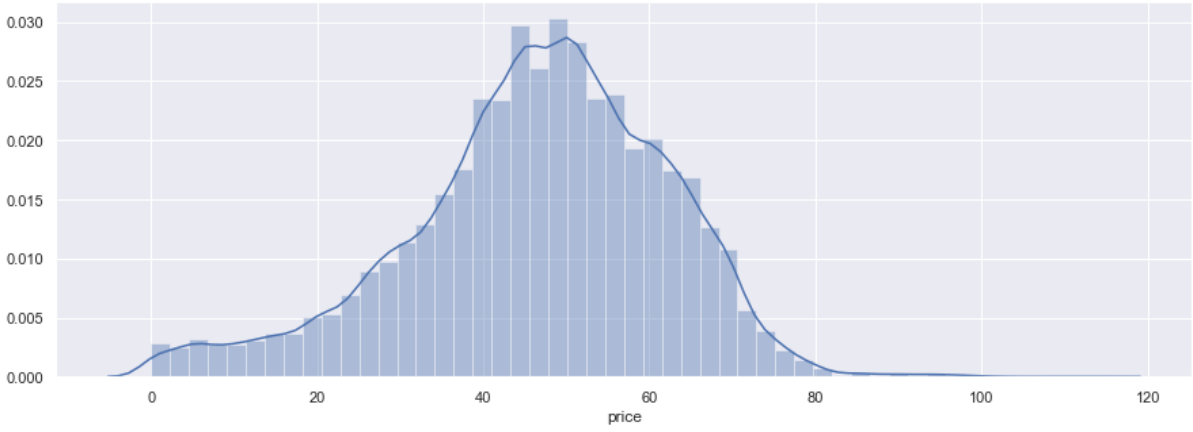


Imagen 1 Outliers demanda y precio

El estudio se puede resumir en:

Variable	demand	solar	wind	price
Outliers	0	403	506	1609
%	0	0.70%	0.88%	2.81%

Tabla 2 Porcentaje de outliers respecto total de datos

Al ser valores en tanto por ciento, respecto el total de datos, tan pequeños no se ha considerado necesario hacer nada con ellos y se usa el dataset tal cual.

Se genera un archivo con toda la información recuperada de la API, este tiene la estructura siguiente:

Date	Demand	Wind	Solar	Price
TimeStamp	Value	Value	Value	Value

Tabla 3 Aspecto de fichero final del dataset

3 Metodología

Para esta parte se ha realizado un estudio con diferentes configuraciones y métodos. En ambas usamos ventanas de tiempo para entrenar y predecir.

Dado un tiempo específico, digamos que quieres predecir el precio 6 horas en el futuro. Para hacer esta predicción, eliges usar 5 días de observaciones. Así, crearías una ventana con las últimas 120(5x24) observaciones para entrenar el modelo. Muchas de estas configuraciones son posibles, lo que hace que este conjunto de datos sea bueno para experimentar.

Se desarrolla una función para cada metodología, esta devuelve las ventanas de tiempo descritas anteriormente para que el modelo se entrene. El parámetro `history_size` es el tamaño de la ventana de información pasada. El `target_size` es cuán lejos en el futuro necesita el modelo aprender a predecir. Este `target_size` se aplica al conjunto de datos para obtener la etiqueta que necesita ser predicha.

Por supuesto antes de esto se ha cargado el fichero `Data.csv` con todo el histórico y se ha normalizado. A continuación se describen las distintas metodologías con diferentes configuraciones:

3.1 Series temporales univariables

Primero probamos a entrenar un modelo usando una sola variable, la que queremos predecir, el precio y el modelo lo usará para hacer predicciones de ese valor en el futuro.

3.1.1 Baseline

Antes de construir un modelo entrenable, sería bueno tener un baseline de rendimiento como punto de comparación con los últimos modelos más complicados.

Dado un punto de entrada, el método de línea de base mira toda la historia y predice que el siguiente punto será el promedio de las últimas 20 observaciones. Esta simple línea de base se conoce como la media móvil (MA).

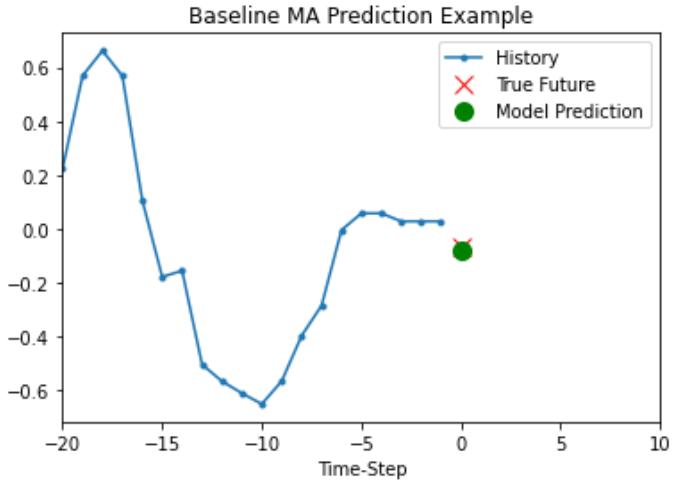


Imagen 2 Predicción con MA

Para poder comparar también se ha creado el WMA (Media Móvil Ponderada), una versión elegante del MA, en la que los últimos datos tienen más peso que los primeros.

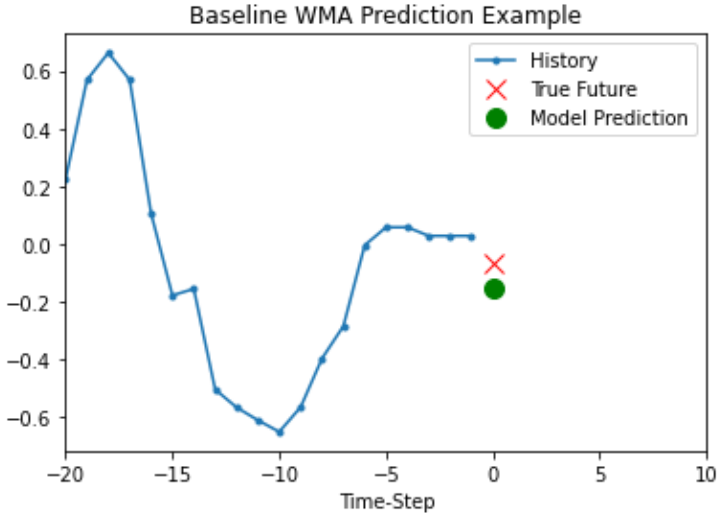


Imagen 3 Predicción con WMA

3.1.2 Red Neuronal Recurrente

Una vez calculadas unas baselines, comenzamos con la primera red neuronal, esta sigue tratando datos univariables y es una Red Neuronal Recurrente (RNN).

Una RNN es un tipo de red neural bien adaptada a los datos de series temporales. Las RNNs procesan una serie temporal paso a paso, manteniendo un estado interno en cada uno. Un importante argumento constructor para todas las capas RNN de keras es el argumento `return_sequences`. Este parámetro puede configurar la capa de dos maneras:

Si es `False` (el valor por defecto), la capa sólo devuelve la salida del último paso temporal, dando al modelo tiempo para calentar su estado interno antes de hacer una sola predicción:

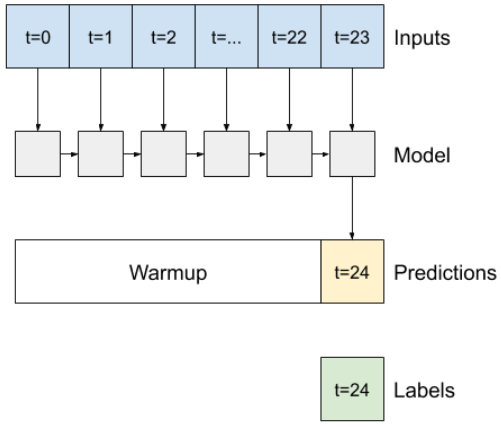


Imagen 4 Esquema red sin retorno de secuencias

Si es `True` la capa devuelve una salida para cada entrada. Esto es útil para apilar capas RNN y entrenar un modelo en múltiples pasos de tiempo simultáneamente.

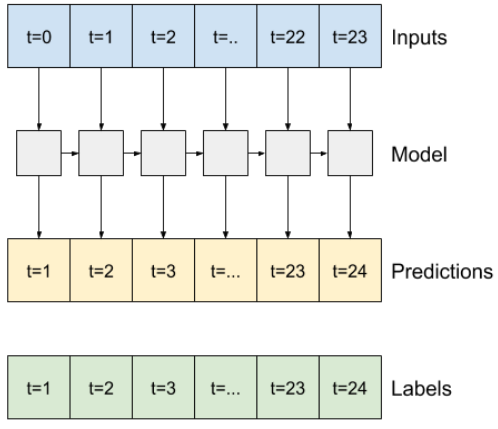


Imagen 5 Esquema red con retorno de secuencias

En este caso para los datos univariables se ha montado una LSTM (Long Short Term Memory) muy sencilla con return_sequences a False. La configuración es la siguiente:

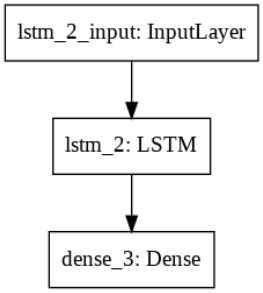


Imagen 5 Esquema red RNN univariable

3.2 Series temporales multivariable

El conjunto de datos original contiene 5 características que son las que se utilizan ahora. Es esencial incluir las fechas ya que el precio de la electricidad tiene una clara periodicidad diaria y anual.

El problema es que en el dataset la fecha es un string, algo que la red no entiende, un enfoque simple para convertirlo en una señal utilizable es usar `sin` y `cos` para convertir la hora en señales claras de "Hora del día" y "Hora del año":

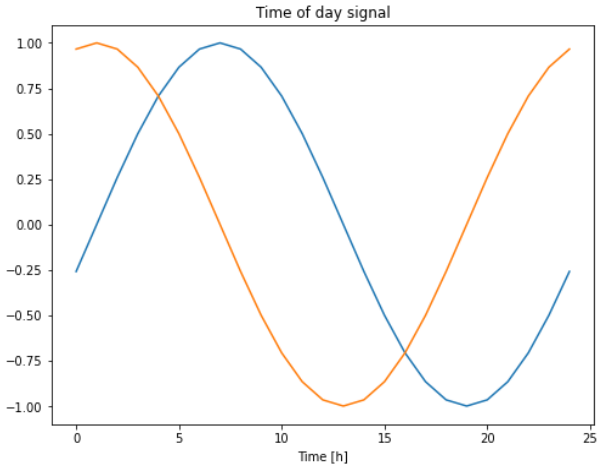


Imagen 6 Representación de señal diaria en sen cos

Tras incluir las nuevas columnas el dataset queda:

Demand	Wind	Solar	Price	Day sin	Day cos	Year sin	Year cos
Value	Value	Value	Value	Value	Value	Value	Value

Tabla 4 Aspecto del dataset final

Tras esto normalizamos el conjunto de datos utilizando la media y la desviación estándar.

3.2.1 Modelo de un solo paso

En una configuración de un solo paso, el modelo aprende a predecir un solo punto en el futuro basado en alguna historia proporcionada.

Como se mencionaba anteriormente una función realiza la misma tarea de ventana que la creada para los datos univariables sin embargo, aquí muestrea la observación pasada basada en el tamaño del paso dado. En este caso ese paso es 1.

En la red se muestran datos de los últimos cinco (5) días, es decir, 120 observaciones que se muestrean cada hora y representan la historia de los últimos cinco días. Para el modelo de predicción de un solo paso, la etiqueta de un datapoint es el precio a 12 horas en el futuro.

En este caso se vuelve a usar una RNN con la siguiente configuración:

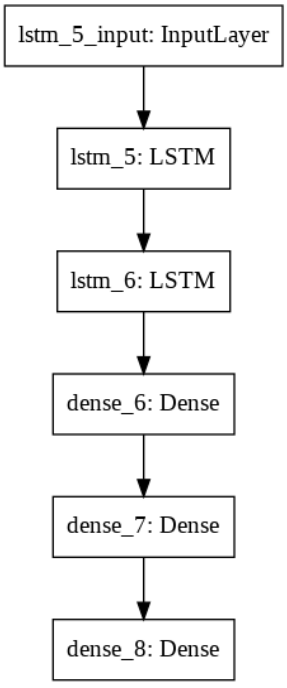


Imagen 7 Esquema red RNN multivariable de un paso

3.2.2 Modelo de múltiples pasos

En un modelo de predicción de múltiples pasos, dada una historia pasada, el modelo necesita aprender a predecir un rango de valores futuros. Así, a diferencia de un modelo de un solo paso, en el que sólo se predice un único punto futuro, un modelo de varios pasos predice una secuencia del futuro.

Para el modelo de múltiples pasos, los datos de entrenamiento consisten de nuevo en registros de los últimos cinco días muestreados cada hora. Sin embargo, en este caso, el modelo necesita aprender a predecir el precio para las próximas 12 horas. Para esta tarea, el conjunto de datos debe prepararse en consecuencia, por lo que el primer paso es simplemente crearlo de nuevo, pero con una ventana objetivo diferente.

3.2.2.1 Modelo LSTM simple

En primer lugar se prueba con una red LSTM sencilla, se vuelve a usar una RNN con la siguiente configuración:

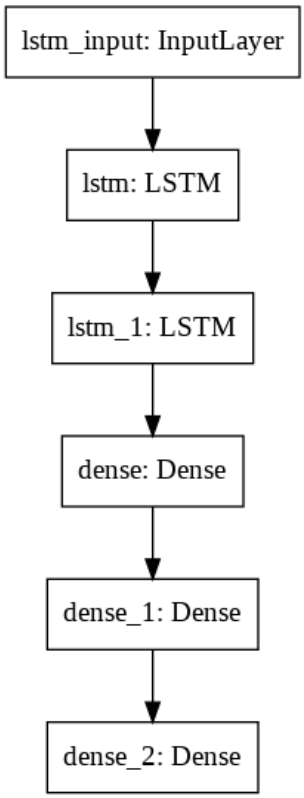


Imagen 8 Esquema red RNN multivariable de un paso

En un primer momento este fué el modelo elegido para hacer las predicciones de la APP al ser el que mejores resultados aportaba. Tras pedir modificaciones se probó con otras configuraciones.

3.2.2.2 Modelo Sequence to Sequence LSTM

Un modelo típico de sequence to sequence tiene dos partes: un **encoder** y un **decoder**. Ambas partes son dos modelos de red neuronal diferentes y son combinados en una red gigante. Es decir, el modelo estará compuesto por dos submodelos, el encoder para leer y codificar la secuencia de entrada, y el decoder que leerá la secuencia de entrada codificada y hará una predicción de un paso para cada elemento de la secuencia de salida.

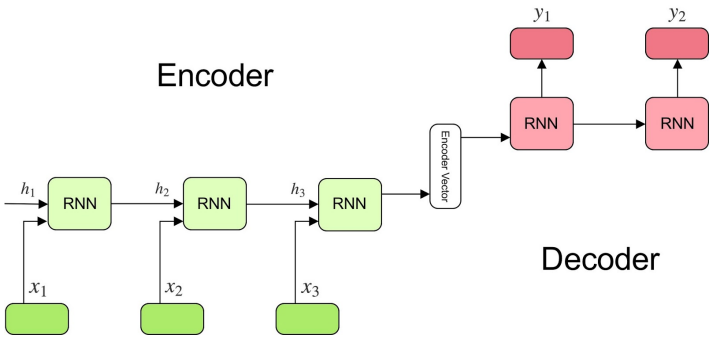


Imagen 9 Esquema Sequence to Sequence

Para ello se define una capa oculta de LSTM con 64 unidades. Posteriormente la representación interna de la secuencia de entrada se repite varias veces, una por cada paso de la secuencia de salida. Esta secuencia de vectores será la entrada al decoder del LSTM.

Luego definimos el decoder como una capa oculta de LSTM con 64 unidades. Lo importante es que el decoder emitirá la secuencia completa, no sólo la salida al final de la secuencia como hicimos con el encoder.

Por último usamos una capa fully connected para interpretar cada paso de la secuencia de salida antes de la capa de salida final. Para lograr esto, envolveremos la capa de interpretación (Dense) y la capa de salida (Dense) en una envoltura de “TimeDistributed” que permite que las capas envueltas sean usadas para cada paso de tiempo del decodificador. Quedando la red:

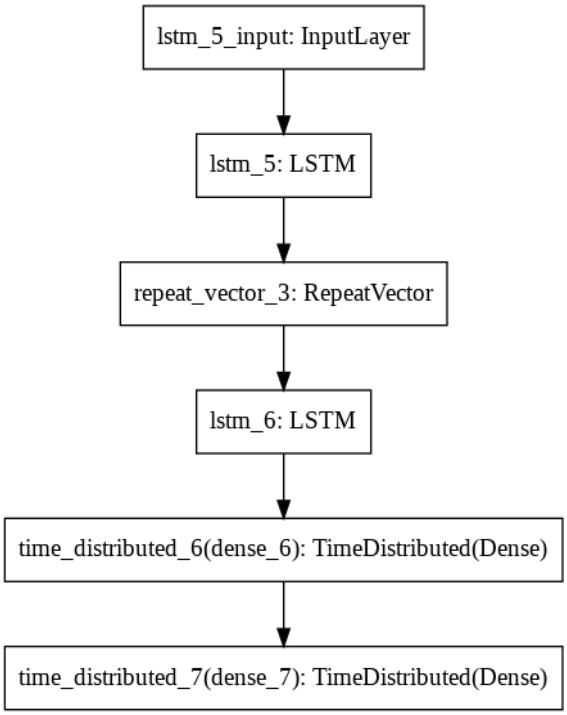


Imagen 10 Esquema red S2S multivariable

3.2.2.3 Modelo Sequence to Sequence CNN-LSTM

Intentando ir un poco más allá se decide probar con una configuración Sequence to Sequence incluyendo Redes Convolucionales (CNN) ya que se pueden usar estas como encoders, la CNN no soporta directamente la entrada de secuencias, sino que una CNN 1D es capaz de leer a través de la entrada de secuencias y aprender automáticamente las características más importantes. Estas, habitualmente, pueden ser interpretadas por un decodificador LSTM.

Para ello se define una arquitectura simple de CNN para el codificador que está compuesto de dos capas convolucionales seguidas de una capa de agrupación máxima, cuyos resultados son luego “aplanados” con una capa Flatten. La capa “MaxPooling” sirve para simplificar los mapas de características manteniendo 1/4 de los valores con la señal más grande (máxima), estos después de la capa de agrupación se aplanan en un vector largo que puede utilizarse como entrada en el proceso de decodificación.

El decoder usado es el mismo que en el modelo anterior una LSTM de 64 capas seguido de la capa de interpretación (Dense) y la de salida (Dense) envueltas en una TimeDistributed para que estas puedan ser usadas en cada paso de tiempo.

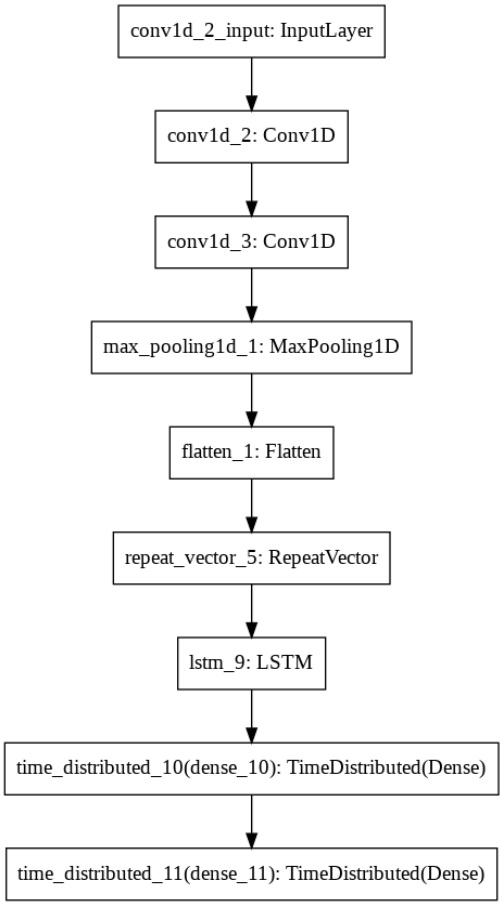


Imagen 11 Esquema red CNN-LSTM Multivariable

3.3 Aplicación para generar las predicciones

Una vez elegido el modelo con el que vamos a realizar las predicciones, se ha creado un script `deep_price.py` en el que se toma la fecha del dia de ejecución y los 20 días anteriores para hacer la predicción, se hace el procesado de los datos, se carga el modelo multivariable y se predicen las últimas 12h, finalmente a través de `steamlit` se genera un gráfico de predicciones.

4 Resumen de los resultados

Para evaluar los resultados se ha obtenido de cada modelo una comparativa con el baseline del MAE evaluando las predicciones con los datos de validación, se muestran las predicciones gráficamente también:

RNN univariable para un único paso:

- MAE para baseline MA 0.320
- MAE para baseline WMA 0.2884
- MAE para LSTM 0.0961

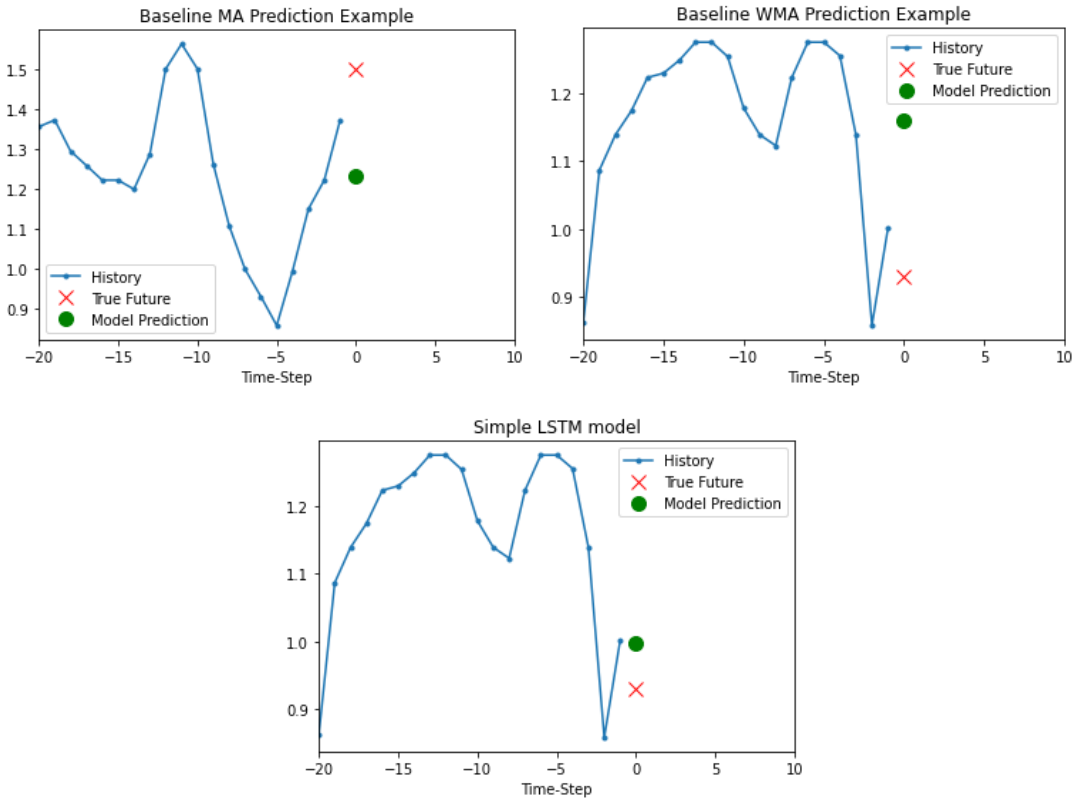


Imagen 12 Predicciones univariables de un paso

Se muestra una clara mejora con la RNN en este caso, siendo el MAE de 0.0961 frente a los 0.32 de MA y los 0.288 del WMA.

RNN multivariable para un único paso:

- MAE para baseline MA 0.7907
- MAE para baseline WMA 0.8452
- MAE para LSTM 0.3105

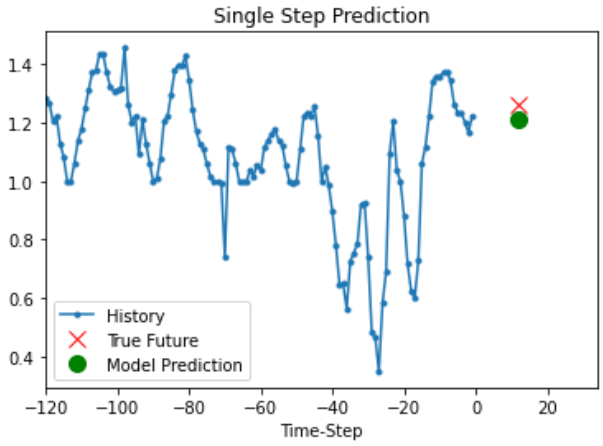


Imagen 13 Predicciones multivariables de un paso

Se muestra una clara mejora con la RNN en este caso, siendo el MAE de 0.3105 frente a los 0.7907 de MA y los 0.8452 del WMA.

RNN multivariable con 12 predicciones: al tener más de una predicción no se puede calcular el MA y el WMA

- MAE para LSTM 0.2307

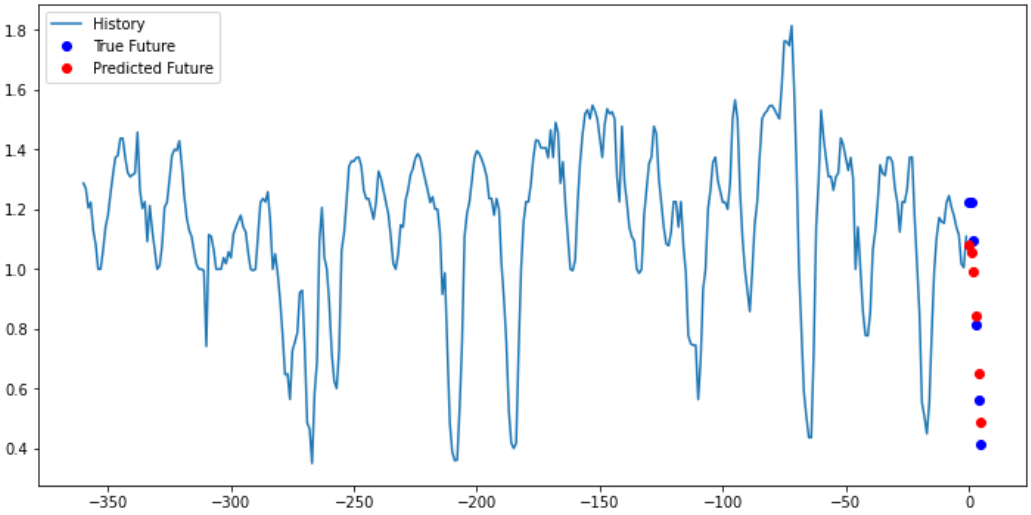


Imagen 14 Predicciones multivariables de 12 pasos

Se ve una clara mejora respecto a la RNN multivariable de un solo paso en el MAE además de estar estar prediciendo 12 h en vez de una.

Además de estas configuraciones se ha mejorado el modelo RNN multivariable con 12 predicciones modificando la ventana de predicción a 144 observaciones (5.8 días), teniendo esta un MAE de 0.2290:

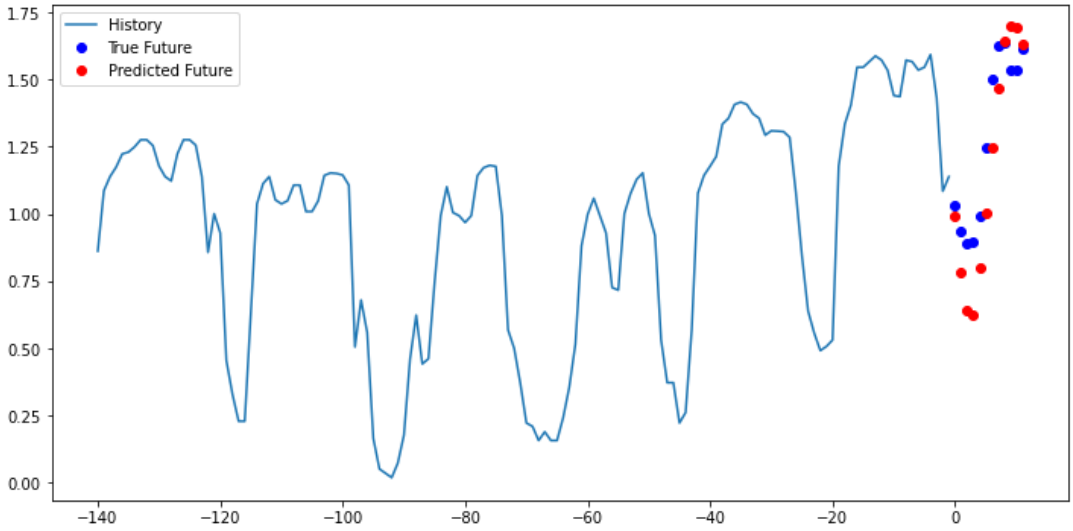


Imagen 15 Predicciones multivariables de 12 pasos mejorada

Sequence to Sequence LSTM:

- MAE para LSTM 0.2295

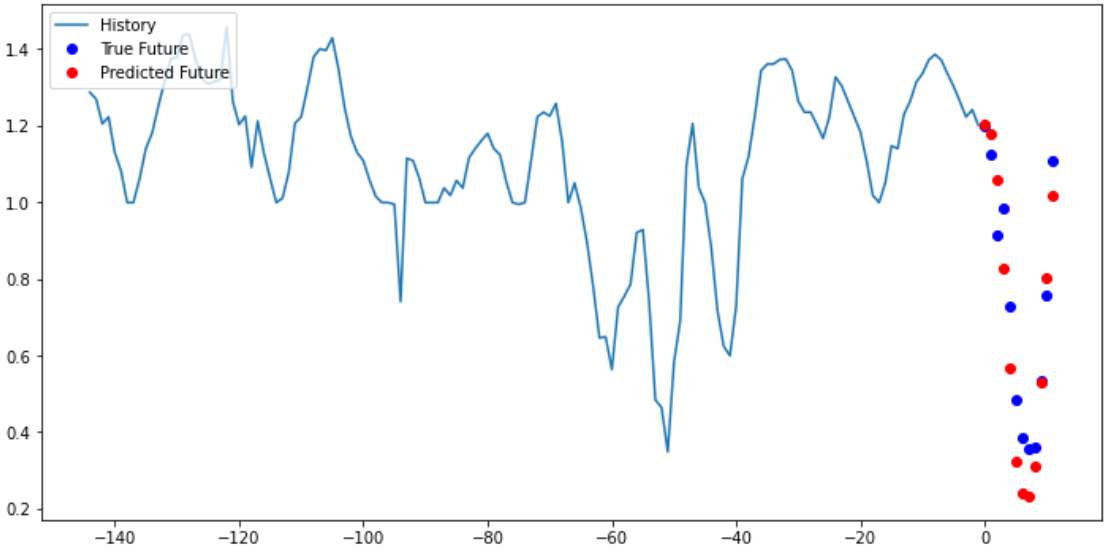


Imagen 16 Predicciones multivariables Seq2Seq LSTM simple

Manteniendo las configuraciones de la ventana de tiempo iguales al modelo mejorado (, no se muestra una clara diferencia en los resultados comparando con la RNN

multivariable multi predicción teniendo esta un MAE de 0.2290 y obteniendo un MAE de 0.2295 con el modelo Seq2Seq con LSTM.

Sequence to Sequence CNN-LSTM:

- MAE para LSTM 0.1061

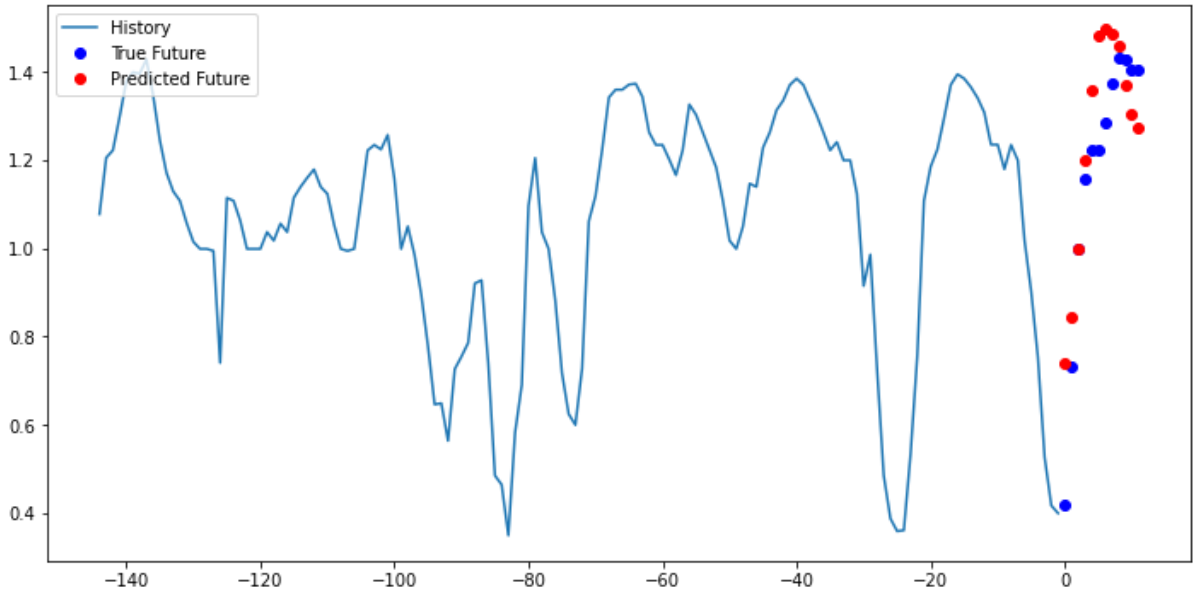


Imagen 16 Predicciones multivariables Seq2Seq CNN-LSTM

En este caso la aplicación como encoder de la CNN muestra una clara mejoría, pasando de un 0.2471 usando la LSTM como encoder a un 0.1061 de MAE con la CNN.

Estos han sido los mejores resultados entre múltiples pruebas realizadas variando desde funciones de activación, número de neuronas en cada capa, modificaciones en la ventana de datos y optimizadores.

Finalmente el modelo de esta última red es el que se ha guardado para hacer las predicciones en la app.

5 Conclusiones

Para predecir series temporales las RNN funcionan realmente bien, siendo una predicción como la del precio eléctrico, la mejor configuración tanto por cantidad de predicciones como resultado en MAE es la multivariable con distintos pasos, pero se puede mejorar con un paso más incluyendo modelos sequence to sequence y en este caso usar como encoders CNNs.

Aún así el modelo puede ser mejorado con distintas configuraciones que no han sido probadas, puede avanzarse mucho más en la predicción modificando la ventana, incluyendo más histórico y modificando la red con distintas capas.

6 Manual de usuario

Para poder ejecutar la APP deep_price es necesario clonar el repositorio de GitHub [deep_price_forecast] y ejecutar el `requirements.txt` para contar con los paquetes requeridos:

```
> pip install -r requirements.txt
```

Ahora ya solo es necesario ejecutar en la terminal la app:

```
> streamlit run deep_price.py
```

Se abrirá una pestaña en tu navegador con la predicción de las próximas 12h del precio eléctrico en España.

Si además quieres reentrenar el modelo solo tienes que:

- **Generar el dataset**: para eso tendrás que solicitar un token a REE como indiqué en la introducción y cambiar la fecha de fin a una más actual si quieres entrenar con más datos. Para ello ejecuta:

  ```
  > jupyter-notebook Get_data.ipynb
  ```

- **Ejecutar el modelo:** como yo no cuento con ordenadores con GPU todo el código está generado para que se ejecute en CPU, por lo que no tiene requerimientos de máquina. Para ello ejecuta:

  ```
  > jupyter-notebook model_deep_price.ipynb
  ```

- Ejecutar la APP: como el modelo se guarda automáticamente en la misma carpeta en la que lo busca la app tan solo hay que volver a ejecutar:

  ```
  > streamlit run deep_price.py
  ```

7 Bibliografía

Los recursos usados para la realización de este Trabajo de Fin de Máster han sido:

- Kschool (Juan Arevalo, RNN aplicado a series temporales)
- Kschool (Victor Peinado, Streamlit)

- Towards Data Science
- TensorFlow tutorials
- Wikipedia
- Data Science Central