

Informe proyecto módulo 2, borrador de traspaso a inglés.

Nombre estudiantes:

* Agustín Kuskinen Gimeno (NRC: 8074)
* Eduardo Correa Marchant (NRC: 8074)
* Matthias Seiltgens Trouve (NRC: 8074)
* Yuan Huang Fan (NRC: 8074)

Docente(s) encargado(s):

* PhD. Pablo Schwarzenberg Riveros
* PhD. Billy Peralta Márquez

Carrera:

Ingeniería Civil Informática

Contenido

[Introducción y resumen del primer informe 3](#_Toc165310743)

[Descripción de las características que constituyen a la entrada del modelo. 4](#_Toc165310744)

[Descripción de los parámetros utilizados para entrenar el modelo. 7](#_Toc165310745)

[Parámetros de ARIMA (modelo ganador): 7](#_Toc165310746)

[Análisis estadístico con dataset con outliers (ARIMA). 12](#_Toc165310747)

[Análisis estadístico con dataset sin outliers (ARIMA). 16](#_Toc165310748)

[Parámetros de RNN con ADAM (modelo mención especial). 22](#_Toc165310749)

[Summary RNN con outliers: 23](#_Toc165310750)

[Summary RNN sin outliers: 23](#_Toc165310751)

[Descripción de la arquitectura del modelo utilizado 24](#_Toc165310752)

[Arquitectura algorítmica y matemática de ARIMA (modelo ganador) con datos y gráficos. 24](#_Toc165310753)

[Análisis matemático. 24](#_Toc165310754)

[Gráficos dados por ARIMA, valores MSE-MAE y estacionario/no estacionario en dataset con outliers: 25](#_Toc165310755)

[Gráficos dados por ARIMA, valores MSE-MAE y estacionario/no estacionario en dataset sin outliers: 32](#_Toc165310756)

[Arquitectura algorítmica y matemática de RNN con Adam (modelo con mención especial) con datos y gráficos. 42](#_Toc165310757)

[Análisis matemático. 42](#_Toc165310758)

[Parámetros orientados a la arquitectura de RNN. 43](#_Toc165310759)

[MSE Y MAE, ¿Qué son? 43](#_Toc165310760)

[Gráficos dados por RNN, valores MSE-MAE y gráficos epoch vs loss en dataset con outliers: 44](#_Toc165310761)

[Gráficos dados por RNN, valores MSE-MAE y gráficos epoch vs loss en dataset sin outliers: 55](#_Toc165310762)

[Explicación de cómo se llegó al modelo ganador (ARIMA) con gráficos: 68](#_Toc165310763)

[Comparación con outliers. 69](#_Toc165310764)

[Comparación sin outliers. 70](#_Toc165310765)

[Conclusiones respecto al trabajo, limitaciones del modelo y propuestas acerca de cómo mejorar su desempeño en el futuro. 72](#_Toc165310766)

[Conclusiones: 72](#_Toc165310767)

[Limitaciones del Modelo: 72](#_Toc165310768)

[Propuestas para Mejorar el Desempeño Futuro: 73](#_Toc165310769)

[Resumen de conclusión, limitaciones y propuestas: 74](#_Toc165310770)

[BIBLIOGRAFÍA Y MATERIAL DE AYUDA: 75](#_Toc165310771)

# Introducción y resumen del primer informe

El desafío que enfrentamos implica examinar una amplia base de datos relacionada con el consumo de energía eléctrica en diferentes áreas urbanas y puntos de distribución durante un período específico. Esta información detallada incluye registros únicos, horarios de mediciones, cantidades de energía consumida, indicadores de calidad de datos y la infraestructura eléctrica subyacente.

Para comprender mejor este panorama, se han proporcionado imágenes que ilustran los aspectos clave de los datos, como identificadores únicos, marcas de tiempo y distribución del consumo de energía. Estas representaciones visuales son fundamentales para comprender la complejidad de la red eléctrica y la variabilidad en los patrones de consumo.

Se sugiere mejorar la visualización de los datos mediante una reestructuración que simplifique su interpretación. Dividir la información en 10 columnas facilitaría la comprensión de los datos y destacaría las relaciones entre las distintas variables.

El análisis de la calidad de los datos será una parte esencial de nuestra investigación, junto con el uso de estadísticas descriptivas y gráficos para identificar tendencias significativas. Además, propondremos una selección de variables para un modelo de aprendizaje automático, así como la división del conjunto de datos en conjuntos de entrenamiento y prueba para evaluar la eficacia del modelo.

En cuanto a las técnicas de aprendizaje profundo, exploraremos cuáles son las más adecuadas para evaluar y seleccionar el mejor modelo generado, proporcionando razones fundamentadas para nuestra elección.

Finalmente, describiremos de manera conceptual el tipo de modelo neuronal que podría ser desarrollado utilizando las entradas y salidas propuestas, ofreciendo una visión clara de cómo las redes neuronales podrían abordar este desafío de manera efectiva.

# Descripción de las características que constituyen a la entrada del modelo.

Antes de describir las características que constituyen a la entrada del modelo, es totalmente fundamental aclarar que en nuestros datos de tipo ‘DATETIME’, se ha usado el periodo 2018 – 2021 para training, y 2022 para testing, el motivo de ello es porque independiente que nuestro dataset tenga o no outliers, tanto los datos de 2017 como de 2023 están incompletos, no da una información sólida o son informaciones totalmente fuera de alcance o contexto.

A la vez ya resumiendo sobre nuestros resultados, ARIMA fue el modelo ganador tanto con outliers como sin outliers en nuestros datasets, aun así, daremos una mención especial a la red neuronal recurrente (RNN) usando un optimizador de ADAM, para explicar otras características que, como grupo, consideramos que se puede complementar en la información y análisis.

Tanto para el dataset original con Outliers, como el dataset sin outliers, se consideró que el modelo ganador para los dos datasets es ARIMA, ahora bien, si bien con ARIMA no hay un análisis explícito acerca de las características, si es totalmente considerado por los resultados tomados por ello (análisis de resultados en [Descripción de los parámetros utilizados para entrenar el modelo (Justificar elección)](#_Descripción_de_los)), es importante mencionar que ARIMA es un modelo más orientado como modelo estadístico que un modelo de red neuronal. A la vez igual daremos una mención especial usando red neuronal recurrente (RNN) usando optimizador ADAM.

Sobre ARIMA, podríamos mencionar alguno de los datos utilizados (características) en ARIMA, para cada subestación.

Texto

Descripción generada automáticamente

Imagen 1: Datos de entrada (General) para ARIMA.

Como se menciona explícitamente en la imagen anterior, se usaron de características de entrada las variables de ‘año’, ‘mes’, ‘dia’, ‘hora’, tanto para el set de training como de testing, y usando la variable ‘consumo’ como mi variable dependiente, tanto para entrenamiento como de testing.

Es importante igual aclarar que a diferencia de MLP y RNN, ARIMA no necesariamente es necesario, o incluso no es recomendable usarlo con variables DUMMY, por lo que ende acá no se da uso de variables DUMMY.

Al mismo tiempo, para verificar sobre en que barra es mejor para entrenamiento y para predicción, es ideal que se trabaje con todo tipo de barras posible:

Texto

Descripción generada automáticamente

Imagen 2: Separación por barra y ordenamiento.

En la imagen anterior, consideramos, separamos las barras para entrenamiento y para validación para el futuro uso de ARIMA, y a la vez se decide usar la variable de ‘horario’ como INDEX para nuestro dataset, tanto para dataset original como dataset modificado.

Es importante recordar que en un modelo ARIMA, las características de entrada son los valores pasados de la serie temporal que estás tratando de predecir. En nuestro caso, se utiliza los valores pasados de:

* Consumo: para predecir los valores futuros.
* año, mes, dia, hora, barra: se utilizan para dividir tus datos en conjuntos de entrenamiento y validación y para organizar tus datos en el orden correcto.

Sobre RNN con ADAM, si nos guiamos con el código proporcionado:

Texto

Descripción generada automáticamente

Imagen 3: Separación datos entrenamiento y testing para RNN, sumado con creación de diccionarios.

Si apreciamos la imagen anterior, aún así, podemos fijarnos que usa una estructura idéntica al de ARIMA en el uso de diccionarios y la importancia de las subestaciones (guardada en barras).

Texto

Descripción generada automáticamente

Imagen 4: Preparación datos para uso de RNN para predicción.

El código anterior prepara los datos de entrenamiento y división para nuestra red neuronal recurrente (RNN) y divide la serie temporal en ventanas de tamaño ‘window\_size’ para generar secuencias de entrada y de salida. Convertir los datos a tipo ‘float32’ es esencial para buen funcionamiento de RNN.

Dada esa información, podemos mencionar que nuestras características de entrada serían:

* Ventanas de tiempo (window\_size = 40): Eso significa que nuestro modelo utiliza 40 valores anteriores de ‘consumo’ para predecir el valor actual. Esas ventanas de tiempo logran definir cuánta información temporal está utilizando el modelo para hacer las predicciones.
* Consumo: La variable objetivo, se utiliza los datos históricos de consumo para entrenar modelo y hacer predicciones futuras.
* Barra: Se utiliza para esperar patrones diferentes de consumo para cada barra (nombre subestación).

Ahora bien, es importante considerar que ‘año’, ‘mes’, ‘dia’, ‘hora’ no se utilizan como características de entrada directas al modelo RNN, pero está implícitamente en nuestro índex ‘HORARIO’.

# Descripción de los parámetros utilizados para entrenar el modelo.

## Parámetros de ARIMA (modelo ganador):

Antes podemos mencionar que tanto en nuestro Dataset, tanto con o sin Outliers, es importante aclarar cómo se mencionó en el [punto anterior](#_Descripción_de_las) de nuestros datasets modificados para un Índex con horario de 2018 a 2022, donde se tenga en las columnas todas las subestaciones tipo ‘barra\_nombreSubEstación, Subestación pertenece a {AJAHUEL, BUIN, CHENA, CNAVIA, ELSALTO, FLORIDA, LOSALME}’, los datasets orientado para el uso de modelo ARIMA tiene forma como se representa en la siguiente:

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Imagen 5: Estructura Datasets orientados a ARIMA

Como se aprecia en la imagen anterior, usaremos el horario como INDEX, y usaremos las barras y consumo como nuestras variables dependientes, donde vamos a predecir posibles futuros valores de consumo, en cada barra (nombre de subestación) en función de nuestro horario. Recordar que como se mencionó con anterioridad, se usará el horario de 2018 a 2021 como training y 2022 como testing, para ARIMA usaremos nuestras variables independientes para training y para testing usando ‘año’, ’mes’, ’dia’, ’hora’, y nuestra variable dependiente para training y para testing usaremos ‘consumo’.

Como se recuerda en la parte 1 de la tarea, los gráficos de autocorrelación y de autocorrelación parcial nos ha dado un valor de para ARIMA, por tema de defecto, se ha usado a la vez el valor de , es importante aclarar el uso, recordar que esos valores se usaron gracias al análisis gráfico de autocorrelación y autocorrelación parcial visto en el código de la parte 1:

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza media

Imagen 6: Gráficos de ARIMA para AJAHUEL

Gráfico

Descripción generada automáticamente

Imagen 7: Gráficos de ARIMA para BUIN

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente con confianza media

Imagen 8: Gráficos de ARIMA para CHENA

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Imagen 9: Gráficos de ARIMA para CNAVIA

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Imagen 10: Gráficos de ARIMA para ELSALTO

Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente

Imagen 11: Gráficos de ARIMA para FLORIDA

Gráfico

Descripción generada automáticamente

Imagen 12: Gráficos de ARIMA para LOSALME

Al apreciar cada gráfico de autocorrelación y autocorrelación parcial, apreciamos que los valores óptimos para usarlos como parámetros de ARIMA sería un , esos, tomando a luego serán nuestros parámetros utilizados para usar para nuestras predicciones.

Sobre los valores que se puede dar con los resúmenes estadísticos con *model.summary(),* igual se tiene su total importancia paranuestro análisis de parámetros que será analizado, por barra (nombre subestación).

Antes de ir al análisis estadístico, es importante mencionar algunos parámetros que tendremos en cada tabla SARIMAX Results, y su relación con ARIMA:

Log Likelihood: Es el logaritmo de la función de verosimilitud. Cuanto mayor sea este valor, mejor se ajusta el modelo a los datos.

AIC (Akaike Information Criterion): Es un criterio de selección de modelos. Los modelos con menor AIC son preferibles.

BIC (Bayesian Information Criterion): Al igual que el AIC, el BIC es un criterio de selección de modelos. Los modelos con menor BIC son preferibles.

coef: Son los coeficientes estimados para cada término del modelo.

std err: Es el error estándar de los coeficientes estimados. Cuanto menor sea este valor, más confiable será la estimación del coeficiente.

Jarque-Bera (JB): Es una prueba de normalidad de los residuos. Si el valor p asociado (Prob (JB)) es pequeño (generalmente menor a 0.05), puedes concluir que los residuos no son normales.

Esos parámetros serán considerados para análisis de cada tabla para cada subestación.

### Análisis estadístico con dataset con outliers (ARIMA).

En el caso de AJAHUEL tenemos ese resumen estadístico:

Interfaz de usuario gráfica

Descripción generada automáticamente

Imagen 13: Resumen estadístico de AJAHUEL

En AJAHUEL apreciamos que tenemos:

* Log Likelihood: -178581.090 (no tiene un ajuste esperado para AJAHUEL).
* AIC: 357168.1
* BIC: 357194.1
* Std error: 0.265 (orientado a sigma2).
* Jarque-Bera: 22963182.2

En el caso de BUIN tenemos ese resumen estadístico:

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Imagen 14: Resumen estadístico de BUIN

En BUIN apreciamos que tenemos:

* Log Likelihood: -71812.2(no tiene un ajuste esperado para BUIN).
* AIC: 143630.5
* BIC: 143656.6
* Std error: 0.000(orientado a sigma2).
* Jarque-Bera: 607431297802.6

En el caso de CHENA tenemos ese resumen estadístico:

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Imagen 15: Resumen estadístico de CHENA

En CHENA apreciamos que tenemos:

* Log Likelihood: -182723.9(no tiene un ajuste esperado para CHENA).
* AIC: 365453.9
* BIC: 365480.03
* Std error: 0.286 (orientado a sigma2).
* Jarque-Bera: 23132270.91

En CNAVIA apreciamos que tenemos:

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Imagen 16: Resumen estadístico de CNAVIA

En CNAVIA apreciamos que tenemos:

* Log Likelihood: -212846.3 (no tiene un ajuste esperado para CNAVIA).
* AIC: 425698.6
* BIC: 425698.6
* Std error: 1.539 (orientado a sigma2).
* Jarque-Bera: 8014114.06

En ELSALTO apreciamos que tenemos:

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Imagen 17: Resumen estadístico de ELSALTO

En ELSALTO apreciamos que tenemos:

* Log Likelihood: -212354.3 (no tiene un ajuste esperado para ELSALTO).
* AIC: 424714.6
* BIC: 424714.6
* Std error: 0.958 (orientado a sigma2).
* Jarque-Bera: 32694744.02

En FLORIDA apreciamos que tenemos:

Interfaz de usuario gráfica

Descripción generada automáticamente

Imagen 18: Resumen estadístico de FLORIDA

En FLORIDA apreciamos que tenemos:

* Log Likelihood: -129488.175 (no tiene un ajuste esperado para FLORIDA).
* AIC: 258982.35
* BIC: 259008.4
* Std error: 0.017 (orientado a sigma2).
* Jarque-Bera: 45537301.6

En LOSALME apreciamos que tenemos:

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Imagen 19: Resumen estadístico de LOSALME

En LOSALME apreciamos que tenemos:

* Log Likelihood: -177604.3 (no tiene un ajuste esperado para FLORIDA).
* AIC: 355214.7
* BIC: 355240.8
* Std error: 0.262 (orientado a sigma2).
* Jarque-Bera: 23670295.7

Hasta ahora, con outliers en ARIMA, tenemos FLORIDA como barra ideal para entrenamiento y testing.

### Análisis estadístico con dataset sin outliers (ARIMA).

En el caso de AJAHUEL tenemos ese resumen estadístico:

Pantalla de computadora con letras

Descripción generada automáticamente con confianza media

Imagen 20: Resumen estadístico de AJAHUEL

En AJAHUEL apreciamos que tenemos:

* Log Likelihood: -178353.1 (no tiene un ajuste esperado para AJAHUEL).
* AIC: 356712.2
* BIC: 356738.3
* Std error: 0.271 (orientado a sigma2).
* Jarque-Bera: 19443947.31

En el caso de BUIN tenemos ese resumen estadístico:

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Imagen 21: Resumen estadístico de BUIN

En BUIN apreciamos que tenemos:

* Log Likelihood: -234428.41 (no tiene un ajuste esperado para BUIN).
* AIC: 468862.83
* BIC: 468888.8
* Std error: 6.061 (orientado a sigma2).
* Jarque-Bera: 496473.67

En el caso de CHENA tenemos ese resumen estadístico:

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Imagen 22: Resumen estadístico de CHENA

En CHENA apreciamos que tenemos:

* Log Likelihood: -237385.3(no tiene un ajuste esperado para CHENA).
* AIC: 474776.6
* BIC: 474802.6
* Std error: 7.016 (orientado a sigma2).
* Jarque-Bera: 603068.59

En CNAVIA apreciamos que tenemos:

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Imagen 23: Resumen estadístico de CNAVIA

En CNAVIA apreciamos que tenemos:

* Log Likelihood: -218026.2 (no tiene un ajuste esperado para CNAVIA).
* AIC: 436058.559
* BIC: 436084.613
* Std error: 1.920 (orientado a sigma2).
* Jarque-Bera: 5546041.53

En ELSALTO apreciamos que tenemos:

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Imagen 24: Resumen estadístico de ELSALTO

En ELSALTO apreciamos que tenemos:

* Log Likelihood: -219819.569 (no tiene un ajuste esperado para ELSALTO).
* AIC: 439645.139
* BIC: 439671.192
* Std error: 1.872 (orientado a sigma2).
* Jarque-Bera: 7588057.31

En FLORIDA apreciamos que tenemos:

Interfaz de usuario gráfica

Descripción generada automáticamente

Imagen 25: Resumen estadístico de FLORIDA

En FLORIDA apreciamos que tenemos:

* Log Likelihood: -129488.175 (no tiene un ajuste esperado para FLORIDA).
* AIC: 258982.351
* BIC: 259008.404
* Std error: 0.017 (orientado a sigma2).
* Jarque-Bera: 45537301.61

En LOSALME apreciamos que tenemos:

Interfaz de usuario gráfica

Descripción generada automáticamente

Imagen 26: Resumen estadístico de LOSALME

En LOSALME apreciamos que tenemos:

* Log Likelihood: -178097.691 (no tiene un ajuste esperado para FLORIDA).
* AIC: 356201.382
* BIC: 356227.436
* Std error: 0.270 (orientado a sigma2).
* Jarque-Bera: 22049415.72

Hasta ahora, sin outliers en ARIMA, tenemos FLORIDA como barra ideal para entrenamiento y testing.

## Parámetros de RNN con ADAM (modelo mención especial).

Como se mencionó, se hará un análisis también de nuestra red neuronal recurrente (RNN) con ADAM, como modelo de mención honorífica para dar tener un análisis más completo y así, tener complementar información de ARIMA con información de RNN.

Sobre los parámetros a mencionar de manera referente, debido que se analizará más en [Descripción de la arquitectura del modelo utilizado](#_Descripción_de_la), se dará una referencia para dar una mención a lo básico del código visto en el punto anterior:

* window\_size: Este es el tamaño de la ventana que se utiliza para crear las secuencias de entrada para el modelo RNN. En este caso, el tamaño de la ventana es de 40.
* Input shape: La forma de entrada para el modelo es (window\_size, 1). Esto significa que el modelo espera secuencias de longitud 40 y cada elemento de la secuencia es un vector de dimensión 1.
* SimpleRNN: Este es el tipo de capa recurrente que se utiliza en el modelo. En este caso, se utiliza una capa SimpleRNN con un tamaño de salida igual al tamaño de la ventana.
* Dense: Esta es la capa de salida del modelo. Tiene una sola unidad con una función de activación lineal.
* Optimizer: El optimizador utilizado para entrenar el modelo es Adam.
* Metrics: Las métricas utilizadas para evaluar el rendimiento del modelo durante el entrenamiento son el error cuadrático medio (mse), el error absoluto medio (mae) y el error porcentual absoluto medio (mape).
* Epochs: El número de épocas para las que se entrena el modelo es de 36.

Ahora bien, para el análisis estadístico con summary() en ARIMA, este comando es diferente para RNN el cual sirve para ver como está estructurado nuestro modelo, es importante explicar el significado de cada uno (los que será escritos explícitamente):

* Layer (type): Es el tipo de capa en el modelo, de ahí existe 2 tipos:
  + SimpleRNN: La capa de red neuronal recurrente Simple, describiendo es una capa donde las salidas de ciertas neuronas se retroalimentan a la entrada para ayudar a la red a “recordar” información a lo largo que pasa el tiempo.
  + Dense: Capa densa, capa de red neuronal donde cada neurona recibe entrada de todas las neuronas de la capa anterior.
* Output Shape: Forma de la salida de cada capa.
* Param: Número de parámetros que se puede aprender por cada etapa.
* Total Params: Este es el total de parámetros entrenables en el modelo.

Tenemos los siguientes valores a considerar para nuestro summary en RNN:

### Summary RNN con outliers:

Texto

Descripción generada automáticamente

Imagen 27: Summary de RNN con outliers

Dada la imagen anterior, apreciamos la siguiente información:

* En Layer (type): Como se mencionó son 2 capas: SIMPLERNN y dense\_3.
* En Output Shape tiene una salida de capa (None,40), o séase dado un número de lotes de entrada, producirá una salida de tamaño 40, mientras que (None, 1) significa que la capa dado un número de lotes de entrada producirá una de salida de tamaño 1.
* Param: SimpleRNN tiene 1680 parámetros y dense\_3 tiene 41 parámetros.
* Total Params: En total son 1721 parámetros.

Como se puede apreciar en el summary está definido como está construido nuestro modelo

### Summary RNN sin outliers:

Texto

Descripción generada automáticamente

Imagen 28: SUMMARY DE RNN SIN OUTLIERS

Dada la imagen anterior, apreciamos la siguiente información:

* En Layer (type): Como se mencionó son 2 capas: SIMPLERNN y dense\_19.
* En Output Shape tiene una salida de capa (None,40), o séase dado un número de lotes de entrada, producirá una salida de tamaño 40, mientras que (None, 1) significa que la capa dado un número de lotes de entrada producirá una de salida de tamaño 1.
* Param: SimpleRNN tiene 1680 parámetros y dense\_3 tiene 41 parámetros.
* Total Params: En total son 1721 parámetros.

# Descripción de la arquitectura del modelo utilizado

## Arquitectura algorítmica y matemática de ARIMA (modelo ganador) con datos y gráficos.

### Análisis matemático.

Como recordamos, ARIMA su objetivo es analizar y predecir series temporales para predecir el futuro basándose en los valores pasados.

Si recordamos matemáticamente, ARIMA se puede expresar como:

Donde es la serie temporal, es el operador de retraso, serían mis parámetros del modelo, el orden de diferenciación, son los parámetros del modelo de media móvil, y es el error de tiempo , se asume que sigue una distribución normal con media 0, con una varianza constante.

En nuestro caso, ∈

O séase, representa a nuestra serie temporal que estamos modelando, orientado a nuestras subestaciones.

Como mencionamos, representa a nuestro operador de retraso, en la práctica eso significa que estás utilizando valores pasados de tu serie temporal para hacer predicciones, en nuestro código, se maneja internamente en la función ARIMA cuando especificamos siendo aquello nuestros parámetros de ARIMA.

Sobre , son los parámetros del modelo autorregresivo, en este caso con , se aprende automáticamente, en nuestro caso ya que para cada subestación, tendríamos un solo parámetro autorregresivo , en la que se puede interpretar como la influencia que podría tener el valor inmediatamente de la serie temporal, sobre la predicción actual.

El orden de diferenciación , sería parte de nuestros parámetros de modelo de media móvil, debido que su valor es 1, se refiere que estamos realizando una diferenciación de primer orden en mi serie, con un retraso de 1 de manera donde es mi resultado de diferenciación, es mi valor actual de la serie temporal en tiempo , mientras que lo mismo, pero valor anterior en un tiempo

Los valores , sería los parámetros del modelo de media móvil. Se aprenden automáticamente al igual que los parámetros, en este caso su función es ayudar una determinación de la influencia de cada error de predicción pasado sobre la predicción actual, como , podemos mencionar una función de modelo de media móvil de donde es la media de la serie temporal, mientras que el resto de la función son ya explicados, lo que si, como ya sabemos, se refiere a nuestro error de tiempo, también calculado con donde se hace un cálculo de

.

Donde es el valor real, el valor predicho por el modelo de tiempo.

## Gráficos dados por ARIMA, valores MSE-MAE y estacionario/no estacionario en dataset con outliers:

Importante aclaración que MSE y MAE su explicación se encuentra en el punto de [MSE Y MAE, ¿Qué son?](#_MSE_Y_MAE,), para tener una idea, al hablar de esos valores.

A la vez para mezclar los gráficos con mayor análisis, se puede comparar con los [resultados estadísticos tomados con ARIMA.](#_Análisis_estadístico_con)

En AJAHUEL, para predecir 2022 se obtuvo el siguiente gráfico:

Gráfico, Gráfico de líneas

Descripción generada automáticamente

Imagen 29: Valores reales vs predicciones para 2022 en AJAHUEL con ARIMA

En el gráfico anterior, se puede lograr apreciar aparte de los sesgos, que, aun así, igual las predicciones logran cubrir gran parte de los valores reales, a pesar del desbalance que se aprecia entre 2022-03 a 2022-05.

A la vez, se obtuvo los siguientes valores:

* MAE 0.11106657372352198
* MSE 1.5690193730856743
* AIC 143630.5687315037

En BUIN, para predecir 2022 se obtuvo el siguiente gráfico:

Gráfico, Histograma

Descripción generada automáticamente

Imagen 30: Valores reales vs predicciones para 2022 en BUIN con ARIMA

En el gráfico anterior, se aprecia que no hay un excelente balanceo en 2022-03 tanto para valores reales como en predicciones, aún así, en el resto de los meses, se ve un buen ajuste.

A la vez, se obtuvo los siguientes valores:

* MAE 8.612872117551614
* MSE 252.6910514107488
* AIC 365453.9844477476

En CHENA, para predecir 2022 se obtuvo el siguiente gráfico:

Gráfico

Descripción generada automáticamente

Imagen 31: Valores reales vs predicciones para 2022 en CHENA con ARIMA

Se aprecia que en esta ocasión al igual que en AJAHUEL, tiene un buen comportamiento entre relación de valores reales y predicciones, a pesar del desajuste entre 2022-03 y 2022-05.

A la vez, se obtuvo los siguientes valores:

* MAE 8.612872117551614
* MSE 252.6910514107488
* AIC 365453.9844477476
* En CHENA, para predecir 2022 se obtuvo el siguiente gráfico:

En CNAVIA, para predecir 2022 se obtuvo el siguiente gráfico:

Gráfico

Descripción generada automáticamente

Imagen 32: Valores reales vs predicciones para 2022 en CNAVIA con ARIMA

Apreciamos por la imagen anterior, que al igual que en AJAHUEL y en CHENA, tenemos una estructura casi similar a las mencionadas.

A la vez, se obtuvo los siguientes valores:

* MAE 19.122630174875386
* MSE 1003.6770766618824
* AIC 425698.6492081627

En ELSALTO, para predecir 2022 se obtuvo el siguiente gráfico:

Gráfico

Descripción generada automáticamente

Imagen 33: Valores reales vs predicciones para 2022 en ELSALTO con ARIMA

Apreciamos por la imagen anterior, que al igual que en AJAHUEL y en CHENA y CNAVIA, tenemos una estructura casi similar a las mencionadas, aunque con mayor sesgo en 2022-03.

A la vez, se obtuvo los siguientes valores:

* MAE 17.54472937275069
* MSE 979.7530138569809
* AIC 424714.6714744745

En FLORIDA, para predecir 2022 se obtuvo el siguiente gráfico:

Gráfico, Histograma

Descripción generada automáticamente

Imagen 34: Valores reales vs predicciones para 2022 en FLORIDA con ARIMA

Apreciamos esta vez ya no es mismo comportamiento como se corresponde, con las barras anteriores, aún así, se aprecia que igual sigue al margen de los valores reales.

A la vez, se obtuvo los siguientes valores:

* MAE 1.6953630055610456
* MSE 22.01202381204552
* AIC 258982.35080836996

En LOSALME, para predecir 2022 se obtuvo el siguiente gráfico:

Gráfico

Descripción generada automáticamente

Imagen 35: Valores reales vs predicciones para 2022 en LOSALME con ARIMA

Apreciamos buen comportamiento en la imagen anterior, sin embargo, se aprecia demasiado desajuste en 2022-03 y 2022-05 que no aporta buena confianza.

A la vez, se obtuvo los siguientes valores:

* MAE 8.210784909277896
* MSE 200.46075057900393
* AIC 355214.7373781807

Apreciamos que en las subestaciones con menor índice de error posible tanto en MAE como en MSE, sería en AJAHUEL y en FLORIDA, siendo AJAHUEL primera opción y FLORIDA como segunda opción.

Sobre el análisis de si las series anteriores son o no estacionarios, se probó con la prueba KPSS:

Texto

Descripción generada automáticamente

Imagen 36: Test KPSS en ARIMA, algoritmo hecho.

Obteniendo los siguientes valores:

--------------------------------

BARRA AJAHUEL

Estadístico KPSS: 2.938176669701572

Valor p: 0.01

Valores críticos: {'10%': 0.347, '5%': 0.463, '2.5%': 0.574, '1%': 0.739}

La serie es estacionaria.

--------------------------------

--------------------------------

BARRA BUIN

Estadístico KPSS: 2.5546466699244013

Valor p: 0.01

Valores críticos: {'10%': 0.347, '5%': 0.463, '2.5%': 0.574, '1%': 0.739}

La serie es estacionaria.

--------------------------------

--------------------------------

BARRA CHENA

Estadístico KPSS: 7.129680832611945

Valor p: 0.01

Valores críticos: {'10%': 0.347, '5%': 0.463, '2.5%': 0.574, '1%': 0.739}

La serie es estacionaria.

--------------------------------

--------------------------------

BARRA CNAVIA

Estadístico KPSS: 6.212401899408551

Valor p: 0.01

Valores críticos: {'10%': 0.347, '5%': 0.463, '2.5%': 0.574, '1%': 0.739}

La serie es estacionaria.

--------------------------------

--------------------------------

BARRA ELSALTO

Estadístico KPSS: 3.5247620942521523

Valor p: 0.01

Valores críticos: {'10%': 0.347, '5%': 0.463, '2.5%': 0.574, '1%': 0.739}

La serie es estacionaria.

--------------------------------

--------------------------------

BARRA FLORIDA

Estadístico KPSS: 1.4560520749754136

Valor p: 0.01

Valores críticos: {'10%': 0.347, '5%': 0.463, '2.5%': 0.574, '1%': 0.739}

La serie es estacionaria.

--------------------------------

--------------------------------

BARRA LOSALME

Estadístico KPSS: 0.8370722283844385

Valor p: 0.01

Valores críticos: {'10%': 0.347, '5%': 0.463, '2.5%': 0.574, '1%': 0.739}

La serie es estacionaria.

Dando a ver que, en la mayoría de los casos, p-valor es 0.01, dando por final que en todas las series son estacionarias.

## Gráficos dados por ARIMA, valores MSE-MAE y estacionario/no estacionario en dataset sin outliers:

En AJAHUEL, para predecir 2022 se obtuvo el siguiente gráfico:

Gráfico, Gráfico de líneas

Descripción generada automáticamente

Da un resultado similar a la referencia de la imagen 29 con el mismo desbalance entre 2022-03 a 2022-06

A la vez, se obtuvo los siguientes valores:

MAE 8.297201255604781

MSE 206.70822250295922

AIC 356712.28298433626

En BUIN, para predecir 2022 se obtuvo el siguiente gráfico:

Gráfico, Histograma

Descripción generada automáticamente

En comparación con el dataset con outliers este si pudo ajustarse mejor a los valores reales aunque este tenga muchos desbalance en distinta horario

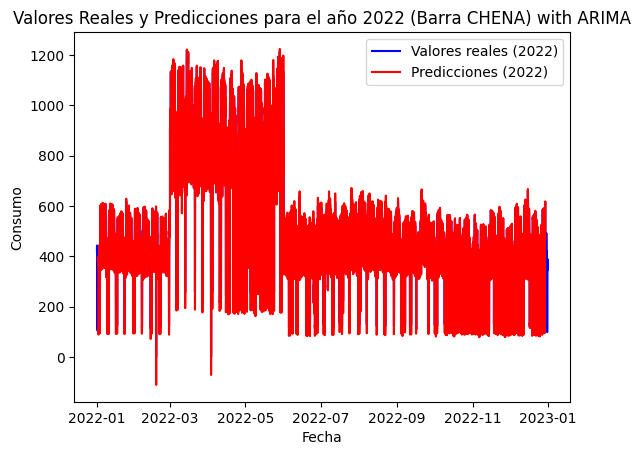
A la vez, se obtuvo los siguientes valores:

MAE 20.281763076291867

MSE 2690.13135331007

AIC 468862.83658613433

En CHENA, para predecir 2022 se obtuvo el siguiente gráfico:



Igual que el anterior con el dataset con outliers este tiene un desbalance entre 2022-03 a 2022-06 pero igual se ajusta bien a los valores reales

A la vez, se obtuvo los siguientes valores:

MAE 28.677905213181383

MSE 3082.606762608241

AIC 474776.63603023964

En CNAVIA, para predecir 2022 se obtuvo el siguiente gráfico:

Gráfico, Histograma

Descripción generada automáticamente

Igual que el anterior con el dataset con outliers este tiene un desbalance entre 2022-03 a 2022-06 pero igual se ajusta bien a los valores reales

A la vez, se obtuvo los siguientes valores:

MAE 20.754829447741425

MSE 1271.623770474885

AIC 436058.5589394837

En ELSALTO, para predecir 2022 se obtuvo el siguiente gráfico:

Gráfico, Gráfico de líneas

Descripción generada automáticamente

Igual que el anterior con el dataset con outliers este tiene un desbalance entre 2022-03 a 2022-06 pero igual se ajusta bien a los valores reales

A la vez, se obtuvo los siguientes valores:

MAE 20.876823846181058

MSE 1378.5956163559124

AIC 439645.13883211475

En FLORIDA, para predecir 2022 se obtuvo el siguiente gráfico:

Gráfico, Histograma

Descripción generada automáticamente

Igual que el anterior con el dataset con outliers este tiene un desbalance entre 2022-03 a 2022-06 pero igual se ajusta bien a los valores reales

A la vez, se obtuvo los siguientes valores:

MAE 1.6953630055791638

MSE 22.012023812044585

AIC 258982.35080836833

En LOSALME, para predecir 2022 se obtuvo el siguiente gráfico:

Gráfico

Descripción generada automáticamente

Igual que el anterior con el dataset con outliers este tiene un desbalance entre 2022-03 a 2022-06 pero igual se ajusta bien a los valores reales

A la vez, se obtuvo los siguientes valores:

MAE 8.264317675812723

MSE 205.0145630240507

AIC 356201.3821182895

Podemos decir que las subestaciones FLORIDA y AJAHUEL tuvieron mejores resultados en MAE y MSE para este modelo de ARIMA.

Sobre el análisis de las series estacionarias/no estacionarias usando el test KPSS:

BARRA AJAHUEL

Estadístico KPSS: 0.9794760789016098

Valor p: 0.01

Valores críticos: {'10%': 0.347, '5%': 0.463, '2.5%': 0.574, '1%': 0.739}

La serie es estacionaria.

--------------------------------

--------------------------------

BARRA BUIN

Estadístico KPSS: 0.8747997965945181

Valor p: 0.01

Valores críticos: {'10%': 0.347, '5%': 0.463, '2.5%': 0.574, '1%': 0.739}

La serie es estacionaria.

--------------------------------

--------------------------------

BARRA CHENA

Estadístico KPSS: 2.152343848551646

Valor p: 0.01

Valores críticos: {'10%': 0.347, '5%': 0.463, '2.5%': 0.574, '1%': 0.739}

La serie es estacionaria.

--------------------------------

--------------------------------

BARRA CNAVIA

Estadístico KPSS: 4.292372256555857

Valor p: 0.01

Valores críticos: {'10%': 0.347, '5%': 0.463, '2.5%': 0.574, '1%': 0.739}

La serie es estacionaria.

--------------------------------

--------------------------------

BARRA ELSALTO

Estadístico KPSS: 0.9528920240212954

Valor p: 0.01

Valores críticos: {'10%': 0.347, '5%': 0.463, '2.5%': 0.574, '1%': 0.739}

La serie es estacionaria.

--------------------------------

--------------------------------

BARRA FLORIDA

Estadístico KPSS: 1.2942786316813313

Valor p: 0.01

Valores críticos: {'10%': 0.347, '5%': 0.463, '2.5%': 0.574, '1%': 0.739}

La serie es estacionaria.

--------------------------------

--------------------------------

BARRA LOSALME

Estadístico KPSS: 4.612052789120463

Valor p: 0.01

Valores críticos: {'10%': 0.347, '5%': 0.463, '2.5%': 0.574, '1%': 0.739}

La serie es estacionaria.

--------------------------------

Todas las subestaciones son estacionarias con un valor p 0.01

## Arquitectura algorítmica y matemática de RNN con Adam (modelo con mención especial) con datos y gráficos.

### Análisis matemático.

Una red neuronal recurrente (RNN) tiene una o más capas de entrada, ocultas y una de salida. La particularidad del RNN que las capas ocultas tienen conexiones de retroalimentación (O séase, la salida de “alimenta” de nuevo a la misma capa, a medida que avanza el tiempo).

En RNN, se toma una entrada y una salida anterior para producir una nueva salida , mediante esa fórmula:

Donde es la matriz de peso para las conexiones de retroalimentación, es la matriz de pesos para las conexiones de entrada, es el vector de sesgo, es la función de activación, donde , y son mis parámetros de RNN.

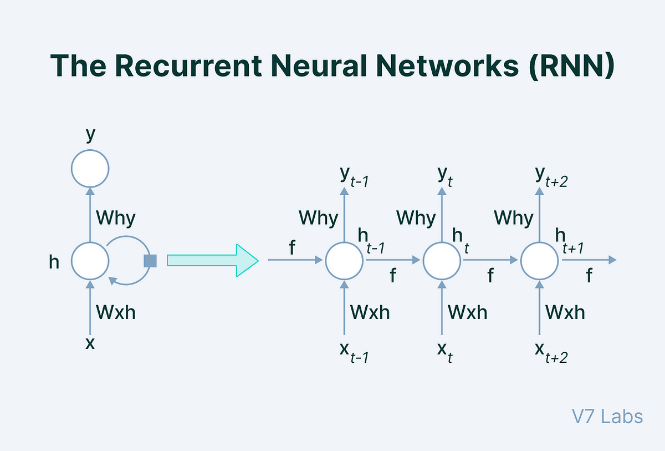


Imagen 37: Referencia de una RNN, con los parámetros explicados antes matemáticamente.

Dentro del código, son los elementos de entrada que correspondería a un elemento en la secuencia de entrada, las secuencias de entrada que yacen en las listas yw. Sobre , y son parámetros internos de la capa SimpleRNN que se aprende durante el entrenamiento, eso es manejado automáticamente por Keras. Sobre es la función de activación que lleva SimpleRNN, la cual por defecto usa la función de activación que en este caso usamos el activation = ‘linear’. Sobre representa el estado oculto en un tiempo , lo cual sirve para usarse en cómo se va a “alimentar” la secuencia de entrada a la RNN, eso ocurre usando model. compile() mientras que corresponde a la capa actual de salida usando SIMPLERNN, que con el cálculo visto anterior (función) usando los parámetros vistos, se calcula convirtiéndose luego en en el siguiente paso de tiempo.

### Parámetros orientados a la arquitectura de RNN.

Los parámetros en este informe que estarán presentes serán sobre:

* Entrada: Datos/Información que le entrego a RNN.
* Codificación: Formato a los datos para que sea recibido por RNN.
* Capas ocultas: Capas que yacen entre la(s) capa(s) de entrada y la de salida.
* Número de neuronas: Cantidad de neuronas que yacen en RNN.
* Dimensión Capa salida: Cantidad de neuronas que yacen en la capa de salida de RNN.
* Función de activación: Que función se utilizó para el aprendizaje y trabajo de RNN.
* Función de costo: Función para analizar el trabajo del modelo, como funciona el modelo, y sobre análisis de errores y como disminuir el error.

### MSE Y MAE, ¿Qué son?

También a la vez, se tomará en cuenta demasiado acerca de los siguientes errores que se puede encontrar:

* MSE: Error cuadrático medio, medida de error que se utiliza para comparar las estimaciones o predicciones de un modelo con los valores reales usando un cuadrado, se calcula como:
* MAE: Error absoluto medio, medida de error que se utiliza para comparar las estimaciones o predicciones de un modelo con los valores reales usando valor absoluto, se calcula como:

Acá es la cantidad total observaciones, es el valor real dado observación , e el valor predicho dado observación .

SE utilizará las dos medidas de error, lo cual será importante para a futuro, entender la elección del mejor modelo (ARIMA VS RNN).

## Gráficos dados por RNN, valores MSE-MAE y gráficos epoch vs loss en dataset con outliers:

Para AJAHUEL se tuvo los siguientes resultados:

Gráfico, Gráfico de líneas

Descripción generada automáticamente

Imagen 38: Gráfico predicciones vs valores reales de AJAHUEL con RNN

Veremos que tiene un cierto parecido en el comportamiento con el gráfico de AJAHUEL tomado con ARIMA.

Los valores MAE y MSE son:

* Final MAE: 17.747880935668945
* Final MSE: 1421.915771484375

Sobre el gráfico de epoch vs loss tomado para AJAHUEL, gracias a tensorboard:

Gráfico, Gráfico de líneas

Descripción generada automáticamente

Imagen 39: Gráfico epoch vs loss en AJAHUEL

A partir desde la época 10 el error empieza a bajar muy despacio lo cual la mejor época para detenerse es entre el 10 y 13

Nuestro elemento es:

* Entrada: la entrada que se le entrego fue una secuencia de tiempo con un batch de 30.
* Codificación: entra en formato float32.
* Capas ocultas: solo tiene una capa oculta de tipo rnn simple.
* Número de neuronas: 40.
* Dimensión Capa salida: 1.
* Función de activación: lineal.
* Función de costo: mse

Para BUIN se tuvo los siguientes resultados:

Gráfico

Descripción generada automáticamente

Imagen 40: Gráfico predicciones vs valores reales en BUIN con RNN

Acá, sin embargo, se puede apreciar un gráfico con demasiado desbalance y desajuste entre valores reales y las predicciones, por ende, por si no es un gráfico de confianza.

Los valores MAE y MSE son:

* Final MAE: 1.0272787809371948
* Final MSE: 5.204549789428711

Y sobre el gráfico de epoch vs loss obtenemos que, gracias a tensorboard:

Gráfico, Gráfico de líneas

Descripción generada automáticamente

Imagen 41: Gráfico epoch vs loss de BUIN

En esta imagen se puede apreciar que nuestra mejor época con el mejor valor posible para mse y mae fue la del 17

Nuestro elemento es:

* Entrada: la entrada que se le entrego fue una secuencia de tiempo con un batch de 30.
* Codificación: entra en formato float32.
* Capas ocultas: solo tiene una capa oculta de tipo rnn simple.
* Número de neuronas: 40.
* Dimensión Capa salida: 1.
* Función de activación: lineal.
* Función de costo: mse

Para CHENA se obtuvieron los siguientes resultados:

Gráfico

Descripción generada automáticamente

Imagen 42: Gráfico predicciones vs valores reales en CHENA con RNN

Apreciamos al igual que en ARIMA, un buen seguimiento de patrón en el gráfico.

Los valores MAE y MSE son:

* Final MAE: 36.221317291259766
* Final MSE: 5499.35888671875

Y sobre el gráfico de epoch vs loss obtenemos que, gracias a tensorboard:

Gráfico

Descripción generada automáticamente con confianza media

Imagen 43: Gráfico epoch vs loss para CHENA

Se puede apreciar que su error no disminuye mucho después de la 4 época

Nuestro elemento es:

* Entrada: la entrada que se le entrego fue una secuencia de tiempo con un batch de 30.
* Codificación: entra en formato float32.
* Capas ocultas: solo tiene una capa oculta de tipo RNN simple.
* Número de neuronas: 40.
* Dimensión Capa salida: 1.
* Función de activación: lineal.
* Función de costo: MSE

Para CNAVIA se obtuvieron los siguientes resultados:

Gráfico, Gráfico de líneas

Descripción generada automáticamente

Imagen 44: Gráfico predicciones vs valores reales en CNAVIA con RNN

Vemos comportamiento tal parecido con ARIMA.

Los valores MAE y MSE son:

* Final MAE: 74.43292236328125
* Final MSE: 22694.056640625

Y sobre el gráfico de epoch vs loss obtenemos que, gracias a tensorboard:

Gráfico, Gráfico de líneas

Descripción generada automáticamente

Imagen 45: Gráfico epoch vs loss para CNAVIA

En este caso fue desde la época 5 que dejo de disminuir drásticamente su error

Nuestro elemento es:

* Entrada: la entrada que se le entrego fue una secuencia de tiempo con un batch de 30.
* Codificación: entra en formato float32.
* Capas ocultas: solo tiene una capa oculta de tipo rnn simple.
* Número de neuronas: 40.
* Dimensión Capa salida: 1.
* Función de activación: lineal.
* Función de costo: mse

Para ELSALTO se obtuvieron los siguientes resultados:

Gráfico, Histograma

Descripción generada automáticamenteVemos comportamiento tal parecido con ARIMA.

Los valores MAE y MSE son:

* Final MAE: 87.69651794433594
* Final MSE: 29617.1015625

Y sobre el gráfico de epoch vs loss obtenemos que, gracias a tensorboard:

Gráfico, Gráfico de líneas

Descripción generada automáticamente

No hay mucha mejora respecto al inicio y final que sigue casi una recta horizontal

Nuestro elemento es:

* Entrada: la entrada que se le entrego fue una secuencia de tiempo con un batch de 30.
* Codificación: entra en formato float32.
* Capas ocultas: solo tiene una capa oculta de tipo RNN simple.
* Número de neuronas: 40.
* Dimensión Capa salida: 1.
* Función de activación: lineal.
* Función de costo: mse

Para FLORIDA se obtuvieron los siguientes resultados:

Gráfico, Gráfico de líneas, Histograma

Descripción generada automáticamente

Vemos comportamiento tal parecido con ARIMA en este en particular se ajusta mejor que las otras subestaciones.

Los valores MAE y MSE son:

* Final MAE: 4.054068565368652
* Final MSE: 92.0399169921875

Y sobre el gráfico de epoch vs loss obtenemos que, gracias a tensorboard:

Gráfico, Gráfico de líneas

Descripción generada automáticamente

Acá se puede apreciar que el error en el entrenamiento disminuye, pero no disminuye para la validación

Nuestro elemento es:

* Entrada: la entrada que se le entrego fue una secuencia de tiempo con un batch de 30.
* Codificación: entra en formato float32.
* Capas ocultas: solo tiene una capa oculta de tipo RNN simple.
* Número de neuronas: 40.
* Dimensión Capa salida: 1.
* Función de activación: lineal.
* Función de costo: MSE

Para LOSALME se obtuvieron los siguientes resultados:

Gráfico

Descripción generada automáticamente

Vemos comportamiento tal parecido con ARIMA pero al igual que los otros no ajusta bien entre los 1500-3000 de la muestra.

Los valores MAE y MSE son:

* Final MAE: 13.163113594055176
* Final MSE: 636.6898803710938

Y sobre el gráfico de epoch vs loss obtenemos que, gracias a tensorboard:

Gráfico, Gráfico de líneas

Descripción generada automáticamente

No hay mucha mejora después de la época 3

Nuestro elemento es:

* Entrada: la entrada que se le entrego fue una secuencia de tiempo con un batch de 30.
* Codificación: entra en formato float32.
* Capas ocultas: solo tiene una capa oculta de tipo rnn simple.
* Número de neuronas: 40.
* Dimensión Capa salida: 1.
* Función de activación: lineal.
* Función de costo: mse

## Gráficos dados por RNN, valores MSE-MAE y gráficos epoch vs loss en dataset sin outliers:

Para AJAHUEL se tuvo los siguientes resultados:

Gráfico, Gráfico de líneas

Descripción generada automáticamente

Imagen 47: Gráfico predicciones vs valores reales de AJAHUEL con RNN

Veremos que tiene un cierto parecido en el comportamiento con el gráfico de AJAHUEL tomado con RNN con outliers.

Los valores MAE y MSE son:

* Final MAE: 17.461612701416016
* Final MSE: 1447.413818359375

Sobre el gráfico de epoch vs loss tomado para AJAHUEL, gracias a tensorboard:

Gráfico, Gráfico de dispersión

Descripción generada automáticamente

Imagen 48: Gráfico epoch vs loss en AJAHUEL

Acá se puede observar que desde el 10 en adelante no mejora mucho su error lo cual quiere decir que alcanza su mejor punto en la época 10

Nuestro elemento es:

* Entrada: la entrada que se le entrego fue una secuencia de tiempo con un batch de 30.
* Codificación: entra en formato float32.
* Capas ocultas: solo tiene una capa oculta de tipo rnn simple.
* Número de neuronas: 40.
* Dimensión Capa salida: 1.
* Función de activación: lineal.
* Función de costo: mse

Para BUIN se tuvo los siguientes resultados:

Gráfico, Histograma

Descripción generada automáticamente

Imagen 49: Gráfico predicciones vs valores reales en BUIN con RNN

Acá, sin embargo, se puede apreciar un gráfico con demasiado desbalance y desajuste entre valores reales y las predicciones, por ende, por si no es un gráfico de confianza.

Los valores MAE y MSE son:

* Final MAE: 1.0272787809371948
* Final MSE: 5.204549789428711

Y sobre el gráfico de epoch vs loss obtenemos que, gracias a tensorboard:

Gráfico

Descripción generada automáticamente con confianza baja

Imagen 50: Gráfico epoch vs loss de BUIN

No tiene muchas mejoras desde el principio hasta el final

Nuestro elemento es:

* Entrada: la entrada que se le entrego fue una secuencia de tiempo con un batch de 30.
* Codificación: entra en formato float32.
* Capas ocultas: solo tiene una capa oculta de tipo rnn simple.
* Número de neuronas: 40.
* Dimensión Capa salida: 1.
* Función de activación: lineal.
* Función de costo: mse

Para CHENA se obtuvieron los siguientes resultados:

Gráfico, Histograma

Descripción generada automáticamente

Imagen 51: Gráfico predicciones vs valores reales en CHENA con RNN

Apreciamos al igual que en ARIMA, un buen seguimiento de patrón en el gráfico excepto entre el rango 1500-3000.

Los valores MAE y MSE son:

* Final MAE: 97.60198211669922
* Final MSE: 32242.763671875

Y sobre el gráfico de epoch vs loss obtenemos que, gracias a tensorboard:

Gráfico, Gráfico de líneas, Gráfico de dispersión

Descripción generada automáticamente

Imagen 52: Gráfico epoch vs loss para CHENA

El error de entrenamiento deja de disminuir a partir de la época 3 pero su error en el entrenamiento disminuye mucho hasta la época 5

Nuestro elemento es:

* Entrada: la entrada que se le entrego fue una secuencia de tiempo con un batch de 30.
* Codificación: entra en formato float32.
* Capas ocultas: solo tiene una capa oculta de tipo rnn simple.
* Número de neuronas: 40.
* Dimensión Capa salida: 1.
* Función de activación: lineal.
* Función de costo: mse

Para CNAVIA se obtuvieron los siguientes resultados:

Gráfico, Histograma

Descripción generada automáticamente

Imagen 53: Gráfico predicciones vs valores reales en CNAVIA con RNN

En este caso a comparación de las otras predicciones de rnn este se desajusta demasiado entre el rango 1500-3000 sin poder seguir el patron.

Los valores MAE y MSE son:

* Final MAE: 88.85162353515625
* Final MSE: 30633.515625

Y sobre el gráfico de epoch vs loss obtenemos que, gracias a tensorboard:

Gráfico

Descripción generada automáticamente con confianza media

Imagen 54: Gráfico epoch vs loss para CNAVIA

Su mejora en el error de entrenamiento de CNAVIA es mínima, en cambio su error de validación oscila un poco

Nuestro elemento es:

* Entrada: la entrada que se le entrego fue una secuencia de tiempo con un batch de 30.
* Codificación: entra en formato float32.
* Capas ocultas: solo tiene una capa oculta de tipo RNN simple.
* Número de neuronas: 40.
* Dimensión Capa salida: 1.
* Función de activación: lineal.
* Función de costo: mse

Para ELSALTO se obtuvieron los siguientes resultados:

Gráfico

Descripción generada automáticamente

Mismo fenómeno que CNAVIA pero aún más evidente el desajuste en el mismo rango.

Los valores MAE y MSE son:

* Final MAE: 77.2448501586914
* Final MSE: 21839.796875

Y sobre el gráfico de epoch vs loss obtenemos que, gracias a tensorboard:

Gráfico, Gráfico de líneas

Descripción generada automáticamente

Al igual que el anterior este no tiene mucha mejora en el error del entrenamiento, pero desde la época 32 se puede apreciar como baja drásticamente su error de la validación

Nuestro elemento es:

* Entrada: la entrada que se le entrego fue una secuencia de tiempo con un batch de 30.
* Codificación: entra en formato float32.
* Capas ocultas: solo tiene una capa oculta de tipo rnn simple.
* Número de neuronas: 40.
* Dimensión Capa salida: 1.
* Función de activación: lineal.
* Función de costo: mse

Para FLORIDA se obtuvieron los siguientes resultados:

Gráfico, Histograma

Descripción generada automáticamente

Vemos comportamiento tal parecido con ARIMA en este en particular se ajusta mejor que las otras subestaciones.

Los valores MAE y MSE son:

* Final MAE: 3.005073308944702
* Final MSE: 54.2440299987793

Y sobre el gráfico de epoch vs loss obtenemos que, gracias a tensorboard:

Gráfico, Gráfico de líneas

Descripción generada automáticamente

El error del entrenamiento deja de crecer a partir de la época 6, además su error dé la validación empieza a oscilar a medida que pasa la épocas

Nuestro elemento es:

* Entrada: la entrada que se le entrego fue una secuencia de tiempo con un batch de 30.
* Codificación: entra en formato float32.
* Capas ocultas: solo tiene una capa oculta de tipo rnn simple.
* Número de neuronas: 40.
* Dimensión Capa salida: 1.
* Función de activación: lineal.
* Función de costo: mse

Para LOSALME se obtuvieron los siguientes resultados:

Gráfico, Histograma

Descripción generada automáticamente

Misma observación con el mismo modelo usando datos outliers.

Los valores MAE y MSE son:

* Final MAE: 14.747123718261719
* Final MSE: 859.4706420898438

Y sobre el gráfico de epoch vs loss obtenemos que, gracias a tensorboard:

Gráfico, Gráfico de líneas

Descripción generada automáticamente

Se puede apreciar que el error de la validación empieza a crecer desde que empieza el entrenamiento, pero por el 32 o 33 se puede apreciar que llega un poco mas bajo que el error inicial que teníamos, pero este vuelve a crecer después.

Nuestro elemento es:

* Entrada: la entrada que se le entrego fue una secuencia de tiempo con un batch de 30.
* Codificación: entra en formato float32.
* Capas ocultas: solo tiene una capa oculta de tipo RNN simple.
* Número de neuronas: 40.
* Dimensión Capa salida: 1.
* Función de activación: lineal.
* Función de costo: mse

# Explicación de cómo se llegó al modelo ganador (ARIMA) con gráficos:

Para hacer la explicación del modelo ganador (que como ya se mencionó, es ARIMA) se hizo el análisis gracias al uso de MSE y de MAE:

A menor MSE y menor MAE entre los modelos, significa que el modelo sería el que tiene una mejor predicción en comparación con otros modelos o con una versión anterior del mismo modelo. Para recordar un poco de conceptos:

- Un MSE menor indica que las predicciones del modelo tienden a estar más cerca de los valores reales, ya que el MSE mide el promedio de los cuadrados de las diferencias entre las predicciones del modelo y los valores reales.

- Un MAE menor indica que las predicciones del modelo tienen errores absolutos más pequeños, es decir, las diferencias absolutas entre las predicciones y los valores reales son menores

Entonces veremos (si bien ARIMA es el modelo ganador, se mencionará su comparación con RNN con ADAM, y a la vez, pondremos una comparación implícita con MLP (perceptrón multicapa).

Si bien donde obtuvimos mejores resultados fue en AJAHUEL y en FLORIDA dado los valores de MSE y MAE, consideraremos FLORIDA como mejor barra (nombre subestación) para entrenamiento y testing, debido a que se consideró que el comportamiento del gráfico tanto con ARIMA como con RNN blinda más confianza que con AJAHUEL.

A la vez si bien MLP no consideramos para informe, si igual está hecho en el código, entonces será mencionado implícitamente, para comparación de cual es mejor modelo, entre “MLP vs ARIMA vs RNN”.

## Comparación con outliers.

En FLORIDA dado dataset con outliers, obtuvimos esos valores con MLP, ARIMA y RNN:

mse\_mlp = 1072.09

mse\_arima = 22.012

mse\_rnn = 92.04

mae\_mlp = 16.99

mae\_arima = 1.69

mae\_rnn = 4.05

mape\_mlp = 224.03

mape\_arima = 1.12

mape\_rnn = 1.35

Donde si apreciamos, también calculado los valores de MAPE también, vemos que el más conveniente entonces es utilizando ARIMA, en donde si vemos los gráficos de comparación utilizando gráfico de barras:

Gráfico, Gráfico de barras

Descripción generada automáticamente

Imagen 55: Comparación rendimiento modelos usando MSE

Gráfico, Gráfico de barras

Descripción generada automáticamente

Imagen 56: Comparación rendimiento modelos usando MAE

Apreciamos que ARIMA explícitamente tiene mejor resultado por bajo MAE y bajo MSE.

## Comparación sin outliers.

Lo mismo analizado sin outliers, si vemos los valores obtenidos en MLP vs ARIMA vs RNN:

mse\_mlp = 1106.525

mse\_arima = 22.012

mse\_rnn = 54.244

mae\_mlp = 17.517

mae\_arima = 1.695

mae\_rnn = 3.005

mape\_mlp = 255.696

mape\_arima = 1.124

mape\_rnn = 1.158

También apreciamos que ARIMA tiene o da menor valores de MSE, MAE y MAPE.

Gráfico

Descripción generada automáticamente

Imagen 57: Comparación rendimiento modelos usando MSE

Gráfico, Gráfico de barras

Descripción generada automáticamente

Imagen 58: Comparación rendimiento modelos usando MAE

Apreciamos lo mismo, ARIMA siendo modelo con menos errores tanto en MAE como en MSE, con FLORIDA con mejor opción para entrenamiento y testing, si se desea ver los valores de MAE y de MSE completos, pueden ver en [Descripción de la arquitectura del modelo utilizado](#_Descripción_de_la).

# Conclusiones respecto al trabajo, limitaciones del modelo y propuestas acerca de cómo mejorar su desempeño en el futuro.

## Conclusiones:

El análisis comparativo entre modelos de predicción, particularmente ARIMA y RNN con el optimizador Adam, proporciona una visión detallada sobre la capacidad de estos modelos para predecir la demanda de energía en subestaciones eléctricas. Los resultados destacan la superioridad general de ARIMA sobre RNN en términos de precisión y estabilidad, especialmente evidenciada en la subestación FLORIDA. Esta superioridad se refleja en los valores más bajos de MSE y MAE obtenidos con ARIMA en comparación con RNN, tanto en conjuntos de datos con outliers como sin ellos.

El rendimiento superior de ARIMA sugiere su utilidad como modelo de referencia para la predicción de demanda de energía en subestaciones eléctricas. Sin embargo, es importante tener en cuenta las limitaciones y consideraciones asociadas con el modelo ARIMA, así como las áreas de mejora potencial para futuros estudios y aplicaciones.

## Limitaciones del Modelo:

A pesar de su efectividad, el modelo ARIMA presenta ciertas limitaciones que pueden afectar su utilidad y aplicabilidad en entornos del mundo real. Estas limitaciones incluyen:

Sensibilidad a la calidad de los datos: El rendimiento del modelo ARIMA puede verse afectado por la calidad de los datos de entrada, incluida la presencia de outliers, datos faltantes o errores de medición. La falta de datos históricos completos y precisos puede limitar la capacidad del modelo para capturar patrones temporales significativos y realizar predicciones precisas.

Dependencia de la selección de hiperparámetros: La eficacia del modelo ARIMA está influenciada por la selección adecuada de los parámetros p, d y q, que determinan la cantidad de autocorrelación y estacionalidad que se tiene en cuenta en el modelo. La selección manual de estos parámetros puede ser subjetiva y requerir experiencia en el dominio, mientras que los enfoques automáticos de selección de hiperparámetros pueden ser computacionalmente intensivos y no siempre garantizan la mejor configuración.

Incapacidad para capturar patrones complejos: ARIMA se basa en supuestos lineales y estacionarios sobre los datos, lo que puede limitar su capacidad para capturar patrones no lineales o complejos presentes en algunos conjuntos de datos. Esto puede conducir a predicciones inexactas o sesgadas, especialmente en entornos con comportamientos dinámicos y variables.

## Propuestas para Mejorar el Desempeño Futuro:

Para abordar estas limitaciones y mejorar el rendimiento del modelo ARIMA en el futuro, se pueden considerar las siguientes propuestas:

Investigación de modelos híbridos: Explorar la posibilidad de combinar las fortalezas de diferentes modelos, como ARIMA y redes neuronales, en un enfoque híbrido. Este enfoque podría aprovechar las capacidades de modelado de tendencias y estacionalidad de ARIMA, mientras se utilizan técnicas de aprendizaje profundo para capturar patrones más complejos y no lineales en los datos. La combinación de modelos podría permitir una mejor adaptación a la complejidad y variabilidad de los datos de demanda de energía.

Mejora de la calidad de los datos: Realizar una limpieza y preprocesamiento adecuados de los datos para mejorar el rendimiento del modelo. Esto implica identificar y corregir anomalías, outliers y datos faltantes antes de entrenar el modelo, lo que puede mejorar significativamente la precisión de las predicciones. Además, la recopilación de datos adicionales y la mejora de la frecuencia y la granularidad de los datos pueden proporcionar información adicional para el modelado y la predicción.

Optimización de hiperparámetros: Explorar el uso de técnicas automatizadas de optimización de hiperparámetros para encontrar los valores óptimos de p, d y q en ARIMA, así como otros parámetros en modelos de aprendizaje profundo. Esto puede ayudar a mejorar la precisión y la generalización del modelo, asegurando que esté bien ajustado a los datos. La utilización de enfoques de optimización avanzados, como la optimización bayesiana o la búsqueda en malla, puede ayudar a explorar de manera más eficiente el espacio de hiperparámetros y encontrar configuraciones óptimas.

Evaluación continua del rendimiento del modelo: Realizar una evaluación regular del rendimiento del modelo utilizando métricas adecuadas y validar el modelo con datos nuevos y actualizados. Esto permitirá identificar cualquier degradación en el rendimiento y tomar medidas correctivas según sea necesario, asegurando que el modelo sea robusto y confiable en diferentes escenarios. Además, la implementación de técnicas de monitoreo en tiempo real y la retroalimentación continua pueden ayudar a mantener el modelo actualizado y adaptado a cambios en los datos y las condiciones del entorno.

Exploración de modelos alternativos: Además de ARIMA y RNN, explorar otros modelos de predicción, como modelos de espacio de estado, modelos de aprendizaje profundo más avanzados (por ejemplo, LSTM o redes neuronales convolucionales), o enfoques basados en técnicas de inteligencia artificial, como redes neuronales generativas adversarias (GAN), para evaluar su idoneidad y rendimiento en la predicción de demanda de energía en subestaciones eléctricas. La diversificación de enfoques puede proporcionar ideas nuevas y perspectivas adicionales sobre la modelización y predicción de series temporales en este dominio.

## Resumen de conclusión, limitaciones y propuestas:

En resumen, aunque el modelo ARIMA ha demostrado ser una herramienta efectiva para la predicción de demanda de energía en subestaciones eléctricas, es importante reconocer sus limitaciones y explorar estrategias para mejorar su desempeño en el futuro. La combinación de enfoques tradicionales como ARIMA con técnicas más avanzadas de aprendizaje automático y la adopción de prácticas de preprocesamiento de datos rigurosas pueden ayudar a superar algunas de las limitaciones identificadas y mejorar la precisión y robustez del modelo.

Además, la investigación continua en este campo es fundamental para mantenerse al tanto de los avances en modelos de predicción y técnicas de análisis de datos, y para adaptar estas innovaciones a las necesidades específicas del sector eléctrico. La colaboración entre investigadores, profesionales de la industria y responsables de políticas también puede contribuir a la identificación de desafíos clave y soluciones efectivas para la gestión de la demanda de energía y la planificación de la infraestructura eléctrica.

En última instancia, la aplicación exitosa de modelos de predicción como ARIMA en entornos del mundo real depende no solo de la calidad y el rendimiento del modelo en sí, sino también de la integración efectiva con sistemas de gestión de energía y la capacidad de los usuarios para interpretar y actuar sobre las predicciones generadas. Al abordar las limitaciones del modelo y explorar nuevas oportunidades para mejorar su desempeño, podemos avanzar hacia una gestión más eficiente y sostenible de la demanda de energía eléctrica, contribuyendo así a la construcción de un futuro energético más resiliente y sostenible.

Principio del formulario

# BIBLIOGRAFÍA Y MATERIAL DE AYUDA:

* Código máquina. (2023, 26 de junio): Análisis Exploratorio de Series de Tiempo con Gráficas de Autocorrelación y Retardo usando Python (Video), YouTube: <https://www.youtube.com/watch?v=U-vdLHdyDLI>
* Ritvikmath. (2020, 20 de marzo): ACF & PACF code example: Time Series Talk (Video), YouTube: <https://www.youtube.com/watch?v=y8opUEd05Dg>
* Schwarzenberg, Pablo: (2023): Repositorio de GitHub (CINF104): <https://github.com/pabloschwarzenberg/CINF104>
* HackersRealm. (2023, 11 de julio): Detect and Remove Outliers in the Data| Machine Learning| Python (Página educativa): <https://www.hackersrealm.net/post/detect-and-remove-outliers-python>
* SuperDataScience Team, María Santos, Juan Gabriel Molina. (noviembre 2023): Machine Learning de A a la Z: R y Python para Data Science (Udemy): [https://www.udemy.com/course/machinelearning-es/learn/lecture/14334826?start=60#overview](https://www.udemy.com/course/machinelearning-es/learn/lecture/14334826?start=60%23overview)
* Santiago Hernández. (): Inteligencia Artificial y Deep Learning desde cero en Python (Udemy): [https://www.udemy.com/course/deep-learning-desde-cero-en-python/learn/lecture/23889650?start=150#overview](https://www.udemy.com/course/deep-learning-desde-cero-en-python/learn/lecture/23889650?start=150%23overview)
* Sebastián Maldonado (2011, 21 de diciembre): Preprocesamiento de Datos (pptx): <https://www.u-cursos.cl/diplomados/2011/2/DIP78J/1/material_docente/bajar?id_material=401889>
* OpenAI. (s.f.). ChatGPT (GPT-3.5). OpenAI: <https://openai.com/chatgpt>
* Joanby (2023, agosto): Repositorio de GitHub (Deep Learning de A a la Z: Redes Neuronales al Completo): <https://github.com/joanby/deeplearning-az>
* BitBoss (2020, 27 de febrero): Redes neuronales: De la neurona al perceptrón multicapa en 9 minutos. El problema XOR (Video), YouTube: <https://www.youtube.com/watch?v=MU3cLsSfnME&t=280s>
* InformatiCódigo (2022, 4 de agosto): SimpleRNN, GRU y LSTM| Redes neuronales RECURRENTES Python Keras 2022 (Video), YouTube:

<https://www.youtube.com/watch?v=bGcxrZMlJ2M>

* Joaquín Amat Rodrigo, Javier Escobar Ortiz (2023, septiembre): Modelos ARIMA y SARIMAX con Python (Página educativa): <https://cienciadedatos.net/documentos/py51-modelos-arima-sarimax-python>

# Nuestro GITHUB:

* El github donde se encuentra todos los archivos se encuentra en el siguiente enlace:
* <https://github.com/FenixCompany/Tarea-1-CINF-104>