Práctica Métodos Clásicos para la Predicción

(AI Generated)

UNIVERSIDAD POLITÉCNICA DE MADRID

ETSI SISTEMAS INFORMÁTICOS

Realizado por:

Alonso Ruiz Palomo

Juan Moreno Segura

Índice

PROBLEMA 1: PROTEGIENDO LA ETSISI DE ATAQUES DE RED

- Estudio cualitativo del dataset.
- Estudio cuantitativo del dataset.
- Aplicar reducción de dimensionalidad para visualizar los datos.
- Encontrar la reducción que consigue una mejor clasificación de las trazas de la red evaluando con distintos modelos de predicción.
- Visualización final de los resultados con las distintas reducciones.

PROBLEMA 2: MEJORANDO LA EFICIENCIA ENERGÉTICA EN LA UPM

- Estudio cualitativo y cuantitativo del dataset.
- Visualizaciones del consumo por campus y por bloques (días, semanas, meses y años).
- Modelos de predicción de los últimos 6 meses.
- Análisis de series temporales individualizado por cada bloque.

PROBLEMA 1: PROTEGIENDO LA ETSISI DE ATAQUES DE RED

- Estudio cualitativo del dataset:

El dataset contiene datos sobre ataques de red a la ETSISI, contiene una columna label que especifica el tipo de ataque producido, que será nuestra variable a predecir. Podemos encontrar distintos tipos de DDoS, DoS, ataques del malware mirai, VulnerabilityScan, injecciones de comandos y sql, ataques de fuerza bruta por medio de un diccionario y un largo etcetera.

Entre el resto de características encontramos varias que representan los distintos protocolos de red (IRC, Telnet, DHCP, HTTP, HTTPS…). Estas indican la presencia o actividad de estos protocolos en una conexión particular. Otras columnas como ack_flag_number, psh_flag_number, syn_count, fin_flag_number están asociadas a los flujos TCP/IP y sus características. Las columnas de métricas como Rate, Tot sum, Srate, Drate, Duration, y flow_duration indican métricas de rendimiento en los flujos de datos de la red.

Tenemos información de más de 700000 ataques con 47 características cada uno, por lo que tendremos que quedarnos con aquellos relevantes para las predicciones. Además, todas nuestras columnas son de tipo float64, menos label que es de tipo object, la tendremos que categorizar.

Tenemos 34 valores distintos para nuestra label. Son demasiados, así que vamos a hacer una agrupación para reducir este número. Para ello, todas las etiquetas que empiezan por DDoS se agrupan, al igual que las DoS, Mirai, Recon y demás. Vamos a juntar “Malware” y “Backdoor” en “Malware”, ya que un ataque Backdoor es un tipo de Malware. El resto de etiquetas que tienen muy pocos valores, los meteremos en una etiqueta “Otro”. Así, hemos reducido a 12 etiquetas distintas, un número más razonable para hacer las predicciones.

- Estudio cuantitativo del dataset:

No tenemos valores nulos en el dataset. Dividimos el dataframe en las características y en la label a predecir.

Vamos a tratar las columnas. Tenemos 46 columnas en nuestro dataframe con las características que utilizaremos para entrenar. Como tenemos alguna columna con un único valor, las eliminamos ya que no aportan ningún tipo de información. Hemos reducido a 41 columnas.

Hacemos una matriz de correlaciones para ver cómo de relacionadas están las distintas características. Tenemos características muy correlacionadas positiva o negativamente, esto significa que cuando los valores de una columna crecen, también crecerán los de aquellas que están muy correladas positivamente con ella, y decrecerán las correladas

muy negativamente con la dicha. Por ello, no nos interesa quedarnos con estas columnas ya que tienen información redundante, así que eliminaremos una de cada par de las que tengan un coeficiente de correlación mayor a 0.9 en valor absoluto. Vemos que ejemplos de columnas muy relacionadas directamente son la desviación típica con el máximo y el radio, que cuando una crece, las demás también, o el rate con el srate. Relacionadas inversamente tenemos IPv con ARP porque cuando una es True, la otra es False, o LLC con ARP, que ocurre lo mismo. Con esta reducción, ahora tenemos 31 columnas.

Por último, para la reducción de columnas vamos a aplicar un VarianceThreshold de sklearn para eliminar aquellas columnas que tengan una varianza demasiado baja, poniendo el umbral en 0.01. Los datos no varían demasiado para estas columnas por lo que no aportarán demasiada información. Finalmente, tenemos 25 columnas con las que haremos reducción de dimensionalidad.

- Aplicar reducción de dimensionalidad para visualizar los datos.

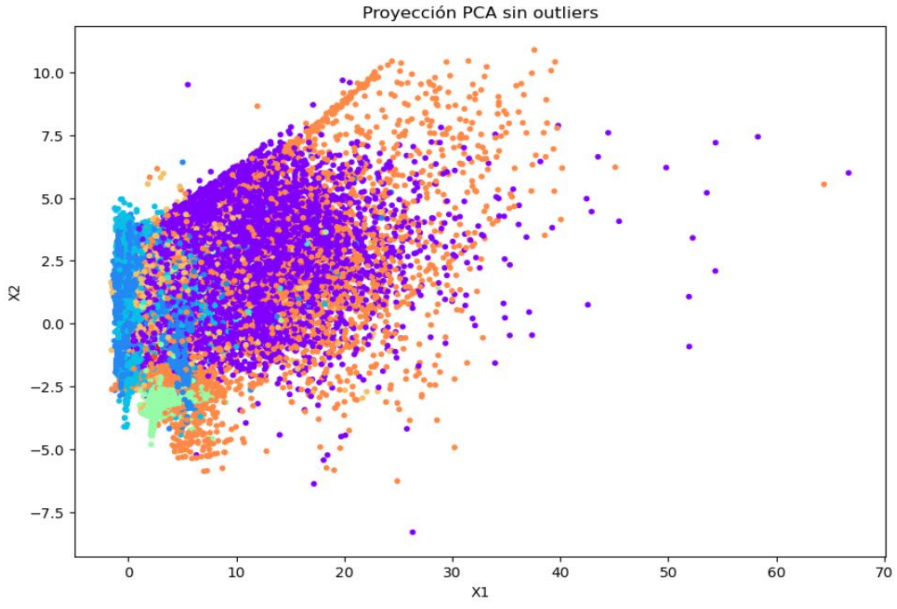
Primero de todo vamos a estandarizar nuestros datos para poder aplicar reducción de dimensionalidad. Para ello aplicaremos un StandardScaler de sklearn en nuestro dataframe con las características a entrenar.

Para nuestras labels, aplicaremos un LabelEncoder para poder pasar de tipo object a numérico, convirtiendo las clases en números enteros.

A continuación, vamos a aplicar los distintos métodos de reducción de dimensionalidad vistos en clase y visualizar los datos.

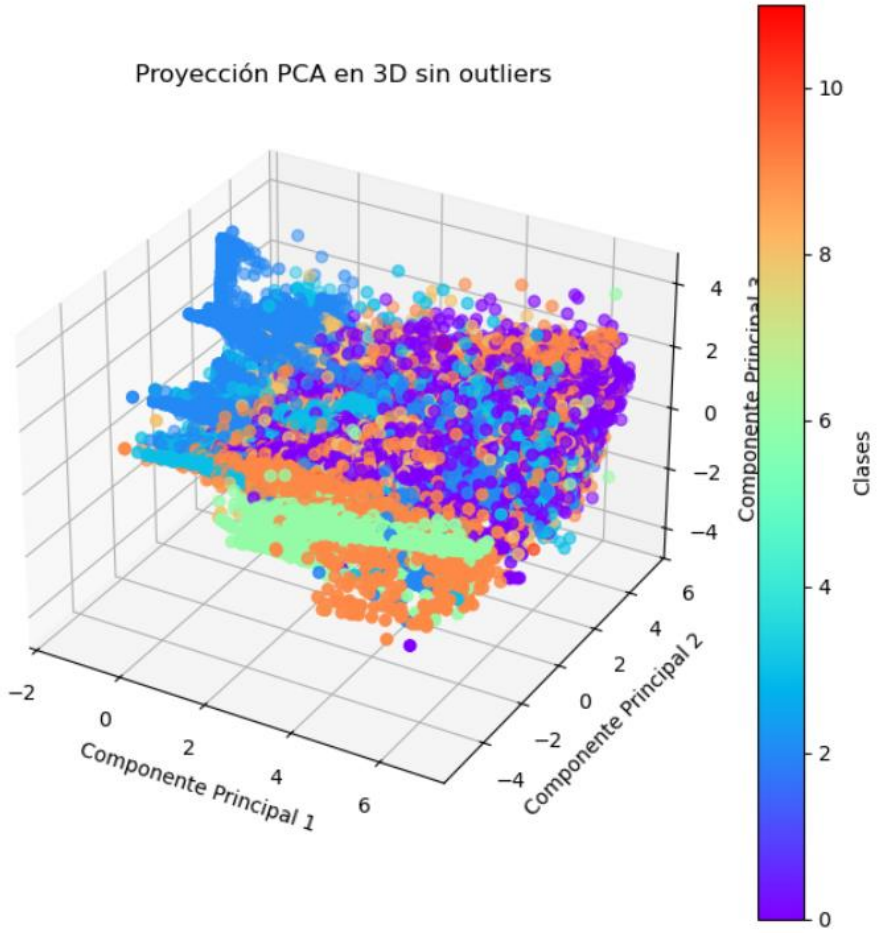
- PCA (Principal Component Analysis):

  Aplicamos PCA para una reducción de los datos en 2 dimensiones y nos quitamos los outliers con una máscara. Resultado (cada clase representada en un color distinto):



Hay clases que si separa bien como la verde o las azules de la izquierda, pero la morada y la naranja están muy mezcladas, por lo que será difícil una buena clasificación.

Ahora vamos a hacer una reducción PCA a 3 dimensiones, eliminando outliers. Resultados:



La clase verde la separa bien, pero las azules se mezclan más con las moradas y naranjas, aunque la naranja está más diferenciada.

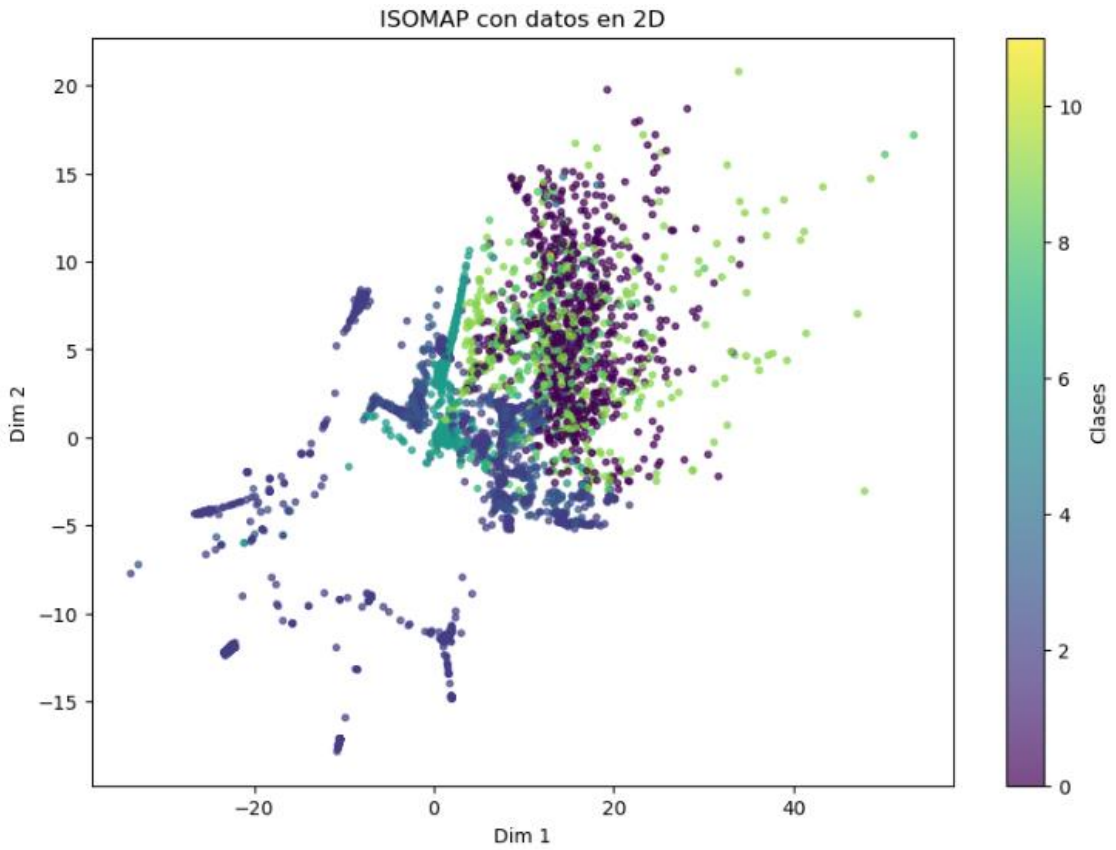
- Truncated – SVD (Singular Value Decomposition):

Haciendo la reducción a 2 y 3 características con Truncated – SVD obtenemos unas visualizaciones muy similares a las de PCA, son prácticamente indistinguibles por lo que no pondremos aquí las visualizaciones.

Esto ocurre porque este es un método que funciona de manera muy similar al PCA en datos centrados, que son los que tenemos. El Truncated SVD se puede ver como una generalización de PCA que también funciona bien con datos dispersos o no centrados.

- ISOMAP:

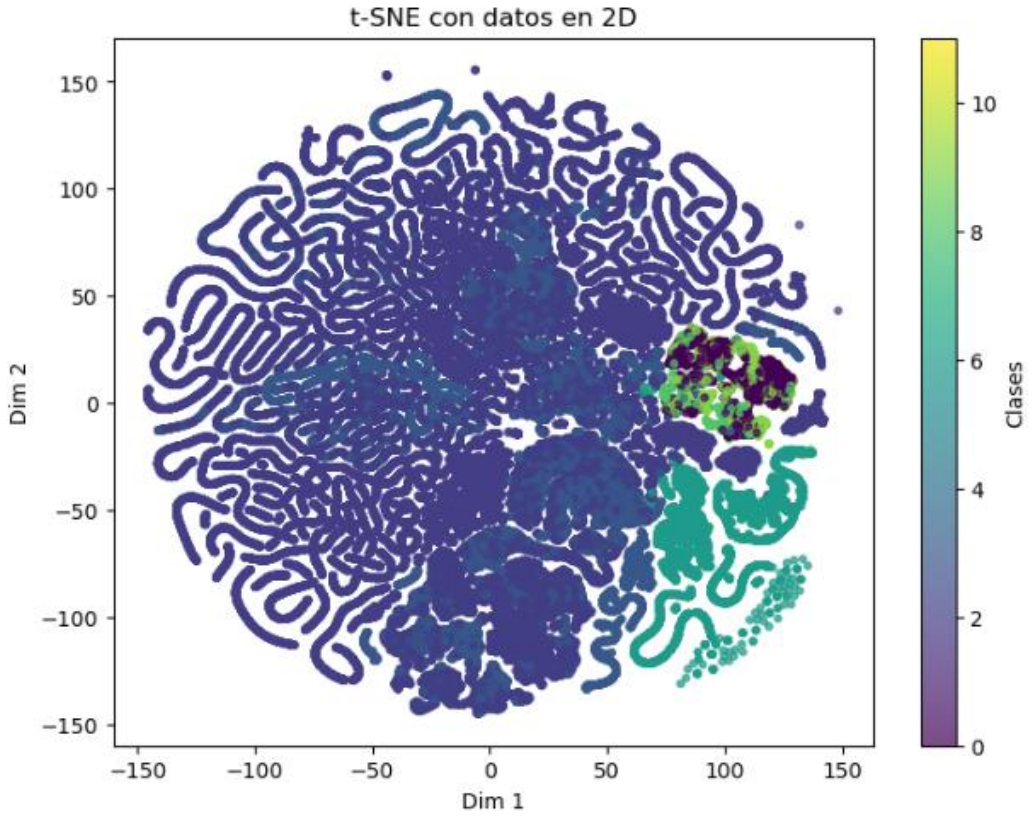
Para la reducción ISOMAP vamos a hacerla con solo el 5% de los datos del dataset debido a que esta es muy costosa. Reducimos a 2 dimensiones con el parámetro n_neighbors igual a 28, lo que indica que, para la reducción, nos fijaremos en los 28 vecinos más cercanos de cada punto. Resultados:



Obtenemos una división de clases que, aparentemente es similar a las anteriores. Tenemos clases que son difíciles de separar como las verdes y moradas, pero otras que sí se diferencian mejor como las azules y azul claro.

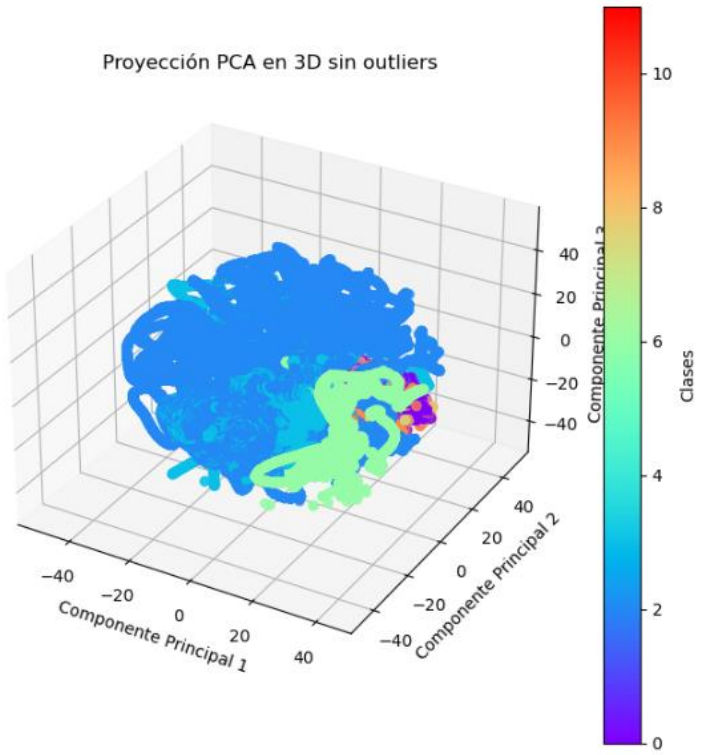
- T-SNE (T-distributed Stochastic Neighbor Embedding):

Esta reducción también es costosa, aunque no tanto como la ISOMAP. Por lo que para la reducción en 2 dimensiones vamos a hacerla con solo el 20% de los datos que tenemos. El parámetro perplexity está con un valor de 30, indicando el número de vecinos cercanos considerados para la reducción de cada punto. Resultados:



Podemos observar la mejor separación de clases, ya que las azules claro están bien separadas. La verde y morada siguen mezcladas, pero están concentradas en una pequeña área. Por último tenemos la clase azul oscuro predominante en datos.

Haciendo reducción t-SNE a 3 Dimensiones obtenemos:



En esta reducción también vemos una buena separación de clases, como la verde. Predomina la azul oscuro y seguimos viendo mezcladas otras como la morada y naranja. Lo bueno de esta reducción, al igual que la de 2 dimensiones, es que este método no crea outliers, los datos están juntos dentro de unos rangos establecidos.

- Encontrar la reducción que consigue una mejor clasificación de las trazas de la red evaluando con distintos modelos de predicción.

Primero de todo, hemos entrenado con el dataset con las 25 características antes de hacer reducción de dimensionalidad para ver qué modelos nos dan mejores resultados. Para ello, entrenamos un RandomForestClassifier, un MLP de sklearn con 2 capas ocultas y un KNN. Decidimos escoger estos modelos debido a que no son demasiado costosos (como una red neuronal muy profunda o un SVM) pero tampoco muy simples (como una regresión lineal o polinomial). Evaluamos los modelos con la métrica f1_score, debido a que es más fiable que el accuracy al ser una mezcla de la precisión y el recall, y tener en cuenta los datos desbalanceados.

Obtenemos el mejor f1 score con el RandomForest (0.90), quedando un poco por debajo el MLP (f1_score = 0.89) y después el KNN (f1_score = 0.88). Por lo que, descartamos el MLP debido a que es más costoso y no obtenemos los mejores resutlados. Por lo que vamos a entrenar nuestros datos con la reducción de dimensionalidad hecha con los siguientes modelos:

- RandomForestClassifier: Debido a su rapidez y facilidad de encontrar relaciones complejas entre los datos. Además, hemos obtenido el que será nuestro mejor f1_score (0.90) entrenando con los datos sin reducción.
- KNN: No hemos obtenido los mejores resultados entrenando con los datos sin reducción, pero tiene sentido que este modelo funcione bien con los datos reducidos a 2 y 3 dimensiones, ya que el objetivo de esto es hacer una mejor separación de las clases. Debido a la naturaleza del KNN de realizar la clasificación fijándonos en las clases de los vecinos cercanos a nuestro dato, este modelo funcionará bien con las reducciones.

Hemos encapsulado el código de ambos modelos en las funciones RandomForest y KNN, donde el primero utiliza 100 árboles (parámetro n_estimators) y luego combina los resultados de clasificación, y el segundo se fijará en los 15 vecinos más cercanos de cada punto para la clasificación (parámetro n_neighbors) y lo hará dando distintos pesos a los vecinos, en función de su distancia al punto a clasificar (parámetro weights).

Tomamos todos los datos con las reducciones de dimensionalidad que hemos mencionado antes hechas, los dividimos en train y test sets (proporción 80-20%) y los entrenamos con ambos modelos. (Resultados en siguiente apartado)

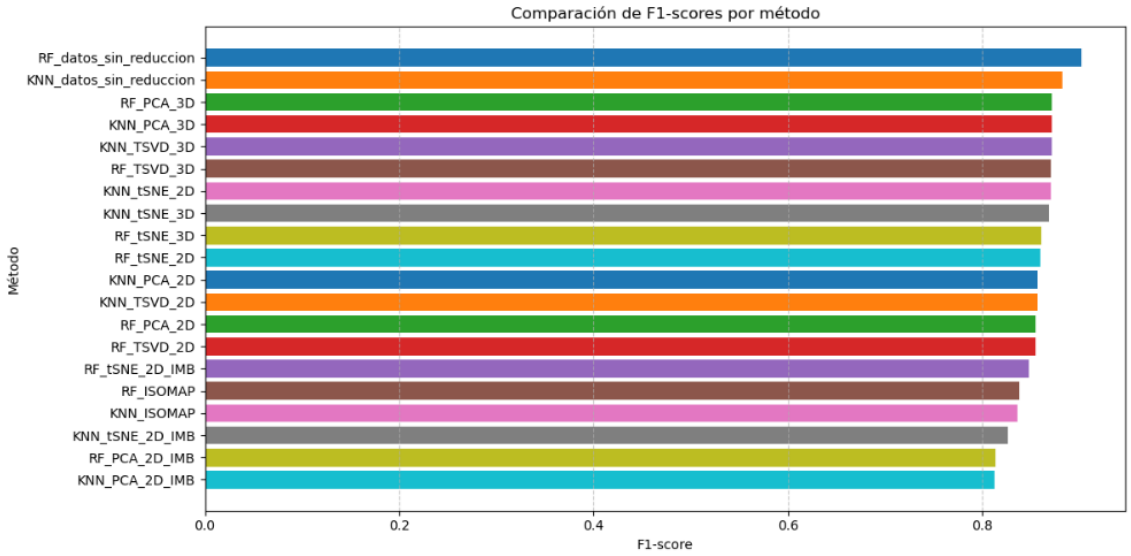
Debido a que los resultados f1 obtenidos han empeorado con reducción de dimensionalidad, pensamos que esto puede darse a que hay datos desbalanceados. Por

ello vamos a aplicar imbalanced learning a 2 conjuntos de datos que han obtenido los mejores f1 scores:

- PCA 3D: aplicamos un RandomUnderSampler para eliminar datos aleatorios de las dos clases mayoritarias y luego aplicamos SMOTE para aumentar las muestras del resto de clases infrarrepresentadas.
- T-SNE 2D: RandomUnderSampler para la clase mayoritaria y RandomOverSampler para aumentar las muestras del resto de clases debido que no podemos utilizar SMOTE puesto que tenemos alguna clase con muy pocas muestras y este método necesita un mínimo de datos para poder hacer el oversampling.

Entrenamos con los 2 modelos mencionados ambos conjuntos de datos tras aplicar imbalanced learning.

- Visualización final de los resultados con las distintas reducciones.



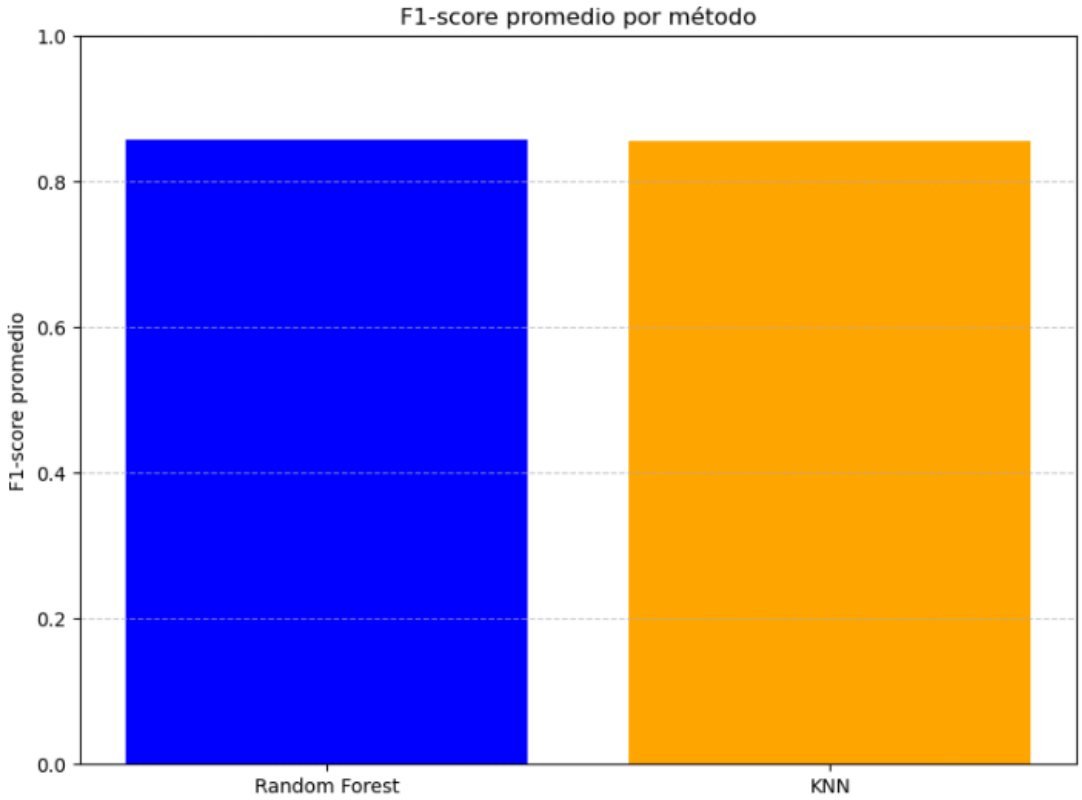
Como podemos ver, la reducción de dimensionalidad en este caso no ha ayudado a obtener una mejor clasificación debido a que nuestros modelos con f1 scores más altos son el RandomForest y KNN con los datos sin ningún tipo de reducción.

Aplicar Imbalanced learning tampoco ha ayudado, puesto que nuestros modelos que entrenan con los datos ya balanceados obtienen los peores resultados. El problema no estaba ahí.

Las reducciones que mejores resultados han obtenido son la PCA a 3 dimensiones al igual que Truncated SVD a 3 dimensiones debido a la gran similitud de estos métodos. La reducción t-SNE a 2 y 3 dimensiones es la siguiente mejor, muy poco por debajo de las anteriores, ya vimos en la visualización que era de las que mejor separaba las clases.

La reducción ISOMAP ha resultado ser bastante mala, además de costosa. Hay que tener en cuenta que nos quedábamos solamente con una pequeña parte de los datos para reducirlos y entrenar.

Antes de ver los resultados, pensábamos que las reducciones a 3 dimensiones iban a ser mejores que las de 2 dimensiones, y que cuantas más dimensiones nos quedáramos de los datos mejores resultados obtendríamos ya que tendríamos más información, pero como podemos ver, esto no ha tenido por qué ser así. Hay algunas reducciones que han resultado mejor evaluar con 2 que con 3 dimensiones, como la del t-SNE.



La comparación entre qué modelo de predicción ha resultado mejor, vemos que el RandomForestClassifier ha clasificado un poco mejor los distintos tipos de ataque de red en general. El KNN también ha obtenido f1 scores promedios muy similares, siendo estos incluso mejores que los del RandomForest en algunas ocasiones.

PROBLEMA 2: MEJORANDO LA EFICIENCIA ENERGÉTICA EN LA UPM

- Estudio cualitativo y cuantitativo del dataset.

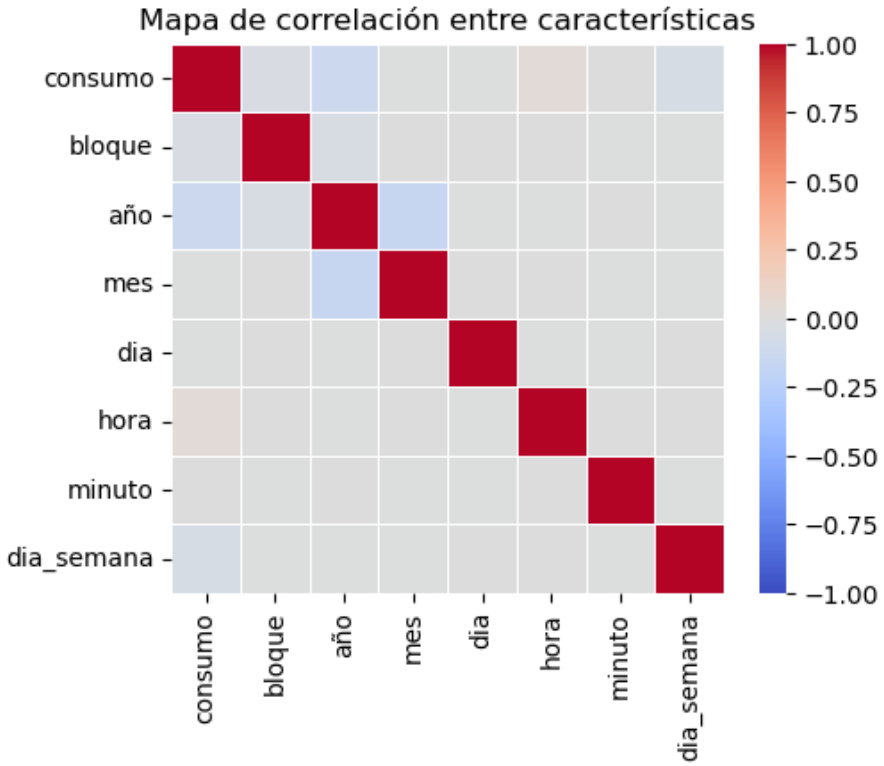
El dataset aunque es extenso en ejemplos (8 millones) presenta muy pocas características (5): Dividimos en objetos y numéricos.

Por una parte, los objetos nos dictan de qué campus, habitación y tanto la fecha como la hora a la que fue hecha la medición. Luego tenemos numerales, que nos dictan la medición de gasto energético (float64) y el bloque del campus en el que se realizaron (int64). Convertimos campus y espacio en categóricos por sus pocos valores únicos y la fecha de medición en datetime.

La columna bloque, aunque sea numérica, tiene solo 4 valores distintos (solo hay 4 bloques). Leyendo el enunciado de la práctica, no vamos a utilizar la columna espacio, por lo que es innecesaria, la eliminamos.

Nos aseguramos que no falten valores y añadimos al dataset 7 columnas que especifican el año, mes, día, hora, minuto, día de la semana y hora:minuto por separado de la fecha_medicion, esto lo usaremos más adelante para visualizar los datos. Nos quedaremos solo con la fecha en fecha_medicion para facilitarnos las cosas más tarde.

La naturaleza de los datos nos dicta que no existen variables altamente correlacionadas



- Visualizaciones del consumo por campus y por bloques (días, semanas, meses y años).

Realizaremos este y todos los estudios posteriores haciendo la diferenciación entre Montegancedo y el campus sur, para ser mas certeros a la hora de dar resultados.

Consumo medio diario de cada bloque por intervalos de 15 minutos: cómo podemos ver en el notebook, tanto en el Campus de Montegancedo como en el Sur, el consumo comienza a elevarse a partir de las 7-8 de la mañana lo cual tiene sentido ya que es cuando empieza la jornada laboral y los empleados, seguidos de los estudiantes comienzan a llegar a la universidad.

El consumo se mantiene elevado durante las horas del día llegando a su pico al medio día y la hora de comer, a partir de las 17:00h comienza a descender el consumo ya que por las tardes parece haber menos gente en las universidades hasta disminuir a mínimos en horarios nocturnos. Hay una clara estacionalidad.

Cabe destacar que en Montegancedo, para el consumo medio por intervalos de 15 minutos, el bloque 1 es el que más consume, seguido del 2, 4 y 3 y en el Campus Sur, en orden descendente de consumición vemos a los bloques 2, 3 y 1.

Consumo medio de cada bloque por cada día de la semana: en este caso vemos que tanto el Campus Montegancedo como el Sur tienen una tendencia de consumo que era de esperar, durante los días laborales este es más elevado y bastante uniforme, en fin de semana desciende.

En el Campus Montegancedo, en orden descendente de consumo medio por día de la semana vemos a los bloques 1, 3, 2 y 4, y en el Campus Sur a los bloques 3, 2 y 1, este orden se mantiene para el resto de las visualizaciones del consumo medio por mes y año.

Consumo medio mensual por cada bloque: en cuanto al consumo mensual, vemos una tendencia de consumo similar en ambos campus, con la diferencia de que en los bloques 1 y 3 del Campus Montegancedo y el 2 del Sur, hay más consumo en el periodo vacacional de verano, lo que nos sorprende, pero imaginamos que es porque se impartirá algún tipo de curso.

Como era de esperar, en periodos vacacionales de Navidad y Semana Santa (Diciembre y Abril) desciende el consumo porque los alumnos no atienden a las aulas, igual que muchos empleados. Meses de Enero y Mayo tenemos un alto consumo por periodo de exámenes ya que los alumnos atienden más a las clases, igual que en Marzo, ya que antes de Semana Santa suele haber parciales. Por lo que hay una tendencia estacional.

Consumo anual por bloque: podemos ver que, en ambos Campus hubo una bajada del consumo en 2020, debido al Covid-19, ya que tanto empleados como alumnos no tenían que acudir a la universidad. En 2021 seguíamos con la recuperación de la pandemia con clases online, por eso también el bajo consumo. Vemos un bajo consumo en ambos

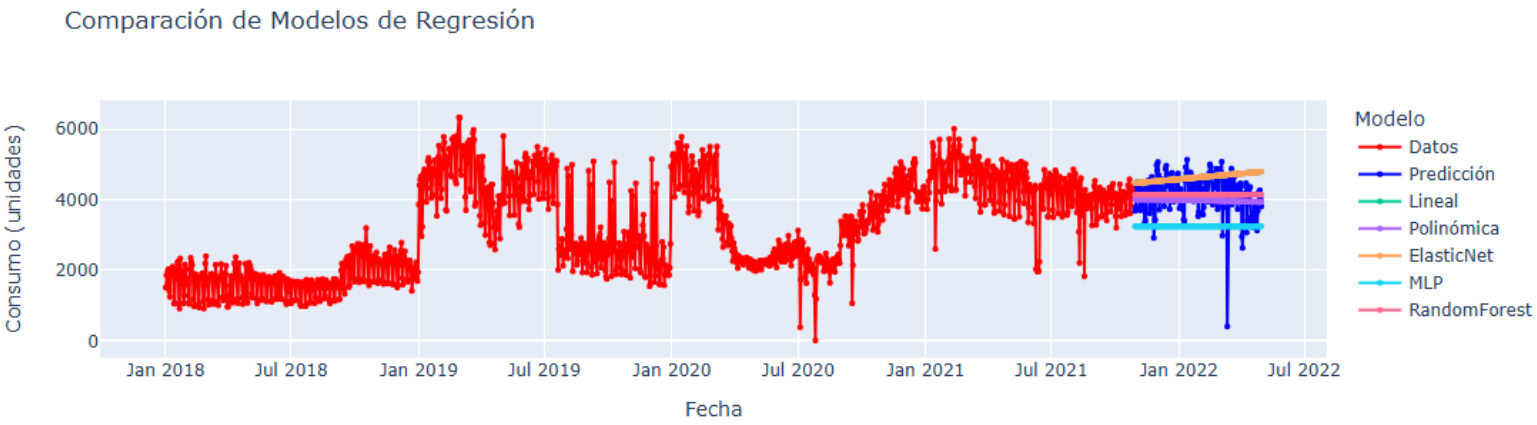
campus en 2022, esto es porque tenemos mucha más información de los años 2018-2021, puesto que del 2022 solo tenemos datos para los primeros 4 meses.

- Modelos de predicción de los últimos 6 meses.

En este apartado, empleamos diversos métodos de predicción como regresiones lineales, polinómicas, redes neuronales y random forests, con el propósito de realizar una predicción aproximada utilizando las fechas y los datos de consumo energético disponibles. Tras entrenar y preparar todos los modelos, obtuvimos los siguientes resultados:

Resultados de MAE (Mean Absolute Error):

- **Regresión Lineal:** 525.14
- **Regresión Polinómica (grado 5):** 461.52
- **ElasticNet:** 525.14
- **Random Forest:** 411.96
- **MLP (Red Neuronal):** 986.98



El único motivo por el que la regresión lineal no aparece, es porque está sobrepuesta a la regresión lineal, pues al parecer ambas llegaron a la misma conclusión.

Como se puede observar, los resultados obtenidos son relativamente pobres, especialmente para la red neuronal, cuyo error es significativamente más alto. Esto sugiere una posible sobrecarga computacional o un mal ajuste debido a la falta de datos relevantes.

Uno de los factores clave en este bajo rendimiento es la naturaleza compleja y errática de los datos. En el contexto de series temporales, es crucial tener en cuenta variables externas que no se incluyeron en nuestro análisis. Por ejemplo, factores como días festivos, huelgas, cambios estacionales, incluso fenómenos globales como pandemias, pueden afectar el consumo de energía de manera impredecible. Además, la naturaleza humana, con sus decisiones y comportamientos cambiantes, agrega un nivel de incertidumbre difícil de modelar con métodos tradicionales de regresión y predicción generalista.

En cuanto a los problemas para obtener mejores resultados y como solucionarlos: estamos usando algoritmos de una índole más general para predecir series temporales, por lo que una posible solución sería usar algoritmos más adecuados. Existen modelos como SARIMAX, los cuales son capaces de capturar patrones complejos al considerar tendencias, estacionalidades y otras dinámicas temporales. Su capacidad para manejar variables dependientes del tiempo y datos secuenciales los hace mucho más efectivos para este tipo de tareas que los métodos utilizados en este análisis.

- Análisis de series temporales individualizado por cada bloque.

Si hablamos de Montegancedo: el bloque uno es el único que no presenta una gran bajada con la aparición de la pandemia, sus datos parecen no variar mucho en general, con picos negativos repentinos probablemente por días festivos. Los bloques del 2 al 3 presentan características similares; con la aparición de la pandemia los datos pasan a tomar valores más negativos, los cuales solamente suben en el bloque 3 cuando esta desaparece. En cuanto a magnitudes, al igual que el bloque 1; presentan picos negativos en dias festivos

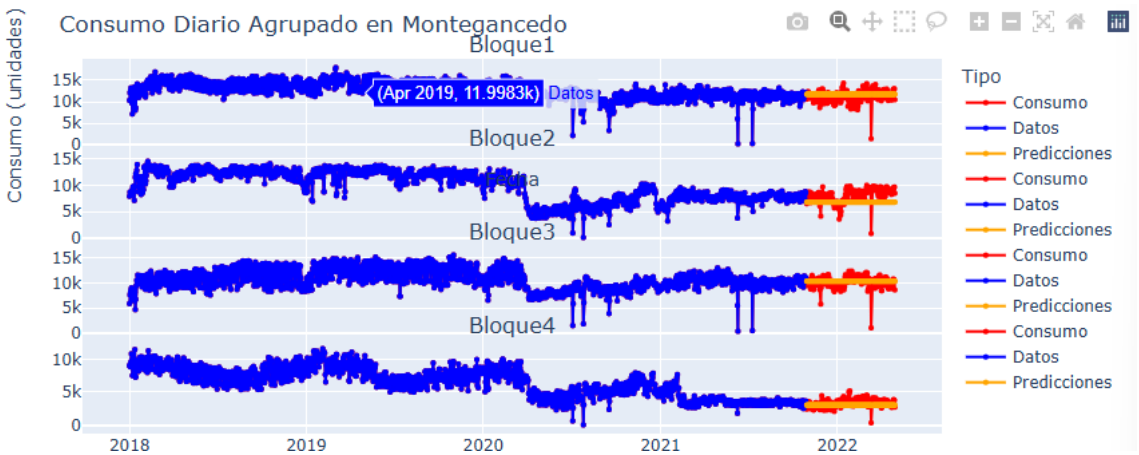
Por otra parte, el campus sur: El bloque 1 presenta datos muy erráticos pre-pandemia, y una recuperación y estabilidad después de la finalización de esta. El bloque 2 presenta una bajada con la aparición del covid que no se recupera al menos hasta el momento de toma de estos datos. El bloque 3 sigue los pasos del bloque 2

Para este apartado emplearemos la misma división; es decir, haremos un estudio para cada campus. Para entender que serie temporal ha comprendido mejor la naturaleza de los datos, hemos creado un diccionario que almacenará la media del error absoluto medio que se generará según se vaya prediciendo.

Le daremos varios enfoques para tratar de garantizar los mejores resultados; entre la lista están: naive-forecasting, regresor simple, exponential smoothing, AR (autoregresion), MA (medias móviles), ARMA (la combinación de ambas con un ruido blanco), ARIMA (integrando para llegar a la estacionalidad) y SARIMAX. Para estos dos últimos, haremos un estudio que nos ayudará a predecir las variables necesarias para que los resultados sean los óptimos.

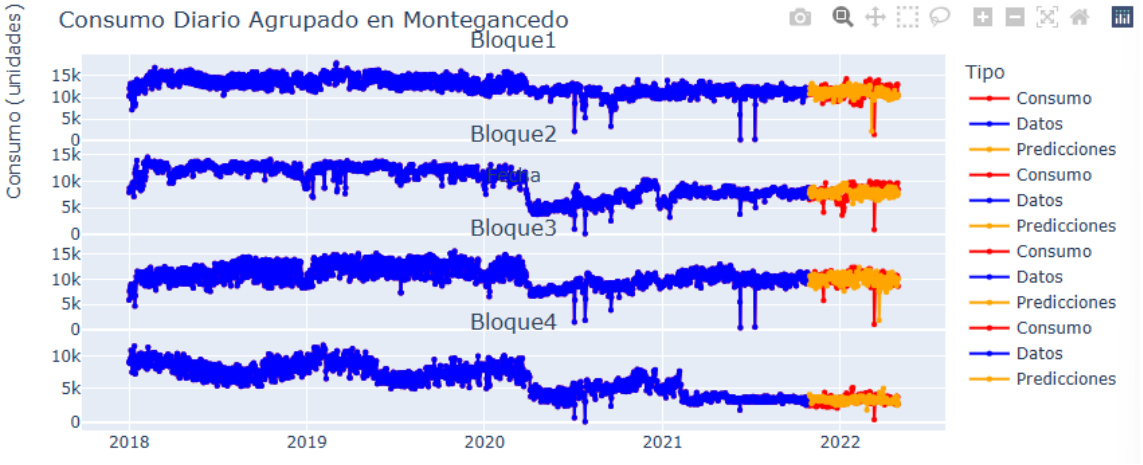
Nuestro objetivo es que se comprenda el funcionamiento de cada algoritmo y se puedan visualizar los resultados; asique adjuntaremos una de las visualizaciones para cada algoritmo para no sobrecargar esta memoria con gráficas.

Naive-forecasting: consiste en usar el último valor de los datos como estimador, por lo cual no es para nada de extrañar que los resultados son pobres



Media de maes: 906.192595904762

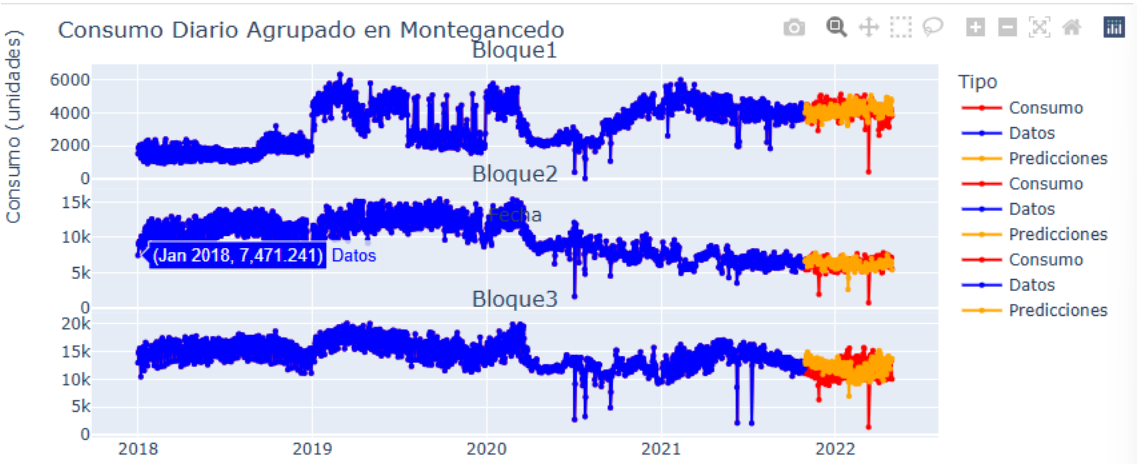
Simple regression: consiste en emplear un modelo sencillo, en nuestro caso hemos elegido un DecisionTreeRegresor, pero se pueden usar otros como un regresor simple. Aún no se esperan grandes resultados pues no usamos uno de los elementos que más información nos brindan, la variante temporal



Media de maes: 801.326218059524

Como era de esperar, el resultado es algo mejor que el anterior, aunque aun le falta afinar mucho para que lo consideremos bueno

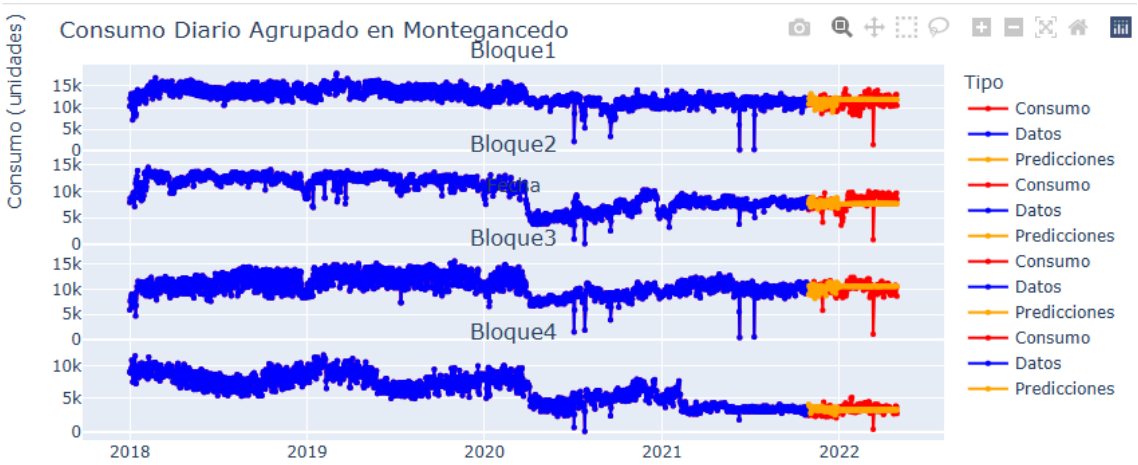
Exponential Smoothing: consiste en calcular una media ponderada de una muestra (cuyo tamaño podemos controlar) de los últimos valores de la serie, como si siguiera



una tendencia.

Media de maes: 671.9188017907173

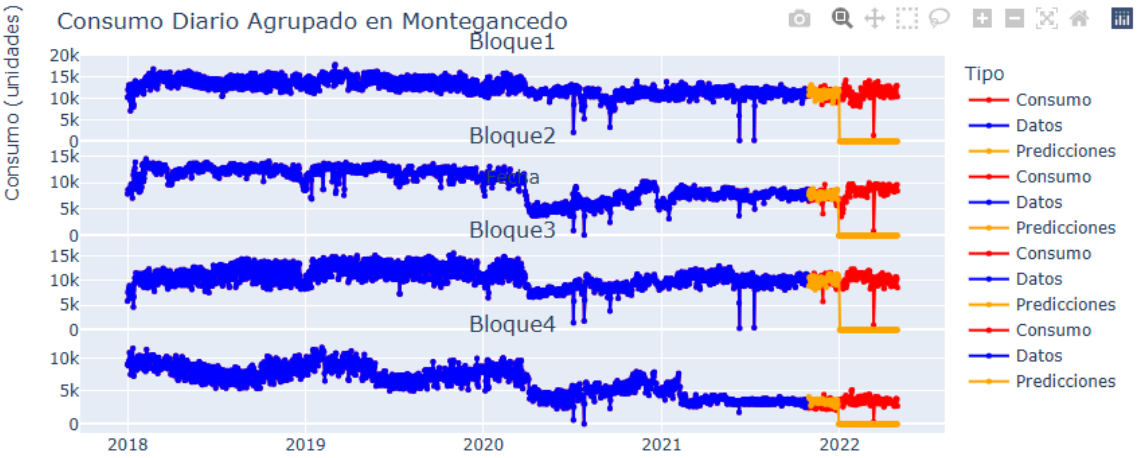
AR: Sigue un concepto parecido al del exponenthial smoothing, el siguiente valor será una suma ponderada al que añadiremos un ruido blanco (término normal aleatorio)



Media de maes: 658.8864395450029

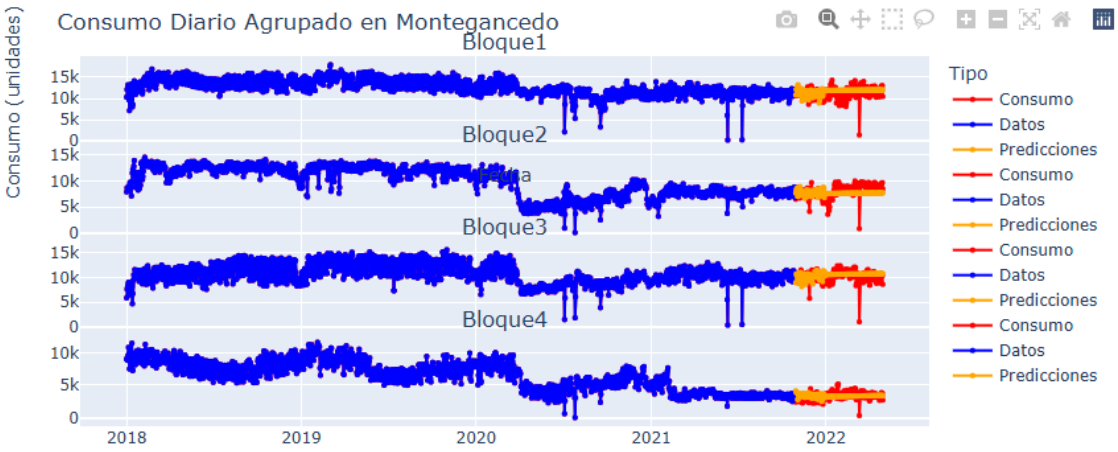
Como veremos más adelante, los datos son algo estacionales y poseen una tendencia, lo cual hace que el modelo AR sea algo mejor que el Exponenthial Smoothing

MA: Usa una combinación de ruidos blancos para predecir el siguiente valor, sin tomar en cuenta los anteriores, lo cual es muy probable que haga de este un modelo pésimo en si mismo



Media de maes: 5918.678745563225

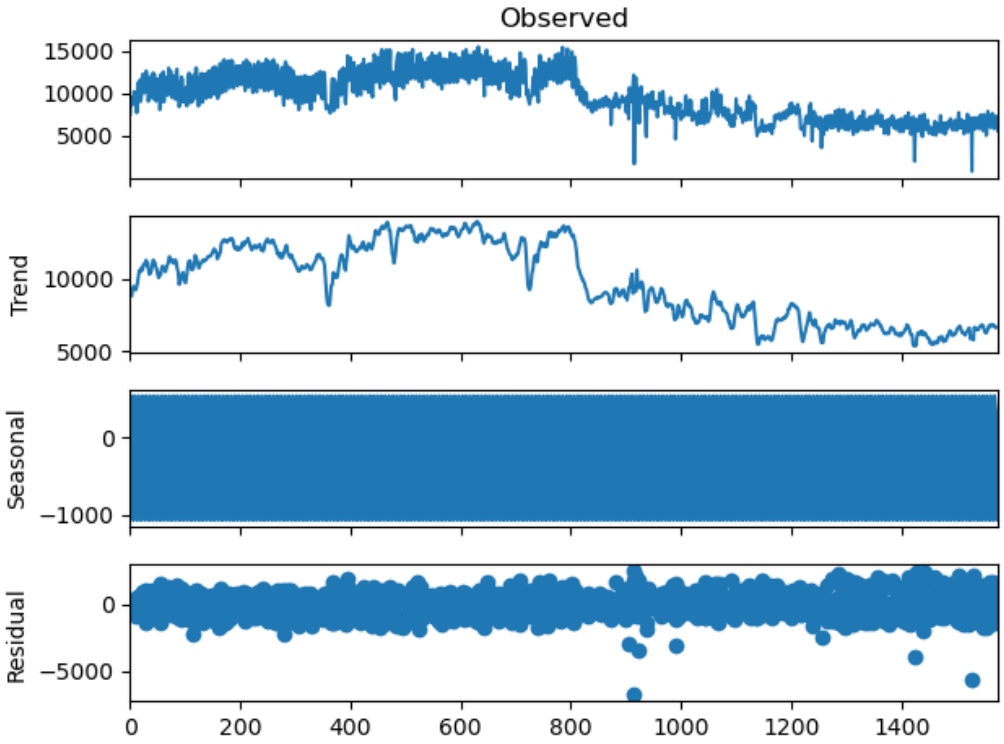
ARMA: Combina los dos últimos modelos para ganar precisión



Media de maes: 656.5076481774306

Antes de terminar con ARIMA y SARIMAX, precisamos información adicional para sacarles el máximo partido a estos modelos tan avanzados. Para esto, estudiaremos la antes mencionada estacionalidad y la estacionariedad de los modelos.

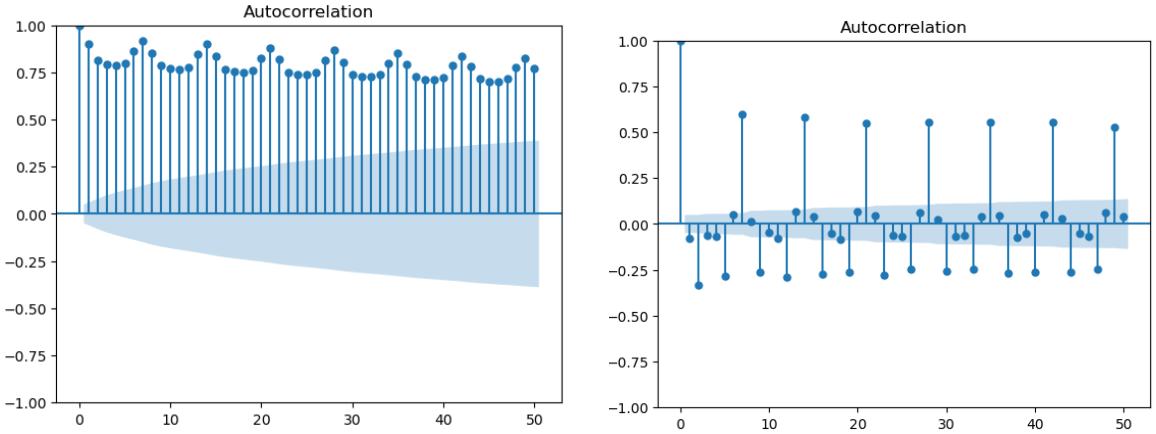
Análisis estacional



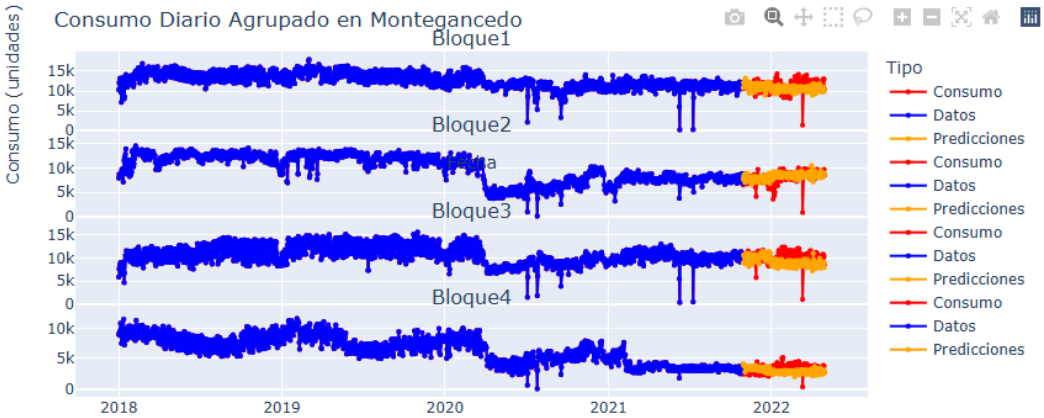
La serie temporal muestra cambios importantes a lo largo del tiempo, con una tendencia general y patrones estacionales bien marcados. El gráfico de tendencia indica una dirección clara que podría modelarse, incluso después de la caída causada por la pandemia de COVID-19. El patrón estacional es repetitivo y mantiene una amplitud constante, lo que sugiere una estacionalidad fuerte y la posibilidad de usar un modelo como SARIMAX para capturarla. Sin embargo, el análisis del residuo muestra que el ruido aumenta en la segunda mitad del periodo (post-COVID), lo que podría hacer que las predicciones futuras sean más difíciles.

Análisis de estacionariedad

En el gráfico de la izquierda tenemos el gráfico con el intervalo de confianza al 95% para estudiar la correlación. A la derecha, el mismo gráfico pero para la función derivada en un primer intento de alcanzar la estacionariedad. Como podemos observar, una única vez nos permite alcanzarla y nos muestra repeticiones cada 7 puntos. Esta información nos da muchas pistas sobre los valores que tendremos que usar como hiperparámetros para los próximos algoritmos

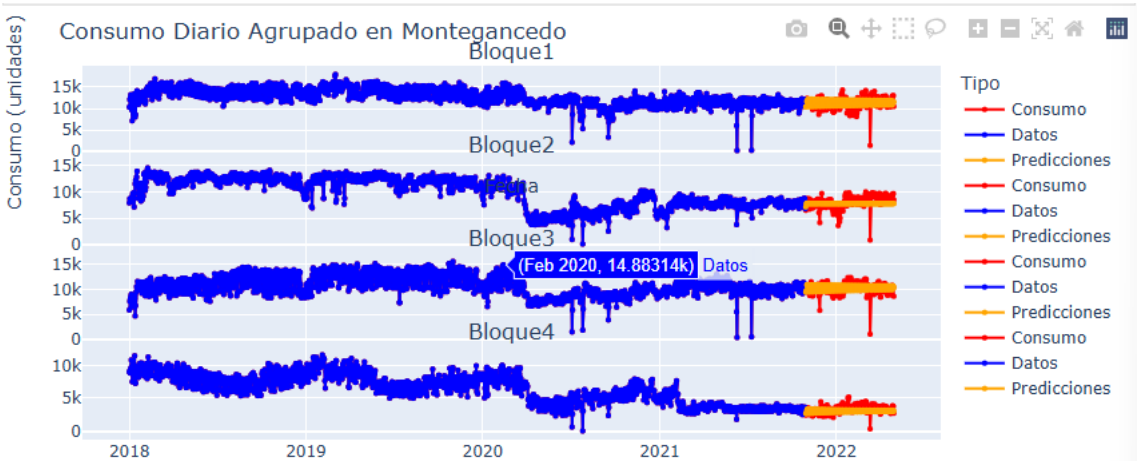


ARIMA: le añade la componente de integración al modelo ARMA para hacer los datos estacionarios

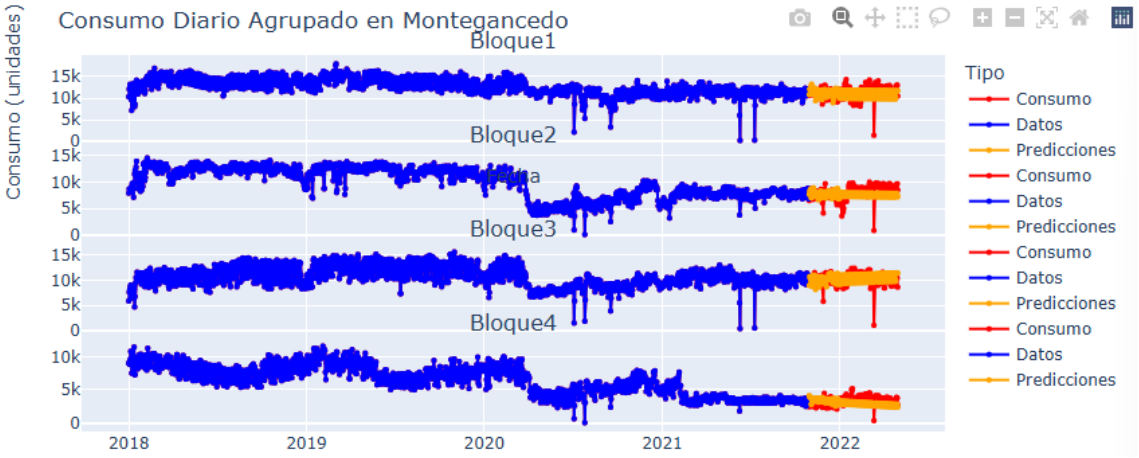


Media de maes: 953.9061323264042

Estos resultados fueron obtenidos con una implementación más rudimentaria. Tras esto recurrimos a la herramienta auto_arima, para que entrenara un modelo para cada bloque de acuerdo a las necesidades de estos (en todos se muestra que la diferenciación debe ser uno, por lo que nuestro análisis previo parece acertado)

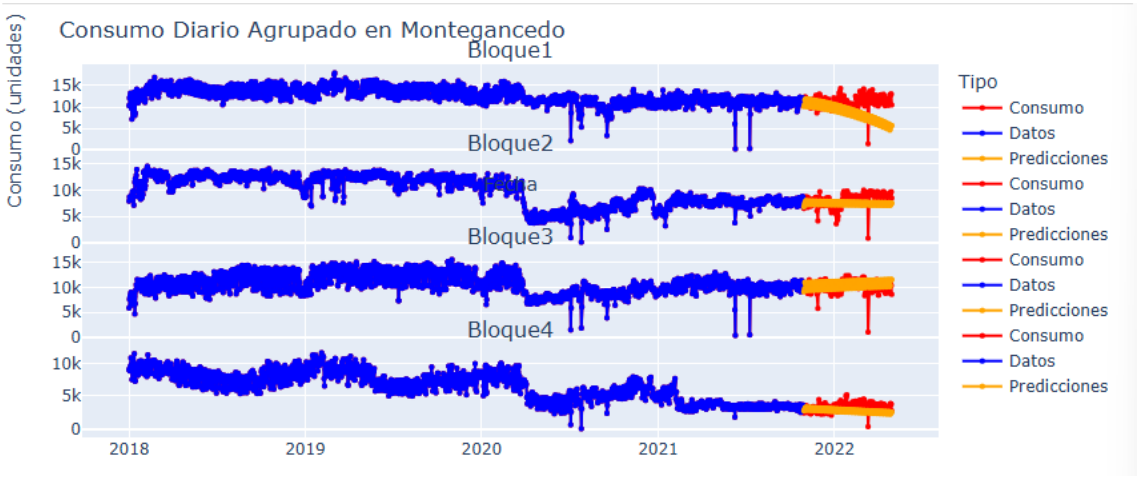


SARIMAX: igual que ARIMA pero añade una componente estacional



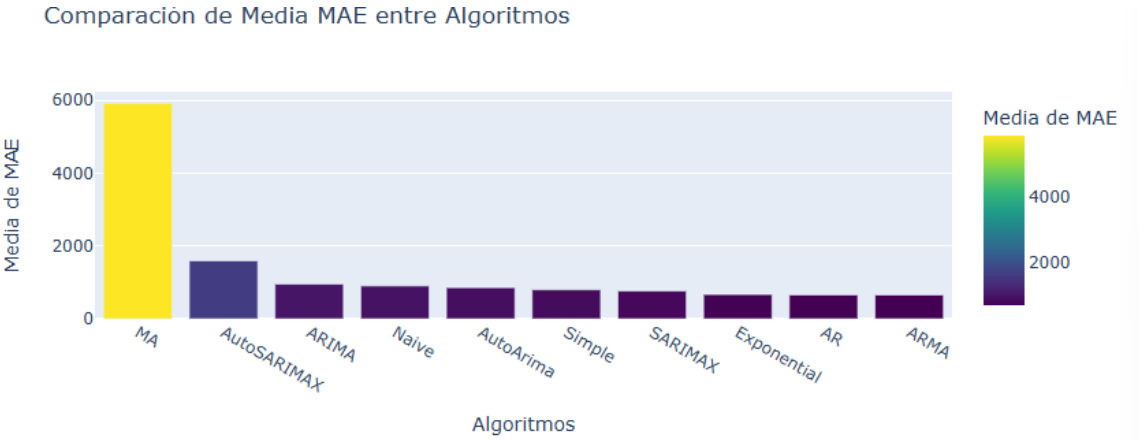
Media de maes: 765.9569398000048

Al igual que para ARIMA, entrenamos uno para cada modelo con los parámetros que más adecuados considera el algoritmo, dándole una estacionalidad de 7 (se repite cada 7 puntos, que equivalen a 7 días)



Media de maes: 821.3336478303333

Veamos estas cantidades en una gráfica para comprender de mejor manera cuales han sido los mejores algoritmos.



Resalta sobre todo el ARMA, venciendo incluso a algoritmos más minuciosos como el SARIMAX y el ARIMA, aun con un autotuning. Esto puede deberse a que estos últimos presenten un sobreajuste en el entrenamiento y a la hora de predecir, no sean tan capaces como pensábamos que serían