Universidad Autónoma de Madrid

Escuela Politécnica Superior


Máster Universitario en Ciencia de Datos

Trabajo Fin de Máster

# Predicción de patrones de uso de Spotify mediante la aplicación de técnicas de series temporales y aprendizaje automático

Guillermo Hoyo Bravo
Tutor: Alvaro Manuel Ortigosa Juarez

Marzo 2023

# Predicción de patrones de uso de Spotify mediante la aplicación de técnicas de series temporales y aprendizaje automático

Guillermo Hoyo Bravo
Tutor: Alvaro Manuel Ortigosa Juarez

Escuela Politécnica Superior
Universidad Autónoma de Madrid
Marzo 2023

# Resumen

En la era digital actual, las plataformas de streaming de música como Spotify han revolucionado la forma en la que los usuarios consumen y descubren nueva música. Comprender y predecir los patrones de uso de los usuarios en estas plataformas es fundamental para mejorar la experiencia y comprensión del usuario. En este Trabajo de Fin de Master, nos enfocamos en la predicción de varios aspectos del comportamiento del usuario en Spotify utilizando técnicas avanzadas de series temporales y aprendizaje automático.

El objetivo principal de nuestra investigación tratar de predeccir rasgos comportamentales de un usiario, asociado a este tipo de plataformas (por ejemplo: número de streamings escuchados por día, cantidad de descansos y tipo de cada uno de estos, tipo de streaming.) a partir de los datos recopilados por la aplicación. Para lograr esto, recopilamos, procesamos y transformamos los datos originales de uso de Spotify. Luego, realizamos un análisis exploratorio de datos (EDA) exhaustivo para obtener una comprensión más profunda de las tendencias, patrones y características presentes en los datos. Realizamos las predicciones con técnicas de series temporales (TS) y algoritmos de aprendizaje por refuerzo, evaluando como afectan diferentes variables al metodo seleccionado y su rendimiento.

Como parte de nuestra investigación, también exploramos de forma global la forma en la que el usuario escucha cada stream. Con esto nos referimos a medir la cantidad de veces que el usuario escucha el stream cada semana, y la racha de semanas seguidas que el usuario escucha este stream, pudiendo sacar conclusiones del patrón de consumo de Spotify del usuario. Esto puede ser afectado por características demográficas y contextuales de los usuarios, como la edad, el género y la ubicación geográfica.

El resultado de esta investigación es un conjunto de modelos predictivos sólidos y eficientes que permiten anticipar el comportamiento del usuario en Spotify. Estos modelos tienen el potencial de mejorar significativamente la calidad del servicio, personalizar la experiencia del usuario en función de sus preferencias individuales y entender el estado mental del usuario.

# Palabras clave

Spotify, predicción, series temporales, aprendizaje automático, comportamiento del usuario, análisis exploratorio de datos, patrones de uso, personalización.

# Abstract

## Keywords

# Agradecimientos

Antes que a nadie, quiero agradecerle a mi familia su confianza y apoyo incondicional en mi y en cada una de mis decisiones tomadas.

También quiero agradecer a mis amigos mas cercanos, los cuales además de mis alegrias han compartido conmigo los momentos mas duros de presión y me han hecho levantar la cabeza. Especialmente a Alejandro Peña, que me ha sido mi guía e inspiración en la vida academica y profesional.

Además, me gustaría dar las gracias a todos mis compañeros del máster por aportarme apoyo e ideas. A mi profesora Alicia Lozano Diaz por su docencia en Series Temporales y en Aprendizaje Profundo para Procesamiento de Señales de Audio, así como su apoyo constante y atento cada vez que se lo he pedido.

Finalmente, quiero dar las gracias a mi tutor Álvaro Manuel Ortigosa Juarez por apoyar mi idea y seguir hasta el final. De igual manera, quiero agradecer

Cabe hablar sobre la sociedad actual y las nuevas IAs con un potencial increible. Desde la llegada del COVID ha existido la crisis en este pais, con suerte, estas IAs puedan ayudar a resolverlas, el problema, es que parece que van a encontrar el modo de automatizar muchos puestos de trabajo y se va a agrandar. Es raro tener 24 años y que el futuro sea tan incierto y existan tantas posibilidades. Da miedo, pero a la vez dan ganas de crecer con ello.

No se sabe lo que va a pasar, pero mientras yo pueda seguiré luchando por un mundo mejor y más ordenado.

*""Piensa a la ligera en ti mismo y profundamente en el mundo".*

Miyamoto Musashi

# Índice general

**Resumen** v

**Abstract** vii

**Agradecimientos** ix

**Índice de Figuras** xiv

**Índice de Tablas** xvii

**1. Introducción** **1**

1.1. Motivación . . . . . 1

1.2. Objetivos . . . . . 5

1.3. Organización de la memoria . . . . . 5

1.4. Contribuciones . . . . . 6

**2. Estado del arte** **7**

2.1. Series Temporales, Regulares e Irregulares . . . . . 7

2.1.1. Series Temporales . . . . . 7

2.1.2. Series Temporales Irregulares . . . . . 8

2.1.3. Series Temporales Regulares . . . . . 9

2.2. Transformación de STI a STR . . . . . 11

2.2.1. Proceso de transformación . . . . . 11

2.2.2. Consecuencias . . . . . 12
2.2.3. Extracción de características . . . . . 13
2.3. Análisis de Series Temporales . . . . . 15
2.3.1. Análisis Exploratorio de Datos (EDA) en Series Temporales . . . 15
2.3.2. Métodos de Validación Cruzada (CV) . . . . . 15
2.3.3. Métricas de Evaluación . . . . . 17
2.4. Preparación previa a la predicción . . . . . 19
2.4.1. Codificación de variables categóricas . . . . . 20
2.5. Aspectos Psicológicos del Trabajo . . . . . 21
2.5.1. Psicología en los Descansos de Música . . . . . 21
2.5.2. Conclusiones para predecir el tiempo de escucha en datos no lineales 23

**3. Diseño del sistema** **25**
3.1. Dataframe Inicial . . . . . 25
3.1.1. Limpieza de datos . . . . . 26
3.2. Tratamiento de datos temporales . . . . . 28
3.2.1. Problemas . . . . . 28
3.2.2. Representación Visual y Busqueda de Patrones . . . . . 30
3.3. Dataframe Diario . . . . . 33
3.4. Dataframe Descansos . . . . . 38
3.5. Rest Type . . . . . 39
3.6. Nombre de la canción . . . . . 41

**4. Experimentos y Resultados** **43**
4.1. EDA de Series Temporales . . . . . 43
4.1.1. Daily Dataframe . . . . . 43
4.1.2. Rest Number . . . . . 44
4.1.3. Rests Type Dataframes . . . . . 44

4.1.4. Song Name Dataframe . . . . . 45
4.2. Predicciones . . . . . 46
4.2.1. Predicciones Dataframe Diario . . . . . 47
4.2.2. Predicciones Dataframe Number of Rests . . . . . 51
4.2.3. Predicciones Dataframe TYPE of Rests . . . . . 52
4.2.4. Predicciones Dataframe Songs Names . . . . . 53

**5. Conclusiones y trabajo futuro** **59**

**Glosario** **61**

**Bibliografía** **62**

**Anexo:** **65**

**Anexo:** **69**

**Anexo:** **76**

# Índice de figuras

2.1. Diferentes tipos de visualizaciones de STs . . . . . 8
2.2. TSCV y BCV . . . . . 16

3.1. Fractura DF . . . . . 29
3.2. Daily Visuals . . . . . 30
3.3. Daily Singles DF . . . . . 31
3.4. Full Daily . . . . . 32
3.5. Mes Visuals . . . . . 32
3.6. Month Singles DF . . . . . 33
3.7. Full Superposed Month . . . . . 34
3.8. Full Superposed Month . . . . . 34
3.9. Year Visuals . . . . . 35
3.10. Years Singles DF . . . . . 36
3.11. Full Superposed Years . . . . . 37
3.12. Daily DF Correlation matrix . . . . . 37
3.13. Daily DF Correlation matrix . . . . . 38
3.14. Histogram of Type of Rests DF . . . . . 39
3.15. Type of Rests correlation Matrix . . . . . 40

4.1. N Streamings ACF and PACF . . . . . 44
4.2. N Rests ACF and PACF . . . . . 45
4.3. Rest Type ACF and PACF . . . . . 46

4.4. Song Name ACF and PACF . . . . . 46
4.5. Variable Importance NLags Example . . . . . 49
4.6. Variable Importance TimeFeatures Example . . . . . 50

1. Churn Rate Number of times streamed DF . . . . . 67
2. Churn Rate weeks streaks . . . . . 67
3. Churn Rate Distribution . . . . . 68
4. Churn Rate Distribution without outlayers and common values . . . . . 69
5. visuals N Streams . . . . . 71
6. Rolling mean (a) N Streams (b) N podcasts . . . . . 72
7. STR N Streangs Decomposition . . . . . 73
8. N Streamings ACF and PACF weekly . . . . . 74
9. N Rests Visual Data . . . . . 75
10. N Rests Rolling Mean . . . . . 75
11. N Rests Decomposition . . . . . 76
12. N Rests weekly ACF and PACF . . . . . 77
13. Rest Type Visual . . . . . 77
14. Rests Type Rolling Mean . . . . . 78
15. Rests Type Decomposition . . . . . 78
16. Rest Type weekly ACF and PACF . . . . . 79

# Índice de tablas


2.1. ST Eventos Astronómicos . . . . . 9

2.2. STR Eventos Astronómicos Transformado . . . . . 10

2.3. Basic Time Features . . . . . 13

2.4. More Time Features . . . . . 13

2.5. Lag Time Features . . . . . 14

3.1. Example of the Original Dataframe . . . . . 26

3.2. Varianza Daily DF . . . . . 35

3.3. Varianza N Rests DF . . . . . 38

3.4. Varianza Rests Type DF . . . . . 39

4.1. Dummy Results Daily DF . . . . . 47

4.2. Example of Lag Generation for Evaluation . . . . . 48

4.3. Baseline Results Daily Lags DF . . . . . 48

4.4. Baseline Results Daily Lags TF DF . . . . . 51

4.5. Baseline method Results of the predictions for the variable Number of Rests 51

4.6. Random Forest Regresor Results Number of Rests . . . . . 52

4.7. Random Forest Classifier Results Type of Rests . . . . . 52

4.8. Rest Type - Dummy Regressors - Baseline Prediction . . . . . 53

4.9. Random Forest Classifier Results Type of Rests . . . . . 53

4.10. Random Forest Classifier Results Type of Rests Different Lags . . . . . 54

4.11. Random Forest Classifier Results Type of Rests for all time features . . . 54

4.12. Dummy Regressors MxN One Hot Encoding and N Labeled Encoding . . 55

4.13. Random Forest Classifier MxN One Hot Encoding and N Labeled Encoding 55

4.14. Random Forest Classifier MxN One Hot Encoding . . . . . 56

4.15. Random Forest Classifier N Labeled Encoding . . . . . 56

4.16. Random Forest Classifier MxN One Hot Encoding Best TF . . . . . 57

4.17. Random Forest Classifier N Labeled Encoding Best TF . . . . . 57

4.18. Random Forest Classifier MxN One Hot Encoding Selenium Features . . 58

4.19. Random Forest Classifier N predictions Labeled Encoding Selenium Features 58

4.20. Random Forest Classifier MxN predictions One Hot Encoding, TF y Selenium Features . . . . . 58

4.21. Random Forest Classifier N predictions Labeled Encoding, TF y Selenium Features . . . . . 58

1. Baseline Results Daily Lags CY TF DF . . . . . 82

2. Baseline Results Daily Lags MIX TF DF . . . . . 83

# Capítulo 1

# Introducción

## 1.1. Motivación

En las últimas décadas, hemos sido testigos de notables progresos en la inteligencia artificial (IA), gracias a la creciente capacidad de computo y la inmensa cantidad de datos proveniente de cualquier fuente. Con ello, la IA cada día es mas cercana a nosotros, la podemos encontrar en los anuncios por los sitemas de recomendación [1] y servicios de streaming [2], al hacer una simple búsqueda en google e incluso al grabar un video o sacar una foto.

Para el público en general, los avances tecnológicos de esta era se traducen en una mayor accesibilidad, debido a las facilidades existentes para que cualquier persona pueda generar contenido de cualquier tipo [3], y a una mayor profundidad en el contenido generado.

Por ejemplo, antiguamente grabar una canción de forma profesional era practicamente inpensable para una persona normal y corriente. Actualmente, cualquiera puede tener un estudio de grabación en su casa con todos los programas de edición de audio profesionales y generar canciones o podcasts de gran calidad. Como es mas accesible, se genera mas contenido, se descubren nuevos elementos y se crean nuevos subtemas (en este ejemplo serían nuevos subgeneros de música), generando mayor profundidad.

La comodidad que estos avances proporcionan fomentan su aceptación, apostando por un futuro de la inmediatez en las cosas, donde cada vez se recogerán mas datos. De forma natural, los sucesos ocurren sin aviso previo, al monitorizarlos y ponerles fecha y hora, la recopilación de estos eventos que se repiten a lo largo del tiempo de forma arbitraria se conoce como Series Temporales Irregulares [4].

Su naturaleza hace que sea muy difícil manejar la información, encontrar patrones y conseguir predicciones con resultados muy ajustados a la relidad. Estas series temporales suelen tratarse con diferentes técnicas para convertirlas en Series Temporales Regulares, de modo que el algoritmo obtenga una mejor comprensión de los patrones de los dátos y mejores resultados. [5].

Esta suma creciente de información almacenada, es muy positiva para procesar con algoritmos pero es algo contraproducente para los humanos. Cada día, el número de expertos aumenta, y existe tal variedad en cada campo que necesitamos especialistas para cada aspecto de la vida cotidiana. Un claro ejemplo de esto es el mencionado anteriormente acerca de crear nueva música, otro mas casual sería cuando queremos adquirir un teléfono móvil o cualquier electrodoméstico; incluso en casos menos personales, buscamos, comparamos y consumimos tiempo en material audiovisual para entender la gran variedad de opciones existentes y tomar una decisión informada [6].

Es cierto que, en ocasiones, disponer de una gran cantidad de información puede resultar abrumador. El hecho de tener múltiples opciones dentro de un mismo tema puede llevarnos a experimentar confusión. La toma de decisiones puede requerir mucho esfuerzo y, además, la inseguridad no escoger la opción óptima puede ser un obstáculo a la hora de decidir. Las marcas, conscientes de esta situación, intentan vendernos sus productos como experiencias únicas y, en muchas ocasiones, esto puede llevarnos a tomar una decisión apresurada o a abandonar el objetivo inicial. En definitiva, la falta de decisión o el exceso de opciones pueden resultar contraproducentes si no sabemos manejar correctamente la información disponible. [7] [8]

La forma y la influencia de las marcas en nuestra sociedad crea la denominación de sociedad de consumo, en la cual hay que consumir para ser feliz [9]. De este modo, podemos volvernos adictos, un problema de la sociedad actual [10]

La música es un elemento clave a la hora de consumir. Las técnicas de marketing promueven el uso de la música para afectar al comportamiento de los usuarios en sus tiendas, así como apelar ciertos sentimientos en anuncios o para crear un sonido que sea unicamente suyo, como por ejemplo: Qechua [1].

Además, la musica también es un elemento que consumimos, cada día en mayor medida y de forma mas sencilla con los servicios de streaming. Los artistas o grupos de música también se indentifican como una marca [11], y esto nos genera fidelidad a ellos, nos pueden enganchar a su vida o sus nuevas canciones. La atención es la moneda de cambio más utilizada mundialmente, quien tenga tu atención, gana dinero, aunque no le pagues de forma explicita.

De este modo, cuanto más tiempo reproduzcamos sus canciones, más dinero ganarán y más cerca nos sentimos a ellos. El bombardeo constante de estímulos generados por los rítmos, melodías y rimas nos genera grandes sentimientos: Desde el sentirse comprendido cuando te sientes triste o enfadado, hasta el de estar enamorado. Las canciones, además, nos recuerdan experiencias y personas, esto hace que sea incluso mas poderoso todavía el gran vínculo que formamos con ellas [12].

Esto implica que la música controla o se asocia con nuestros sentimientos y moldea nuestra personalidad. Dicho de otra manera, las emociones condicionan nuestras acciones y forma de ver la vida, y esta, a su vez, generan diferentes emociones segun la canción que se escuche. Para cada persona es un caso distinto, y cada día es único, eso es lo que

[1] `https://www.decathlon.es/es/browse/b/quechua/_/N-14j0fuw`

hace tán complicado porner barreras o límites a la música.

La música puede tener tanto efectos positivos como negativos en nuestro cerebro y estado emocional. En el articulo 'Do Somber Songs Make Teens More Depressed?' acerca del impacto de escuchar música triste en adolescentes [13], se concluye que si bien esta puede consolar, escucharla en exceso,y según que tipo, puede llevar a la rumiación y a la depresión [14]

En otro estudio llamado "Music and People with Tendencies to Depression-[15], se investiga con una escala el resultado de escuchar música triste en alumnos de universidad, en la cual se demuestra que para gente con tendencia a la depresion, además de gustarles mas la música que les evoca sentimientos tristes, por lo que escuchan este típo de música, este típo de canciones les incrementan el sentimiento de depreseión.

En el artículo Çan You Hear the Music?-[16], hablan de estudios que enseñan que la música puede afectar a nuestro estado de ánimo a largo plazo, incrementando los sentimientos de ansiedad y depresión, según la letra de la canción y su mensaje.

También, en la revista "Los Angeles Times", encontramos un articulo donde se afirma por neurologs que la música nos genera dopamina, generandonos un placer intenso como el que nuestro cerebro consigue con otras recompensas como comida o drogas. Lo que puede llevar a un consumo excesivo de esta [17].

Escuchamos mas música a diario de la que imaginamos, alguna de manera consciente, y otra de forma inconsciente, pero es parte de nuestro entorno y nos afecta. En el centro comercial, al ver la televisión, mientras trabajamos, incluso en los ascensores. Cuántas veces os habéis sorprendido tarareando o silvando una canción mientras caminabais, incluso suenan en eventos deportivos, en los supermercados, y hasta en los ascensores. Se estima que escuchamos 20 [18] horas semanales de forma consicente.

Al ser algo que está integrado y normalizado en una gran variadad de aspectos y acividades diarias de nuestra vida, y con la corta duración de estas, es importante entender cómo lo consumimos, o al menos poder tener alguna herramienta que nos pueda ayudar a gestionarlo. Ya que tendemos a ser menos conscientes y darle poca importancia.

La mayoría de sistemas de recomendación de hoy en día se basan en lo que les gusta a personas .ªfines a ti", pero en un caso particular, puede que dos personas cercanas no sean afines, lo que se puede comprobado con spotify fusion [2]. Lo que mas nos gusta es poder compartir nuetros gustos y los de la gente que queremos.

Los sitemas de recomendación en plataformas de streaming son algoritmos diseñados para sugerir contenido relevante y personalizado a los usuarios, basandose en sus preferencias, comportamiento de navegación y patrones de consumo. Diferentes enfoques son:

- Filtrado colaborativo: Este enfoque se basa en la idea de que los usuarios que han

[2]https://support.spotify.com/mx/article/social-recommendations-in-playlists/

mostrado preferencias similares en el pasado también tendrán gustos similares en el futuro. Puede ser de dos tipos [19]:

1. Basado en usuarios: Identifica usuarios similares al usuario objetivo y recomienda contenido que otros usuarios similares han consumido.
2. Basado en elementos: Sugiere contenido similar al que el usuario ha consumido previamente, basándose en la similitud entre los elementos.

- Filtrado basado en contenido: Este enfoque sugiere contenido basado en las características del contenido consumido por el usuario. Utiliza técnicas de procesamiento de lenguaje natural y aprendizaje automático para extraer características y determinar la similitud entre diferentes elementos [20].
- Filtrado híbrido: Combina los enfoques de filtrado colaborativo y basado en contenido, aprovechando las ventajas de ambos para mejorar la calidad de las recomendaciones.

Para modelar el comportamiento de usuario, se recolectan y analizan diversas fuentes de datos, como historiales de reproducción, calificaciones, búsquedas, interacciones sociales y más. Esto permite crear perfiles de usuario y grupos de usuarios similares, y agruparlos en clusters. Algunas técnicas empleadas en esta modelación incluyen el aprendizaje supervisado, no supervisado y por refuerzo, como son: Machine Learning (ML) o Procesamiento de Lenguaje Natural (NLP).

En este caso, Spotify utiliza un sistema colaborativo llamado Bart (acrónimo de Bandits for recommendations as treatments), una IA diseñada para predecir, según las acciones del usuario como pueden ser, el historial de escucha, saltar, agregar a playlist o los atributos de estas canciones [21].

Esto les permite construir conjuntos de datos y modelos de interacción del usuario para crear experiencias atractivas y personalizadas para sus usuarios, de manera basada en datos basada en las interacciones y señales de retroalimentación de los usuarios.

Exiten distintos retos para detectar emociones con sistemas de recomendaciones [22], como la falta de datos específicos, requiriendo una gran cantidad un etiquetado a mano el cual es poco fiable, Ambiguedad en la interpretación, debido a que cada persona reacciona a diferentes canciones de modo diferente, así como el sesgo de la IA emocional. En el estudio 'The Risks of Using AI to Interpret Human Emotions', nos explican como gente de diferente etnia tiene mayor probabilidad de asignación de emociones negativas que otra.

Con todo esto en mente, en este Trabajo de Fin de Máster partimos de una premisa fundamental, nuestro estado emocional es una parte vital y el consumismo y el exceso de posibilidades que existen globalmente para generar nuevo contenído es demasiado para que el espacio de una persona esté limpio y ordenado. Es responsable aprender a manejar y guardar la información que es mas relevante para cada persona de manera eficiente, dejando atrás las cosas que no lo son, algo dificil de ver en un mar de posibilidades.

Con este TFM se busca generar conciencia sobre la salud mental, generado por el abuso de consumo y la abrumación del exceso de información de la época actual, especialmente en la música, que en si es muy relevante en nuestro estado de ánimo. Generando información relevante del método de consumo y sus patrones de un único usuario, de manera que el mismo tome conciencia. Al tratar datos objetivos como son el número de reproducciones y sin tener información previa del usuario como su edad o etnia, quitamos posibles sesgos y nos centramos unicamente en los patrones generados por el, de forma esta información individual y personalizada pueda ser tratada para casos concretos como una herramienta extra, y pueda hacer más consciente al usuario.

## 1.2. Objetivos

Como se menciona en la Sección 1.1, el objetivo de este Trabajo de Fin de Máster es profundizar en el conocimiento de los patrones o de la forma de consumo de un usuario, prediciendo los resultados de un día en el futuro mediante clasificadores basados en Random Forest (RF). Para ello, es necesario tratar uns serie temporal irregular, extraer caraterísticas temporales y encontrar las que generan mejores resultados, así como nuevas *features* útiles para entender cómo se comporta el usuario en su día a día.

Con este objetivo principal en mente, podemos definir una serie de objetivos parciales que buscamos alcanzar en este trabajo:

1. Revisión del Estado del Arte en Series Temporales, poniendo especial interés en qué son, como transformarlas y sus conseccuencias, así cómo en las formas de predicción de series temporales Irregulares o Regulares, análisis y EDAs de estas.

2. Revisión del Estado del Arte en extracción de características temporales, métodos de validación cruzada para series temporales, y métodos de evaluación.

3. Estudio de la base de datos de Spotify, de cómo tratar sus datos, así como extraer características web de estos.

4. Revisión del métodos de aprendizaje automático para series temporales con Random Forest.

## 1.3. Organización de la memoria

Esta memoria de Trabajo de Fin de Máster está organizada en los siguientes capítulos:

- *Capítulo 1. Introducción*: Este capítulo introduce la motivación, los objetivos, la organización general del trabajo y las principales contribuciones realizadas.

- *Capítulo 2. Estado del Arte*: En este capítulo se examinan las tecnologías y estudios relevantes en relación a las Series Temporales Irregulares y Regulares.
- *Capítulo 3. Diseño*: Este capítulo aborda el proceso de limpieza de datos, transformación de dataframes de Series Temporales Irregulares a Regulares, depuración de dataframes y extracción de nuevas características tanto del dataframe como en línea.
- *Capítulo 4. Experimentos y resultados*: A lo largo de este capítulo, se presentan y analizan los distintos experimentos llevados a cabo en el contexto de este trabajo, mostrando los resultados obtenidos.
- *Capítulo 5. Conclusiones y trabajo futuro*: En este capítulo se resumen las principales conclusiones del estudio, así como las posibles direcciones para futuras investigaciones.
- *Glosario.*
- *Bibliografía.*
- *Anexos*

## 1.4. Contribuciones

Las principales contribuciones de este Trabajo de Fin de Máster a la comunidad investigadora son las siguientes:

- El estudio de extracción de características para detectar patrones en el consumo de usuarios y atributos personales que definan al un único usuario.
- El desarrollo y estudio de la extracción de características para series temporales con Random Forest.
- El desarrollo y predicciones futuras de variables con diferentes naturalezas (categóricas y numéricas), comprobando su eficiencia.

# Capítulo 2

# Estado del arte

En este capítulo, se examina el Estado del Arte en tecnologías relacionadas con el enfoque de este trabajo. La Sección 2.1 proporciona una introducción breve a las Series Temporales Irregulares (STI) y Regulares (STR). En la Sección 2.2 se aborda la transformación de STI a STR, sus implicaciones y la extracción de características temporales o Time Features (TF). La Sección 2.3 se centra en el análisis de series temporales o Análisis Exploratorio de Datos (EDA), junto con diversos métodos de Validación Cruzada (Cross Validation, CV) especializados para Series Temporales y los distintos métodos de evaluación disponibles. La Sección 2.4 trata sobre métodos clásicos de limpieza de datos, así como las posibles transformaciones de variables categóricas a numéricas con fines predictivos. Por último, la Sección 2.5 aborda brevemente la psicología humana en relación con las decisiones de diseño tomadas en el proyecto.

## 2.1. Series Temporales, Regulares e Irregulares

### 2.1.1. Series Temporales

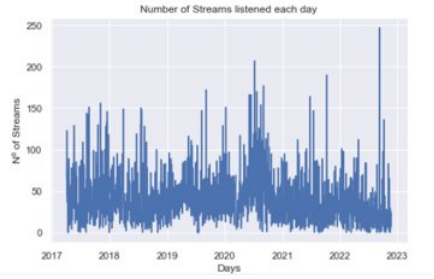
Las Series Temporales (ST) son secuencias de datos ordenados cronológicamente. Si son Regulares, estos datos se obtienen con una frecuencia de muestreo constante. Por ejemplo, si la frecuencia es diaria, se obtendrá una muestra cada día. El proceso de predicción de un valor futuro que aún no ha ocurrido suele realizarse modelando el comportamiento de la ST con los valores pasados, es decir, su comportamiento autorregresivo (AR). Si esto no fuera suficiente, también se utilizan variables externas o exógenas (X) que ayuden a modelar el comportamiento.

En relación con los tipos de series temporales, según sus diferentes características, se pueden encontrar las siguientes [23]:

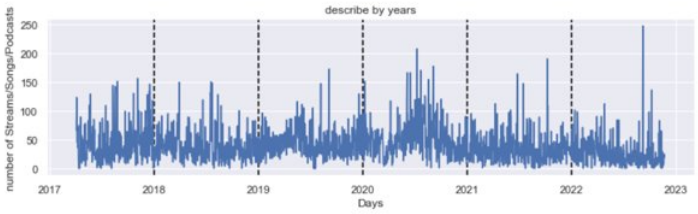
1. **Series Temporales Regulares:** Aquellas con frecuencia regular en la recopilación de datos, como mediciones diarias o mensuales.

| | day_n_streams | week_day | day_n_songs | day_n_podcast | day_new_stream | day_new_song | day_new_podcast |
|---|---|---|---|---|---|---|---|
| **2017-04-04** | 66 | 1 | 66 | 0 | 0 | 0 | 0 |
| **2017-04-05** | 123 | 2 | 123 | 0 | 0 | 0 | 0 |
| **2017-04-06** | 63 | 3 | 63 | 0 | 0 | 0 | 0 |
| **2017-04-07** | 45 | 4 | 45 | 0 | 0 | 0 | 0 |
| **2017-04-08** | 56 | 5 | 56 | 0 | 0 | 0 | 0 |
| **2017-04-09** | 36 | 6 | 36 | 0 | 0 | 0 | 0 |
| **2017-04-10** | 73 | 0 | 73 | 0 | 0 | 0 | 0 |
| **2017-04-11** | 18 | 1 | 18 | 0 | 0 | 0 | 0 |

(a)




(b)




(c)

Figura 2.1: Ejemplos visuales de varias STs obtenidas de Spotify: (a) 6 STs con frecuencia de muestreo diaria y el día de la semana; (b) visualización gráfica de la evolución del número de streams por dia; (c) Evolución del número de streams por día separado por años

2. **Series Temporales Irregulares:** Aquellas sin frecuencia regular en la recopilación de datos, como mediciones realizadas en fechas aleatorias.

3. **Series Temporales Estacionarias:** Aquellas cuya media y varianza son constantes a lo largo del tiempo.

4. **Series Temporales No Estacionarias:** Aquellas cuya media y varianza cambian a lo largo del tiempo, lo que puede dificultar su análisis y modelado.

En la imagen 2.1 podemos observar diferentes representaciones de Series Temporales Regulares.

### 2.1.2. Series Temporales Irregulares

Las STIs son aquellas que se encuentran de forma natural, es decir, las STs cuyos eventos se dan sin una frecuencia de muestreo fija, pero ordenadas cronológicamente. Como se mencionó anteriormente, las STIs presentan desafíos específicos debido a la falta de una frecuencia fija en la recopilación de los datos, lo que puede dificultar el análisis y la predicción.

| Timestamp | Suceso Astronómico |
|---|---|
| 02-02-2022 02:02:02 | Estrella fugaz |
| 02-10-2022 13:57:45 | Supernova |
| 03-25-2022 12:36:59 | Estrella fugaz |
| 04-05-2022 01:18:24 | Cometa visible |
| 04-21-2022 05:39:12 | Eclipse lunar |
| 05-02-2022 09:55:11 | Lluvia de meteoritos |
| 06-14-2022 22:47:05 | Estrella fugaz |
| 08-23-2022 07:35:20 | Estrella variable |
| 08-03-2022 10:24:48 | Supernova |
| 08-19-2022 20:17:32 | Eclipse solar |

Tabla 2.1: Ejemplo de Serie Temporal Irregular con Eventos Astronómicos

A continuación, se presenta un ejemplo de sucesos astronómicos para ilustrar este concepto. Predecir cuándo ocurrirá el siguiente evento astronómico es una tarea complicada, debido a que los marcas temporales (Timestamps) de los datos medidos carecen de frecuencia constante. Un ejemplo de este tipo de datos se muestra en la Tabla 2.1:

En relación con las técnicas más adecuadas para tratar Series Temporales Irregulares, algunos de los métodos más comunes incluyen:

1. **Modelos de Espacio de Estados (SSM):** [24] Estos modelos pueden ser especialmente útiles para STIs, ya que consideran factores no observables o latentes que podrían influir en la serie. Pueden adaptarse a cambios en la frecuencia de muestreo y manejar *missing values*.

2. **Redes Neuronales Recurrentes (RNN):** [**?** ] Las RNN, especialmente las variantes como las *Long Short-Term Memory* (LSTM) y las *Gated Recurrent Unit* (GRU), son adecuadas para STIs, ya que pueden capturar dependencias temporales de largo alcance y manejar secuencias de longitud variable.

Las series Temporales Irregulares son complejas de tratar y para predecir con ellas [25].

### 2.1.3. Series Temporales Regulares

Las Series Temporales Regulares, como se mencionó anteriormente, son aquellas Series Temporales que tienen una frecuencia temporal constante en sus datos. A continuación, presentamos un ejemplo de transformación de los datos de la Serie Temporal Irregular mostrada en 2.1 en una Serie Temporal Regular, cuyo resultado se puede observar en la Tabla 2.2.

Los modelos más utilizados y con mejores resultados a la hora de realizar predicciones para Series Temporales Regulares incluyen los modelos que pueden componer el

| Mes | Frecuencia | Suceso mas probable | Timestamp medio |
|---|---|---|---|
| 02 | 2 | Estrella Fugaz | 06-07:30:25 |
| 03 | 1 | Estrella Fugaz | 05-12:36:59 |
| 04 | 2 | Eclipse lunar | 13-03:29:18 |
| 05 | 1 | Lluvia de meteorítos | 21-09:55:11 |
| 06 | 1 | Estrella fugaz | 14-22:47:05 |
| 07 | 0 | Null | 1-00:00:01 |
| 08 | 3 | Supernova | 15-12:25:46 |

Tabla 2.2: Ejemplo de una posible transformación de una STI a una STR de Eventos Astronómicos

modelo SARIMAX: Seasonal (S) AutoRegressive (AR) Integrated (I) Moving Average (MA) Exogenous (X), las Redes Neuronales Recurrentes (RNN) o *Short Term Memory* (LSTM) [26] y algoritmos clásicos como Random Forests (RF) [27].

1. **ARIMA:** Este modelo ofrece el enfoque estadístico combinando de los modelos AR y MA; además, para volver la serie estacionaria, el modelo también integra (I). Este modelo funciona correctamente con tendencias y estacionalidad, es sencillo de interpretar y ajustar, pero su mayor problema es la asunción de relaciones lineales entre los datos pasados y futuros, y le cuesta comprender las dependencias a largo plazo, además de requerir estacionariedad en los datos.

2. **LSTM/RNN:** Los modelos Deep Learning estan diseñados para detectar relaciones temporales. Es capaz de modelar relaciones lineales y no lineales con dependencias a largo plazo, también funciona para alta dimensionalidad y se adapta bien a grandes conjuntos de datos. Por desgracia, requiere una gran cantidad de datos y tiempo de entrenamiento, sus interpretaciones no son claras para los humanos, y es muy sensible al ajuste de sus hiperparametros.

3. **RF:** Es un algoritmo de Machine Learning (ML) basado en árboles de decisión que, gracias a las características temporales como el Timestamp, puede adaptarse a las Series Temporales. Es un algoritmo robusto ante ruido y valores atípicos, puede manejar variables categóricas y numéricas y además de proveernos información de la importancia de las variables de entrada, es resistente al sobreajuste. Lamentablemente, este algorimo no esta preparado para trabajar con STs y modelar este típo de relciones, pero afortunadamente, se pueden extrar características temporales aumentando su eficacia.

La comparación de resultados entre estos métodos dependerá del conjunto de datos y el problema específico. En general, LSTM/RNN podría ser más adecuado para series temporales con estructuras complejas y no lineales, mientras que ARIMA podría funcionar mejor en STs con tendencias y/o estacionalidades. Random Forests puede ser una buena opción cuando se desean incorporar variables adicionales y se busca robustez ante ruido y valores atípicos.

BIBLIO:

https://riunet.upv.es/bitstream/handle/10251/159161/Morer

....

## 2.2. Transformación de STI a STR

### 2.2.1. Proceso de transformación

En este apartado, se presentan los pasos a seguir para pasar de una STI a una STR de forma correcta [28].

1. Interpolación: Para convertir una serie temporal irregular en una regular, a menudo es necesario interpolar los *missing values*. Se pueden utilizar varios métodos de interpolación, como interpolación lineal, polinomial o basada en splines, dependiendo de la naturaleza de los datos y el dominio específico.

2. Resolución temporal: Al transformar una STI en una STR, es importante elegir una resolución temporal adecuada. Por ejemplo, si la STI tiene datos recopilados en intervalos de tiempo muy diferentes, es posible que sea necesario reducir la resolución temporal para obtener una serie regular.

3. Resultado de la STR: Una vez que se ha interpolado y seleccionado una resolución temporal, se puede convertir la STI en una STR mediante técnicas de remuestreo o agregación.

Para ilustrar mejor este proceso, podemos tomar como ejemplo las Tablas 2.1 y 2.2 del apartado anterior, donde la variable "Suceso Astrológico" es una variable categórica.

Para interpolar, en este caso, necesitamos crear una nueva entrada en los datos para rellenar la información del mes 07, en el cual no sucedieron sucesos astronómicos y no aparece en la tabla. Por lo que creamos esta entrada con valores de null o estándares definidos. Otras técnicas son utilizar la moda o la media del valor a completar. Con ello, asumimos resultados, o añadimos variabilidad, pudiendo ser problemático si el número de *missing values* es muy grande.

La resolución temporal puede tener cualquier valor temporal, es decir, de segundos, minutos, horas, días, meses, años, o incluso periodos de meses, o una combinación entre varias. Lo importante es que la información contenga una frecuencia constante. De este modo, lo más apropiado para estos datos es una frecuencia mensual.

Por último, los datos se presentarán como una STR. Para ello también se pueden aplicar una gran variedad de técnicas para intentar perder el mínimo de información

posible, pero es imposible mantenerla completa, de ahí el gran problema que existe para predecir en STIs. En este caso, contamos el número de eventos astronómicos de ese mes en la variable "Frecuencia", la variable "Suceso más probable" contiene el evento que más ha pasado ese mes o en caso de empate, el que más ha ocurrido contando todas las anteriores veces, y la variable "TimeStamp medio" contiene la media de las horas de los eventos astronómicos.

De esta manera, podemos predecir de manera segura y fiable el número de eventos astronómicos que ocurrirán el próximo mes. A cambio, será muy complicado concretar el evento que sucederá después, o la hora en la que ocurrirá.

### 2.2.2. Consecuencias

Transformar una serie temporal irregular en una regular tiene varias ventajas y desventajas, por lo que es importante analizar el caso concreto antes de hacerlo [29]. Vamos a mencionarlas a continuación:

Ventajas:

1. Compatibilidad con modelos: La mayoría de los modelos estadísticos y de aprendizaje automático están diseñados para trabajar con datos regulares. Al convertir una STI en una STR, se puede utilizar una amplia gama de modelos para analizar y predecir los datos.
2. Mejor manejo de la dependencia temporal: Los modelos de STs pueden capturar más fácilmente las relaciones temporales en datos regulares, lo que permite realizar predicciones más precisas y confiables.
3. Detección de estacionalidad y tendencias: Las STRs facilitan la identificación de patrones estacionales y tendencias en los datos, lo que puede mejorar la precisión de las predicciones y permitir un análisis más profundo.
4. Reducción de ruido y manejo de *missing values*: Es posible suavizar el ruido y llenar los *missing values* mediante técnicas de interpolación, lo que mejora la calidad de los datos y las predicciones resultantes.
5. Comparabilidad: Permite estandarizar la resolución temporal y mejorar la comparabilidad entre diferentes series, lo cual es útil cuando se trabaja con múltiples conjuntos de datos.

Desventajas:

1. Pérdida de información: En este proceso puede haber pérdida de información si los datos originales contienen patrones específicos relacionados con la irregularidad del tiempo. La interpolación y el remuestreo pueden introducir errores y distorsiones en los datos.

| Año | Mes | Día | Hora | minuto | segundo |
|---|---|---|---|---|---|
| 2022 | 12 | 15 | 04 | 00 | 00 |

Tabla 2.3: Ejemplo de una posible combinación de Time Features muy básicos y muy efectivos para mejorar resultados

| estacion | dia-año | Semana | Día-semana |
|---|---|---|---|
| Verano | 349 | 52 | Jueves |

Tabla 2.4: Ejemplo de posibles derivaciones de Time Features de la tabla 2.1

2. Elección de la resolución temporal: Elegir una resolución temporal adecuada puede ser un desafío. Una resolución demasiado alta puede resultar en una gran cantidad de *missing values* que deben ser interpolados, mientras que una resolución demasiado baja puede ocultar patrones y detalles importantes en los datos.

3. Coste computacional: Se Aumentar el coste computacional, especialmente si la resolución temporal elegida es alta y se requiere una gran cantidad de interpolación y remuestreo.

4. Sobreajuste: Se puede aumentar el riesgo de sobreajuste si se utilizan demasiados datos interpolados en lugar de datos reales.

En resumen, la transformación de una STI en una STR puede facilitar el análisis y la predicción, pero también presenta desafíos y riesgos específicos.

### 2.2.3. Extracción de características

Como se ha mencionado en la Sección 2.1.3, es importante para mejorar los resultados de los modelos extraer características temporales [30], estas nuevas características ayudan a capturar dependencias temporales. Por ejemplo, en la Tabla 2.1, en vez de tener el Timestamp, es decir, la fecha y la hora juntas, podemos crear una variable para cada variable temporal, acabando con las 6 siguientes representadas en la Tabla 2.3:

De igual manera podemos extraer nuevas variables temporales también muy intuitivas relacionadas con estas 6 originales e intuitivas. La Tabla 2.4 muestra nuevas características extraídas de las 6 iniciales de la Tabla 2.3. Se puede formar una nueva tabla que agrupe las caracteristicas que contienen ambas tablas.

A su vez, la extracción de Time Features puede seguir complicándose. Por ejemplo, el modelo de predicción de series temporales desarrollado por Facebook[1]; Prophet, es un gran modelo fácil de usar para el usuario y robusto ante *missing values*, el cual nos genera Time Features Cíclicos automáticamente.

[1] `https://github.com/facebook/prophet`

| año | mes | Día | Día- Lag1 | Día-Lag2 | Día-Lag3 |
|---|---|---|---|---|---|
| 2022 | 12 | 18 | 17 | 16 | 15 |

Tabla 2.5: Ejemplo de realizar la operación .shift() para los valores 1, 2 y 3 en la variable Día

Los Time Features Cíclicos [31] son relevantes ya que el día número 365 está igual de cerca del día número 1 que del día número 2. Pero en este formato numérico los modelos no lo entienden, y es necesario utilizar técnicas especializadas.

Aun así, las Time Features Cíclicas no son beneficiosas para todos los modelos. Por ejemplo, Random Forests es incapaz de recoger 2 variables a la vez, por lo que una representación con 2 valores (e.g. seno y coseno) para cada día de la semana, puede ser ineficiente. Es decir, Random Forests procesa las variables de una en una afectando a las anteriores, no es capaz de representar una única variable como un conjunto de varias.

Otra Time Feature muy importante y completamente obligatoria a generar son los lags. Las variables lags únicamente son aquellas que recogen el valor anterior de una variable en una serie temporal. Por ejemplo, si creamos 3 lags para la variable día de la Tabla 2.3, nos queda laTtabla 2.5.

De este modo, los primeros 3 valores del Dataframe se pierden al tener la necesidad de desplazar la variable. Es decir, si el primer valor de la variable día es el 15, su lag 1, 2 y 3 son NaN, y al desplazarla 1 vez, es decir al realizar la operación en Python: .shift(1). Para el siguiente valor, el día 16, el lag1 es 15, pero el lag2 y lag3 siguen siendo NaN, porque no hay información de estos.

Estas variables lags son completamente cruciales en caso de utilizar el modelo Random Forests ya que este no tiene en cuenta de forma intrínseca la estructura temporal de los datos y son las que permiten que capture los patrones y las relaciones temporales de los datos.

Para ayudarnos a calcular el número de lags a utilizar se define según los valores de la función de Autocorrelación (ACF) y de Autocorrelación Parcial (PACF), las cuales se describen detalladamente más adelante en el Apartado 2.3. También es importante tener en cuenta la complejidad del modelo o la capacidad de cómputo, ya que añadir nuevos lags y variables solo incrementará la complejidad y el tiempo de entrenamiento.

Para comprobar la importancia y relevancia de todas estas nuevas variables o Time Features generadas, nos ayudaremos de la propia función de Random Forest de la librería Sklearn, que nos da la importancia que tienen para el modelo cada variable: feature_importance(). De este modo, podemos reducir la complejidad del modelo, maximizando los resultados de este.

## 2.3. Análisis de Series Temporales

### 2.3.1. Análisis Exploratorio de Datos (EDA) en Series Temporales

El análisis de STs es un enfoque estadístico y de aprendizaje automático utilizado para estudiar y predecir patrones en datos recopilados secuencialmente en el tiempo. El objetivo es identificar tendencias, estacionalidades, ciclos y otros patrones en los datos y utilizar esta información para predecir valores futuros de la serie temporal. A continuación, se describe un enfoque general para el Análisis Exploratorio de Datos (EDA) [32] en series temporales.

Un EDA en series temporales consiste en:

- Visualización de la serie temporal: Para ello, utilizaremos la función plot de matplotlib.pyplot, obteniendo los resultados que hemos observado en la imagen 2.1: (b) y (c). Además, analizaremos su composición, representada en la siguiente imagen. Como sus propios nombres indican, la tendencia nos muestra si la serie temporal tiende a una subida o una bajada constante. Por otro lado, la estacionalidad nos indica si existen siempre los mismos picos en los mismos valores, por ejemplo, para las agencias de vuelo existe una estacionalidad anual, en la cual se compran más vuelos durante el verano y menos en invierno. El ruido son los valores que la suma de los dos componentes recién explicados no es capaz de explicar.
- Análisis de estacionariedad: Para realizarlo, podemos utilizar diferentes técnicas. Podemos aproximar un resultado con el método de media y desviación típica móviles. Además, los test de Dickey Fuller (ADF) o Kwiatkowski-Phillips-Schmidt-Shin (KPSS) nos indican si la serie es o no estacionaria, rechazando o no rechazando esta hipótesis nula con cierto p-value.
- Análisis de autocorrelación: Realizado mediante las ya comentadas funciones de autocorrelación (ACF) y autocorrelación parcial (PACF), de manera que podamos encontrar relaciones entre los valores de la serie temporal y sus lags.

### 2.3.2. Métodos de Validación Cruzada (CV)

La validación cruzada [33] es una técnica esencial en el aprendizaje automático para evaluar el rendimiento y la robustez de los modelos cuando se enfrentan a nuevos datos. En el contexto de las STRs, es importante investigar diferentes tipos de métodos de validación cruzada debido a la estructura temporal inherente de los datos y a la necesidad de preservar la dependencia temporal durante el proceso de evaluación.

La validación cruzada estima el rendimiento del modelo para diferentes subconjuntos de datos, ayudando a prevenir el sobreajuste y asegurando una buena generalización para

nuevos datos. Esta es aplicable a la hora de seleccionar los hiperparámetros, obteniendo los mejores resultados posibles. Este método consiste en partir el DF original en subconjuntos de Test y Train, extrayendo métricas de rendimiento para cada uno de ellos y evaluandolas independientemente de los subconjuntos usados.

Sin embargo, como el modelo se entrena múltiples veces en diferentes conjuntos de datos, esto hace que el proceso sea más costoso computacionalmente y que el tiempo de entrenamiento sea mayor.

El problema de utilizar la técnica clásica de CV es que perdemos la dependencia temporal, por eso necesitamos utilizar otro tipo de CV especial. A continuación, se muestran dos tipos diferentes de CV especializados en series temporales en la Figura 2.2 :

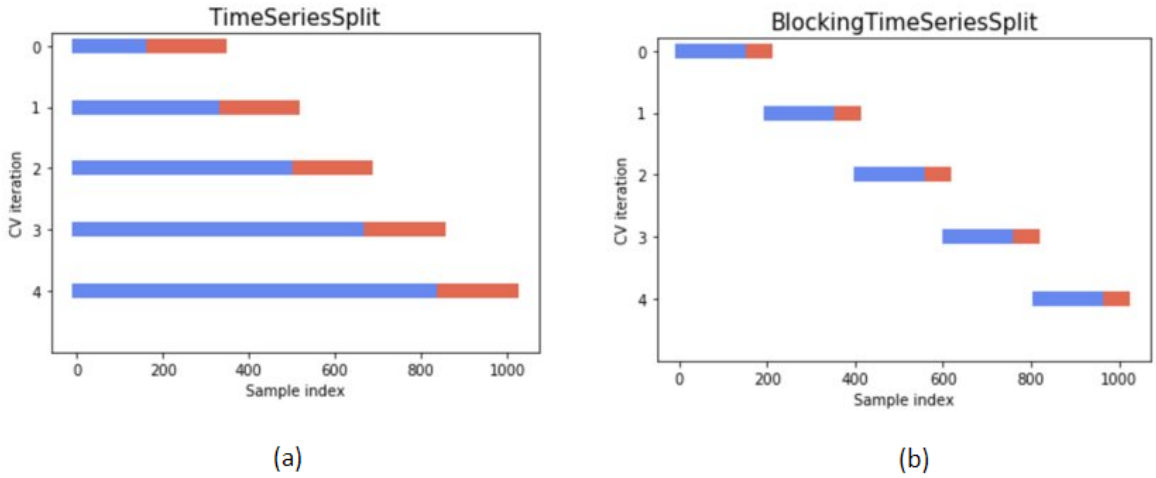



Figura 2.2: Ejemplo Visual del método Time Serie Cross Validation (TSCV) y Block Cross Validation (BCV)

Como podemos observar en la imagen, Time Series Cross Validation (TSCV) es una técnica que nos permite entrenar con todos los datos anteriores para predecir el conjunto futuro, manteniendo todas las relaciones temporales posibles.

De manera similar, Block Cross Validation (BCV) nos permite entrenar con los datos anteriores, pero esta vez con un conjunto limitado de ellos, perdiendo los más antiguos. De este modo, la etapa de entrenamiento tiene menos datos y el tiempo de entrenamiento es menor, pero según la relevancia y las relaciones que existan entre los datos, se pueden perder relaciones importantes.

BIBLIO: .....

### 2.3.3. Métricas de Evaluación

Las métricas de evaluación para validar y seleccionar modelos son bien conocidas [34], y existen diferentes tipos según los intereses o la naturaleza de los datos. Entre los tipos, podemos clasificarlos en dos grandes grupos principales que son:

1. **Métricas para Variables Numéricas**

   - **Error cuadrático medio (MSE, Mean Squared Error)**:
     Mide la diferencia media entre las predicciones del modelo y los valores reales, elevada al cuadrado.

     $$MSE = \frac{1}{M}\sum_{i=1}^{M}(real_i - estimado_i)^2$$

     *Ventajas*: Penaliza errores grandes, lo que puede ser útil si esos errores son particularmente problemáticos en el contexto del problema.
     *Desventajas*: Sensible a valores atípicos y la escala de la métrica depende de la variable objetivo.
   - **Error cuadrático medio (RMSE, Root Mean Squared Error)**:
     Mide la diferencia media entre las predicciones del modelo y los valores reales, elevada al cuadrado y con raíz cuadrada aplicada.

     $$RSME = \sqrt{\frac{1}{M}\sum_{i=1}^{M}(real_i - estimado_i)^2}$$

     *Ventajas*: Penaliza errores grandes, lo que puede ser útil si esos errores son particularmente problemáticos en el contexto del problema.
     *Desventajas*: Sensible a valores atípicos y la escala de la métrica depende de la variable objetivo.
   - **Error absoluto medio (MAE, Mean Absolute Error)**:
     Mide la diferencia media en magnitud absoluta entre las predicciones del modelo y los valores reales.

     $$MAE = \frac{1}{N}\sum_{i=1}^{N}|real_i - estimado_i|$$

     *Ventajas*: Menos sensible a valores atípicos en comparación con RMSE, interpretable en términos de la variable objetivo.
     *Desventajas*: No penaliza tanto los errores grandes como el RMSE.
   - **Error absoluto porcentual medio (MAPE, Mean Absolute Percentage Error)**:
     Mide el error absoluto medio como un porcentaje de los valores reales.

$$MAPE = \frac{100}{N}\sum_{i=1}^{N}|\frac{real_i - estimado_i}{real_i}||$$

*Ventajas*: Escala independiente, facilita la comparación entre diferentes series temporales y unidades.

*Desventajas*: Indefinido o no válido cuando el valor real es cero, sesgado hacia series temporales con valores bajos.

- **Coeficiente de determinación ($R^2$)**:
  Mide la proporción de la variabilidad total de los datos que se explica mediante el modelo.

$$FORMULA$$

*Ventajas*: Fácil de interpretar, valores entre 0 y 1, con 1 indicando un ajuste perfecto.

*Desventajas*: Puede ser engañoso en el caso de series temporales, ya que un alto $R^2$ no necesariamente implica buenas predicciones.

2. **Métricas para Variables Categóricas**

- **Exactitud (Accuracy) (ACC)**:
  Mide la proporción de predicciones correctas entre el número total de predicciones realizadas.

$$ACC = \frac{\#\quad Correct\quad Predictions}{\#Total\quad Predictions}$$

*Ventajas*: Fácil de interpretar, útil cuando las clases están equilibradas.

*Desventajas*: Puede ser engañoso en casos de clases desequilibradas.

- **Precisión**:
  Mide la proporción de verdaderos positivos identificados correctamente entre el número total de predicciones positivas realizadas por el modelo.

$$Precision = \frac{TruePositives}{TruePositives + FalsePositives}$$

*Ventajas*: Útil cuando es importante minimizar los falsos positivos.

*Desventajas*: No tiene en cuenta los falsos negativos.

- **Recall (Sensibilidad)**:
  Mide la proporción de verdaderos positivos identificados correctamente entre todos los verdaderos positivos reales.

$$Recall = \frac{TruePositives}{TruePositives + FalseNegatives}$$

*Ventajas*: Útil cuando es importante identificar correctamente los verdaderos positivos.

*Desventajas*: No tiene en cuenta los falsos positivos.

- **Especificidad**:

  Mide la proporción de verdaderos negativos identificados correctamente entre todos los verdaderos negativos reales.

$$Especifidad = \frac{TrueNegatives}{TrueNegatives + FalsePositives}$$

  *Ventajas*: Útil cuando es importante identificar correctamente los verdaderos negativos.

  *Desventajas*: No tiene en cuenta los falsos negativos.

- **F1-score**:

  La media armónica de la precisión y el recall, equilibrando la importancia de ambos.

$$F1 = \frac{2 * ACC * Recall}{ACC + Recall}$$

## 2.4. Preparación previa a la predicción

Antes de realizar las técnicas mencionadas anteriormente, es necesario garantizar la consistencia y precisión de los datos [35]. A continuación, se describen los métodos utilizados en este proceso:

- **Recolección de datos:** Llevada a cabo directamente por el propio servicio de Spotify y gracias a usuarios que nos han prestado sus datos. Además, se utiliza Selenium para extraer características adicionales de la web tunebat.com.

- **Exploración de datos:** Este proceso es necesario para comprender las características y la naturaleza básica de los datos, identificar errores, *missing values* o atípicos.

- **Detección de problemas en los datos:** Detecta problemas en los datos, cómo *missing values*, datos duplicados, errores tipográficos, inconsistencias en la codificación y formato. También incluye la eliminación de duplicados y el análisis de los riesgos de las decisiones tomadas.

- **Creación de variables y dataframes derivados:** Se crean nuevas variables a partir de las existentes para mejorar la comprensión del problema o facilitar el análisis, como la extracción de características de texto, agregación de datos temporales, creación de variables dummy, etc. Así como la creación de nuevos dataframes con estas nuevas variables u otros, transformando las series temporales irregulares en regulares.

### 2.4.1. Codificación de variables categóricas

La codificación de variables categóricas [36] convierte estas variables en representaciones numéricas para que los modelos de aprendizaje automático puedan trabajar con ellas (one-hot encoding, ordinal encoding, etc.).

- **One-hot encoding:**

  *Descripción:* Consiste en representar cada valor de una variable categórica como un vector binario en el que un único elemento es 1 y el resto son 0. Por ejemplo, si tenemos tres categorías A, B y C, se representarán como $[1, 0, 0]$, $[0, 1, 0]$ y $[0, 0, 1]$, respectivamente.

  *Ventajas:*

  - No introduce un orden artificial entre las categorías.
  - Funciona bien con la mayoría de los algoritmos de aprendizaje automático.

  *Desventajas:*

  - Puede aumentar significativamente la dimensionalidad de los datos, lo que puede llevar a un mayor tiempo de entrenamiento y riesgo de sobreajuste.
  - No es eficiente en términos de espacio de almacenamiento.

- **Label encoding:**

  *Descripción:* Consiste en asignar un número entero único a cada categoría de una variable categórica. Por ejemplo, si tenemos tres categorías A, B y C, podrían representarse como 0, 1 y 2, respectivamente.

  *Ventajas:*

  - Es simple y fácil de implementar.
  - Es eficiente en términos de espacio de almacenamiento.

  *Desventajas:*

  - Introduce un orden artificial entre las categorías, lo que puede ser problemático para algoritmos basados en la distancia o regresión.
  - No es ideal para algoritmos que no pueden manejar relaciones ordinales espurias entre las categorías.

- **Ordinal encoding:**

  *Descripción:* Es similar al *label encoding*, pero se utiliza cuando las categorías tienen un orden natural o jerárquico [37]. En este caso, los números asignados a cada categoría siguen el orden de las categorías. Por ejemplo, si tenemos tres categorías de ”bajo”, ”medio” y ”alto”, podrían representarse como 1, 2 y 3, respectivamente.

  *Ventajas:*

- Es simple y fácil de implementar.
- Conserva la información del orden jerárquico de las categorías.
- Es eficiente en términos de espacio de almacenamiento.

*Desventajas:*

- Asume un orden entre las categorías, lo que no es apropiado para variables categóricas sin orden.
- La distancia entre los números asignados puede no reflejar la verdadera distancia o similitud entre las categorías.

- **Embeddings:**

  *Descripción:* Los embeddings [38] son representaciones vectoriales densas y continuas de las categorías en un espacio de baja dimensión. Los embeddings se aprenden junto con el modelo de aprendizaje automático y capturan relaciones semánticas entre las categorías. Son comunes en el procesamiento del lenguaje natural, donde se usan para representar palabras o frases.

  *Ventajas:*

  - Pueden capturar relaciones semánticas complejas entre las categorías.
  - Reducen la dimensionalidad de los datos y son más eficientes en términos de espacio en comparación con el one-hot encoding.
  - Son adaptables, ya que se aprenden junto con el modelo.

  *Desventajas:*

  - No son interpretables fácilmente, ya que las representaciones vectoriales densas no tienen un significado claro.
  - Requieren un mayor tiempo de entrenamiento y más potencia de cómputo en comparación con otros métodos de codificación.
  - Son más apropiados para grandes conjuntos de datos, ya que se necesita suficiente información para aprender representaciones significativas.
  - Pueden ser sensibles a la arquitectura del modelo y los hiperparámetros, lo que puede afectar su eficacia en función del problema específico.

## 2.5. Aspectos Psicológicos del Trabajo

### 2.5.1. Psicología en los Descansos de Música

La música es una parte importante de la vida de muchas personas, pero el consumo excesivo de música puede tener un impacto negativo en nuestra salud mental. Por lo tanto,

es importante encontrar un equilibrio adecuado y tomar descansos regulares para permitir que nuestro cerebro se recupere y procese la música que hemos estado escuchando.

Existen diferentes tipos de descansos, y cada uno tiene un propósito específico en relación con el consumo de música:

- **Descanso muy breve (microdescanso): 5-15 minutos** Estos descansos pueden ser útiles para recuperarse rápidamente de la fatiga auditiva o para cambiar de enfoque mental. Sin embargo, si se necesita tomar múltiples microdescansos en un corto período, podría ser una señal de que la persona está usando la música como una distracción constante, lo que puede indicar un comportamiento compulsivo.
- **Descanso breve: 30 minutos - 1 hora**
  Estos descansos permiten una recuperación más completa de la fatiga auditiva y brindan la oportunidad de realizar otras actividades. Si los descansos breves son muy regulares, podría ser una señal de un consumo equilibrado de música. Sin embargo, si alguien necesita descansos breves con frecuencia, podría indicar dependencia o dificultad para concentrarse sin la música.
- **Descanso medio: 2-4 horas**
  Estos descansos son adecuados para permitir que el cerebro procese y asimile la música escuchada. Un patrón regular de descansos medios sugiere un consumo saludable de música. Si estos descansos son raros o irregulares, puede indicar un patrón de consumo compulsivo o adictivo.
- **Descanso largo: 6-12 horas**
  Un descanso de esta duración permite un período de recuperación más prolongado y puede ser especialmente útil si has estado escuchando música durante un período prolongado o a alto volumen. Tomar descansos largos regularmente puede ser un signo de un consumo equilibrado de música.
- **Descanso muy largo: 1-2 días**
  Estos descansos pueden ser beneficiosos para reducir la fatiga auditiva crónica o el estrés relacionado con la música. Si estos descansos son muy infrecuentes, podría ser una señal de dependencia o adicción a la música.
- **Descanso extenso: 3 días - varias semanas**
  Un descanso extenso puede ser necesario en casos de fatiga auditiva crónica, estrés o pérdida de interés en la música debido al consumo excesivo. La necesidad de un descanso extenso puede ser una señal de que ha habido un consumo inadecuado de música y puede indicar un patrón adictivo.

En resumen, la regularidad y duración de los descansos pueden proporcionar información sobre el consumo y la posible adicción a la música. Un patrón equilibrado de descansos de diferentes duraciones puede indicar un consumo saludable, mientras que la necesidad frecuente de descansos cortos o la necesidad de descansos extensos puede ser una señal de un comportamiento adictivo o compulsivo. La clave está en encontrar un

equilibrio que permita disfrutar de la música y, al mismo tiempo, mantener un estilo de vida saludable y equilibrado.

### 2.5.2. Conclusiones para predecir el tiempo de escucha en datos no lineales

El comportamiento humano es, en general, no lineal [39]. Los individuos son sistemas complejos y adaptativos que están influenciados por una multitud de factores, como las emociones, el entorno, las experiencias pasadas y las relaciones sociales, entre otros. Estos factores interactúan de maneras no lineales y a menudo impredecibles, lo que dificulta modelar y predecir el comportamiento humano.

Dada la naturaleza no lineal y compleja del comportamiento humano, es esencial utilizar enfoques y modelos apropiados para analizar y predecir dicho comportamiento en diferentes contextos. Estos modelos pueden incluir técnicas de aprendizaje automático, redes neuronales, teoría del caos, sistemas dinámicos y simulaciones basadas en agentes, entre otros.

Sin una cantidad considerable de datos, el uso de aprendizaje profundo (Deep Learning) no es justificado y, dada la naturaleza humana y no lineal en la que abundan los valores atípicos generados por emociones, el modelo ARIMA no es apropiado. Por lo tanto, nuestro enfoque consistirá en maximizar los resultados de las predicciones utilizando un algoritmo de aprendizaje automático, como el Random Forests. Para lograrlo, será necesario desarrollar y agregar variables, así como aplicar otras técnicas previamente mencionadas, con el fin de optimizar los resultados obtenidos.

# Capítulo 3

# Diseño del sistema

En esta sección, se presentan los datos utilizados y las diversas decisiones de diseño tomadas a lo largo del proyecto. También se discute cómo se obtuvieron y transformaron los datos originales, así como las herramientas empleadas y los métodos mencionados en el estado del arte.

## 3.1. Dataframe Inicial

El Dataframe original se obtiene a través de los servicios de la plataforma Spotify, la cual ofrece los datos de la cuenta al usuario en la página web de Spotify [1]. Gracias a la colaboración de varios voluntarios, quienes proporcionaron sus datos y consentimiento, hemos obtenido la información necesaria para este estudio, como se menciona en los agradecimientos.

Spotify también ofrece información detallada sobre cada variable y archivo de datos en la página web [2]. Para este trabajo de fin de máster, los datos de mayor interés se encuentran en la sección denominada "Extended streaming history". A continuación, se presentan las columnas del Dataframe original y sus respectivos tipos:

Para generar este Dataframe (DF) en Python, utilizamos la librería pandas. Este DF se compone de varios archivos JSON, por lo que lo primero que hacemos es leerlos todos y comprobar su compatibilidad, es decir, si tienen el mismo número de columnas, en el mismo orden, con el mismo tipo de datos y con el mismo nombre.

[1] https://www.spotify.com/us/account/privacy/
[2] https://support.spotify.com/us/article/understanding-my-data/

| Name | Data Example | Data Type |
|---|---|---|
| ts | 2017-04-08T03:34:25Z | object |
| username | JC | object |
| platform | samsung | object |
| ms_played | 51641 | int64 |
| conn_country | ES | object |
| ip | 79.146.217.142 | object |
| user_agent | unknown | object |
| master_metadata_track_name | El Rito | object |
| master_metadata_album_artist_name | Gondwana | object |
| master_metadata_album_album_name | Reggae N Roll | object |
| spotify_track_uri | spotify:track:2OPdYnaZV | object |
| episode_name | None | object |
| episode_name_show | None | object |
| episode_show_name | None | object |
| spotify_episode_uri | None | object |
| reason_start | trackdone | object |
| reason_end | endplay | object |
| shuffle | False | bool |
| skipped | NaN | float64 |
| offline | False | bool |
| offline_timestamp | 1593558698134 | int64 |
| incognito_mode | False | bool |

Tabla 3.1: Ejemplo de los datos originales, del dataframe sin tratar proporcionado por Spotify de un solo usuario

### 3.1.1. Limpieza de datos

Una vez que hemos cargado toda la información, de los diferentes archivos, en diferentes DFs, eliminamos los datos duplicados, los fusionamos en un único DF y volvemos a eliminar los duplicados.

De este modo, el Dataframe original que utilizamos consta de las siguientes características: 106745 entradas o filas y 21 columnas

Lo inspeccionamos con las funciones .info() y .describe() para observar que existen varias columnas con datos nulos y obtener información sobre las medias, desviaciones típicas y otras estadísticas en las columnas numéricas.

Es común encontrar datos nulos o erróneos en este tipo de proyectos. Lo primero que debemos hacer es analizarlos y buscar una forma de tratarlos. Primero, identificamos las columnas relevantes y las que no lo son. Al hacer esta inspección, nos damos cuenta de que hay dos grupos específicos de 3 columnas. Uno de estos grupos está reservado para la información de los streamings que son canciones, y el otro para los podcasts. Si simplemente eliminamos todas las filas que tienen datos nulos, perderíamos mucha

información relevante, ya que las canciones tienen datos nulos en las columnas de podcasts y viceversa.

La primera decisión que tomamos es unificar ambas informaciones en un único grupo de los dos mencionados anteriormente, en este caso, se hace en el grupo de canciones y eliminamos el otro grupo. Además, generamos una columna con valores booleanos, 0 si es canción, 1 si es podcast, llamada is_song.

Antes de eliminar los valores nulos restantes, eliminamos otras columnas que no sean de gran interés. Para ello, contamos el número de valores únicos que tiene cada columna y analizamos su relevancia en función de la información que proporcionan. Por ejemplo, la columna llamada Skipped solo contiene los valores 0 y 1, y observamos que cada valor se repite aproximadamente 4,000 veces, lo que significa que aproximadamente el 90 % de los datos en esta columna son nulos. Por lo tanto, concluimos que eliminar esta columna es beneficioso. Además, la información resumida que proporciona esta columna se puede encontrar de manera más detallada en la columna reason_end.

Repetimos este proceso para el resto de las variables y eliminamos aquellas que carecen de relevancia o son problemáticas debido a la falta de información que contienen. También aprovechamos para eliminar las canciones que se han reproducido durante menos de 12 segundos, una medida que hemos considerado óptima y que elimina muchos elementos que realmente no han tenido impacto en el usuario. Además, sabemos que Spotify trata de manera similar este tipo de reproducciones [3]

Otra transformación realizada es en los dispositivos de reproducción. En este caso, existen 53 dispositivos diferentes desde donde el usuario ha realizado streamings en la plataforma. Al observarlos detenidamente, podemos ver que estas reproducciones provienen de diversos dispositivos Android e iOS, lo que representa el 90 % de las reproducciones totales. Otro 5 % se ha realizado desde dispositivos Windows u ordenadores, y el último 5 % se asocia con otro tipo de dispositivos como tablets o televisiones. Por ello, convertimos estos 53 valores diferentes en 4 categorías: Teléfono, Ordenador, Dispositivos y Otros.

Después de limpiar y tratar todas las columnas de datos para obtener una mejor comprensión y reducir su complejidad, observamos que ya no existen valores vacíos o nulos. Las características de nuestro nuevo dataframe, que continúa siendo una STI, son: 83,000 entradas o filas y 10 columnas con una duración total de 7 años y 8 meses: de 2015-03-24 a 2022-11-24.

Esto representa una pérdida significativa de información, el 17 %, pero es un paso necesario para mejorar la calidad de los datos y facilitar el análisis posterior. Hemos eliminado las repeticiones en los datos, y además hemos eliminado valores vacíos, los cuales complican mucho las predicciones.

[3] `https://www.reddit.com/r/spotify/comments/rctp8i/how_spotify_wrapped_is_calculated_explained/`

## 3.2. Tratamiento de datos temporales

Ahora nos vamos a centrar en el apartado de información o características temporales, y vamos a extraer y analizar nuevas variables, todas ellas dependientes de la variable original TS.

Lo primero que se realiza es pasar esta variable a formato datetime para poder trabajar con ella de manera más sencilla. Además, le cambiamos el nombre a end_streaming, el cual es más representativo para la información que contiene.

La variable ms_played la transformamos a s_played, lo que también nos va a facilitar extraer nueva información de los datos.

La primera nueva variable o columna relevante que extraemos es la variable llamada start_streaming. Esta es extraída restando las dos mencionadas anteriormente: end_streaming - s_played.

La segunda variable fundamental que extraemos es el delta_time, es decir, extraemos la diferencia entre el inicio de un streaming y la finalización del anterior, es decir: end_streaming[0] - start_streaming[1]. El valor del primer delta_time se trata como 0, ya que no ha existido la reproducción de un elemento anterior a este. El resto se tratan como se ha mencionado. Esta variable contiene una variabilidad inmensa, alrededor de 273,686,380. Debido a esto, y como será la referencia para poder calcular y predecir el número de descansos que realizará el usuario y de qué duración, pasará a convertirse en una variable con 7 tipos de descansos diferentes, que englobarán todos los valores posibles que contiene actualmente.

También aprovechamos para crear características temporales más sencillas y estándar como son: año, estación, mes, mes-y-año, semana, semana-y-año, día, días-para-el-siguiente-mes, fecha, hora, minuto, mes, tiempo, tiempo-en-segundos. Cada variable nos será útil en futuros apartados, y algunas tendrán un mayor peso que otras. Así nos quedamos con un dataframe de 83,000 entradas o filas, 10 columnas, de las cuales 15 son nuevas características temporales extraídas.

Al mismo tiempo, encontramos varios problemas en la información y el DF. Estos problemas serán abordados y solucionados en las siguientes etapas del proceso de análisis y modelado.

### 3.2.1. Problemas

En primer lugar, podemos observar que existe un valor de un delta_time absurdamente alto, este valor es: 22,472,086, lo que pasado a días se traduce como: 260 días, y a meses como: 8,5 meses. Este es un claro punto de ruptura del DF, y es necesario tratarlo. Durante estos 8 meses de información faltante, el usuario ha podido cambiar por completo sus gustos y sus hábitos o patrones, pudiendo volverse irrelevante, o incluso contraproducente, analizar e intentar predecir en base a la información antes de este

punto.

Se plantea la solución de generar diferentes dataframes, donde se realicen las mismas predicciones para cada uno, sin embargo, se consideró una mejor opción eliminar o generar nueva información antes de este punto, y crear en el DF un nuevo punto de inicio.

Encontramos un total de 5 puntos con esta naturaleza. Estos 5 puntos tienen valores de: 40, 260, 211, 36 y 81 días sin escuchar música. Es lógico o razonable pensar que si un usuario no va a utilizar un servicio de suscripciones durante más de un mes, pase a anularla. Además, podemos observar cómo todos los puntos de esta naturaleza ocurren al principio del DF, entre el año 2015 y 2017, como podemos ver en 3.1 (a).

Ambas soluciones planteadas acerca de eliminar los datos antiguos y añadir los *missing values*, supondrían eliminar un 1 % de los datos o añadir un 1 % de datos sparse al dataframe.

Como tal, se considera que es mejor eliminar los datos a añadir datos sparse. De esta manera, el problema se soluciona eliminando la información anterior en el dataframe al último salto entre meses generados, es decir, si los meses donde se ha reproducido música son de la siguiente manera: 2022-12, 2023-01, 2023-03, 2023-04, 2023-05. El DF pasa a quedarse únicamente con los siguientes meses de información: 2023-03, 2023-04, 2023-05. Podemos ver como queda el dataframe en la figura como podemos ver en 3.1 (b).

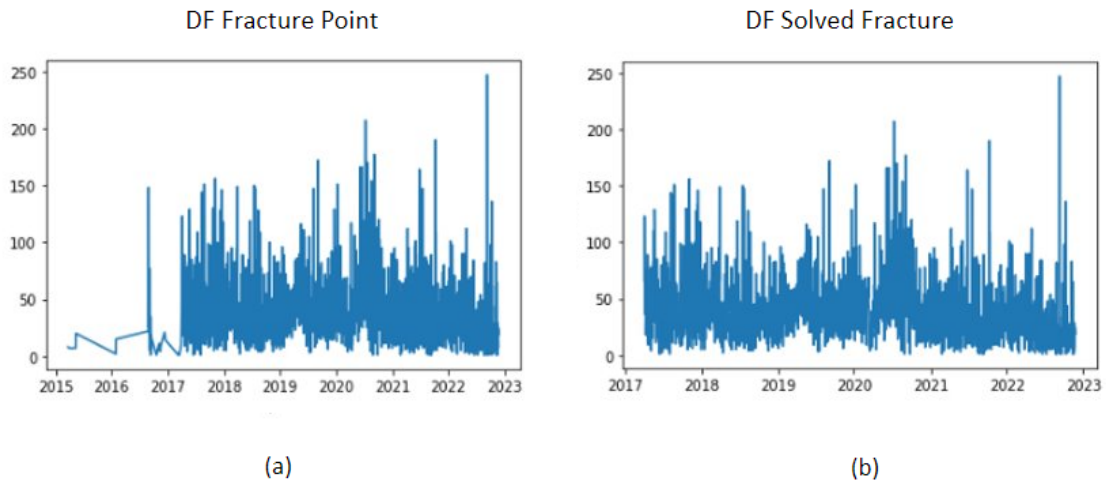



Figura 3.1: Ejemplo Visual de la ruptura temporal en la información recopilada de un usuario y cómo queda la información al solucionar este problema

A su vez, también aparece un nuevo problema, el problema de los delta_times negativos, es decir, existen streamings que han empezado antes de que acabase el streaming anterior, e incluso se han reproducido a la vez. Esto es claramente imposible, aún así, son un gran número de datos, por lo que se realizará una prueba y se comprobará por qué esto sucede.

PRUEBA DELTA NEGATIVOS

Es muy normal obtener datos corruptos o erróneos en la vida real, este proceso es completamente normal. Se toma la decisión de igualar los valores negativos a 0, es decir, como máximo el streaming se ha reproducido justo después, por lo que para no perder datos, pasamos de delta_time negativo a 0, solucionando el problema sin perder una cantidad de información relevante. De esta manera, nos aseguramos de mantener la integridad de los datos y mejorar la calidad del análisis posterior sin sacrificar una cantidad significativa de información. Estos ajustes son esenciales para garantizar que los modelos de predicción y análisis que se desarrollen a partir de estos datos sean precisos y útiles.

### 3.2.2. Representación Visual y Busqueda de Patrones

Ahora pasamos a plotear los valores obtenidos de delta time, de forma que podamos comprender mejor el modo de escuchar música del usuario. Además, una representación visual nos ayudará a comprender patrones de forma sencilla.

Vamos a plotear los valores por días, meses y años.

**Diario**:

La figura 3.2 nos representa el valor del número de streamings escuchadas cada día.

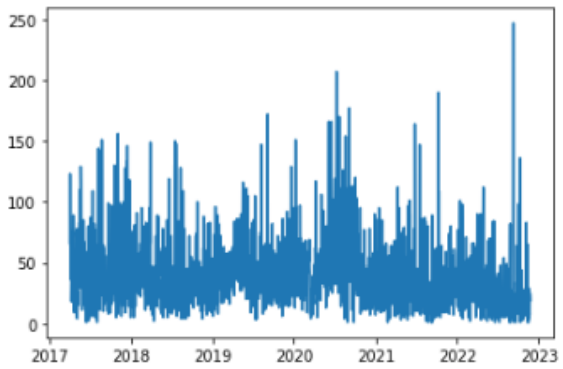



Figura 3.2: Ejemplo Visual del número de streamings realizados cada día

La figura 3.3 nos representa los diferentes valores de la variable delta_time en 3 días diferentes, representado cada uno por un color.

Como podemos observar, hay gráficas generadas muy similares, indicando que las probabilidades de encontrar patrones son altas. Además, aunque el número de picos en el día no son los mismos y varían, la suma total de sus valores no es tan dispar.

Si ploteamos un día encima de otro obtenemos la figura 3.4, para los 10 días:

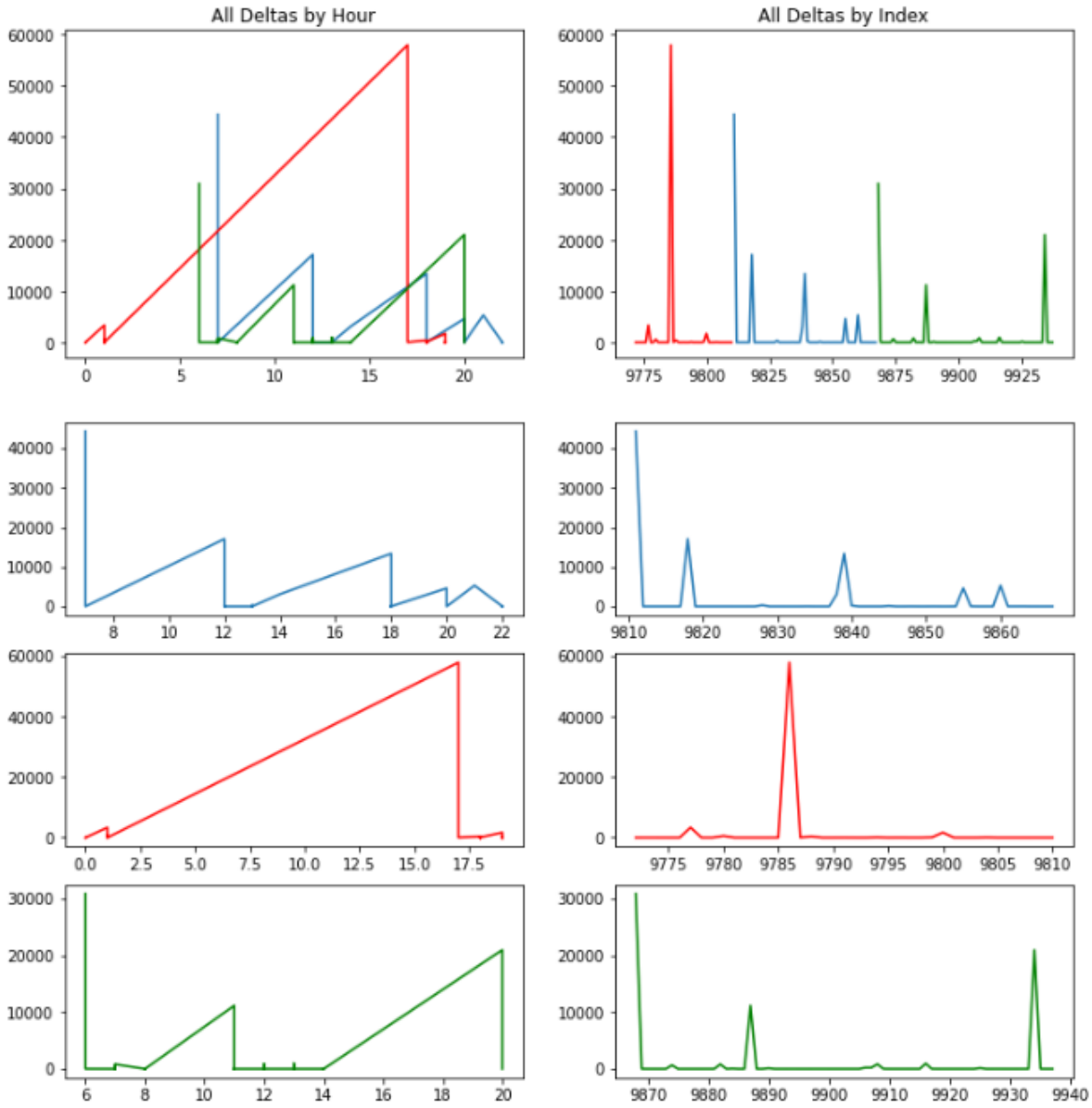



Figura 3.3: Ejemplo Visual de los valores de Delta_time para 3 días diferentes

**Mensual**:

La figura 3.5 nos representa el valor del número de streamings escuchadas cada mes del DF.

La figura 3.6 nos representa los diferentes valores de la variable delta_time en 3 meses diferentes, representado cada uno por un color:

Ahora ploteamos todos los meses juntos con diferentes colores pero sin suporponerse, uno seguido de otro en la figura 3.11, para todos los meses del DF:

Además, ploteamos un mes encima de otro superpuestos, obtenemos la figura 3.8, para todos los meses del DF:

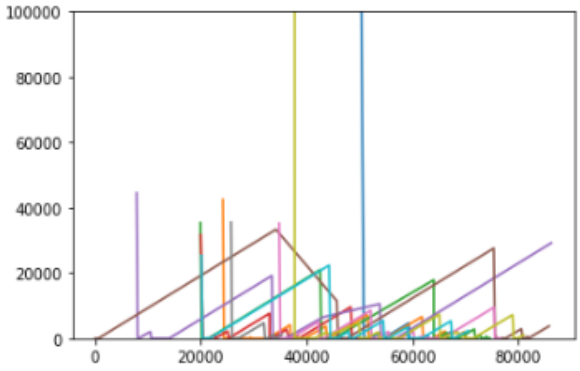



Figura 3.4: Ejemplo Visual de varios valores de Delta_Time diarios superpuestos

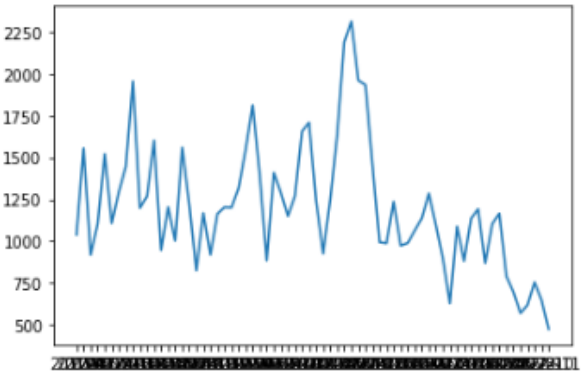



Figura 3.5: Ejemplo Visual del número de streamings realizados cada mes

**Anual**:

La figura 3.5 nos representa el valor del número de streamings escuchadas cada año del DF.

La figura 3.6 nos representa los diferentes valores de la variable delta_time en 3 años diferentes, representado cada uno por un color:

Ahora ploteamos todos los meses juntos con diferentes colores pero sin suporponerse, uno seguido de otro en la figura 3.11, para todos los meses del DF:

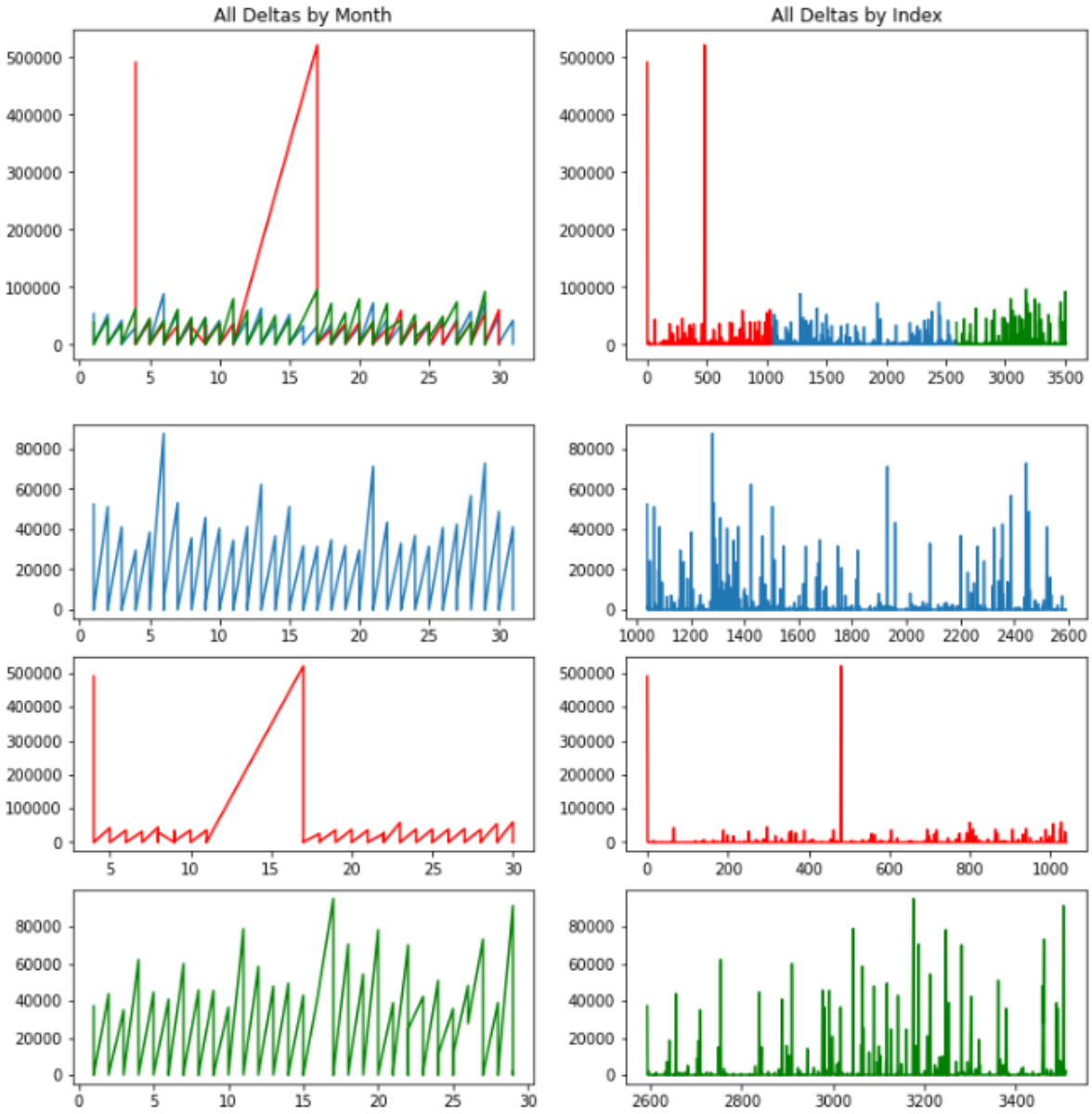



Figura 3.6: Ejemplo Visual de los valores de Delta_time para 3 meses diferentes

**Dataframes para Predicciones** Después de estos análisis y de la creación de nuevas variables, vamos a empezar a generar nuevos dataframes donde se realizarán las predicciones.

## 3.3. Dataframe Diario

Este dataframe surge al convertir el dataframe original, que es una STI, en una STR. Específicamente, contiene información sobre el número de canciones, podcasts, streamings y los nuevos elementos escuchados, uno por día: 2061 filas = nº días y

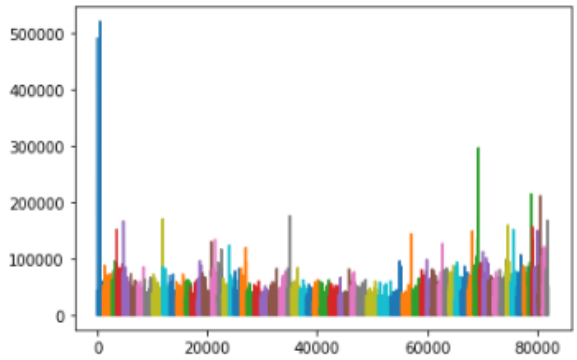



Figura 3.7: Ejemplo Visual de varios valores de Delta_Time mensuales

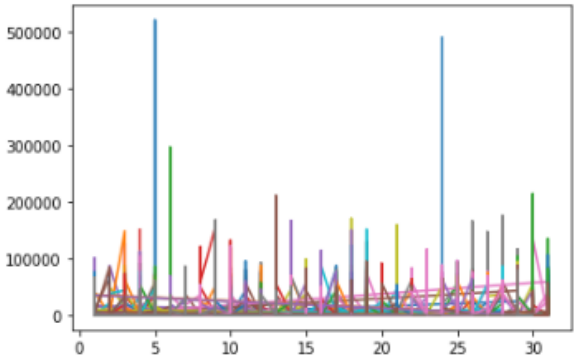



Figura 3.8: Ejemplo Visual de varios valores de Delta_Time mensuales superpuestos

6 columnas = *day_n_streaming, day_n_songs, day_n_podcast, day_new_streaming, day_new_songs, day_new_podcast* de 2017-04-04 a 2022-11-24.

En cuanto a las columnas mencionar que:

$$\#StreamsPredicted = \#Streaming - \#NewStreaming$$

De forma que utilizar cualquiera de ellas para predecir otra es "hacer trampas", ya que le estamos dando parte de la información al algoritmo.

**Corr matrix**

**Variabilidad**

**Sparcity**

- day_n_streaming: 0.01 - 28 ceros
- day_new_songs: 0.01 - 29 ceros
- day_new_streaming: 0.98 - 2016 ceros

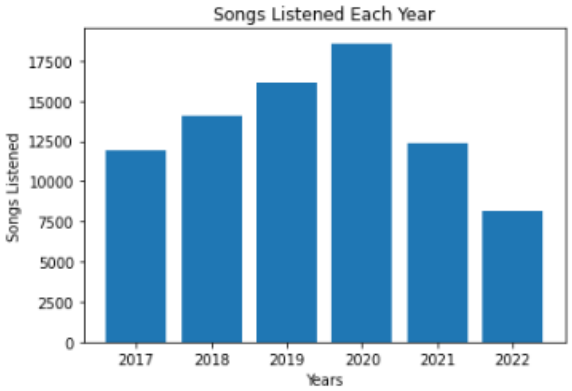



Figura 3.9: Ejemplo Visual del número de streamings realizados cada año

| Variable | Varianza | Std | Var_Coef |
|---|---|---|---|
| day_n_streaming | 806.53 | 28.39 | 0.71 |
| day_new_songs | 806.63 | 28.40 | 0.71 |
| day_new_streaming | 0.44 | 0.66 | 11.35 |
| day_n_podcast | 97.25 | 9.86 | 1.51 |
| day_n_songs | 97.12 | 9.85 | 1.51 |
| day_new_podcast | 0.02 | 0.14 | 8.80 |

Tabla 3.2: Varianza, Std y Coeficiente de la Varianza del dataframe de la serie temporal con datos diarios

- day_n_podcast: 0.27 - 547 ceros
- day_n_songs: 0.27 - 553 ceros
- day_new_podcast: 0.99 - 2032 ceros

Con estos resultados, entendemos que las variables n_streamings y n_songs son prácticamente idénticas, con una correlación extremadamente alta, de 1. Además, comprendemos la dificultad que tendrá la predicción debido a la varianza, y entendemos que para las variables dependientes de podcasts es mejor predecir con ceros debido a su dispersión.

**Proceso de creación:**

Para crear este dataframe a partir del original, se han agrupado el número de streamings por día, contando también por separado el número de podcasts y de canciones, así como los elementos que aparecen por primera vez ese día, para poder obtener la información de las tres variables llamadas "NEW".

De igual modo, para hacer más realista estos datos, se asume que las 3 primeras semanas, ningún elemento es nuevo, y todos han sido escuchados anteriormente. Esto se realiza para suavizar las gráficas, ya que durante el primer mes casi todos los elementos eran nuevos, lo que influye en la media y varianza móvil a la hora de realizar el EDA para la Serie Temporal Regular y su análisis de Estacionariedad.

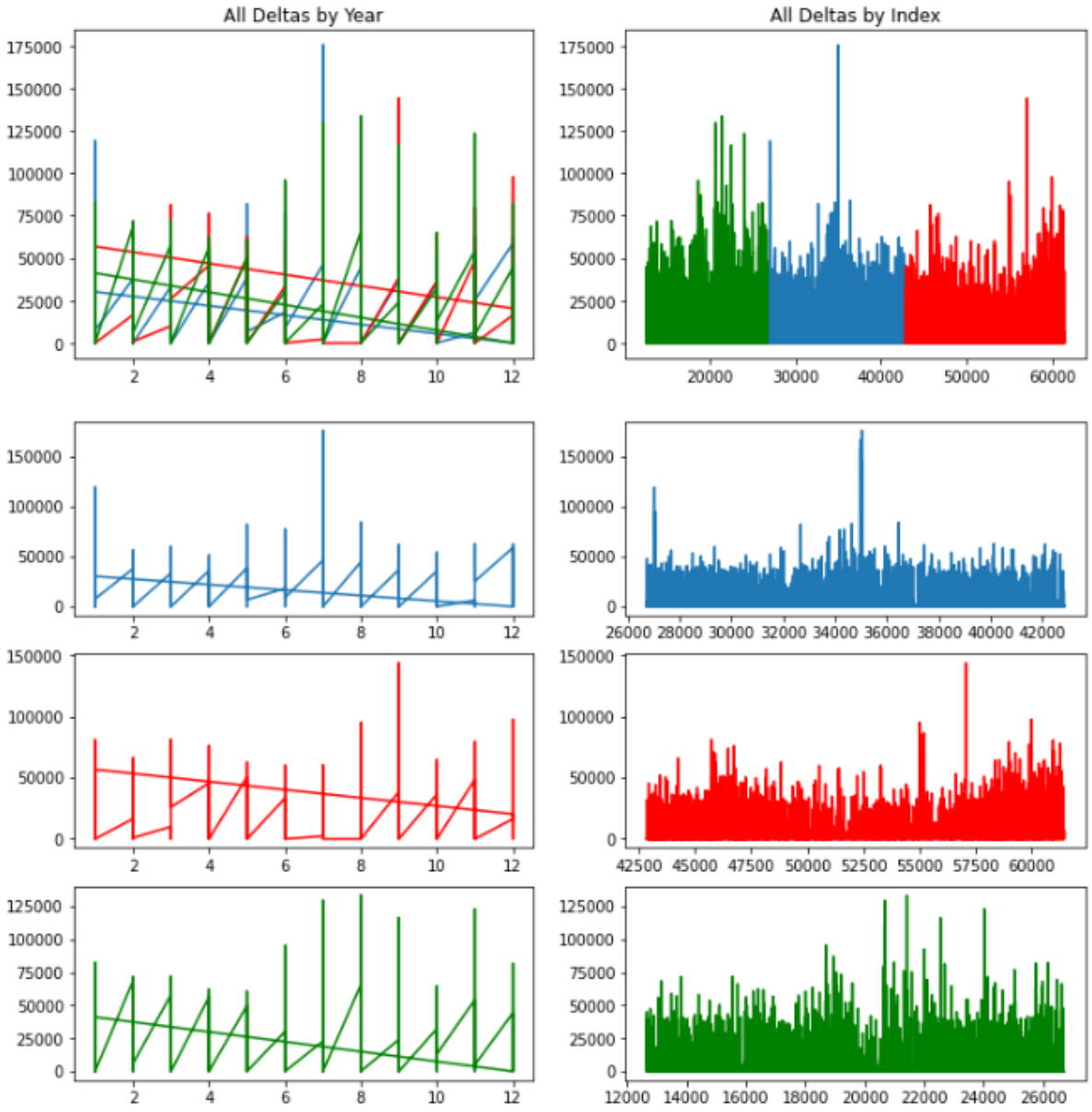



Figura 3.10: Ejemplo Visual de los valores de Delta_time para 3 años diferentes

Se asume que el usuario seguramente ya conoce sus primeros streamings de antes y no es la primera vez que los ha escuchado, solo está trasladando sus conocimientos a la nueva plataforma.

Además, hacemos uso del método de interpolación para completar los datos de los días en los que no se ha escuchado ninguna canción. Esto se realiza creando la lista de días entre el primero y el último. Una vez la tenemos, comprobamos qué días no están en el dataset e incluimos una fila con estos y con los datos inicializados a 0, ya que ese día el usuario no ha escuchado música.

De este modo, tenemos un dataframe de una Serie Temporal Regular completo en el cual podemos analizar y realizar predicciones.

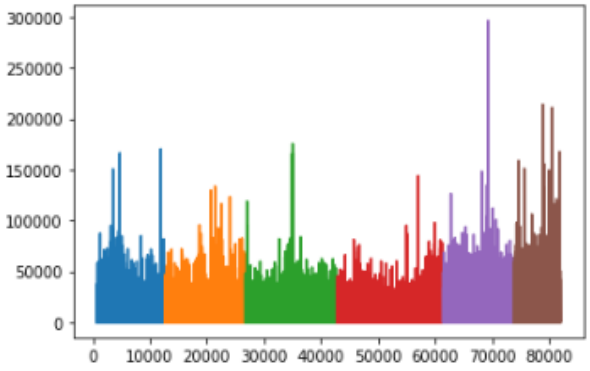



Figura 3.11: Ejemplo Visual de varios valores de Delta_Time anualmente

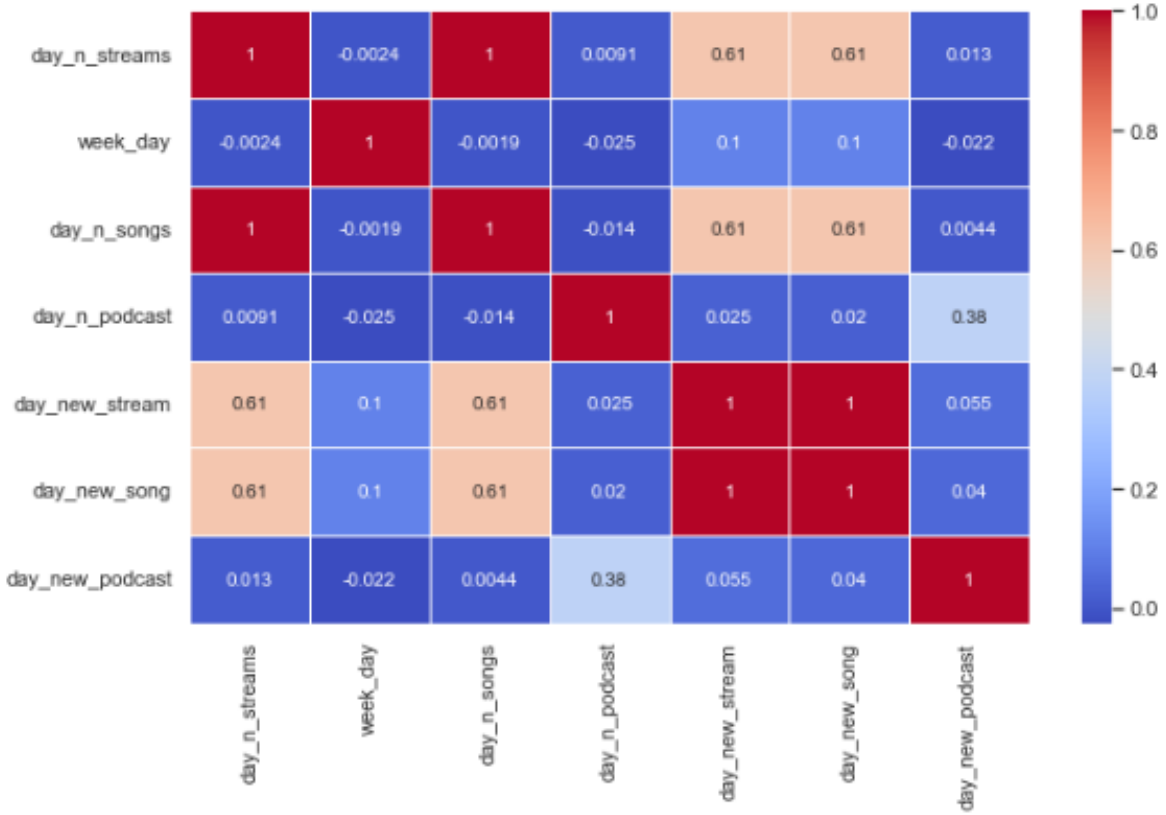



Figura 3.12: Daily DF Correlation matrix

24h Dataframe??

| Variable | Varianza | Std | Var_Coef |
|---|---|---|---|
| n_rests | 7.99 | 2.82 | 0.50 |

Tabla 3.3: Varianza, Std y Coeficiente de la Varianza del dataframe de la serie temporal con datos Numero de descansos

## 3.4. Dataframe Descansos

Este Dataframe es el resultado de contar el número de veces que el usuario deja de escuchar música por un tiempo superior a 10 minutos, estos descansos son mencionados en la sección 2.5. Contiene el número de descansos diarios que realiza el suuario, por lo que también es una STR:

2061 filas = nº días, 4 columnas = *n_rests, n_streams_listened, delta_time, total_time_listened*

Estas cuatro variables son muy similares de 2 a 2, ya que una calcula el número y la otra el tiempo de streams/descansos. Igual que se hace con el dataframe anterior, aquí la variable n_streams_listened puede servir para predecir cuantos streamings consumirá el usuario entre cada descanso del día. De forma que si en un día hace 6 descansos, realizando 6 predicciones de esta variable y sumandolas, daría el número de streamings en un día.

Como esto se ha calculado en el dataframe anterior, solo nos quedamos con la varibal n_rests, que es la variable objetivo a predecir.

| | rest_type | n_streams_listened | delta_time | total_time_listened |
|---|---|---|---|---|
| rest_type | 1.00 | -0.01 | 0.79 | -0.02 |
| n_streams_listened | -0.01 | 1.00 | -0.01 | 0.96 |
| delta_time | 0.79 | -0.01 | 1.00 | -0.01 |
| total_time_listened | -0.02 | 0.96 | -0.01 | 1.00 |

Figura 3.13: Daily DF Correlation matrix

**Variabilidad**

**Sparcity**

- n_rests: 0.01 - 28 ceros

Visual Dataframe + Dataframe

| Variable | Varianza | Std | Var_Coef |
|---|---|---|---|
| n_rests | 2.84 | 1.68 | 0.68 |

Tabla 3.4: Varianza, Std y Coeficiente de la Varianza del dataframe de la serie temporal con los tos de descansos

## 3.5. Rest Type

Este Dataframe se crea a partir del dataframe original, y gracias a este hemos podido crear el dataframe n_rests. Este dataframe es una STI, conteniendo cada descanso realizado del usuario, cuando empieza, (Last End Streaming Timestamp), y cuando termina (Next Starting Streaming Timestamp). Para saber que tipo de descanso es, mencionar que se han hechor eferencia en el estado del arte Sección 2.5. Obteniendo un dataframe de las siguientes dimensiones: 11504 filas = nº días, 3 columna = *delta_time, rest_type, week_day*

En este caso, únicamente se predice la variable rest_type, y se prediecn aproximadamente 6 tipos de descansos. 6 es la media de descansos que se realiza diariamente, extraida del dataframe anterior, pero en verdad se predice el número de descansos predecido desde el dataframe anterior.

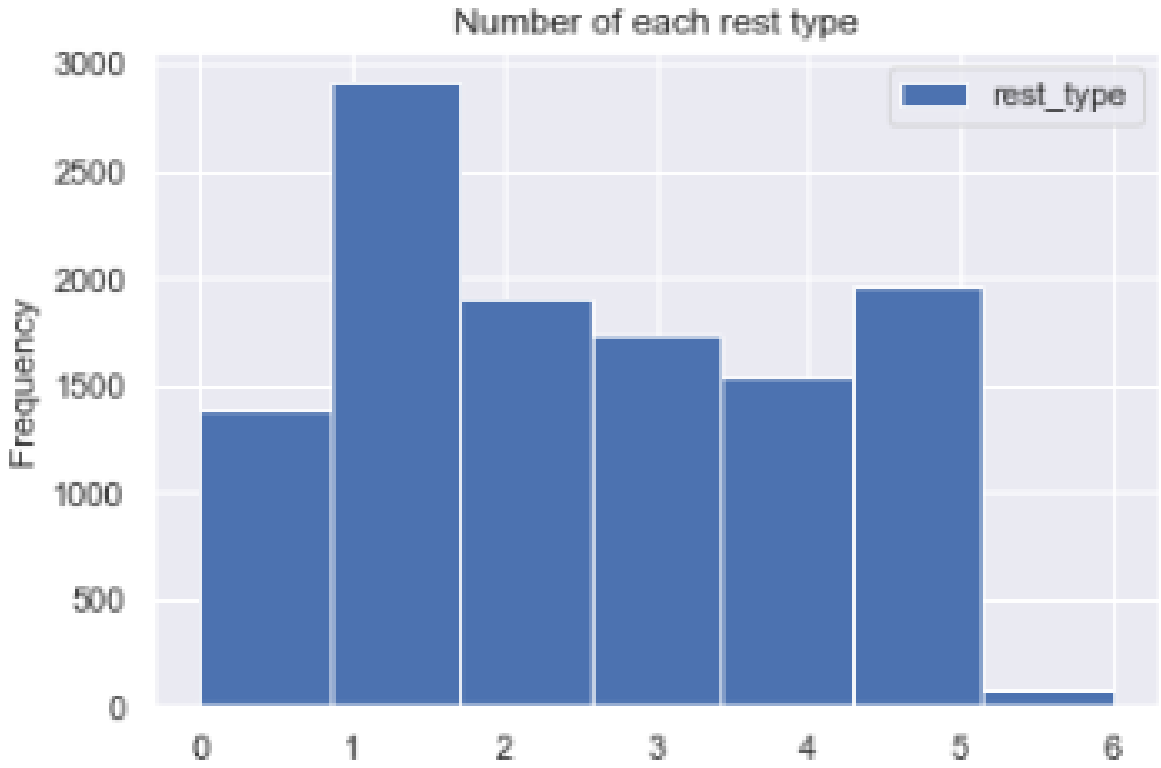



Figura 3.14: Type of Rests DF histogram

**Variabilidad**

| | rest_type | n_streams_listened | week_day |
|---|---|---|---|
| rest_type | 1.00 | -0.01 | 0.05 |
| n_streams_listened | -0.01 | 1.00 | 0.07 |
| week_day | 0.05 | 0.07 | 1.00 |

Figura 3.15: Matrix de correlación del DF Rest Type

En este caso, la varianza es menos útil porque

Visual Dataframe + Varianza

Gracias a esto, podemos considerar ciertas conductas que describen posibles estados de ánimo y salud mental en función de sus descansos o hábitos de consumo de música, desde una perspectiva psicológica. Algunos ejemplos son:

- Si un usuario realiza principalmente microdescansos y no toma descansos más largos durante un período extenso, podría estar utilizando la música como una distracción constante para evitar enfrentar sus emociones o problemas subyacentes. Esto puede conducir a una dependencia emocional de la música y un incremento de la ansiedad y la depresión.
- Si un usuario toma descansos cortos y regulares (30 minutos a 1 hora) con poca frecuencia, puede ser indicativo de una capacidad de concentración y atención saludable. Sin embargo, si el usuario tiende a no tomar descansos regulares o prolongados, podría estar consumiendo música de manera compulsiva o adictiva, lo cual puede impactar negativamente su salud mental a largo plazo.
- Si un usuario toma descansos prolongados de manera regular, podría ser una señal de que está cuidando su salud auditiva y mental al darle a su cerebro y oídos tiempo para descansar y recuperarse. Estos descansos pueden ayudar a prevenir la fatiga auditiva crónica y reducir el estrés relacionado con la música.
- Si un usuario toma descansos muy largos (1-2 días) regularmente, podría ser una señal de que se siente abrumado o estresado por la música. Es importante que el usuario sea consciente de cómo la música afecta su salud mental y trate de encontrar un equilibrio saludable en su consumo.
- Si un usuario toma descansos regulares y variados en términos de duración, puede ser indicativo de una relación saludable con la música. Estos descansos pueden ayudar a prevenir la fatiga auditiva crónica, reducir el estrés y mejorar la capacidad de concentración y atención.

- Si un usuario nunca toma descansos o toma descansos muy cortos e infrecuentes, puede ser una señal de adicción a la música. El usuario puede experimentar síntomas de abstinencia si no escucha música durante un período prolongado, lo que puede afectar negativamente su salud mental y emocional.
- Si un usuario toma descansos muy largos y frecuentes, puede ser una señal de que se siente desconectado o insatisfecho con su relación con la música. El usuario podría estar experimentando una pérdida de interés o placer en la música, lo que puede afectar su salud mental y emocional.

En resumen, los patrones de descanso en el consumo de música pueden ofrecer información valiosa sobre la salud mental y emocional de un usuario. Es crucial que los usuarios sean conscientes de sus patrones de consumo y tomen medidas para asegurar un equilibrio saludable entre la música y su bienestar psicológico. También es importante destacar que cada individuo es único y puede tener diferentes patrones de consumo y descanso con respecto a la música, por lo que siempre se recomienda buscar asesoramiento profesional en caso de preocupaciones o problemas relacionados con la salud mental y emocional.

CORROBORAR CON PROFESIONAL

## 3.6. Nombre de la canción

El último Dataframe en el que se realizan predicciones es un dataframe que consiste en la última semana del dataframe original, siendo este una STI. El porque solo contiene la última semana es explicado en el Anexo A - Churn Rate.

Además, parece ser la tendencia del usuario específico escuchar canciones de forma seguida, lo que significa que agregar más semanas aumentará mucho el número de valores únicos de la variable objetivo, que es categórica, aumentando así la complejidad del Dataframe, dificultando la transformación de las variables categóricas y obteniendo peores resultados aun cuando el número de datos sea mayor. El dataset queda del siguiente modo: 72 filas = canciones escuchadas en la última semana, 9 columnas = *platform, master_metadata_track_name, master_metadata_album_name, master_metadata_artist_name, reason_start, reason_end, is_song, s_played*, 1 semana de duración.

En este dataframe, vamos a predecir la variable 'master_metadata_track_name', que contiene el nombre del streaming realizado. Esta misma variable contiene un total de 36 valores únicos, lo que significa que tenemos la mitad de clases que datos, una mala noticia.

Para realizar la predicción de la variable objetivo de este dataframe, primero necesitamos transformar las variables categóricas en numéricas. De esta manera, y con un

dataframe tan pequeño, se ha decidido emplear las técnicas de one-hot-encoding y labeled encoding, descartando el método de ensemble por su dificultad y la falta de datos.

Después, realizaremos X predicciones, siendo X el número predicho anteriormente con el primer dataframe, siendo X el número de streamings escuchados el próximo día menos los nuevos streamings que se van a escuchar el próximo día: X = day_n_streams_pred - day_new_streams_pred

Además, al comprobar qué método proporciona los mejores resultados, se añadirán al dataframe nuevas características de la canción actual. Estas características se descargarán con el software de selenium[4], recolectados de una base de características de musica [5]

Para intentar mejorar aun mas los resultados. Estas características van del 0 al 100 menos en la última que va de -60 a 0, y son:

- Popularity: según el número y la actualidad de la reproducción de la pista.
- Energy: qué tan intensa y activa es la pista, según la entropía general, la velocidad de inicio, el timbre, el volumen percibido y el rango dinámico.
- Danceability: Qué tan apropiada es la pista para bailar según la regularidad general, la fuerza del ritmo, la estabilidad del ritmo y el tempo.
- Happiness: Que alegre y positiva es la pista.
- Acousticness: Cuán probable es que la pista sea acústica.
- Instrumentalness: La probabilidad de que la pista no contenga voces de palabras habladas.
- Liveness: La probabilidad de que la pista se haya grabado con una audiencia en vivo.
- Speechiness: Qué tan presentes están las palabras habladas en la pista.
- Loudness: La amplitud promedio de decibelios a través de la pista.

[4] `https://www.selenium.dev/`
[5] `https://tunebat.com/`

# Capítulo 4

# Experimentos y Resultados

Antes de nada, es importante mencionar que los experimentos han sido realizados en un ordenador portatil IdealPad L340 Gaminng, con 16 GB de RAM, 6 nucleos de 2,6 GHz, NVIDIA® GeForce® GTX 1650 de 8 GB.

Este proyecto y el procesado de datos ha podido llegar a ser excesivo para este equipo, teniendo problemas con la capacidad de la memoria RAM en los paratados Anexo A - Churn Rate, prediciendo el dataframe Rest Type y el dataframe Song Name.

## 4.1. EDA de Series Temporales

### 4.1.1. Daily Dataframe

En este dataframe vamos a mostrar el proceso de EDA o Exploratory Data Analysis para las variables objetivo: day_n_streams, day_n_songs, day_n_podcast, day_new_streams, day_new_songs, day_new_podcast

El primer paso después de comprobar que no hay datos nulos o repetidos, es visualizar los datos y comprobar si son estacionarios, ploteando también la rolling mean y std, y comprobando que es constante de principio a fín de la gráfica. Esto también es comprobable con diferentes tests estadísticos, entre ellos el test Augmented Dickey-Fuller (ADF)[1] y el test Kwiatkowski–Phillips–Schmidt–Shin[2](KPSS).

Los resultados de ambos test para estas varibales, exceptuando las 2 variables que contienen información sobre los elementos que son podcasts, es la conclusión de que son estacionarias para un *p-value* inferior a 0,05.

Al analizar las funciones de autocorrelación(ACF) y de autocorrelación parcial (PACF),

[1] `https://www.machinelearningplus.com/time-series/augmented-dickey-fuller-test/`
[2] `https://www.statisticshowto.com/kpss-test/`

observamos que para las variabels de canciones y de streamings **??** estos son muy importantes:

Investigando los valores de las funciones ACF y PACF [3], nos hacemos a la idea de los posibles valores a tener en cuenta para escojer un número de lags. Representaicón en figura 4.1

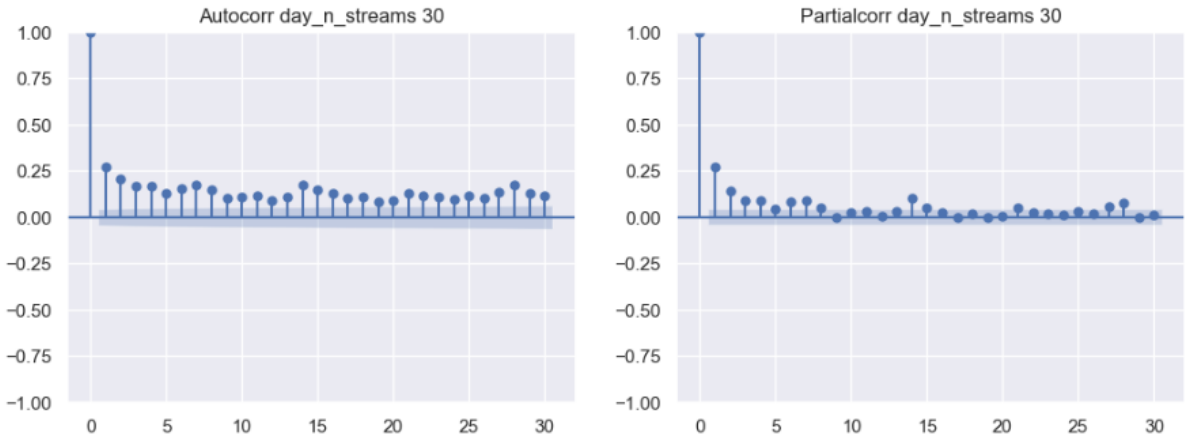



Figura 4.1: Plot de los valores de las funciones ACF y PACF para la variable N Streamings

En este caso, vamos a probar con hasta un total de 8 valores lag a la hora de realizar las predicciones, como indíca la PACF. Los procesos de EDA del resto de dataframes se realizan igual por lo que se explicarán en menor detalle, llegando a los resultados extraidos de forma mas agil.

### 4.1.2. Rest Number

Análisis de la variables *Rest Number* o número de descansos. Es una variable que recoge un dáto al día, sumando el numero total de veces que el usuario ha interrumpido una sesión de música.

El resutlado de los test estadísticos ADF y KPSS coinciden en su estacionariedad.

Plot de los valores de las funciones ACF PACF, figura 4.2:

Concluyendo que el mejor número de lags a tomar como el número 8, pudiendo llegar hasta 15.

### 4.1.3. Rests Type Dataframes

Análisis temporal de la variable Rest type para su futura predicción. Esta variable contiene el típo de descanso que se ha realizado, según el tiempo que ha pasado desde

[3] `https://finanzaszone.com/analisis-y-prediccion-de-series-temporales-con-r-iii-autocorrelacion/`

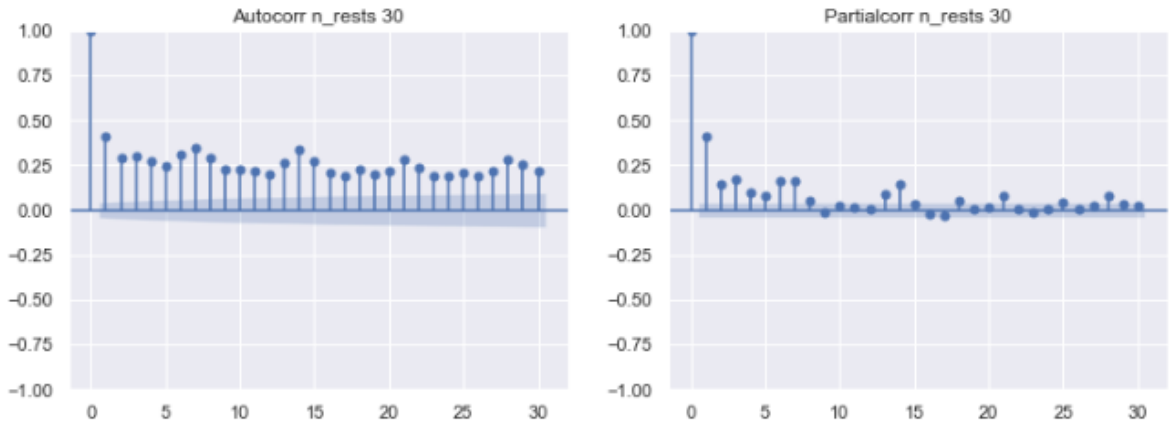



Figura 4.2: Plot de los valores de las funciones ACF y PACF para la variable objetivo N Rests

que escucho el último streaming, hasta que vuelve a reproducir un nuevo streaming.

En este caso, al ser una variable categórica, para saber si es estacionaria o no, es necesario realizar la prueba de xi cuadrado, explciada a continuación:

1. Definir un intervalo de tiempo (1 año).
2. Contar la frecuencia de cada categoría en cada intervalo.
3. Calcular la frecuencia esperada de cada categoría asumiendo que la distribución es constante en el tiempo.
4. Aplicar la prueba xi-cuadrado utilizando la frecuencia observada y esperada de cada categoría.

Se deja un ejemplo de como realizar esta prueba y un código de ejemplo, en el Anexo B, en el apartado del dataframe correspondiente. En este caso la série es estacionaria. El resultado es que es estacionaria.

Plot los valores de la ACF y PACF rest_type, figura 4.4:

Como anteriormente, los lags son muy importante, se toma el número máximo de 15 lags para realizar la investigación.

### 4.1.4. Song Name Dataframe

2 semanas de datos, 72 valores, no son suficientes para realizarlo

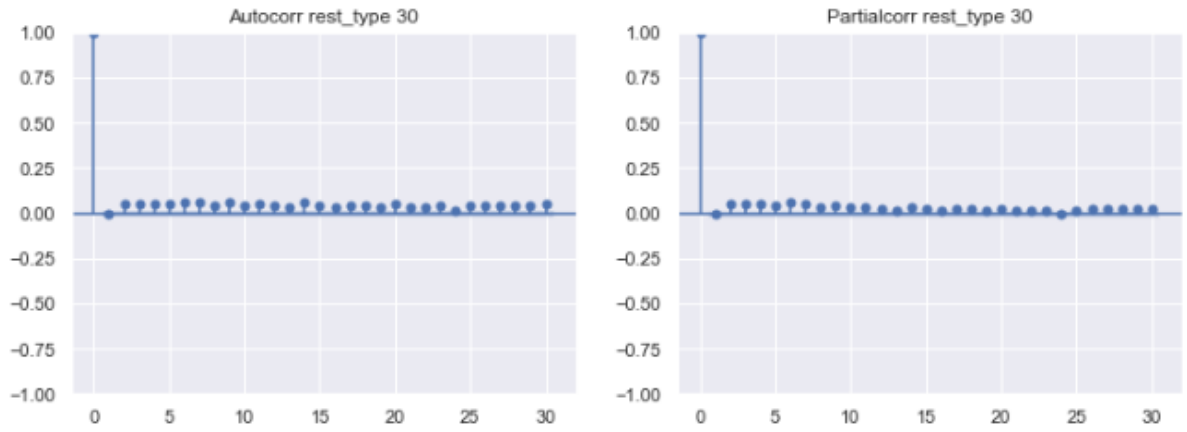



Figura 4.3: Plot de las funciones ACF y PACF de la variable objetivo Rest Type

Esta serie no es estacionaria. es lógico ya que estos valores no se mantienen constante en el timpo

Plot los valores de la ACF y PACF rest_type, figura 4.4:

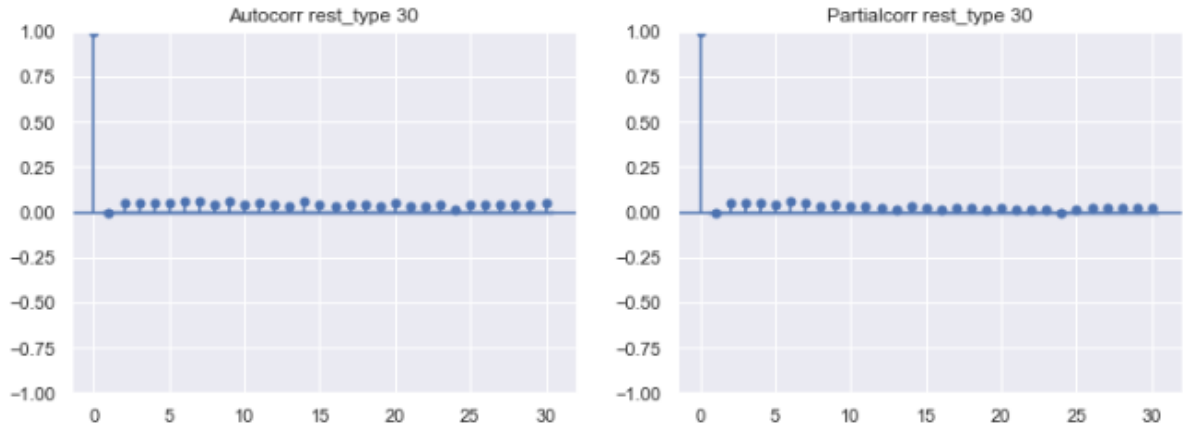



Figura 4.4: Plot de las funciones ACF y PACF de la variable objetivo Song PACF

Para este dataframe vemos que los lags son menos importantes, por lo que seguiremos probando con 8.

## 4.2. Predicciones

El método *Baseline* utilizado es el metedo Dummy Regresors. Este método consiste en crear el lag 1 de la variable objetivo o de las variables objetivo, y asumir que esté es el valore de la predicción actual.

| Eva | streams | songs | podcasts | new_streams | new_songs | new_podcasts |
|---|---|---|---|---|---|---|
| NONE MAE | 2 | 2 | 0 | 4 | 4 | 0 |
| TSCV MAE | 22,68 | 22,59 | 0,45 | 7,04 | 7,01 | 0.09 |
| BCV MAE | 20,45 | 20,45 | 00,9 | 5,95 | 5,95 | 0 |

Tabla 4.1: Resultados de la realización de Dummy Regressors para las variables del DF Diario

En cuanto a métodos de cross validation se utilizan los que se mencionan en el apartado de estado del arte: TSCV y BCV.

Los resultados de las predicciones se muestran con 2 decimales, exceptuando las medias que se muestran con tres, igual que si se consideran poco representativo los resultados con 2 decimales debido a que muchos valores tengan los mismos valores.

### 4.2.1. Predicciones Dataframe Diario

**Variables a predecir**: Vamos a predecir, por separado, las variables: day_n_streams, day_n_songs, day_n_podcast, day_new_streams, day_new_songs, day_new_podcast

**Método de predicción**: Las predicciones para este Dataframe se realizan con Random Forest Regressor de la librería sklearn, y la función .predict(). Además, el horizonte de predicción o los valores futuros a predecir es de uno.

Para obtener los resultados se realiza la media de un total de 75 simulaciones con 14 splits para los métodos de CV (siendo cada split aproximadamente de 3 meses de datos de entreno en BCV):

**Resultados**:

**Baseline**: Para una mejor representación visual de las tablas que encontramos a continuación, se elimina parte del nombre de las variables predichas, como la palabra day.

Resultado de la predicción Dummy Regressors:

**Lags**: En este apartado, se busca el mejor número de lags posibles, según a su ACF y PACF mecionada en el apartado anterior, intentando no incrementar en exceso los costes de cmputación.

De esta manera, para cada variable del DF original, se genera una lista de DF. Esta lista de listas de DFs, contiene los n_lags DFs de la primera variable. La primera posición de la lista contiene únicamente un lag, el segundo contiene los lags número uno y número 2, así van incrementando hasta el último que contiene los n primeros lags.

Ejemplo Tabla 4.2:

| **Streams** |
|---|

| **Streams** | **Streams_lag1** |
|---|---|

| **Streams** | **Streams_lag1** | **Streams_lag2** |
|---|---|---|

| **Streams** | **Streams_lag1** | **...** | **Streams_nlag** |
|---|---|---|---|

Tabla 4.2: Esta es la forma en la que se generan las diferentes variables lags, de un modo aditivo, para así poder comparar los diferentes modelos

| **Var** | **Eva** | **lag1** | **lag2** | **lag3** | **lag4** | **lag5** | **lag6** | **lag7** | **lag8** | **mean** |
|---|---|---|---|---|---|---|---|---|---|---|
| V1 | MAE - NONE | 25,53 | 20,81 | 23,51 | 23,19 | 23,56 | 23,07 | 22,76 | 22,97 | 23,17 |
| | MAE - TSCV | 25,41 | 21,12 | 23,74 | 23,46 | 23,60 | 23,42 | 22,89 | 22,94 | 23,30 |
| | MAE - BCV | 24,23 | 16,03 | 19,31 | 14,46 | 27,12 | 21,25 | 21,25 | 18,28 | 20,24 |
| V2 | MAE - NONE | 25,35 | 22,79 | 22,88 | 23,45 | 23,41 | 23,20 | 22,92 | 23,11 | 23,38 |
| | MAE - TSCV | 25,18 | 23,24 | 22,89 | 23,53 | 23,75 | 23,29 | 22,92 | 23,23 | 23,50 |
| | MAE - BCV | 24,23 | 16,37 | 19,5 | 13,89 | 26,71 | 21,16 | 21,4 | 19,53 | 19,53 |
| V3 | MAE - NONE | 6,45 | 5,77 | 5,47 | 8,08 | 6,22 | 6,42 | 6,42 | 6,46 | 6,41 |
| | MAE - TSCV | 6,46 | 5,76 | 5,47 | 8,14 | 6,25 | 6,45 | 6,97 | 6,47 | 6,49 |
| | MAE - BCV | 6,09 | 6,24 | 4,96 | 6,01 | 5,94 | 9,49 | 8,37 | 7,66 | 6,84 |
| V4 | MAE - NONE | 6,54 | 5,97 | 5,95 | 8,59 | 6,52 | 6,35 | 6,77 | 6,36 | 6,62 |
| | MAE - TSCV | 6,54 | 6,00 | 5,98 | 8,62 | 6,61 | 6,36 | 6,81 | 6,39 | 6,66 |
| | MAE - BCV | 6,11 | 6,32 | 4,9 | 5,99 | 5,92 | 9,39 | 8,33 | 7,87 | 6,853 |

Tabla 4.3: Resultados de la predicción con Random Forest Regressor para las variables del DF y con los lags generados apra encontrar el número óptimo

Con la función de la libreria sklearn y de Random Forest, .variable_importance() podemos observar la importancia de las variables 4.5. A continuación, se muestra un ejemplo de los 7 lags de una de las variables objetivos.

Ahora mostramos la media de las 75 simulaciones para los resultados obtenidos en cada número de lags en la Tabla 4.3. Las variables pasan a llamarse V1, V2... en el orden en el que se han presentado al principio de este subapartado para estilizarla. Además, como hemos visto que predecir para las 2 variables de podcasts es infructuoso, estas variables son eliminadas y se da el método baseline o se asume que la predicción tendrá el valor de 0 como método válido.

**Predicción Lags Daily RF**:

En este caso, los mejores lags son para la variables 1 y 2, el lag 4. Para la variables 3 y 4, el lag 3. En cuanto al método de cross validation, se observa que da mejores resultados el método BCV en las variables 1 y 2, mientras que el TSCV da mejores resultados en las variabels 3 y 4. Esto puede deberse a que las dependencias de las 2 primeras variables son mas a corto plazo, con una mayor variabilidad, mientras que las dependencias temporales

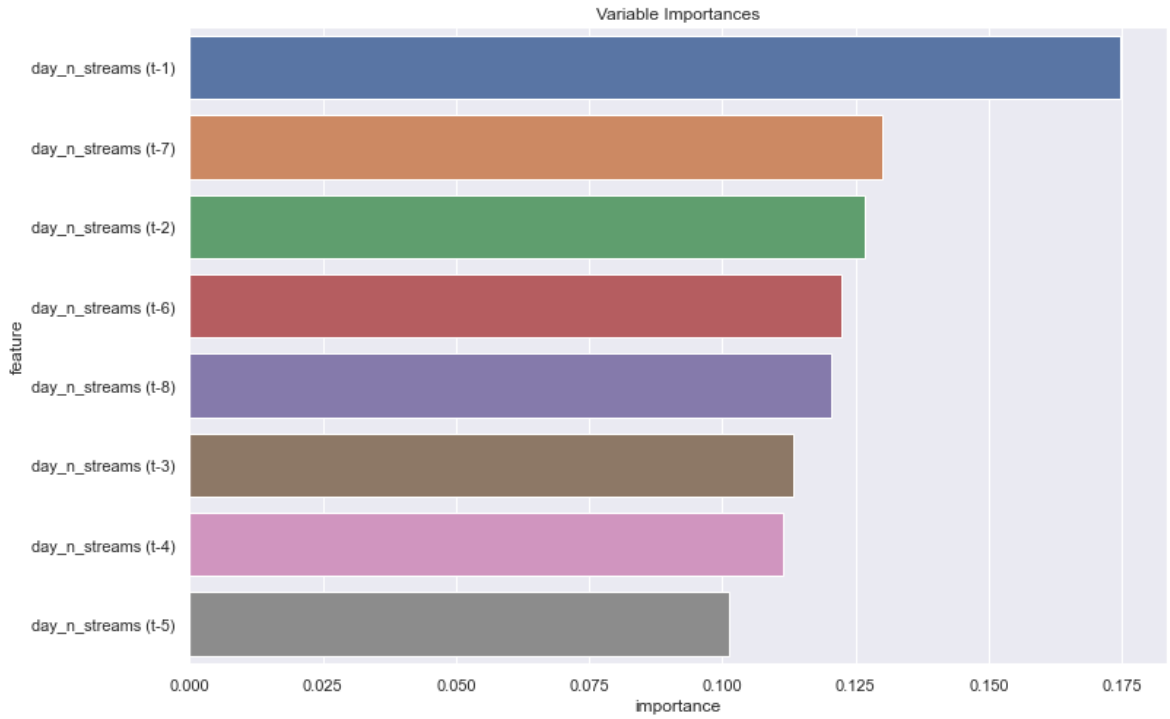



Figura 4.5: Ejemplo visual de la importancia dada por el algoritmo random forest a cada variable según lo útiles que resultan ser para hacer la predicción de la variable objetivo

de las variables 3 y 4 son de un mayor plazo temporal.

Como la diferencia de mejora entre TSCV y BCV en las diferentes variabels es bastante grande, mientras que en TSCV solo mejora un 0,5 en las variables 3 y 4, con BCV vemos una mejora de hasta 9 en cuanto al método de evaluación MAE, utilizaremos este método a la hora de realizar el hyperparameter tunning.

**Tiempos de ejecucción:** Mientras que el algoritmo RF sin CV tarda apenas 12 minutos, con TSCV ha tardado 5 hora y con BCV una.

**TF**: De igual modo, en esta parte generamos varios sets de variables temporales para intentar mejorar los resultados.

El conjunto con todas las nuevas variables contiene las variables: Año, cuarto, mes, semana, dia, dia del año, dia de la semana, es final de mes, es principio de mes, es final de cuarto, es principio de cuarto, es principio de año.

El segundo set con menos variables contiene las variables: Año, cuarto, mes, semana, dia, dia del año, dia de la semana

El tercet set de variables temporales, unicamente contiene las variables: dia, dia del año, dia de la semana, año.

Estos sets se han generado, reduciendo las variables que menos importancia daba el propio algoritmo Random Forests. Se observa en la Figura 4.6.

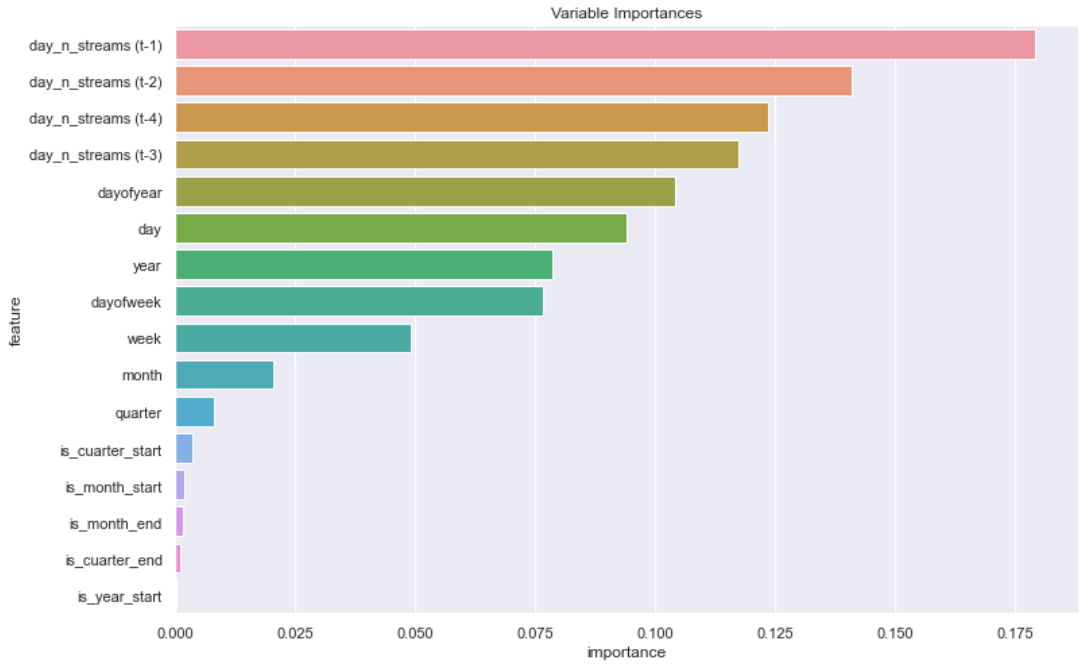



Figura 4.6: Ejemplo visual de la importancia dada por el algoritmo random forest a cada variable temporal según lo útiles que resultan ser para hacer la predicción de la variable objetivo

**Predicción TF**:

**Tiempo de computación:** En este caso, el tiempo de computo para RF sin CVs es aproximadamente de 2 minútos, mientras que con TSCV sube a 50 minútos y con BCV son 8 minutos.

En el anexo C se encuentran los resultados obtenido de transformar el mejor set de TF a TF cíclicas, ya que el día 31 está igual de cerca del día 1, que el día 2.

**Hyperparameter tunning:** Realizando el hyperparameter tunning para el mejor dataframe y las mejores features, extraemos los siguientes resultados:

**N Streams:** Mejor RandomForestRegressor: RandomForestRegressor(max_features='sqrt', min_samples_leaf=2, n_estimators=10) Mejores parámetros: {'bootstrap': True, 'max_depth': None, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 2, 'n_estimators': 10} Error absoluto medio (MAE): 11.001334119310078

**N Songs** Mejor RandomForestRegressor: RandomForestRegressor(bootstrap=False, max_depth=10, max_features='sqrt', min_samples_leaf=2, n_estimators=10) Mejores parámetros: 'bootstrap': False, 'max_depth': 10, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 2, 'n_estimators': 10 Error absoluto medio (MAE): 11.85062418896514

**New Streams** Mejor RandomForestRegressor: RandomForestRegressor(max_depth=10, max_features='auto', min_samples_leaf=4, n_estimators=10) Mejores parámetros: {'boots-

| Var | Eva | TFSet1 | TFSet2 | TFSet3 | mean |
|---|---|---|---|---|---|
| V1 | MAE - NONE | 20,77 | 20,60 | 20,68 | 20,68 |
| | MAE - TSCV | 21,79 | 21,54 | 21,64 | 21,65 |
| | MAE - BCV | 15,49 | 15,38 | 15,28 | 15,38 |
| V2 | MAE - NONE | 21,19 | 20,95 | 21,02 | 21,05 |
| | MAE - TSCV | 22,01 | 21,62 | 21,82 | 21,81 |
| | MAE - BCV | 15,44 | 15,23 | 15,21 | 15,29 |
| V3 | MAE - NONE | 6,28 | 6,21 | 6,18 | 6,22 |
| | MAE - TSCV | 6,15 | 6,14 | 6,25 | 6,18 |
| | MAE - BCV | 3,64 | 3,65 | 3,6 | 3,63 |
| V4 | MAE - NONE | 6,19 | 6,14 | 6,14 | 6,15 |
| | MAE - TSCV | 6,11 | 6,12 | 6,15 | 6,12 |
| | MAE - BCV | 3,71 | 3,66 | 3,63 | 3,66 |

Tabla 4.4: Resultados de la predicción con Random Forest Regressor y diferentes métods CV para todos los sets de Time Features del DF y con el mejor número de lags encontrado

| Var | Eva | NONE | TSCV | BCV |
|---|---|---|---|---|
| Number of rests | MAE | 1.00 | 1.95 | 2.44 |

Tabla 4.5: Resultados de la predicción con Dommy Regressor Base, TSCV y BCV sobre el número de descansos del próximo día

trap': True, 'max_depth': 10, 'max_features': 'auto', 'min_samples_leaf': 4, 'min_samples_split': 2, 'n_estimators': 10} Error absoluto medio (MAE): 4.684260727837251

**New Songs** Mejor RandomForestRegressor: RandomForestRegressor(max_features='sqrt', n_estimators=10) Mejores parámetros: {'bootstrap': True, 'max_depth': None, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 10} Error absoluto medio (MAE): 2.6335762876579203

**Hemos conseguido reducir satisfactoriamente el MAE inicial hasta la mitad.**

### 4.2.2. Predicciones Dataframe Number of Rests

La predicción realizada es la mísma que para el dataframe anterior, un paso en el futuro con random forest regressor.

**Predicciónes Baseline Method**:

Ahora pasamos a investigar los resultados obtenidos con las variabels algs generadas. En este caso se han generado 15 lags, por temas de limpieza solo mostraremos los mas significativos, los 10 primeros, y la media (M):

**Predicciónes Random Forest - Best lags**:

| NRests | CV | lag1 | lag2 | lag3 | lag4 | lag5 | lag6 | lag7 | lag8 | lag9 | lag10 | M |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MAE | NONE | 2.43 | 1.94 | 1.97 | 2.02 | 1.99 | 1.85 | 1.71 | 1.77 | 1.83 | 1.84 | 1,891 |
| | TSCV | 2,43 | 1,94 | 1,97 | 2,03 | 1,99 | 1,87 | 1,73 | 1,78 | 1,84 | 1,85 | 1,898 |
| | BCV | 2,4 | 1,74 | 2,03 | 1,86 | 1,95 | 2,31 | 1,61 | 1,57 | 1,98 | 2,23 | 2,00 |

Tabla 4.6: Resultados de la predicción con Random Forest Regressor Base, TSCV y BCV sobre el número de descansos del próximo día

| Rest_Type | TF0 | TF1 | TF2 | M |
|---|---|---|---|---|
| NONE | 1.760 | 1.757 | 1.758 | 1,758 |
| TSCV | 1.745 | 1.752 | 1.755 | 1,750 |
| BCV | 1.760 | 1.757 | 1.758 | 1,758 |

Tabla 4.7: MAE Resultados de la predicción con Random Forest Classifier Base, TSCV y BCV sobre el Tipo de descansos del próximo día

Como se puede observar, El mejor resultado observado en los MAES es para el LAG nº8 con el método BCV.

En cuanto a los tiempo de ejecución tenemos que: RF sin CV tardó 9 minutos, RF TSCV tardó 2 horas 30 minutos, RF BCV tardó 33 minutos.

**Predicciónes Random Forest Regressor - Best Time Features**:

En cuanto a los tiempos de ejecución tenemos que: RF sin CV tarda 1 minuto, RF TSCV tarda 17 minutos, RF BCV tarda 3 minutos.

De nuevo, los mejores resultados son unicamente con las variables lags.

**Hyperparameter tunning:** Mejor RandomForestRegressor: RandomForestRegressor(bootstrap=False, max_depth=50, max_features='sqrt', min_samples_split=5, n_estimators=10) Mejores parámetros: {'bootstrap': False, 'max_depth': 50, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 5, 'n_estimators': 10}

Error absoluto medio (MAE): 0.4034502354278292

### 4.2.3. Predicciones Dataframe TYPE of Rests

En este caso, la predicción que vamos a realizar es con RF Classifier, varios pasos en el futuro. En caso de tener que predecir multiples variables se utiliza multioutput. El número de pasos futuros que se predicen se determina por la predicción obtenida del DF anterior, o con la media de la variable N_Rests del DF anterior. En este caso se predicen 6 pasos en el futuro, y la métrica de evaluación seleccionada es el *Accuracy* (ACC)

Estas variables han sido codificadas con Ordinal Encoding, debido a la creciente componente temporal por la que se diferencian. Para tomar esta decisión también ha influido

| Rest_Type | Acc |
|---|---|
| NONE | 0.666 |
| TSCV | 0.208 |
| BCV | 0.182 |

Tabla 4.8: Resultados de la prediccion base con el método Dummy Regressors para la variable Rest Type

| Rest_Type | Acc |
|---|---|
| NONE | 0.222 |
| TSCV | 0.119 |
| BCV | 0.127 |

Tabla 4.9: Resultados de la predicción con Random Forest Classifier Base, TSCV y BCV sobre el Tipo de descansos del próximo día

la capacidad y el tiempo de computo.

**Rest Typre - Dummy Regressors - Baseline Prediction**:

**Predicciónes Random Forest Classifier Baseline**:

Ahora mostramos los resultados obtenidos con las variables lags generadas. Se toman en cuenta los 15 primeros lags, por temas de limpieza se muestran los lags mas significativos, los últimos 10 lags, junto con la media (M), en la Tabla 4.10.

**Predicciónes Random Forest Classifier - Best Lags**:

El mejore resultado lo dan los últimos lags, en concreto el lag 15. En este caso, da mejore resultado TSCV que BCV.

Ahora pasamos a observar los resutlados para los distintos tipos de conjuntos de TF en la Tabla 4.11. **Predicciónes Random Forest Classifier - Best Time Features**:

**Hyperparameter Tunning:** Mejor RandomForestClass: RandomForestClassifier(max_depth=10, max_features='auto', min_samples_split=5, n_estimators=800) Mejores parámetros: {'bootstrap': True, 'max_depth': 10, 'max_features': 'auto', 'min_samples_leaf': 1, 'min_samples_split': 5, 'n_estimators': 800} Error absoluto medio (ACC): 0.5685933147632312

### 4.2.4. Predicciones Dataframe Songs Names

En este apartado, realizaremos la predicción de múltiples valores en el futuro, decidido por la media del número de streamings del primer DF, o por la predicción que nos de nuestro algoritmo. En este caso, el número de predicciones futuro es: 17.

Además, como la variable objetivo es categórica, realizamos diferentes transformaciones para poder predecirlas, estas son: *Labeled Encoding* y *One Hot Encoding*. En el

| Rest_Type | lag6 | lag7 | lag8 | lag9 | lag10 | lag11 | lag12 | lag13 | lag14 | lag15 | M |
|---|---|---|---|---|---|---|---|---|---|---|---|
| NONE | 0,38 | 0,28 | 0,19 | 0,27 | 0,35 | 0,35 | 0,34 | 0,39 | 0,46 | 0,46 | 0,56 he |
| 0,324 height | | | | | | | | | | | |
| 0,24 | 0,26 | 0,25 | 0,28 | 0,26 | 0,25 | 0,28 | 0,28 | 0,28 | 0,28 | 0,29 | 0,254 |
| BCV | 0,18 | 0,19 | 0,17 | 0,19 | 0,19 | 0,27 | 0,26 | 0,25 | 0,26 | 0,27 | 0,24 | 0,212 |

Tabla 4.10: Resultados de la predicción con Random Forest Classifier Base, TSCV y BCV sobre el Tipo de descansos del próximo día según diferente número de algs

| Rest_Type | TF1 | TF2 | TF3 | M |
|---|---|---|---|---|
| Base | 0.766 | 0.775 | 0.764 | 0,768 |
| TSCV | 0.337 | 0.347 | 0.343 | 0,342 |
| BCV | 0.287 | 0.287 | 0.284 | 0,283 |

Tabla 4.11: Resultados de la predicción con Random Forest Classifier Base, TSCV y BCV sobre el Tipo de descansos del próximo día según diferentes Time Features

primer caso solo es necesario predecir una variable que contiene todas las posibles clases, en cambio, en el segundo caso, necesitamos predecir tantas variables como clases o valores únicos existan. Se utiliza el algoritmo Random Forest Classifier.

Para realziar las transformaciones mencionada se utilizan las funciones LabelEncoder().transform() y la función de pandas .get_dummies(). Antes de realizar cualquiera de las 2 transformaciones, reducimos el dataframe original a el número de semanas descubiertas como relevantes en el proceso de Churn Rate, descrito en el Anexo A, y cogemos las X ultimas. También es necesario limpiar de caracteres especiales la variable objetivo antes de realzizar las transformaciones.

Resultados de la predicción con el método Baseline, Tabla 4.12.

**Predicciónes Baseline - One Hot and Labeled Encoding**:

Aclaración para los resultados de One Hot. Los N pasos futuros = 17. En el DF existe 1 semana de información que contienen un total de 72 reproducciones y 32 canciones. Esto Significa que el stream a predecir no ha tenido porque haber sido escuchado, y así acertando teoricamente en la predicción e incrementando el ACC, ya que se va a compara un Array de zeros con otra igual, pero en este caso, aunque aumente el ACC, no es viable predecir siempre cero.

Resultados de la predicción en el DF original con Ranfom Forest, Tabla 4.13

**Predicciónes Random Classification Forest - Base One Hot and Labeled Encoding**:

Se continuan mostrando los resultados obtenidos de la generación de Lags, en este caso, por temas de limpieza, se muestran unicamente los mas importantes y que dan mejores resultados, los 10 primeros, y la media (M).

| CV | Base One-Hot ACC | Base Labeled ACC |
|---|---|---|
| NONE | 0.648 | 0.0 |
| TSCV | 0.608 | 0.0 |
| BCV | 0.628 | 0.01 |

Tabla 4.12: Resultados de la predicción base de N pasos futuros y de las M variables objetivos transformadas con one hot encoding y únicamente de N pasos futuros para Labeled encoding, para saber que streamings escuchará el usuario

| CV | Base One-Hot ACC | Base Labeled ACC |
|---|---|---|
| NONE | 0.267 | 0.245 |
| TSCV | 0.462 | 0.459 |
| BCV | 0.108 | 0.275 |

Tabla 4.13: Resultados de la predicción base de N pasos futuros y de las M variables objetivos transformadas con one hot encoding y únicamente de N pasos futuros para Labeled encoding, para saber que canciones escuchará el usuario

Resultados de los lags generados para la transformación *One Hot Encoding*, Table 4.14

**Predicciónes Random Classification Forest - One Hot Best lags**:

Para **Labeled Encoding**, Tabla 4.15 **Predicciónes Random Classification Forest - Labeled Best lags**:

En estos resultados podemos observar una clara tendencia descendente. No solo eso si no que podemos ver que con la cantidad de datos que hay, o con los datos que son, funciona mucho mejor el método de cross validation TSCV. Además podemos comprobar que funciona mucho mejor el metodo one hot encoding que el labeled encoding.

Pasamos a investigar los resutlados obtenidos tras la generación de **Time Features**, la transformación *One Hot Encoding* en la Tabla 4.16, y *Labeled Encoding* en la Tabla 4.17.

**Predicciónes Random Classification Forest - One Hot Best TF**:

**Predicciónes Random Classification Forest - Labeled Best TF**:

Ahora, se añaden variables extraidas de internet de cada canción.

Predicción para *One Hot Encoding* con las neuvas características, Tabla 4.18

**Predicciónes Random Classification Forest - One Hot Lags - Selenium**:

Ahora, con la transformación **Labeled Encoding**, Tabla 4.19

**Predicciónes Random Classification Forest - Labeled Lags - Selenium**:

| CV | Lag1 | Lag2 | Lag3 | Lag4 | Lag5 | Lag6 | Lag7 | Lag8 | Lag9 | Lag10 | M |
|---|---|---|---|---|---|---|---|---|---|---|---|
| NONE | 0,309 | 0,296 | 0,293 | 0,281 | 0,289 | 0,283 | 0,274 | 0,268 | 0,267 | 0,273 | 0,279 |
| TSCV | 0,414 | 0,397 | 0,382 | 0,376 | 0,36 | 0,311 | 0,319 | 0,301 | 0,286 | 0,269 | 0,319 |
| BCV | 0,084 | 0,088 | 0,096 | 0,081 | 0,087 | 0,105 | 0,065 | 0,063 | 0,065 | 0,062 | 0,079 |

Tabla 4.14: Resultados de la predicción de N pasos futuros y de las M variables objetivos transformadas con One Hot Encoding para encontrar que lags dan el mejor resultado

| CV | Lag1 | Lag2 | Lag3 | Lag4 | Lag5 | Lag6 | Lag7 | Lag8 | Lag9 | Lag10 | M |
|---|---|---|---|---|---|---|---|---|---|---|---|
| NONE | 0,241 | 0,239 | 0,222 | 0,241 | 0,215 | 0,229 | 0,240 | 0,234 | 0,214 | 0,205 | 0,221 |
| TSCV | 0,419 | 0,398 | 0,358 | 0,335 | 0,339 | 0,337 | 0,308 | 0,278 | 0,222 | 0,198 | 0,284 |
| BCV | 0,133 | 0,066 | 0,088 | 0,084 | 0,078 | 0,083 | 0,062 | 0,05 | 0,063 | 0,053 | 0,074 |

Tabla 4.15: Resultados de la predicción base de N pasos futuros para la transformación Labeled Encoding, para encontrar el mejor lag

Estos resultados nos llevan a la conclusion de que con *Labeled Encoding* le cuesta mucho mas detectar y darle importancia a relaciones a largo plazo. De igual manera, podemos observar la importancia de las características extraidas con selenium, ocupando los primeros puestos en la importancia de las Features para la predicción según el algorimo, dejando de lado la importancia de los lags en las pruebas sin CV.

Con TSCV obtenemos unos resultados mucho mejores que con BCV, y con *One Hot Encoding* También obtenemos Resultados mejores que con *Labeled Encoding* exceptuando el primer lag con las características de selenium, el cual aporta un 0,002 de mejoría.

Ahora se pueba la introducción de variables temporales o Time Features, para comprobar si aún podemos mejorar más los resultados.

Con *One Hot Encoding* obtenemos la Tabla 4.20

**Predicciónes Random Classification Forest - One Hot y TF - Selenium**:

Con *Labeled Encoding* obtenemos la Tabla 4.21

**Predicciónes Random Classification Forest - Labeled y TF - Selenium**:

Se observa que las time features solo han reducido el ACC obtenido anteriormente sin ellas.

**Hiperparameter Tunning**

Al realizar GridSearchCV para el DF y método que ha obtenido mejores resultados, obtenemos en ambas transformaciones un ACC de 1.0.

Esto claramente se debe a un Overfitting causado por la falta de datos en el Dataframe.

Obteniendo los siguientes resultados:

| CV | TF1 | TF2 | TF3 | M |
|---|---|---|---|---|
| NONE | 0.312 | 0.306 | 0.313 | 0,310 |
| TSCV | 0.412 | 0.412 | 0.410 | 0,411 |
| BCV | 0.079 | 0.079 | 0.080 | 0,079 |

Tabla 4.16: Resultados de la predicción base de N pasos futuros y de las M variables objetivos transformadas con one hot encoding y variables TF

| CV | TF1 | TF2 | TF3 | M |
|---|---|---|---|---|
| NONE | 0,135 | 0,175 | 0,152 | 0,154 |
| TSCV | 0,320 | 0,351 | 0,329 | 0,333 |
| BCV | 0,072 | 0,077 | 0,074 | 0,074 |

Tabla 4.17: Resultados de la predicción de N pasos futuros para Labeled encoding con variables TF

Mejor RandomForestRegressor: RandomForestRegressor(bootstrap=False, max_depth=30, max_features='sqrt', n_estimators=10) Mejores parámetros: {'bootstrap': False, 'max_depth': 30, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 10}

Mejor RandomForestRegressor: RandomForestRegressor(bootstrap=False, max_depth=10, max_features='sqrt') Mejores parámetros: {'bootstrap': False, 'max_depth': 10, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 100}

| CV | lag1 | lag2 | lag3 | lag4 | lag5 | lag6 | lag7 | lag8 | lag9 | lag10 | M |
|---|---|---|---|---|---|---|---|---|---|---|---|
| NONE | 0,332 | 0,332 | 0,332 | 0,332 | 0,332 | 0,332 | 0,332 | 0,332 | 0,332 | 0,332 | 0,332 |
| TSCV | 0,553 | 0,553 | 0,551 | 0,547 | 0,546 | 0,516 | 0,504 | 0,503 | 0,474 | 0,469 | 0,502 |
| BCV | 0,533 | 0,552 | 0,549 | 0,547 | 0,541 | 0,51 | 0,507 | 0,500 | 0,476 | 0,469 | 0.500 |

Tabla 4.18: Resultados de la predicción de M Variales N pasos futuros para One Hot Encoding con features extraidas con selenium.

| CV | lag1 | lag2 | lag3 | lag4 | lag5 | lag6 | lag7 | lag8 | lag9 | lag10 | M |
|---|---|---|---|---|---|---|---|---|---|---|---|
| NONE | 0,332 | 0,332 | 0,332 | 0,332 | 0,332 | 0,332 | 0,332 | 0,332 | 0,332 | 0,332 | 0,332 |
| TSCV | 0,555 | 0,555 | 0,555 | 0,554 | 0,554 | 0,554 | 0,531 | 0,528 | 0,519 | 0,493 | 0,526 |
| BCV | 0,177 | 0,088 | 0,088 | 0,111 | 0,088 | 0,088 | 0,109 | 0,066 | 0,066 | 0,066 | 0,091 |

Tabla 4.19: Resultados de la predicción de N pasos futuros para Labeled Encoding con features extraidas con selenium.

| CV | TF1 | TF2 | TF3 | M |
|---|---|---|---|---|
| NONE | 0,31 | 0,39 | 0,313 | 0,337 |
| TSCV | 0,412 | 0,411 | 0,408 | 0,410 |
| BCV | 0,078 | 0,078 | 0,078 | 0,078 |

Tabla 4.20: Resultados de la predicción de M variables y N pasos futuros para la transformación One Hot Encoding con Time Features y con las features extraidas con selenium.

| CV | TF1 | TF2 | TF3 | M |
|---|---|---|---|---|
| NONE | 0,128 | 0,183 | 0,157 | 0,156 |
| TSCV | 0,317 | 0,346 | 0,335 | 0,332 |
| BCV | 0,071 | 0,084 | 0,081 | 0,078 |

Tabla 4.21: Resultados de la predicción de N pasos futuros para la transformación Labeled Encoding con Time Features y con las features extraidas con selenium.

# Capítulo 5

# Conclusiones y trabajo futuro

Las conclusiones que sacamos de este trabajo es que en este concreto caso de datos donde ni la estacionalidad ni la linealidad de ellos esta garantizada es que RFs es un gran algoritmo para realizar predicciones.

Cuando es necesario realizar Regressión, BCV funciona mucho mejor, y mejoramos mucho los resultados investigando los lags, mucho mas que introduciendo variables temporales, las cuales en pocas ocasiones han mejorado los resultados, únicamente en aquellos con baja variabilidad, como son las variables New Streams o New Songs.

En el caso del primer DF (Daily DF), conseguimos reducir el valor del MAE un 50 %, de 22 a 11 En el caso del segundo DF (N Rests DF), conseguimos reducir el valor del MAE un 80 %, de 2,5 a 0,4.

En los casos que se aplica Classificación, TSCV ha dado mejores resultados, aunque a cambio de mucho mas tiempo de computación, así como da mejores resultados la codificación One Hot Encoding.

En el caso del tercer DF (Rest Type DF), conseguimos incrementar el ACC casi un 90 %, de 0,3 a 0,56. En el cuarto y último DF (Song Name DF), conseguimos llevar el ACC con el proceso de *Hyperparameter Tunning* hasta 1. Esto se debe a la falta de datos por los problemas comentados en el Anexo A Churn Rate. Sin este proceso de ajuste de parámetros, pasamos de predicciónes iniciales con 0,3 de ACC, a 0,55.

Los resultados los problemas de classificación son peores que los resultados obtenidos en los problemasd e regresion, relativamente más sencillos. Por como se comportan los valores de los resultados según el método de CV utilizado, se entiende que los problemas de regresión tienen dependencias a menor plazo, mientras que los de classificación tienen dependencias a largo plazo que son necesarias de extraer para obtener mejores resultados.

En el último datafrmae, extraemos características de cada canción. Estas características han sido claves para mejorar las predicciones, teniendo mayor peso e importancia que los lags y las TF.

Por todo esto, el proyecto puede continuar comparando estos resultados con otros modelos y algoritmos como son las LSTMs, ARIMA o Propher. Además, sería muy interesante repetir el proceso completo con un dataframe por horas, ya que de este modo se puede entender mejor las rutinas de una persona.

Otro punto a investigar es la aportación de nuevas características de la letra de las cancions extraidas con NLP, como el sentimiento de la letra, o el origen del lenguaje y las expresiones, o el idioma.

Finalmente, se podrían extrar también nuevas estadísticas y características del usuario, como se menciona en el Anexo A - Churn Rate. Igual que se enseña una posible manera de etiquetar streamings favoritas del usuario, se podría realizar el opuesto, recomendandole una seríe de elementos de streaming que consume, o que lleva mucho sin consumir, y que pueden estar ocupandole esspacio en listas de reproducción o que le pueden estar distrayendo. Por ejemplo, esto se podría detectar si siempre salta una canción.

# Glosario

**A** *aaa*

**B** *bbb*

# Bibliografía


[1] Ladorian. Sistemas de recomendaciones: La revolución de la inteligencia. http://ladorian.com/actualidad/sistemas-recomendaciones-ia/.

[2] OpenMind. Aplicaciones ia en el comercio: Sistemas de recomendación. https://www.bbvaopenmind.com/economia/finanzas/aplicaciones-de-la-inteligencia-artificial-en-el-comercio-online/.

[3] L. E. Parra-Medina and F. J. Álvarez Cervera. Síndrome de la sobrecarga informativa: una revisión bibliográfica. *Revista de Neurología*, 73(12):421–428, 2021.

[4] IBM. Regular and irregular time series. Available at: https://www.ibm.com/docs/en/streams/5.3?topic=series-regular-irregular-time, 2021.

[5] YouTube. The most satisfying video in the world. Available at: https://www.youtube.com/watch?v=E4NMZyfao2c, 2021.

[6] The Decision Lab. The paradox of choice. Available at: https://thedecisionlab.com/reference-guide/economics/the-paradox-of-choice, 2021.

[7] María de la Paz Toldos Romero. La influencia de la música en el comportamiento del consumidor. *The Decision Lab*, 2019.

[8] La Mente es Maravillosa. ¿cómo nos influye la música en las tiendas de ropa? Available at: https://lamenteesmaravillosa.com/como-nos-influye-la-musica-en-las-tiendas-de-ropa/, 2021.

[9] Javier Garcés Prieto. Consumismo e infelicidad. impacto psicologico de la actual sociedad de consumo. YouTube, 2022. https://www.youtube.com/watch?v=Tq_cLvOPvAM.

[10] El Salto. Consumismo: adicción a la infelicidad. El Salto, 2022. https://www.elsaltodiario.com/consumo-que-suma/consumismo-adiccion-a-la-infelicidad.

[11] Rosa Boschetti. El artista como marca, producto o propuesta. diferencia entre ellos. Rosa Boschetti, 2022. https://rboschetti.com/2022/08/30/el-artista-como-marca-producto-o-propuesta-diferencia-entre-ellos/.

[12] El País. La música activa las mismas áreas cerebrales que el sexo y las drogas. El País, 2017. https://elpais.com/elpais/2017/02/08/ciencia/1486532624_646645.html.

[13] Annemieke J.M. Van den Tol and Jane Edwards. Sad music depresses sad adolescents: A listener's profile. *Psychology of Music*, 2019. https://journals.sagepub.com/doi/full/10.1177/0305735619849622.

[14] ¿puede la música sombría hacer que los adolescentes se depriman? https://discoverymood.com/blog/somber-music-make-teens-depressed/.

[15] Música y personas con tendencias a la depresión.

[16] Efectos negativos y positivos de la música. Embark Behavioral Health, n.d. https://www.embarkbh.com/qa/negative-and-positive-effects-of-music/.

[17] Los Angeles Times. Music has power over mind and body. Los Angeles Times, 2011. https://www.latimes.com/archives/la-xpm-2011-jan-09-la-heb-music-dopamine-20110109-story.html.

[18] El Confidencial. Los españoles escuchan 20 horas de música a la semana: ¿qué géneros prefieren? El Confidencial, 2019. https://www.elconfidencial.com/cultura/2019-09-25/20-horas-musica-semana-espanoles_2253927/.

[19] Filtrado colaborativo y filtrado basado en contenido. https://bigdatamagazine.es/filtrado-colaborativo-y-filtrado-basado-en-contenido. Este artículo de Big Data Magazine explica los conceptos básicos del filtrado colaborativo y basado en contenido.

[20] Enrique Villamarín Fonseca. Desarrollo de un sistema de recomendaciÓn para un caso real. http://www.uhu.es/mecofin/documents/docencia/tfm/20172018/MECOFIN_20172018_tfm_20172018-VFE_DSRC.pdf. Este trabajo de tesis desarrolla un sistema de recomendación basado en un filtrado colaborativo que permite una mejor selección cuando existe un determinado número de afinidades.

[21] Daniel Roy. How spotify's recommender system works. https://www.linkedin.com/pulse/how-spotify-recommender-system-works-daniel-roy-cfa/, May 2019.

[22] The risks of using ai to interpret human emotions. https://hbr.org/2019/11/the-risks-of-using-ai-to-interpret-human-emotions, November 2019.

[23] Bookdown. Estadística y machine learning con r. https://bookdown.org/content/2274/series-temporales.html, 2018.

[24] Statsmodels. Análisis de series temporales por métodos espaciales de estado. https://www.statsmodels.org/stable/statespace.html, 2018.

[25] Inovex. Deep learning for time series. https://www.inovex.de/de/blog/deep-learning-time-series/, 2018.

[26] Sima Siami-Namini, Neda Tavakoli, and Akbar Siami Namin. A comparison of arima and lstm in forecasting time series. In *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 1394–1401. IEEE, 2018.

[27] Jason Brownlee. Random forest for time series forecasting. https://machinelearningmastery.com/random-forest-for-time-series-forecasting/, 2020.

[28] IBM. Transformaciones de los datos de serie temporal. https://www.ibm.com/docs/es/spss-statistics/25.0.0?topic=transformations-time-series-data, 2022.

[29] Universidad Complutense de Madrid. Tema 6: Modelización con datos de series temporales. https://www.ucm.es/data/cont/docs/518-2013-10-25-Tema_6_EctrGrado.pdf, 2013.

[30] Ciencia de Datos. Skforecast: forecasting series temporales con python y scikitlearn. https://www.cienciadedatos.net/documentos/py27-forecasting-series-temporales-python-scikitlearn.html, 2022.

[31] Towards Data Science. Cyclical features encoding: It's about time. https://towardsdatascience.com/cyclical-features-encoding-its-about-time-ce23581845ca, 2021.

[32] Robert H. Shumway and David S. Stoffer. *Time Series Analysis and Its Applications: With R Examples*. Springer, 2017.

[33] Medium. Cross validation in time series. https://medium.com/@soumyachess1496/cross-validation-in-time-series-566ae4981ce4, 2022.

[34] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2009.

[35] Max Kuhn and Kjell Johnson. *Applied Predictive Modeling*. Springer, 2013.

[36] Rahil Shaikh. Choosing the right encoding method-label vs onehot encoder. *Towards Data Science*, 2018.

[37] Ana Fló et al. Evidence of ordinal position encoding of sequences extracted from continuous speech. *Cognition*, 213:104646, 2021.

[38] Hafidz Zulkifli. Understanding entity embeddings and it's application. *Towards Data Science*, 2023.

[39] Muhammad Awais Qasim, Saeed Ul Hassan, Naif Radi Aljohani, and Miltiadis D. Lytras. Human behavior analysis in the production and consumption of scientific knowledge across regions: A case study on publications in scopus. *Library Hi Tech*, 35(4):646–660, 2017.

# Anexo: A - Churn Rate

Para entender mejor la conducta de modo de escucha del usuario generamos una matriz. Esta matriz contiene por fila, cada elemento que se ha stremeado, y por columna, el número de veces que ha escuchado la canción ese semana. Se ha considerado la unidad de medida semana la mas conveniente para el experimento, ya que un día es excesivamente corta, y mas de una semana sería asumir demasiada información para saber como de seguido escucha el usuario el mismo stream Figura 1.

| | w0 | w1 | w2 | w3 | w4 | w5 | w6 | w7 | w8 | w9 | ... | w289 | w290 | w291 | w292 | w293 | w294 | w295 | w296 | first_listened | last_listened |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Excesos | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 |
| Calavera no chilla | 25.0 | 4.0 | 36.0 | 17.0 | 8.0 | 2.0 | 4.0 | 10.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 158.0 |
| Metacrilato | 40.0 | 3.0 | 39.0 | 26.0 | 13.0 | 4.0 | 4.0 | 8.0 | 0.0 | 1.0 | ... | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 292.0 |
| La flauta de Hamelin | 3.0 | 0.0 | 1.0 | 3.0 | 2.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 38.0 |
| 2000 Clavos | 3.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| Need4Speed - Fumado y Hambriento | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | 296.0 | 297.0 |
| HOODS HOTTEST | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | 296.0 | 297.0 |
| somehow she's still here (feat. James Blake) | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | 296.0 | 297.0 |
| WHY AM I STILL IN LA (feat. Shlohmo & D33J) | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | 296.0 | 297.0 |
| je suis | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 2.0 | 296.0 | 297.0 |

14082 rows × 299 columns

Figura 1: Ejemplo de la matriz con el número de escucha semanal para cada elemento, también contiene la primera semana y la última que se ha hecho un stream de ese elemento

Después, de la matriz, se extrae otra nueva que cuenta las rachas de semanas con música escuchada para cada stream. Para mejor comprensión observar la Figura 2

```
[2, 8, 13, 3, 7, 17],
[1, 5, 16, 5, 3, 11],
[1],
[1, 1],
[16, 9, 5, 4],
[1, 3, 3, 2, 1, 1, 8, 9, 6, 8, 11, 4, 4, 7, 3, 5, 5],
```

Figura 2: Ejemplo de la matriz con el número de semanas un mismo elemento ha sido escuchado de seguido sin parar

Gracias a ella, podemos comprender mejor al usuario y su forma de consumir elementos

de streaming. Observando la Figura4 podemos comprender la distribución de sus hábitos de consumo de streams.

Counter$\{17:69, 19:53, 18:50, 20:48, 21:39, 22:39, 23:33, 26:31, 24:28, 25:25, 31:18, 28:17, 27:15, 29:14, 30:11, 36:8, 37:8, 32:8, 34:6, 35:6, 43:5, 44:4, 33:4, 39:3, 47:3, 40:2, 38:2, 42:2, 48:2, 41:2, 53:1, 54:1, 45:1, 50:1, 46:1, 63:1, 51:1, 58:1, 81:1\}$

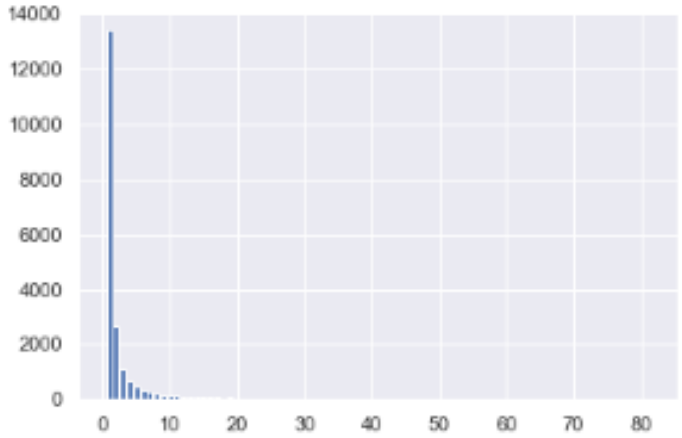



Figura 3: Ejemplo de la matriz con los valores únicos de rachas máximas y el número de elementos que han llegado a ese número número

Analizando los percentiles, así como aplicando técnicas de eliminación de *Outlayers* como son Z-Score y IQR, obtenemos nuevos valores. De los valores mas probables, que son: el 1, el 2 y el 3. Esto se decide estos número hacen que el suelo de la media obtenida den ellos mismos.

De los métodos de outlayer se deciden eliminar los valores: 54: 1, 45: 1, 50: 1, 46: 1, 63: 1, 51: 1, 58: 1, 81: 1

De este modo se han eliminado los valores mas típicos y los atípicos, quedando la siguiente distribución:

Counter$\{4:647, 5:451, 6:344, 7:292, 8:228, 9:165, 10:143, 11:136, 12:108, 14:105, 13:96, 15:80, 16:78, 17:69, 19:53, 18:50, 20:48, 21:39, 22:39, 23:33, 26:31, 24:28, 25:25, 31:18, 28:17, 27:15, 29:14, 30:11, 36:8, 37:8, 32:8, 34:6, 35:6, 33:4, 39:3, 40:2, 38:2, 41:2\}$

De esta manera obtenemos 2 médias que son nuestra base para la extracción de características a largo plazo. Con todos los datos de rachas extraidos, la media inferior tiene el valor 1. Quitando los valores típicos y los *Outlayers*, la media de rachas tiene el valor de 10. Es decir, para tratar de averiguar que canción se escuchará después, se utilizará un dataset con la última semana, y otro con 10 semanas.

Con ente análisis se comprende que 10 semanas es el máximo que el usuario escucha una cación que le gusta, mas allá de eso sería un claro ejemplo de como determinar y

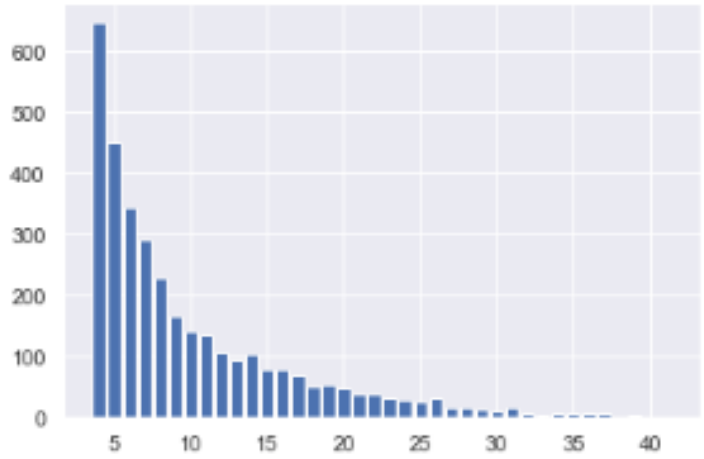



Figura 4: Ejemplo de la matriz con los valores únicos de rachas máximas y el número de elementos que han llegado a ese numero, sin outlayers ni valores típicos

poner la etiqueta de canción favorita para un usuario, además de otras maneras como tener en cuenta el número de reproducciones totales, o el de reproducciones por semana.

Es decir, lo mas común es que el usuario se aburra de escuchar una canción normal, entre una semana y 10. Por ello utilizaremos estos dataframes.

Debido a los recursos de computación, existe el problema que al hacerlo con 10 semanas se genera un error el cual nos indíca que no disponemos de suficiente RAM para realizar las operaciones necesarias. ERROR: MemoryError (indica que el proceso de ajuste del modelo RandomForestClassifier está agotando la memoria disponible).

# Anexo: B - EDAS

**Dataframe Diario:** En la figura 5 se muestran los datos del DF de la variable objetivo day_n_stream.

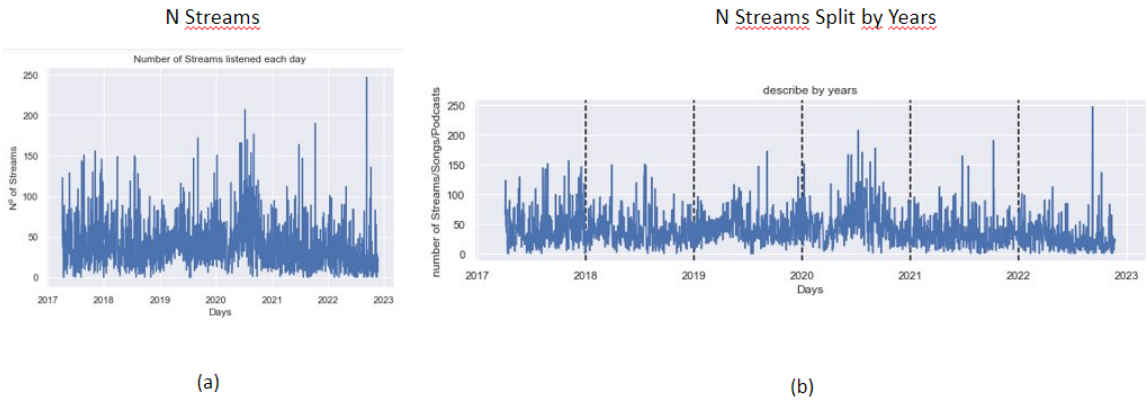



Figura 5: Plot de todos los valores de la variable objetivo N Streams (a), y de estos mismos separados en los diferentes años (b)

El Segundo paso es comprobar si la serie es estacionaria, para ello, vamos a aplicar la rolling mean, si esta es constante durante toda la serie podemos asumir que la ST será estacionaria, imagen 6.

Podemos observar que en la mayoría de variables, la rolling mean es constante. Sin enbargo, en las variables day_n_podcast, day_new_podcast podemos observar ciertas variaciones.

Para saber con certeza si las variables objetivos son o no estacioanria, vamos a realizar 2 tests estadísticos que nos responden esta pregunta. El test ADF y el test KPSS.

Los resultados de estos tests son los esperados. En primer lugar, para las 6 variables el test ADF fuller nos indica que todas ellas son estacioanrias para un p-valor inferior a 0,05.

Sin embargo, el test KPSS nos índica que las variables dependientes de el número de podcasts no lo son. Estas posibles combinaciónes y contradicciones se tratan en los

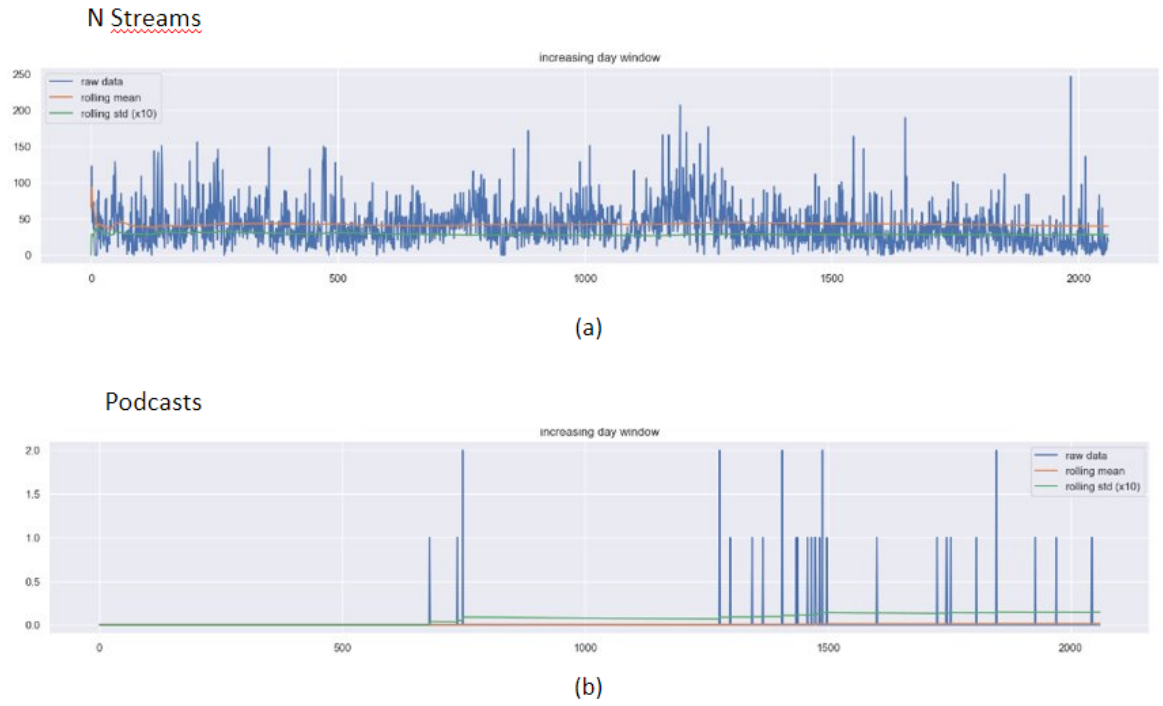



Figura 6: Plot de las rolling mean para las variables (a) N Streams, (b) N Podcasts

siguientes casos [1].

- Caso 1: Ambos test concluyen que la serie no es estacioanria: La serie no es estacionaria.
- Caso 2: Ambos test concluyen que la serie es estacioanria: La serie es estacionaria.
- Caso 3: KPSS indica que es estacionaria mientras que ADF indica que no lo es: Indíca tendencia en la serie estacionaria.
- Caso 4: KPSS indica que es no estacionaria mientras que ADF indíca que lo es: La serie necesita ser diferenciada.

De esta manera, generamos un proceso recursivo en el que se comprueba la estacionariedad de cada variable objetivo por ambos test. Las conclusiones generadas por estos posibles casos se guardan en una lista. Según Los resultados se le aplica a cada variable la transformación necesaria, hasta que todas las variables sean estacionarias.

En este preciso caso aunque apliquemos el proceso, no afectará a las variables objetivo, además Random Forests no necesita obligatoriamente que las ST sean Estacionarias. No es necesario porque el método baseline es suficiente en este usuario para utilizarlo en vez del algoritmo de ML seleccionado. Aún así queda aquí descrito el método:

[1] `https://www.statsmodels.org/dev/examples/notebooks/generated/stationarity_detrending_adf_kpss.html`

1. Realización de ambos test en cada variable objetivo.
2. Computo de casos.
3. En caso de que todas las variables sean asignadas con el caso 2, Fin.
4. Transformación de varibales con casos diferentes al número 2.
5. Vuelta al paso 1

La tercera parte del EDA consiste en descomponer en diferentes componentes la ST [2]. Una ST puede descomponerse en tendencia, periodicidad y ruido, tal como se observa en la Figura 7.

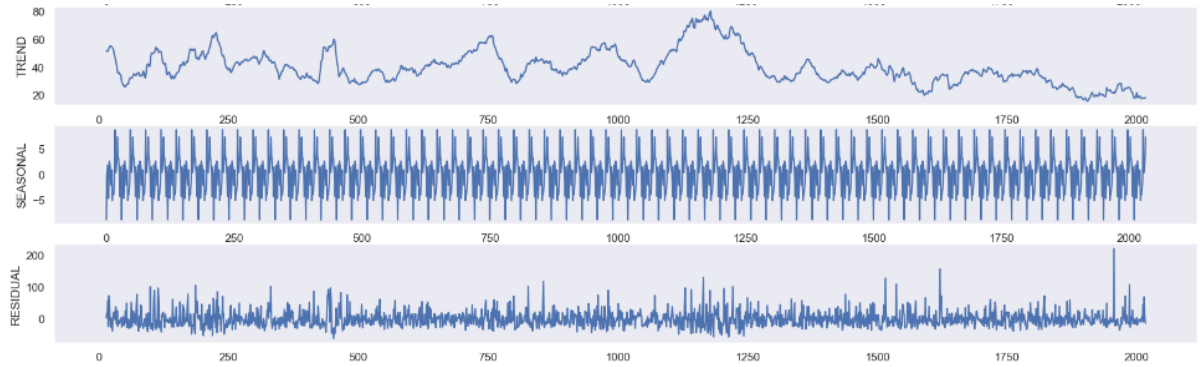



Figura 7: Plot de las STR tendencia, periodicidad y ruido, generadas a partir de la STR Original N Streams

Aprovechamos para ver si un mismo día de la semana tiene gran correlación con ese mismo día en anteriores semanas. Por ejemplo, estudiamos la autocorrelación entre un lunes y los anteriores en la figura 8.

Al realizar esto, se observa que esto hay ciertos días con una alta relevancia por lo que puede ser otro punto de investigación.

Las variables podcast son completamente sparse, por lo que podemos contemplarlas como nulas, y las variables de canciones y streaming son practicamente iguales, por lo que este mísmo análisis se puede realizar unicamente con las dos variables dependientes de los streamings.

**Rest Number**

Plot Visual de la variable objetivo N Rests, figura 9:

Plot Rolling Mean, figura 10

Resultados para los test estadísticos ADF y KPSS Both 0. La serie es estacionaria.

Plot de la descomposición de la ST, figura 11:

[2] `https://rstudio-pubs-static.s3.amazonaws.com/289564_7557e57a8aac42b1a8ca434689ee3cff.html`

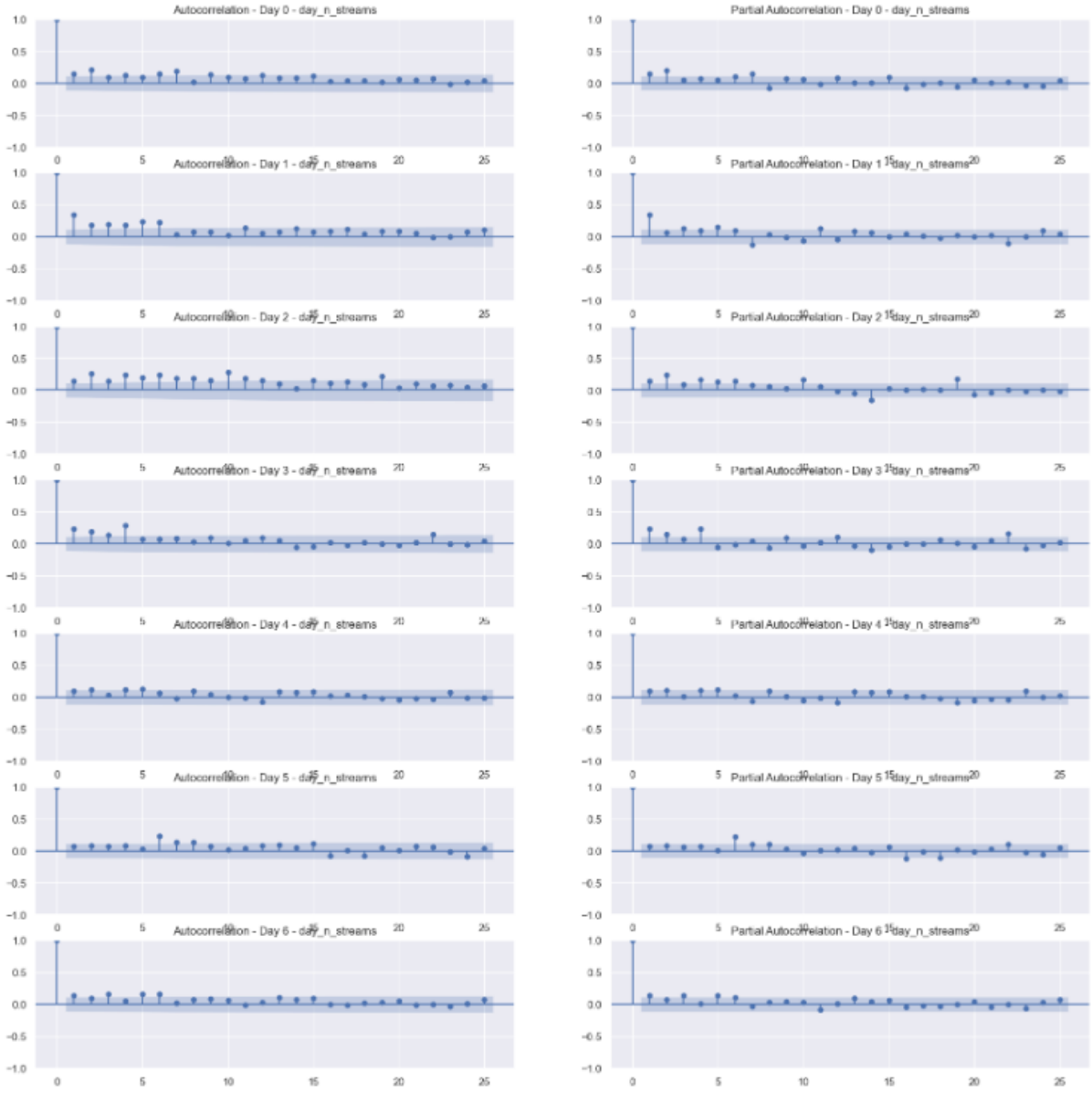



Figura 8: Plot de los valores de las funciones ACF y PACF para la variable N Streamings por días de la semana

Valores de las funciones ACF PACF Semanalmente, figura 12:

**Rest Type**

Plot visual de la variable objetivo rest_type, figura 13:

Plot Rolling Mean para la variable objetivo rest_type, figura 14:

Ejemplo Prueba Xi cuadrado:

*fruits = ["manzana", "naranja", "manzana", "manzana", "pera", 'naranja', 'pera', 'naranja']*


Figura 9: Plot de los valores de la variable N Rests

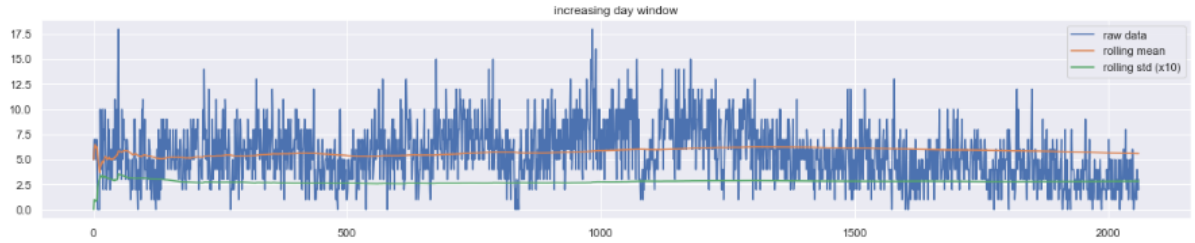



Figura 10: Plot de la media desplazada de la variable N Rests

*freqs = Counter(fruits)*

*print(freqs)*

*Output: Counter("manzana": 3, "naranja": 3, "pera": 2)*

*expected_freqs = dict.fromkeys(freqs.keys(), sum(freqs.values()) / len(freqs))*

*print(expected_freqs)*

*Output: "manzana": 2.66, "naranja": 2.66, "pera": 2.66*

*observed_freqs = list(freqs.values())*

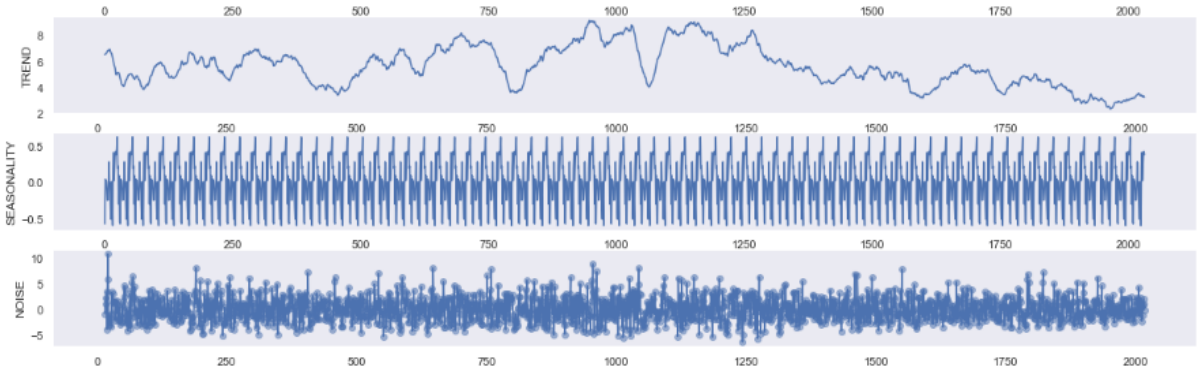



Figura 11: Plot de las STR tendencia, estacionalidad y ruide de la STR N Rests

*expected_freqs = list(expected_freqs.values())*

*_, p_value = chisquare(observed_freqs, f_exp=expected_freqs)*

*print(p_value)*

*Output: 1.0*

Como el output de *p-value* es mayor a 0.05, utilizariamos la función .diff() de numpy para transformar la serie y hacerla estacionaria.

*diff_fruits = np.diff(fruits)*

*print(diff_fruits)*

*Output: "naranja" "manzana" "manzana" "pera" "naranja" "pera" "naranja"*

Plot de la descomposición de rest_type, figura 15:

Plot los valores de la ACF y PACF de rest_type semanalmente, figura 16:

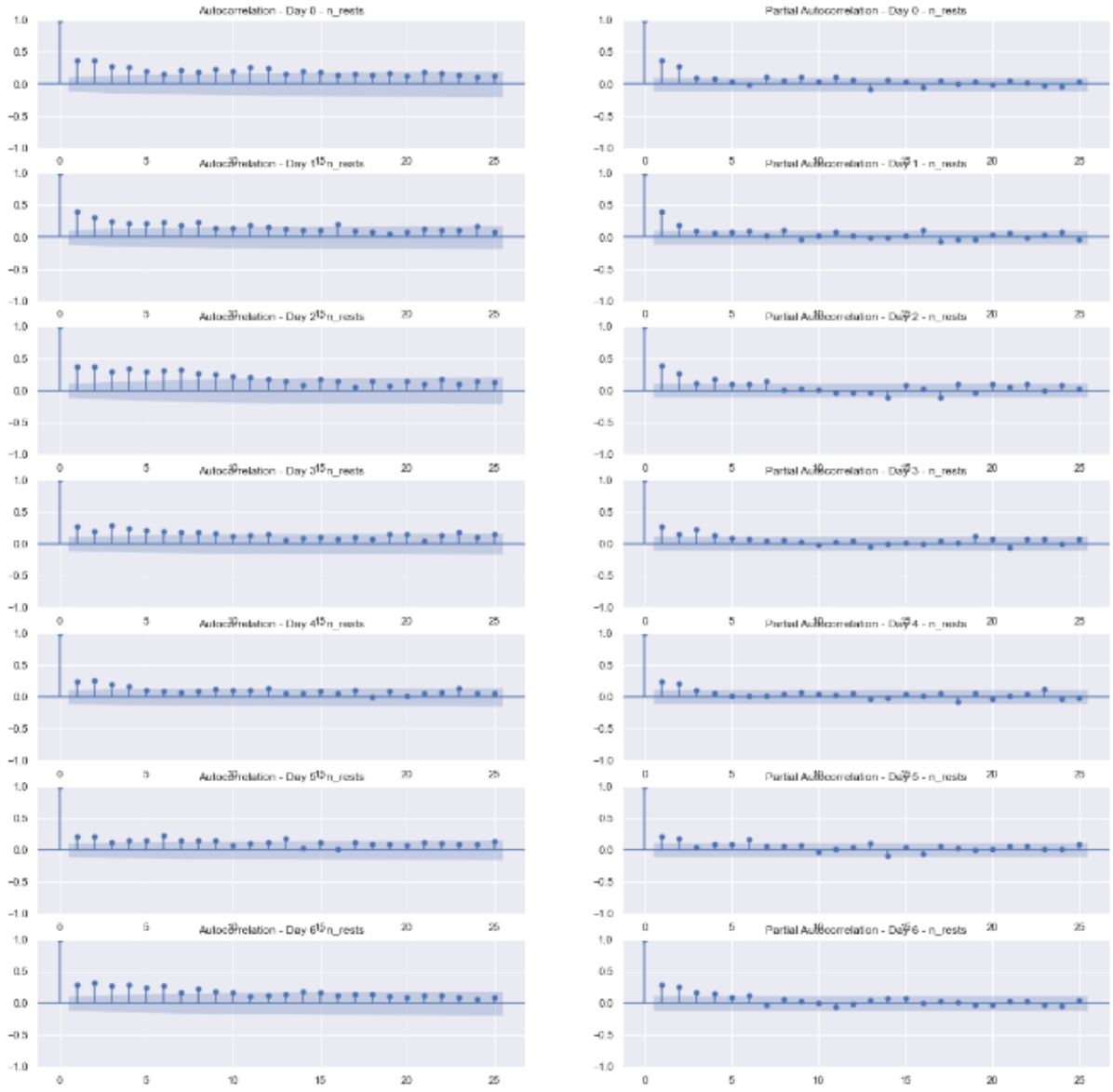


Figura 12: Plot de los valores de las funciones ACF y PACF para la STR semanal de N Rests

| | rest_type | n_streams_listened | week_day |
|---|---|---|---|
| 2017-04-04 14:06:56 | 1 | 2 | 1 |
| 2017-04-04 14:55:10 | 1 | 9 | 1 |

Figura 13: Plot de los valores de la variable objetivo Rest Type

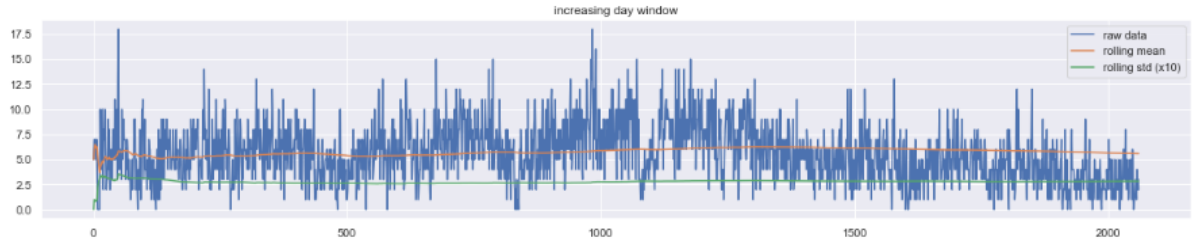



Figura 14: Plot de la media desplazada de la variable Rests Type

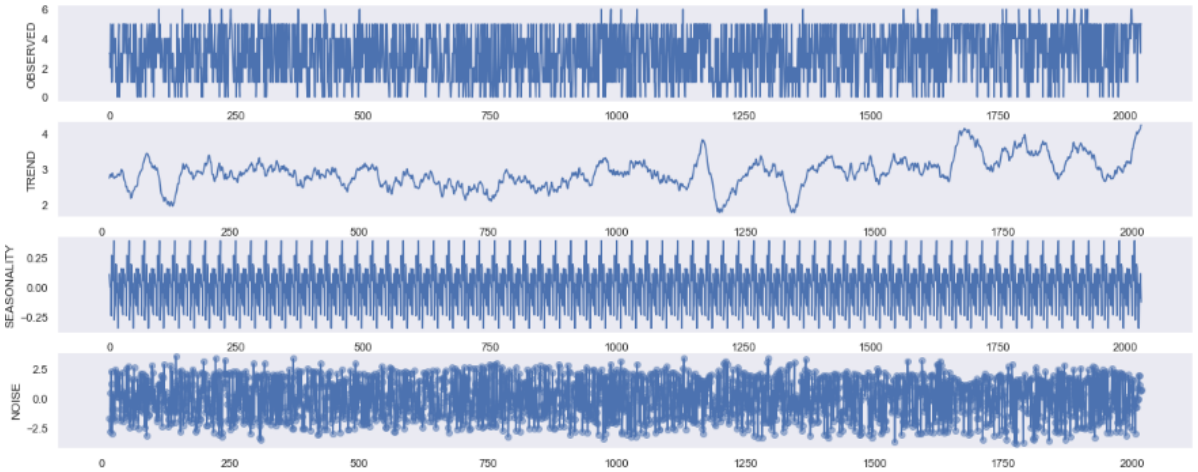



Figura 15: Plot de las STR tendencia, estacionalidad y ruido de la variable objetivo rest_type

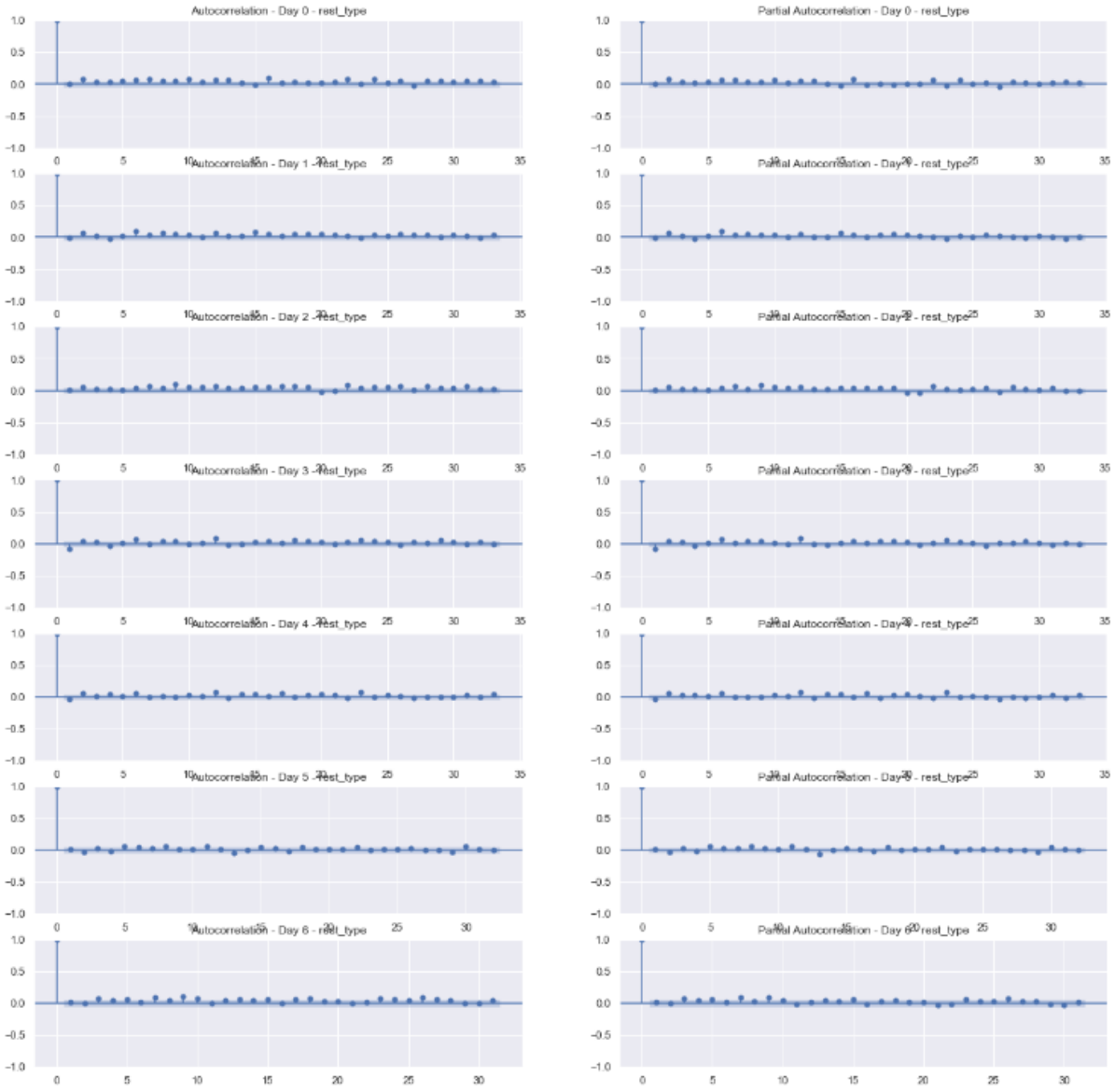



Figura 16: Plot de las funciones ACF y PACF de la variable Rest Type semanal

# Anexo: C - TF Cíclicas

En este anexo se tratan los resultados obtenidos de la transformación de las mejores time features, en variables cíclicas, de diferentes maneras. Para ello generamos 4 sets. Se realizarán 3 posibles transformaciones:

- Basadas en una función radial.

- Basada en valores de seno y coseno.

- Con valores positivos y negativos, de manera que el valor medio sea el 0, así sea posible que exista en punto medio para las variables, ejemplo: [1,2,3,4,5,6] = [1,2,3,-3,-2,-1]

De este modo, las variables temporales seleccioandas han sido las del segundo set, aunque las del tercer set tuvieran una pequeña mejora en el resultado. Esto se debe a que la primera transformación, la ams complicada y normalmente que resulta en mejores resultados, es dificil aplicar a las variables del tercer conjunto, y en cambio, el segundo conjunto si tiene unas cuantas variables que nos permiten testear como funiona.

La primera transformación se aplica a las variable: quarter, month y week

La segunda transformación se aplica a las variables: quarter, month, week, day, dayofweek, dayofyear.

La tercera transformación se aplica a las variables: quarter, month, week, day, dayofweek, dayofyear.

De esta manera, podemos sacar diferentes convinaciones para aplciar técnicas a todas ellas:

- cy + sin: En esta variante se aplica la transformación cíclica a sus variables correspondientes y a las sobrantes la transformación de valores seno y coseno.
- sin/cos: En esta variante se transforman todas las variables con el métodod e seno/coseno.
- cy + positive/negative: En esta variante se aplica la transformación cíclica a sus variables correspondientes y a las sobrantes la transformación de valores positivos y negativos.

| Var | Eva | V1 | V2 | V3 | V4mean | |
|---|---|---|---|---|---|---|
| cy1 | MAE - RF | 20,77 | 20,91 | 6,22 | 6,26 | 13,54 |
| | MAE - TSCV | 20,16 | 20,67 | 6,36 | 6,32 | 13,37 |
| | MAE - BCV | 20,31 | 20,66 | 6,28 | 6,29 | 13,38 |
| cy2 | MAE | 20,62 | 21,030 | 6,03 | 5,90 | 13,95 |
| | MAE - TSCV | 21,82 | 21,93 | 6,08 | 6,05 | 13,97 |
| | MAE - BCV | 21,49 | 22,02 | 6,01 | 6,08 | 13,9 |
| cy3 | MAE | 20,16 | 20,55 | 6,63 | 6,69 | 13,62 |
| | MAE - TSCV | 20,05 | 20,34 | 6,77 | 6,81 | 13,49 |
| | MAE - BCV | 19,94 | 20,42 | 6,76 | 6,79 | 13,47 |
| cy4 | MAE | 22,13 | 22,45 | 6,63 | 6,69 | 14,47 |
| | MAE - TSCV | 22,64 | 22,87 | 6,7 | 6,85 | 14,76 |
| | MAE - BCV | 22,65 | 22,87 | 6,73 | 6,83 | 14,77 |

Tabla 1: Resultados de la predicción con Random Forest Regressor para todos los sets de Time Features Cíclicos del DF y con el mejor número de lags encontrado

- Por puro ámbito cientifico, y ya que es la única que mantiene las variables originales en una única modificada, se prueba también con todas ellas transformadas por el método positivo y negativo.

Explique algo de esto en el Estado del ARTE??

**Predicción Cy Base**:

**Tiempo de computación:** Debido al gran numero de feature existente, sobre todo para el primer set de Time Features Cíclicas, se ha incrementado mucho la complejidad del modelo, y con ello el coste de computación. Por lo que los tiempos para el entreno y predicción en la simulación, has sido de: 20 minutos para RF, 9 horas para TSCV y 6 horas para BCV. Sigue dando mejores resultados BCV, pero los resultados han empeorado bastante, solbre todo para la variable 1 y 2, volviendo a tener un MAE medio igual a 20.

Esto se debe a que RF no diferencia valores negativos de positivos, y tampoco es capaz de computar dos features o columnas diferentes, en esta caso, columna valor seno, columna valor seno, como una única, por lo que da mas importancia a una que a la otra cuando no debería ser así.

**MIX**: Por estas características que se acaban de mencionar, vamos a investigar si es posible mezclar las variables temporales cíclicas con las variables temporales originales, con la esperanza de mejorar el resultado MAE. Se fusionan los métodos de transformación seno/coseno y la transformación radial con Time Features sin transformar y analizamos los resultados.

Para ello, las transformaciones ocurren en las variables menos importantes para el algoritmo: semana, mes y cuarto. Las Time Features originales son: día, día del año, año, dia de la semana.

| **Var** | **Eva** | **MixSet1** | **MixSet2** | **mean** |
|---|---|---|---|---|
| V1 | MAE - RF | 20,47 | 20,80 | 20,63 |
| | MAE - TSCV | 20,11 | 21,6 | 20,855 |
| | MAE - BCV | 20,02 | 21,46 | 20,74 |
| V2 | MAE | 20,99 | 21,25 | 21,12 |
| | MAE - TSCV | 20,18 | 21,68 | 20,93 |
| | MAE - BCV | 20,31 | 21,68 | 20,99 |
| V3 | MAE | 6,23 | 6,06 | 6,14 |
| | MAE - TSCV | 6,17 | 6,17 | 6,17 |
| | MAE - BCV | 6,18 | 6,13 | 6,15 |
| V4 | MAE | 6,19 | 6,09 | 6,14 |
| | MAE - TSCV | 6,16 | 6,11 | 6,13 |
| | MAE - BCV | 6,09 | 6,07 | 6,08 |

Tabla 2: Resultados de la predicción con Random Forest Regressor para las combinaciones de Time Features Cíclicas y no transformadas del DF y con el mejor número de lags encontrado

**Predicción MIX**:

Los resutados entre métodos de validación y entre los diferentes sets son muy similares, también son muy similares a los DF de Time Features Cíclicas. No hemos conseguido mejorar el resultado.