ANÁLISIS Y PREDICCIÓN DE ACCIDENTES DE TRÁFICO EN CALIFORNIA APLICANDO TÉCNICAS DE PREDICCIÓN DE SERIES TEMPORALES.

CONVOCATORIA: 12 de junio de 2024
ALUMNO: Jorge Tesch Torres
TUTORA: María del Mar Angulo Martinez
GRADO: Matemática Computacional

Índice

Índice de figuras

Resumen

El objetivo principal de este Trabajo Fin de Grado es encontrar modelos adecuados que permitan identificar los patrones de la serie de accidentes de tráfico en California e identificar la técnica con mayor capacidad para generar predicciones sobre la misma, considerando un conjunto amplio de condiciones del entorno.

Se realiza un estudio comparativo de diversas técnicas de análisis y predicción de series temporales: enfoque clásico a partir del análisis de componentes, modelización de Box-Jenkins con técnicas SARIMAX, herramienta Prophet, redes neuronales LSTM y redes de Kolmogorov-Arnold (KAN).

Abstract

The main objective of this Final Degree Project is to find suitable models that can identify patterns in the series of traffic accidents in California and identify the technique with the greatest capacity to generate predictions about them, considering a wide range of environmental conditions.

A comparative study of various time series analysis and prediction techniques is conducted: the classical approach based on component analysis, Box-Jenkins modeling with SARIMAX techniques, Prophet tool, LSTM neural networks, and Kolmogorov-Arnold networks (KAN).

1. Introducción

1.1. Motivación y contexto

En el análisis de datos modernos, la predicción de eventos futuros a partir de datos históricos se ha convertido en una herramienta crucial en muchísimos campos, desde la economía o la meteorología hasta la seguridad vial. En particular, el análisis y la identificación de patrones en las series de accidentes de tráfico y las predicciones que pueden generarse a partir de los mismos es un área de gran interés debido a su potencial para salvar vidas y optimizar la infraestructura vial.

En California, una de las regiones más transitadas de Estados Unidos, la predicción de accidentes de tráfico mediante técnicas de series temporales es especialmente relevante. A pesar de los avances tecnológicos, los accidentes de tráfico siguen siendo una causa importante de mortalidad y costos económicos. La capacidad de prever estos eventos puede ayudar a las autoridades a implementar medidas preventivas y a mejorar la seguridad de los conductores.

Existen diversas técnicas para la predicción de series temporales, cada una con sus propias ventajas y desafíos. Este trabajo se centra en evaluar y comparar varias de estas técnicas aplicadas a la predicción de accidentes de tráfico en California, con el objetivo de identificar la más efectiva.

1.2. Planteamiento del problema

El problema central de este estudio radica en construir, a partir de los datos, diversos modelos que nos permitan, comparando técnicas diferentes, presentar los mejores resultados para predecir accidentes de tráfico en California. Al ser el análisis de series temporales un estudio de amplio y creciente interés, se crean y desarrollan diversas técnicas para esta tarea, cada una de ellas con sus propias fortalezas y debilidades.

1. **Modelo clásico**: Estos modelos se basan en la descomposición de la serie temporal en componentes como tendencia, estacionalidad y ruido, permitiendo capturar patrones históricos para hacer predicciones futuras.
2. **Modelos ARIMA (Enfoque de Box-Jenkins)**: La modelización ARIMA (AutoRegressive Integrated Moving Average) basada en la consideración de la serie temporal como una parte de la historia de un proceso estocástico del que procede la serie.

   Un proceso estocástico es una secuencia de variables aleatorias, ordenadas y equidistantes cronológicamente referidas a una (proceso univariante) o a varias (proceso multivariante) características de una unidad observable en diferentes momentos de forma que, si las circunstancias se mantienen estables para otros períodos, entonces las conclusiones obtenidas del análisis de la serie serán aplicables a otros momentos fuera de ese período muestral.

Modelos estacionales: SARIMA (Seasonal ARIMA) y SARIMAX (Seasonal ARIMA with eXogenous variables) extienden el modelo ARIMA al incluir componentes estacionales y variables exógenas, lo que puede mejorar significativamente la precisión de las predicciones en datos con patrones estacionales claros.

3. **Modelos basados en redes neuronales**: Las redes neuronales de memoria a largo plazo (LSTM) y las redes de Kolmogorov-Arnold (KAN) representan enfoques más actuales que ofrecen la capacidad de capturar relaciones complejas y no lineales en los datos. Las LSTM son especialmente útiles para capturar dependencias a largo plazo, mientras que las KAN pueden ofrecer una mayor interpretabilidad y precisión en ciertos contextos.
4. **Prophet**: Esta herramienta desarrollada por Facebook combina componentes de tendencia, estacionalidad y eventos especiales en un marco fácil de ajustar y aplicar, siendo especialmente útil para analistas con conocimientos limitados en estadística.

1.3. Objetivos del trabajo

Los objetivos que se presentan en este trabajo son los siguientes:

1. Evaluar ventajas y desventajas de diversas técnicas de análisis de series temporales a la hora de construir modelos de predicción de accidentes de tráfico en California
2. Realización de un análisis exploratorio y descriptivo de los datos, incluyendo la visualización de gráficas para entender mejor la distribución y las características de los patrones que pueden identificarse en la serie de accidentes de tráfico en la zona estudiada
3. Análisis estadístico de la~~s~~ serie~~s~~ temporale~~s~~ sobre las que se van a realizar las predicciones, identificando tendencias, estacionalidades y componentes residuales. Incluye el estudio de posibles autocorrelaciones para entender mejor la estructura temporal de los datos.
4. Implementación de diferentes modelos de predicción con el fin de comparar y mejorar predicciones mediante el uso de técnicas avanzadas.
5. Comparación de las predicciones obtenidas por cada modelo utilizando los gráficos comparativos entre los datos históricos y las predicciones obtenidas y métricas de evaluación como el error cuadrático medio (MSE), la raíz del error cuadrático medio (RMSE), el error absoluto medio (MAE) y el coeficiente de determinación (R^2).

El objetivo general consiste en identificar qué patrones y elementos definen el comportamiento de los accidentes de tráfico en California para tratar de construir un modelo de predicción efectivo y proporcionar a su vez un marco teórico y práctico que pueda ser utilizado por las autoridades para mejorar la seguridad vial en California.

2. Estado de la Cuestión

2.1. Tráfico y accidentes en California

El tráfico y los accidentes en California son temas de vital importancia debido a su impacto significativo en la seguridad vial y en la economía del estado. California, conocida por su extensa red de carreteras y autopistas, enfrenta desafíos únicos debido a su alta densidad de población y su diversidad geográfica y climática. Este estudio proporciona una visión general del estado del tráfico y los accidentes en California, utilizando datos recientes y estudios relevantes para comprender mejor las tendencias y los factores que influyen en estos incidentes.

2.1.1 Situación actual del tráfico en California

California es el estado más poblado de los Estados Unidos, con una población que supera los 39 millones de habitantes. Este elevado número de residentes, junto con una considerable afluencia de turistas, contribuye a la existencia de un volumen de tráfico excepcionalmente alto. Los centros urbanos como Los Ángeles, San Francisco y San Diego son particularmente conocidos por sus problemas de congestión vehicular. Según el informe de movilidad urbana del Texas A&M Transportation Institute [14], Los Ángeles aparece en casi todas las clasificaciones como una de las ciudades con peor estado de tráfico en los Estados Unidos con un dato significativo que cifra, en promedio, en más de 100 el número de horas al año que los conductores se ven atrapados en atascos.

2.1.2. Estadísticas de accidentes

Los accidentes de tráfico son una preocupación crítica en California, tanto en términos cuantitativos como por su gravedad. El Departamento de Vehículos Motorizados de California (DMV) reporta anualmente miles de accidentes [15], con un número significativo de estos que resultan además en lesiones graves y muertes.

En 2022, hubo más de 3.500 muertes relacionadas con accidentes de tráfico en el estado, una cifra que ha ido en aumento en los últimos años.

Las principales causas de accidentes en California incluyen la conducción bajo la influencia del alcohol y las drogas, el exceso de velocidad, la distracción al volante y la falta de uso del cinturón de seguridad. En particular, el número de accidentes causados por conductores distraídos se ha incrementado notablemente con el uso de teléfonos móviles y otros dispositivos electrónicos mientras se conduce.

2.1.3. Factores contribuyentes y análisis de tendencias

Distracción al Volante

La distracción al volante es una de las principales causas de accidentes en California. La proliferación de dispositivos móviles ha llevado a un aumento en los incidentes de

conductores que utilizan sus teléfonos para mensajes de texto, llamadas o navegación mientras conducen. Según la Administración Nacional de Seguridad del Tráfico en las Carreteras (NHTSA) [16], los conductores que usan teléfonos móviles tienen cuatro veces más probabilidades de verse involucrados en un accidente grave.

Conducción bajo la influencia de alcohol y drogas

La conducción bajo la influencia del alcohol y las drogas sigue siendo un problema significativo en California. Los esfuerzos de las autoridades para reducir estos incidentes incluyen campañas de concienciación, controles de sobriedad y leyes más estrictas. Sin embargo, el número de accidentes relacionados con el alcohol sigue siendo alto, contribuyendo significativamente a la mortalidad en las carreteras.

Condiciones meteorológicas

California, por su diversidad climática, también enfrenta desafíos específicos relacionados con las condiciones meteorológicas. Las lluvias, la niebla y, en algunas áreas, la nieve, pueden aumentar significativamente el riesgo de accidentes. La falta de visibilidad y las carreteras resbaladizas son factores que contribuyen a estos incidentes.

2.1.4. Medidas de prevención y futuras iniciativas

Para abordar el problema de los accidentes de tráfico, el estado de California ha implementado varias medidas, incluyendo:

1. **Campañas de concientización pública**: Programas educativos que enfatizan la importancia de la conducción segura y el uso del cinturón de seguridad.
2. **Mejoras en la infraestructura vial**: Proyectos para mejorar la calidad de las carreteras y la señalización, así como la implementación de tecnologías de tráfico inteligentes para gestionar mejor el flujo vehicular.
3. **Aplicación rigurosa de la ley**: Mayor presencia policial en las carreteras y uso de tecnologías como cámaras de tráfico para detectar y multar a los infractores.
4. **Investigación y desarrollo de tecnologías**: Fomento del uso de vehículos autónomos y sistemas avanzados de asistencia al conductor (ADAS) para reducir el error humano en la conducción.

2.2. Marco teórico

2.2.1. Procesos estocásticos

Un proceso estocástico es una colección de variables aleatorias indexadas por el tiempo o el espacio que representan fenómenos que evolucionan de manera aleatoria. Estos procesos se utilizan para modelar sistemas que aparentemente siguen un patrón, pero tienen una incertidumbre inherente. Al analizar procesos estocásticos, es posible describir la dinámica de sistemas complejos y predecir su comportamiento futuro mediante herramientas probabilísticas y estadísticas. Ejemplos de procesos estocásticos

incluyen el movimiento browniano, las colas de espera y ciertos modelos de crecimiento poblacional.

2.2.2 Series temporales

Una serie temporal [4] es una secuencia cronológica de observaciones, que captura la evolución de una o más variables a lo largo del tiempo. Estas series son fundamentales en diversos campos, desde la economía y las finanzas hasta la meteorología y la demografía, ya que permiten analizar patrones, identificar tendencias y realizar predicciones basadas en datos históricos.

La relación entre los procesos estocásticos y las series temporales es intrínseca, ya que muchas series temporales pueden ser modeladas como procesos estocásticos. Esto permite aprovechar las propiedades matemáticas de los procesos estocásticos para analizar y predecir el comportamiento de las series temporales, incorporando tanto la estructura determinista como la componente aleatoria de los datos observados.

Naturaleza y componentes clave

Las series temporales pueden ser univariantes, cuando se enfocan en una sola variable, o multivariantes, si consideran múltiples variables interrelacionadas. Cada dato de la serie representa el valor de la variable en un momento específico, y la frecuencia de las observaciones puede variar desde intervalos anuales hasta mediciones diarias u horarias.

Estas series poseen cuatro componentes fundamentales:

- **Tendencia $T(t)$:** La tendencia refleja el comportamiento a largo plazo de la variable, indicando si su nivel medio aumenta, disminuye o se mantiene estable a lo largo del tiempo.
- **Estacionalidad $S(t)$**: Muchas series exhiben patrones cíclicos que se repiten en períodos regulares, generalmente inferiores a un año. Estos ciclos estacionales pueden deberse a factores naturales, como las estaciones climáticas, o institucionales, como las vacaciones escolares o los horarios comerciales.
- **Ciclos $C(t)$:** Además de la estacionalidad, algunas series reflejan ciclos más prolongados y de duración variable, superiores a un año. Estos ciclos suelen estar asociados a fluctuaciones económicas, como los períodos de recesión y recuperación.
- **Componente residual o aleatoria $\epsilon(t)$**: Esta componente engloba todas las fluctuaciones impredecibles y aleatorias que no se ajustan a ningún patrón discernible. Pueden ser causadas por eventos extraordinarios, como huelgas o catástrofes naturales, y representan el ruido inherente a cualquier serie temporal.

De esta manera, la estructura general de una serie temporal puede ser representada como:

$$Y(t) = T(t) + S(t) + C(t) + \epsilon(t)$$

Donde:

- $Y(t)$ representa el valor observado de la serie temporal en el tiempo t.
- $T(t),\ S(t),\ C(t), \epsilon(t)$ son los componentes de tendencia, estacionalidad, ciclo y residual, respectivamente.

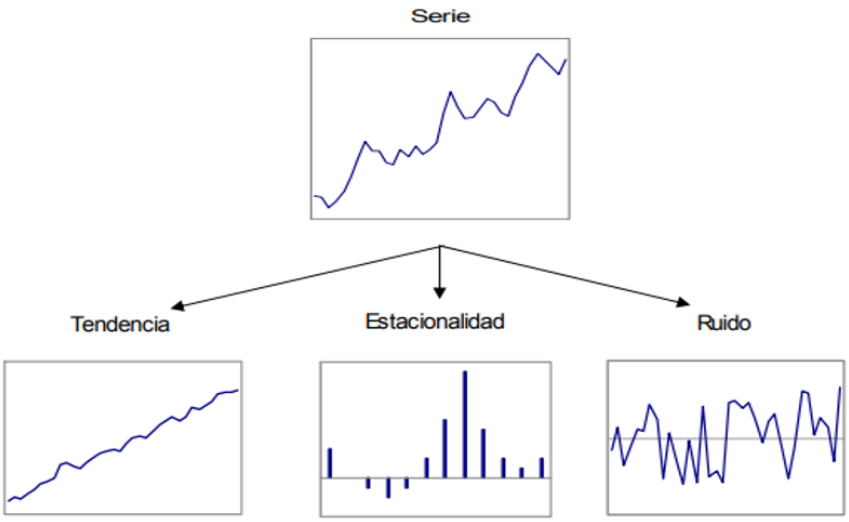

Figura 1 Componentes de una serie temporal. Fuente: [4]

Tipos y propiedades de las series temporales

Las series temporales pueden clasificarse de diversas formas según sus características:

- **Series discretas o continuas**: Las series discretas recopilan datos en momentos específicos, mientras que las continuas registran valores acumulados a lo largo de un período.
- **Series de flujo o stock**: Las series de flujo miden cantidades que se acumulan con el tiempo (por ejemplo, la producción mensual), mientras que las de stock reflejan valores puntuales (como la población a una fecha determinada).
- **Series homocedásticas o heterocedásticas**: Las series homocedásticas mantienen una variabilidad constante, mientras que, en las heterocedásticas, la varianza cambia a lo largo del tiempo.

Además, las series temporales pueden exhibir propiedades clave, como la estacionariedad y la ergodicidad, que influyen en el enfoque analítico empleado.

2.2.3. Autocorrelación simple y parcial

Para analizar las relaciones dentro de una serie temporal, es importante entender los conceptos de autocorrelación simple y autocorrelación parcial.

- **Autocorrelación simple**: Mide el grado de correlación entre los valores de la serie temporal en distintos momentos del tiempo. La función de autocorrelación (ACF) se utiliza para identificar el nivel de correlación entre $Y(t)$ y $Y(t-k)$, donde k es el retardo (lag). Esta función es útil para detectar patrones repetitivos o estacionales en los datos.
- **Autocorrelación parcial**: La función de autocorrelación parcial (PACF) mide la correlación entre dos puntos de la serie temporal, $Y(t)$ y $Y(t-k)$, eliminando el efecto de las observaciones intermedias. La PACF ayuda a identificar el número de retardos significativos en un modelo AR(p), lo que es esencial para la especificación de modelos ARIMA.

2.2.4. Retardo (lag)

El retardo o lag es el desfase temporal entre los valores de una serie temporal. Es una herramienta clave en el análisis de series temporales, ya que permite explorar la relación entre observaciones separadas por diferentes intervalos de tiempo. En los modelos de series temporales, como ARIMA, los retardos se utilizan para capturar la dependencia temporal en los datos y para desarrollar modelos predictivos efectivos.

2.2.5. Estacionariedad

Una serie se considera estacionaria cuando sus propiedades estadísticas (media, varianza y covarianza) se mantienen estables a lo largo del tiempo, sin presentar tendencias ni cambios significativos. Estas series son más fáciles de analizar y predecir, ya que sus fluctuaciones se deben únicamente a factores aleatorios.

- $\boldsymbol{E[Y(t)] = \mu}$: Indica que la media esperada de la serie temporal $Y(t)$ es constante y no varía con el tiempo.
- $\boldsymbol{Var\big(Y(t)\big) = \sigma^2}$: Indica que la varianza de la serie temporal $Y(t)$ es constante y no varía con el tiempo.
- $\boldsymbol{Cov\big(Y(t), Y(t+k)\big) = \gamma(k)}$: Indica que la covarianza entre los valores de la serie temporal $Y(t)$ y $Y(t+k)$ depende únicamente del desfase k, y no del tiempo t.

Por otro lado, las series no estacionarias presentan cambios en su media, varianza o ambas, lo que dificulta su análisis y requiere transformaciones previas para lograr la estacionariedad.

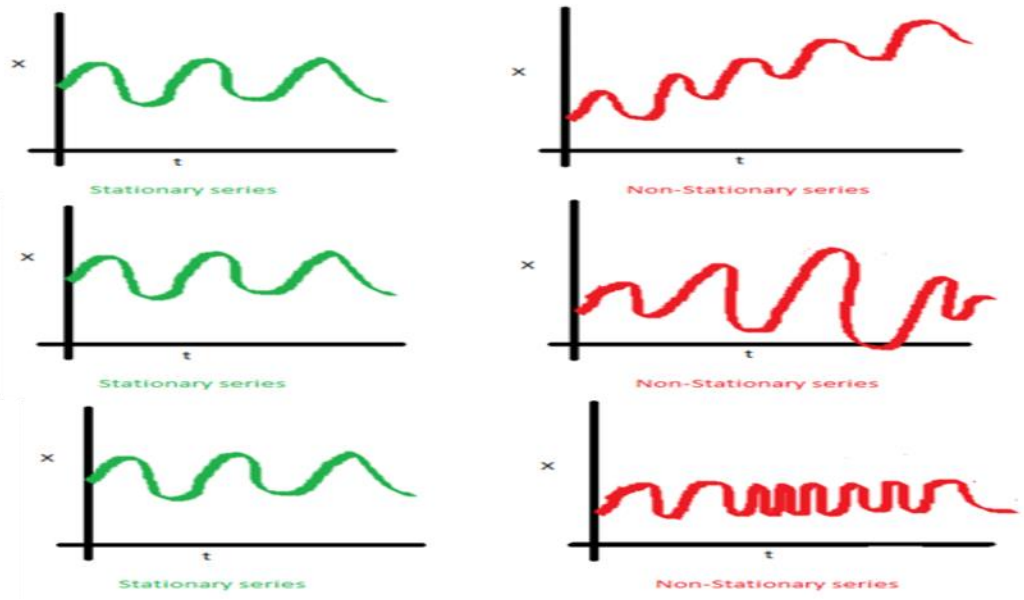

Figura 2 Series estacionarias y no estacionarias. Fuente: [4]

2.2.6. Ergodicidad

Una serie temporal se considera ergódica cuando la interdependencia entre observaciones disminuye a medida que aumenta el tiempo transcurrido entre ellas. En otras palabras, las observaciones más recientes tienen mayor influencia en el comportamiento futuro de la serie que las observaciones más antiguas.

- $\boldsymbol{lim_{k\to\infty}\,\gamma(k)=0}$: Indica que, a medida que el desfase k tiende a infinito, la covarianza entre $Y(t)$ y $Y(t+k)$tiende a cero, reflejando la disminución de la interdependencia entre las observaciones a lo largo del tiempo.

Esta propiedad es fundamental para realizar predicciones confiables, ya que permite asignar mayor peso a los datos más actuales y minimizar el impacto de observaciones distantes en el tiempo.

2.2.7. Objetivos y enfoques del análisis de series temporales

El análisis de series temporales persigue dos objetivos principales:

- **Descripción y comprensión**: Identificar y describir los patrones subyacentes en la evolución de la variable, así como las relaciones contemporáneas y dinámicas entre variables en series multivariantes. Este análisis descriptivo se realiza mediante gráficos, funciones estadísticas y técnicas de descomposición en componentes.
- **Predicción**: Desarrollar modelos capaces de predecir el comportamiento futuro de la variable, basándose en su evolución histórica y en los patrones identificados.

2.2.8. Modelos de predicción de series temporales

El objetivo de este trabajo, sin embargo, no radica solamente en el estudio de las series temporales que ofrecen los datos, sino que trata sobre todo de aplicar diferentes modelos para realizar predicciones realistas de la evolución de la serie o del fenómeno en cuestión en el futuro. Por ello, es de crucial importancia comprender en qué se fundamenta la predicción de series temporales y las diferentes técnicas que giran a su alrededor.

Las series temporales, conjuntos de observaciones ordenadas cronológicamente, ofrecen una ventana invaluable para comprender el comportamiento pasado y proyectar escenarios venideros. Sin embargo, el verdadero desafío radica en dominar las técnicas adecuadas para extraer insights valiosos de estos datos y traducirlos en predicciones precisas.

2.2.9. Enfoque clásico

En el enfoque clásico [7][17], se asume que una serie temporal es una función del tiempo, donde la variable de interés depende del factor temporal. Este marco conceptual descompone la serie en cuatro componentes clave: tendencia, variaciones estacionales, variaciones cíclicas y variaciones residuales o aleatorias.

2.2.9.1. Modelos de descomposición

El análisis clásico de series temporales se basa en dos modelos principales de descomposición: el modelo aditivo y el modelo multiplicativo. Estos esquemas suponen que la serie observada es la suma o el producto, respectivamente, de sus componentes constituyentes.

Modelo aditivo

En el modelo aditivo, la serie temporal se expresa como la suma de sus componentes:

$$Y(t) = T(t) + S(t) + C(t) + \epsilon(t)$$

Donde:

- $Y(t)$ es el valor observado de la serie en el instante t.
- $T(t)$ es la componente de tendencia.
- $S(t)$ es la componente estacional.
- $C(t)$ es la componente cíclica.
- $\epsilon(t)$ es la componente residual o aleatoria.

Este modelo es adecuado cuando la amplitud de las variaciones estacionales y cíclicas es independiente del nivel de la serie, es decir, cuando estas fluctuaciones son aproximadamente constantes a lo largo del tiempo.

Modelo multiplicativo

En el modelo multiplicativo, la serie temporal se expresa como el producto de sus componentes:

$$Y(t) = T(t) * S(t) * C(t) * \epsilon(t)$$

El modelo multiplicativo es más apropiado cuando la amplitud de las variaciones estacionales y cíclicas depende del nivel de la serie, es decir, cuando estas fluctuaciones aumentan o disminuyen proporcionalmente a medida que la serie crece o decrece.

La elección entre el modelo aditivo o multiplicativo depende de la naturaleza de los datos y se puede determinar mediante técnicas gráficas, análisis de variabilidad o criterios estadísticos.

2.2.9.2. Estimación de la tendencia

La estimación de la tendencia es un paso crucial en el análisis de series temporales, ya que permite visualizar el patrón subyacente a largo plazo y eliminar las fluctuaciones de corto y mediano plazo. Existen varios métodos para estimar la tendencia, cada uno con sus ventajas y limitaciones.

Método gráfico

El método gráfico consiste en representar la serie temporal en un sistema de coordenadas y trazar una línea suave que capture la dirección general del fenómeno. Esta técnica, aunque subjetiva, puede ser útil para obtener una primera aproximación visual de la tendencia.

Ajuste analítico

El ajuste analítico implica ajustar una función matemática a los datos de la serie temporal. Las funciones más comunes utilizadas son:

- Tendencia lineal: $T(t) = a + bt$
- Tendencia logarítmica: $T(t) = a + b * ln(t)$
- Tendencia polinómica: $T(t) = a + bt + ct\text{^}2 + \ldots + nt\text{^}n$
- Tendencia exponencial: $T(t) = a \times b\text{^}t$
- Tendencia potencial: $T(t) = a \times t\text{^}b$

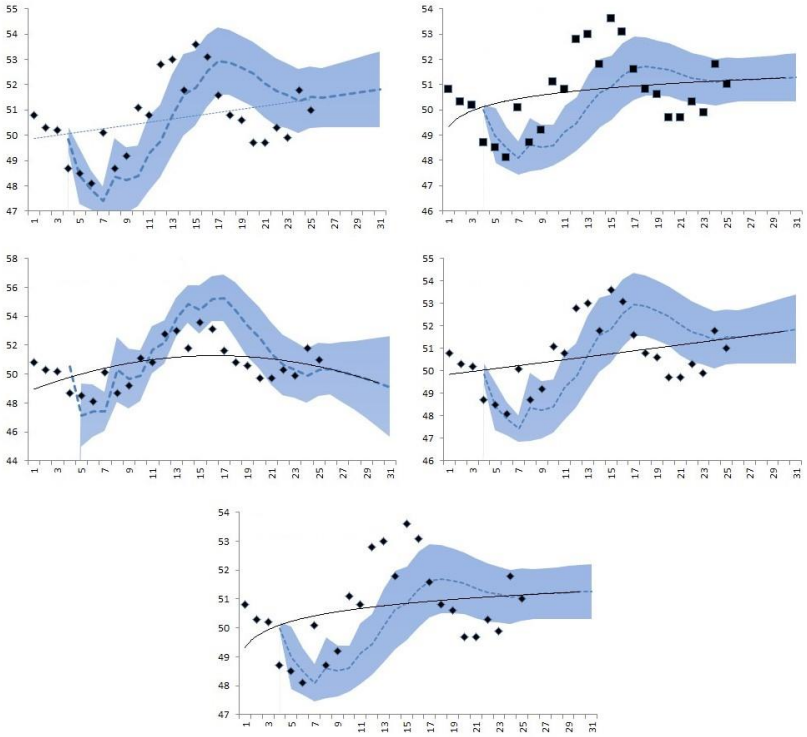

Figura 3 Ajuste analítico, en orden de izquierda a derecha y de arriba abajo, tendencia lineal, tendencia logarítmica, tendencia polinómica (cuadrática), tendencia exponencial y tendencia potencial. Fuente: [18]

El ajuste analítico se realiza mediante técnicas de regresión, como el método de mínimos cuadrados, y el modelo seleccionado dependerá del patrón observado en los datos y de la capacidad de la función para capturar adecuadamente la tendencia.

Método de las medias móviles

El método de las medias móviles consiste en suavizar la serie original reemplazando cada valor por el promedio de un número determinado de observaciones adyacentes. Este enfoque elimina las fluctuaciones irregulares y estacionales, dejando solo la tendencia subyacente.

El cálculo de las medias móviles puede ser centrado o descentrado, dependiendo de si el número de observaciones promediadas es impar o par, respectivamente. El número óptimo de observaciones a promediar dependerá de la frecuencia de los datos y de la duración de las variaciones estacionales y cíclicas.

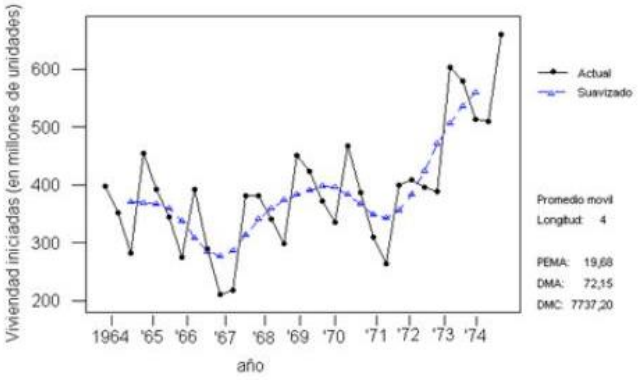

Figura 4 Ajuste de las medias móviles, el segmento negro representa la serie original y el segmento azul representa la serie suavizada. Fuente: [17]

2.2.10. Modelos ARIMA, SARIMA y SARIMAX

En el ámbito del análisis de series temporales, la metodología conocida como Box-Jenkins, nombrada en honor a los reconocidos estadísticos George E. P. Box y Gwilym Jenkins, se utiliza para identificar y ajustar modelos autorregresivos de media móvil (ARMA) o modelos autorregresivos integrados de media móvil (ARIMA). Esta metodología tiene como objetivo principal encontrar el mejor ajuste posible de una serie temporal de datos para que los pronósticos generados sean lo más precisos y confiables posibles.

2.2.10.1. Modelos AR (Autoregressive)

Un modelo autorregresivo (AR) de orden p se denota como AR(p). Este modelo supone que el valor de la serie temporal en un momento dado es una combinación lineal de los valores anteriores de la serie y un término de error. La fórmula general es:

$$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_p y_{t-p} + \epsilon_t$$

Donde:

- y_t es el valor de la serie en el tiempo.
- c es una constante.
- ϕ_i son los coeficientes AR, que indican la relación entre los valores pasados y el valor actual de la serie.
- ϵ_t es el error en el tiempo t, que se asume que es ruido blanco (con media cero y varianza constante).

2.2.10.2. Modelos MA (Moving Average)

Un modelo de promedio móvil (MA) de orden q se denota como MA(q). Este modelo supone que el valor de la serie temporal es una combinación lineal de los errores pasados (ruido blanco). La fórmula general es:

$$y_t = c + \epsilon_t + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + \cdots + \theta_p \epsilon_{t-q}$$

Donde:

- y_t es el valor de la serie en el tiempo.
- c es una constante.
- θ_i son los coeficientes MA, que indican la relación entre los errores pasados y el valor actual de la serie.
- ϵ_t es el error en el tiempo t.

2.2.10.3. Modelos ARMA (Autoregressive Moving Average)

Un modelo ARMA combina los componentes AR y MA. Se denota como ARMA(p, q) y su fórmula general es:

$$y_t = +\phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_p y_{t-p} + \epsilon_t + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + \cdots + \theta_p \epsilon_{t-q}$$

Donde los términos son los mismo que en los modelos AR y MA.

2.2.10.4. ARIMA

Los modelos ARIMA (Autoregressive Integrated Moving Average) combinan tres componentes clave: un proceso autorregresivo (AR), una integración (I) y un promedio móvil (MA). Capturan la dinámica subyacente de una serie temporal para realizar predicciones precisas.

Notación y componentes de un modelo ARIMA

Un modelo ARIMA se denota como ARIMA(p,d,q), donde:

- **p**: orden del componente autorregresivo (AR).
- **d**: grado de diferenciación para lograr la estacionariedad.
- **q**: orden del componente de promedio móvil (MA).

La fórmula general del modelo ARIMA es:

$$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_p y_{t-p} + \epsilon_t + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + \cdots + \theta_q \epsilon_{t-q}$$

Donde:

- y_t es el valor de la serie en el tiempo t.
- c es una constante.
- ϕ_i son los coeficientes AR.
- θ_i son los coeficientes MA.
- ϵ_t es el error en el tiempo t.

La diferenciación se aplica para transformar una serie no estacionaria en estacionaria:

$$y'_t = y_t - y_{t-1}$$

Aplicaciones y beneficios de los modelos ARIMA

Los modelos ARIMA son útiles en economía, finanzas, ingeniería y ciencias ambientales. Su principal beneficio es su enfoque basado en datos, minimizando el sesgo y maximizando la objetividad. Sin embargo, tienen limitaciones, especialmente su incapacidad para incorporar factores externos.

2.2.10.5. SARIMA

Los modelos SARIMA (Seasonal Autoregressive Integrated Moving Average) extienden los modelos ARIMA agregando componentes estacionales.

Notación y componentes de un modelo SARIMA

El modelo SARIMA se denota como SARIMA(p,d,q)(P,D,Q)_m, los parámetros p, d, y q representan las componentes autorregresivas, de diferenciación y de media móvil, respectivamente. P, D, y Q son las mismas componentes para la parte estacional del modelo y m es el número de períodos en cada temporada.

- **p** es el orden (número de lags temporales) de la parte autorregresiva del modelo.
- **d** es el grado de diferenciación (el número de veces que se han restado los valores consecutivos de la serie).
- **q** es el tamaño de la media móvil del modelo.
- **P** es el orden (número de lags temporales) de la parte estacional del modelo.
- **D** es el grado de diferenciación de la parte estacional del modelo.
- **Q** es el tamaño de la media móvil de la parte estacional del modelo.
- **m** es el parámetro estacional adecuado

Cuando los términos P, D, Q, y m son cero, el modelo SARIMA es un modelo ARIMA.

Aplicaciones y beneficios de los modelos SARIMA

Los modelos SARIMA son ideales para series temporales con patrones recurrentes y son aplicables en meteorología, turismo y comercio minorista. Sin embargo, pueden carecer de precisión cuando las series temporales se ven afectadas por factores externos no capturados por el modelo.

2.2.10.6. SARIMAX

Los modelos SARIMAX (Seasonal Autoregressive Integrated Moving Average with Exogenous Variables) extienden los modelos SARIMA al permitir la inclusión de variables exógenas o externas, mejorando la precisión y confiabilidad de las predicciones.

La notación del modelo SARIMAX es la misma que la del modelo SARIMA, incluyendo las variables exógenas, por lo que cuando los términos P, D, Q, y m son cero y no se incluyen variables exógenas, el modelo SARIMAX es un modelo ARIMA.

Ejemplo de SARIMAX

SARIMAX (1,0,2)(2,0,1,5):

- 1,0,2 son los parámetros no estacionales.
- 5 es el ciclo, lo que significa que incluimos los valores de 5 períodos atrás X(t-5) y de 10 períodos atrás X(t-10).
- No hay integración en la parte estacional porque D=0, y añadimos un único error/innovación atrasado (Q=1) a(t-5).

Así, el modelo quedaría:

$$X(t) = \mu + \phi_1 X(t-1) + \zeta_1 a(t-1) + \zeta_2 a(t-2) + \alpha_1[X(t-5) + \phi_1 X(t-6)] + \alpha_2[X(t-10) + \phi_1 X(t-11)] + \beta_2[a(t-5) + \zeta_1 a(t-6) + \zeta_2 a(t-7)] + a(t)$$

En resumen:

- Añadimos dos valores retrasados, cinco y diez días atrás (porque P=2).
- Añadimos un error retrasado cinco días atrás (Q=1).
- Cada uno de estos elementos adicionales tiene un retraso de un día (p=1) y tiene dos errores retrasados (porque q=2).

Aplicaciones y beneficios de los modelos SARIMAX

Los modelos SARIMAX ofrecen mejoras en precisión, flexibilidad e interpretabilidad, siendo útiles en áreas como la predicción de demanda, análisis de series financieras y previsión meteorológica.

2.2.10.4. Selección de variables exógenas

Se requiere de un profundo conocimiento del dominio para seleccionar las variables relevantes.

1. **Multicolinealidad**: La alta correlación entre variables exógenas puede dificultar la estimación precisa de los parámetros del modelo.
2. **No linealidades**: Los modelos SARIMAX asumen relaciones lineales, lo que puede no ser suficiente para capturar dinámicas complejas en la serie temporal.
3. **Requisitos de datos**: Requieren una cantidad sustancial de datos históricos para una estimación precisa, y la calidad de los datos es crucial.
4. **Complejidad computacional**: Aumenta con la inclusión de más componentes y variables exógenas, lo que puede ser un desafío en grandes conjuntos de datos o predicciones en tiempo real.

2.2.11. Prophet

Es un modelo de pronóstico de series temporales diseñado para manejar las características comunes de las series temporales que reflejan características propias de magnitudes empresariales, como múltiples estacionalidades, cambios de tendencia, valores atípicos y efectos de días festivos. Otro elemento característico de Prophet es que está diseñado para tener parámetros intuitivos que pueden ser ajustados sin conocer los detalles del modelo subyacente, lo cual es necesario para poder ajustar el modelo de manera efectiva.

El modelo Prophet [10] combina tres componentes principales: tendencia, estacionalidad y días festivos. La tendencia se modela mediante una curva de crecimiento logístico con puntos de cambio ajustables, o una tendencia lineal por partes. La estacionalidad se modela mediante series de Fourier, lo que permite patrones estacionales suaves y flexibles. Los efectos de días festivos se incorporan mediante la adición de covariables que indican la proximidad a los días festivos.

La combinación de las tres componentes da como resultado la siguiente fórmula:

$$y(t) = g(t) + s(t) + h(t) + \epsilon_t$$

Donde:

- $g(t)$ es la función de tendencia que modela los cambios no periódicos en el valor de la serie temporal.
- $s(t)$ representa los cambios periódicos.
- $h(t)$ representa los efectos de los días festivos que ocurren en horarios potencialmente irregulares durante uno o más días.
- ϵ_t representa cualquier cambio idiosincrásico que no es acomodado por el modelo.

2.2.11.1. Modelo de tendencia

Se han implementado dos modelos de tendencia que cubren muchas aplicaciones: un modelo de crecimiento saturado y un modelo lineal por partes.

Crecimiento saturado no lineal

Para el pronóstico de crecimiento, el componente central del proceso generador de datos es un modelo que refleja cómo ha crecido la población y cómo cabe esperar que continúe creciendo. El modelado del crecimiento a menudo es similar al crecimiento poblacional en ecosistemas naturales, donde hay un crecimiento no lineal que se satura en una capacidad de carga.

Se utiliza un modelo de crecimiento logístico para representar este comportamiento, con una capacidad de carga variable en el tiempo para capturar los cambios en el techo de

crecimiento. Además, se permiten cambios en la tasa de crecimiento en puntos específicos para representar el impacto de nuevos productos o cambios en el mercado.

La fórmula del modelo de crecimiento logístico es:

$$g(t) = \frac{C(t)}{1 + exp\,(-k(t - m))}$$

Donde:

- $y(t)$ es el valor en el tiempo t.
- $C(t)$ es la capacidad de carga en el tiempo t.
- k es la tasa de crecimiento.
- m es el punto medio de la curva logística.

Tendencia lineal con puntos de cambio

Para problemas de pronóstico que no exhiben crecimiento saturado, una tasa de crecimiento constante por partes proporciona un modelo parsimonioso y a menudo útil. Aquí, el modelo de tendencia es una función lineal por partes con puntos de cambio ajustables.

La fórmula del modelo de tendencia lineal con puntos de cambio es:

$$g(t) = (a + bt) + \sum_{j=1}^{J} \delta_j D_j(t)$$

Donde:

- a es el nivel inicial.
- b es la tasa de crecimiento.
- $D_j(t)$ es una función indicadora que toma el valor 1 si t es posterior al punto de cambio t_j, y 0 en caso contrario.
- δ_j es el cambio en la tasa de crecimiento en el punto t_j.

Selección automática de puntos de cambio

Los puntos de cambio pueden ser especificados por el analista utilizando fechas conocidas de lanzamientos de productos y otros eventos que alteran el crecimiento, o pueden seleccionarse automáticamente dados un conjunto de candidatos. La selección automática se puede realizar de manera natural mediante el uso de una distribución previa dispersa en los ajustes de tasa.

2.2.11.2. Estacionalidad

Las series temporales empresariales a menudo presentan estacionalidades de múltiples períodos como resultado de los comportamientos humanos que representan. Para

ajustar y pronosticar estos efectos, se especifican modelos de estacionalidad que son funciones periódicas del tiempo.

Se utilizan series de Fourier para proporcionar un modelo flexible de los efectos periódicos. Los efectos estacionales suaves se pueden aproximar mediante una combinación de términos seno y coseno, donde el número de términos controla la suavidad del patrón estacional.

La fórmula de la estacionalidad usando series de Fourier es:

$$s(t) = \sum_{n=1}^{N} \left(a_n \cos\left(\frac{2\pi n t}{P}\right) + b_n \sin\left(\frac{2\pi n t}{P}\right) \right)$$

Donde:

- P es el período de la estacionalidad (e.g., 365.25 para anual).
- N es el número de términos de Fourier.
- a_n y b_n son los coeficientes de Fourier.

2.2.11.3. Días festivos y eventos

Los días festivos y eventos proporcionan cambios grandes, pero en cierta medida predecibles, en muchas series temporales empresariales y, a menudo, no siguen un patrón periódico.

Se permite que el analista proporcione una lista personalizada de eventos y días festivos pasados y futuros, identificados por su nombre único. Incorporar esta lista en el modelo es sencillo, asumiendo que los efectos de los días festivos son independientes. Para cada día festivo, se agrega una función indicadora que representa si el tiempo t está durante ese día festivo, y se asigna un parámetro correspondiente que captura el cambio en el pronóstico.

La fórmula para incorporar días festivos es:

$$h(t) = \sum_{k} \alpha_k D_k(t)$$

Donde:

- $D_k(t)$ es una función indicadora que toma el valor 1 si el tiempo t es durante el día festivo k, y 0 en caso contrario.

- α_k es el parámetro correspondiente al efecto del día festivo k.

2.2.12. LSTM

Las redes neuronales recurrentes (RNNs) son efectivas para analizar patrones secuenciales en aplicaciones como el procesamiento del lenguaje natural y la predicción de series temporales. Sin embargo, tienen dificultades para capturar dependencias a largo plazo debido al problema de la explosión de gradiente durante el entrenamiento. Este problema obstaculizó el progreso en el modelado de series temporales hasta la introducción del algoritmo Long Short-Term Memory (LSTM) [11] en 1997 por Sepp Hochreiter y Jürgen Schmidhuber.

LSTM supera estas dificultades mediante "celdas de memoria" y "compuertas" que permiten retener información contextual por períodos prolongados. Esto es crucial para series temporales complejas, como precios de acciones o patrones climáticos, que requieren la retención de información a largo plazo que las RNNs convencionales no pueden manejar debido a la explosión de gradiente.

2.2.12.1. Arquitectura de LSTM

LSTM aborda este desafío al introducir una arquitectura de red especializada que facilita el flujo constante de información relevante a través de la secuencia. En lugar de las unidades ocultas convencionales, LSTM utiliza celdas de memoria especiales que pueden mantener su estado interno durante largos períodos de tiempo.

Cada celda de memoria LSTM consta de tres compuertas multiplicativas: una compuerta de entrada, una compuerta de olvido y una compuerta de salida. Estas compuertas controlan el flujo de información hacia y desde la celda, determinando qué datos se almacenan, se olvidan o se utilizan para la salida.

- **Compuerta de entrada**: Decide qué información nueva de la entrada de la secuencia y del estado oculto anterior se escribirá en la celda.
- **Compuerta de olvido**: Determina qué información del estado anterior de la celda se conservará o se eliminará.
- **Compuerta de salida**: Controla qué parte del estado actual de la celda se utilizará como salida y se propagará a las siguientes etapas de la secuencia.

Esta estructura de compuertas inteligentes permite que LSTM capture dependencias temporales muy largas al regular el flujo de información a través de las celdas de memoria. Los gradientes pueden fluir sin obstáculos a través de las celdas, evitando así el problema de la explosión del gradiente que afecta a las RNNs tradicionales.

2.2.12.2. Funcionamiento interno de LSTM

Ahora que tenemos una idea general de la arquitectura LSTM, profundicemos en los detalles de su funcionamiento interno. Cada celda de memoria LSTM realiza una serie de cálculos en cada paso de tiempo de la secuencia de entrada.

Paso 1: Decidir qué información olvidar

La compuerta de olvido determina qué parte del estado anterior de la celda se conservará o se descartará. Esto se logra mediante una operación de multiplicación de punto entre una capa de activación sigmoidea y el estado anterior de la celda.

$$f_t = \sigma\big(W_f \cdot [h_{t-1}, x_t] + b_f\big)$$

Donde:

- f_t es la salida de la compuerta de olvido en el tiempo t.
- σ es la función de activación sigmoidea.
- W_f es la matriz de pesos de la compuerta de olvido.
- h_{t-1} es el estado oculto anterior.
- x_t es la entrada actual.
- b_f es el sesgo de la compuerta de olvido.

Paso 2: Decidir qué información almacenar

La compuerta de entrada decide qué nueva información se agregará al estado de la celda. Primero, una capa sigmoidea decide qué valores se actualizarán. Luego, una capa de activación tangente hiperbólica crea un vector de nuevos valores candidatos, $\tilde{C}_t$, que podrían agregarse al estado.

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

Donde:

- i_t es la salida de la compuerta de entrada en el tiempo t.
- $\tilde{C}_t$ es el vector de estados candidatos en el tiempo t.
- W_i , W_C son las matrices de pesos de las compuertas de entrada y estado candidato, respectivamente.
- b_i, b_C son los sesgos correspondientes.

Paso 3: Actualizar el estado de la celda

El nuevo estado de la celda, C_t, se calcula combinando el estado anterior modificado por la compuerta de olvido y la nueva información candidata modulada por la compuerta de entrada.

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Donde $*$ denota la multiplicación de punto elemento a elemento.

Paso 4: Decidir la salida

Finalmente, la compuerta de salida decide qué parte del estado actual de la celda se propagará como salida y se utilizará para la siguiente etapa de la secuencia. Primero, una capa sigmoidea decide qué partes del estado interno se utilizarán como salida. Luego, se aplica la función tangente hiperbólica al estado interno y se multiplica por la salida sigmoidea para obtener la salida final.

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

$$h_t = o_t * \tanh(C_t)$$

Donde:

- o_t es la salida de la compuerta de salida en el tiempo t.
- h_t es el estado oculto actual (salida de la celda).
- W_o es la matriz de pesos de la compuerta de salida.
- b_o es el sesgo de la compuerta de salida.

Este proceso se repite para cada paso de tiempo de la secuencia de entrada, permitiendo que LSTM capture dependencias temporales complejas y prolongadas de manera eficiente.

2.2.12.3. Ventajas y limitaciones de LSTM

Como cualquier técnica de aprendizaje automático, LSTM tiene sus fortalezas y debilidades. Exploremos algunas de las principales ventajas y limitaciones de este enfoque.

Ventajas

- **Capacidad de capturar dependencias a largo plazo**: La principal ventaja de LSTM es su capacidad para modelar dependencias temporales prolongadas de manera efectiva, superando el problema de la explosión de gradiente que afecta a las RNNs convencionales.
- **Manejo de secuencias de longitud variable**: LSTM puede procesar secuencias de entrada de longitud variable sin la necesidad de recortar o rellenar, lo que lo hace adecuado para una amplia gama de aplicaciones.
- **Robustez ante ruido y datos faltantes**: Gracias a su estructura de compuertas inteligentes, LSTM es más resistente al ruido y los datos faltantes en comparación con otros modelos de series temporales.
- **Capacidad de aprender patrones complejos**: LSTM puede capturar patrones no lineales y relaciones complejas en los datos, lo que lo convierte en una herramienta poderosa para el modelado de series temporales en diversos dominios.

Limitaciones

- **Complejidad computacional**: El entrenamiento de redes LSTM puede ser computacionalmente más costoso que otros modelos de series temporales, especialmente para secuencias muy largas o conjuntos de datos grandes.
- **Sensibilidad a la inicialización y el ajuste de hiperparámetros**: El rendimiento de LSTM puede verse afectado por la inicialización de pesos y sesgos, así como por el ajuste de hiperparámetros como las tasas de aprendizaje y la regularización.
- **Interpretabilidad limitada**: Al igual que con otras redes neuronales profundas, los modelos LSTM pueden ser difíciles de interpretar y explicar, lo que puede ser un desafío en aplicaciones donde se requiere una comprensión clara del proceso de toma de decisiones.
- **Problemas con dependencias muy largas**: Si bien LSTM es capaz de capturar dependencias a largo plazo mucho mejor que las RNNs convencionales, aún puede enfrentar dificultades con secuencias extremadamente largas o dependencias que abarcan miles de pasos de tiempo.

A pesar de estas limitaciones, LSTM sigue siendo una herramienta poderosa y ampliamente utilizada en el modelado de series temporales, y continúa siendo un área activa de investigación y desarrollo.

2.2.13. Redes de Kolmogorov-Arnold (KAN)

Las redes neuronales artificiales han revolucionado numerosos campos, desde el procesamiento de imágenes hasta el reconocimiento del habla. Sin embargo, a pesar de sus impresionantes logros, persisten ciertas limitaciones inherentes a las redes neuronales convencionales, como los perceptrones multicapa (MLP). Las redes de Kolmogorov-Arnold (KAN) [12], un enfoque vanguardista inspirado en el teorema de representación de Kolmogorov-Arnold.

2.2.13.1. El teorema de representación de Kolmogorov-Arnold

El teorema de representación de Kolmogorov-Arnold, establecido por los eminentes matemáticos Andrey Kolmogorov y Vladimir Arnold, sienta las bases teóricas de las KAN. Este teorema afirma que cualquier función continua de varias variables en un dominio acotado puede expresarse como una composición finita de funciones continuas de una sola variable y la operación binaria de suma.

Matemáticamente, para una función suave $f: [0,1]^n \to \mathbb{R}$,el teorema establece que:

$$f(x) = f(x_1, \dots, x_n) = \sum_{q=1}^{2n+1} \Phi_q \left(\sum_{p=1}^{n} \varphi_{q,p}(x_p) \right)$$

Donde $\varphi_{q,p}: [0,1] \to \mathbb{R}$ y $\Phi_q: \mathbb{R} \to \mathbb{R}$ son funciones univariables continuas. Esencialmente, el teorema demuestra que la única función multivariable verdadera es la suma, ya que todas las demás funciones pueden escribirse utilizando funciones de una única variable y sumas.

2.2.13.2. Arquitectura de las redes KAN

Aunque la estructura de conexión global de las redes KAN es similar a los perceptrones multicapa existe una diferencia importante y es que mientras estos aplican funciones de activación fijas en los nodos ("neuronas"), las KAN utilizan funciones de activación en los bordes ("pesos").

En una red KAN, cada parámetro de peso se reemplaza por una función univariable parametrizada como una curva spline. Los nodos simplemente suman las señales entrantes sin aplicar ninguna no linealidad. Esta modificación se traduce en mejoras en precisión e interpretabilidad.

La arquitectura de una red KAN se define por una serie de números enteros $[n_0, n_1, \dots, n_L]$, donde n_i representa el número de nodos en la capa i de la red. Entre las capas l y $l+1$, hay $n_l n_{l+1}$ funciones de activación, de modo que cada una de ellas conecta un nodo de la capa l con un nodo de la capa $l+1$.

2.2.13.3. Simplificación e interpretabilidad de las KAN

Además de su alta precisión, las redes KAN destacan por su interpretabilidad inherente que se logra utilizando algunas técnicas de simplificación:

- **Visualización**: Escalar la transparencia de cada función de activación de acuerdo con su magnitud, resaltando las funciones más importantes.
- **Podado**: Eliminar neuronas "no importantes" según un umbral predefinido.
- **Simbolización**: Ajustar funciones de activación a formas simbólicas conocidas, como seno, coseno o logaritmo.

La combinación de estas técnicas de simplificación permite obtener representaciones que permiten entrever la estructura y composición subyacente de los datos

2.2.13.4. Aplicaciones en tareas sintéticas

Para ilustrar el potencial de las redes KAN, consideremos algunos ejemplos sintéticos:

- **Multiplicación** $\boldsymbol{f(x,y) = xy}$: Una Red KAN $[2,2,1]$ aprende a calcular xy utilizando la identidad $2xy = (x+y)^2 - (x^2+y^2)$.
- **División** $\boldsymbol{f(x,y) = \frac{x}{y}}$: Una Red KAN $[2,2,1]$ aprende a calcular $\frac{x}{y}$ utilizando la identidad $\frac{x}{y} = exp(log(x) - log(y))$.

- **Función especial** $\boldsymbol{f(x,y) = exp(J_o(20x) + y^2)}$: Las KAN pueden aprender funciones especiales como la función de Bessel $J_o(20x)$, que sería imposible para la regresión simbólica sin conocimiento previo.
- **Transición de fase** $\boldsymbol{f(x_1, x_2, x_3) = tanh\left(5\left(x_4^3 - 1\right)\right)}$ **:** Las KAN pueden detectar transiciones de fase e identificar los parámetros de orden correctos, como la dependencia cuartica de x_1, x_2, x_3 y la función $tanh$.

Estos ejemplos demuestran cómo las redes KAN pueden revelar estructuras composicionales en fórmulas simbólicas, aprender funciones univariables complejas y detectar patrones subyacentes en los datos, todo de manera interpretable.

2.2.13.5. Ventajas de las redes KAN en el ajuste de datos

Más allá de las tareas sintéticas, las redes KAN han demostrado un rendimiento superior al de las MLP en tareas de ajuste de datos y resolución de ecuaciones diferenciales parciales (EDP).

Las redes KAN pueden evitar además el "olvido catastrófico", donde una red neuronal entrena en una tarea y luego olvida cómo realizar la tarea anterior al cambiar a una nueva, remodelando solo las regiones donde se presentan nuevos datos y dejando intactas las regiones previamente aprendidas.

Esta propiedad ~~de~~ que algunos autores califican de "plasticidad local" es una característica clave que distingue a las KAN de las MLP convencionales.

2.2.14. Evaluación de los modelos de predicción

Una vez descritos todos los modelos de predicción de series temporales que se van a aplicar en el desarrollo de este trabajo, es crucial evaluar su desempeño para garantizar la confiabilidad y precisión de las predicciones. Este proceso implica una serie de pasos y métricas diseñadas para medir la calidad de los pronósticos y detectar posibles problemas o sesgos en los modelos.

Existen varias métricas comúnmente utilizadas para evaluar el desempeño de los modelos de predicción de series temporales:

- **Error cuadrático medio (MSE):** Mide la magnitud promedio de los errores de predicción al cuadrado, dando mayor peso a los errores grandes. La fórmula del MSE es:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

donde y_i son los valores reales y $\hat{y}_i$ son los valores predichos.

- **Raíz del error cuadrático medio (RMSE):** Similar al MSE, pero en la misma escala que los datos originales, lo que facilita la interpretación. La fórmula del RMSE es:

$$RMSE = \sqrt{MSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

- **Error absoluto medio (MAE)**: Mide la magnitud promedio de los errores de predicción en valor absoluto, lo que da una idea clara del error típico. La fórmula del MAE es:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

- **Error porcentual absoluto medio (MAPE)**: Expresa el error como un porcentaje del valor real, facilitando la comparación entre diferentes series temporales. La fórmula del MAPE es:

$$MAPE = \frac{100\%}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right|$$

- **Coeficiente de determinación (R^2)**: Indica la proporción de la varianza en la variable dependiente que es predecible a partir de la variable independiente. Es una medida de bondad de ajuste de las predicciones a los datos reales. La fórmula del R^2 es:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

2.3. Trabajos relacionados

En este apartado se presentarán algunos trabajos similares al que se desarrolla en este documento, enfocados en la predicción de accidentes de tráfico utilizando técnicas de series temporales y modelos de aprendizaje automático. A continuación, se describen brevemente estos trabajos, señalando el valor añadido que aporta el proyecto presente en comparación con los estudios mencionados.

Un estudio destacado en este ámbito es el de Kumar [1], donde se emplearon modelos ARIMA y redes neuronales para predecir la frecuencia de accidentes de tráfico en India. Los autores encontraron que los modelos ARIMA son efectivos para capturar patrones lineales, mientras que las redes neuronales muestran un mejor rendimiento al manejar relaciones no lineales en los datos. Este trabajo subraya la importancia de combinar métodos tradicionales y modernos para mejorar la precisión de las predicciones.

En otro estudio, Abellán [2] emplearon modelos SARIMA para analizar y predecir accidentes de tráfico en la región de Andalucía, España. Los resultados mostraron que SARIMA es particularmente útil para datos con patrones estacionales claros. La investigación destacó la capacidad del modelo para adaptarse a diferentes periodos estacionales y mejorar las estrategias de prevención de accidentes.

Finalmente, un estudio por Yannis [3] exploró la predicción de accidentes en carreteras urbanas y rurales utilizando modelos de regresión y análisis de series temporales. Este trabajo destacó las diferencias en los patrones de accidentes entre áreas urbanas y rurales y la necesidad de adaptar los modelos a estos contextos distintos.

El proyecto presente no solo utiliza ARIMA y redes neuronales LSTM, sino que también incluye modelos avanzados como Prophet y las novedosas redes de Kolmogorov-Arnold (KAN). Esta inclusión permite una comparación más exhaustiva de técnicas y la identificación de la mejor opción para la predicción de accidentes en California, un contexto geográfico y demográfico diferente.

Además, considera la estacionalidad mediante SARIMAX, incorporando variables exógenas como condiciones meteorológicas, proporcionando una perspectiva más completa y precisa de los factores que influyen en los accidentes de tráfico.

El enfoque práctico del presente trabajo en la implementación de modelos de predicción en un entorno real de tráfico en California, utilizando datos históricos reales, aborda la optimización de parámetros y la validación de modelos en profundidad, garantizando que las técnicas aplicadas sean las más adecuadas para el contexto específico.

Una vez investigada la situación del tráfico y los accidentes en California y los trabajos relacionados, se va a dar paso al estudio de las bases matemáticas en las que se fundamente este trabajo, comenzando con los procesos estocásticos y las series temporales.

3. Desarrollo

3.1. Metodología

En este trabajo, el objetivo principal es analizar y predecir accidentes de tráfico en California utilizando técnicas avanzadas de predicción de series temporales. Para ello, se ha diseñado una metodología estructurada en varias etapas clave que guiarán el desarrollo del notebook de Python, que se puede encontrar en el repositorio de GitHub "TFG_MACO" [19].

La primera fase consiste en la recopilación del conjunto de datos históricos de accidentes de tráfico en California. Se utilizará el dataset "US Accidents (2016 - 2023)" [13][20] que contiene información detallada sobre los accidentes, incluyendo fecha, ubicación, condiciones meteorológicas, y otros factores relevantes. Una vez recopilados los datos, se procederá a la limpieza y tratamiento inicial. Esto incluye la eliminación de valores atípicos, el manejo de datos faltantes, y la transformación de variables necesarias para garantizar la calidad y consistencia del conjunto de datos.

A continuación, se realizará un análisis descriptivo de los datos para comprender mejor la distribución y características de los accidentes de tráfico en California. Esto incluirá el cálculo de estadísticas descriptivas como medias, medianas, desviaciones estándar, y la visualización de distribuciones mediante diferentes gráficos. Se utilizarán técnicas de visualización para explorar relaciones y patrones en los datos.

En la fase de desarrollo de técnicas de predicción de series temporales se implementarán cinco modelos distintos:

1. **Medias móviles (Enfoque clásico**): Este método simple pero efectivo utiliza promedios móviles para suavizar la serie temporal y eliminar fluctuaciones aleatorias, capturando tendencias a corto plazo.
2. **SARIMAX**: Extiende el modelo SARIMA al incluir variables exógenas, adecuado para datos con patrones estacionales, incorporando factores externos que pueden influir en la serie temporal.
3. **Prophet**: Desarrollado por Facebook, es útil para series temporales con múltiples estacionalidades y cambios de tendencia. Facilita el ajuste de modelos complejos mediante parámetros intuitivos, capturando patrones estacionales.
4. **LSTM**: Redes neuronales que capturan dependencias a largo plazo en datos secuenciales, adecuadas para series temporales con patrones dependientes de eventos pasados.
5. **Redes de Kolmogorov-Arnold (KAN)**: Es el modelo más novedoso y ofrece mayor interpretabilidad y precisión al descomponer funciones multivariables en funciones univariables y sumas.

Para evaluar el desempeño de los modelos, se utilizarán métricas como el error cuadrático medio (MSE), la raíz del error cuadrático medio (RMSE), el error absoluto medio (MAE) y el coeficiente de determinación (R2). Estas métricas permitirán comparar la precisión de las predicciones realizadas por cada modelo. Se compararán los resultados obtenidos por cada modelo para identificar el que ofrece las mejores predicciones.

Esta metodología estructurada permitirá no solo la implementación efectiva de técnicas de predicción de series temporales en Python, sino también se asegurará la identificación del modelo más eficaz para este contexto específico.

Definida la metodología que se va a seguir, se da paso el apartado en el que se van a explicar los datos que se van a emplear.

3.2. Datos

Para el desarrollo de este trabajo se va a hacer uso del conjunto de datos “US Accidents (2016 - 2023)” [13][20] que recopila datos históricos de accidentes de tráfico en Estados Unidos entre los años 2016 y 2023. Lo que distingue a este conjunto de datos es su cobertura integral de diversos atributos para cada registro de accidente. Además de la información básica como ubicación, hora y descripción del evento, US Accidents proporciona datos contextuales valiosos sobre las condiciones meteorológicas, el período del día, y las anotaciones de puntos de interés cercanos, como cruces, señales de tránsito e intersecciones.

3.2.1. Importancia y aplicaciones

La disponibilidad de un conjunto de datos de accidentes de tránsito a gran escala y de alta calidad como US-Accidents presenta oportunidades significativas para abordar uno de los desafíos más importantes de seguridad pública a nivel mundial.

US Accidents puede utilizarse para diversas aplicaciones, como la predicción de accidentes en tiempo real, el estudio de puntos críticos de accidentes, el análisis de víctimas y la extracción de reglas de causa y efecto para predecir accidentes, o el estudio del impacto de las precipitaciones u otros estímulos ambientales en la ocurrencia de accidentes.

3.2.2. Proceso de creación del conjunto de datos

La creación de US Accidents implicó un proceso detallado de recolección y mejora de datos de tránsito en tiempo real. Primero, se recopilaron datos de tránsito de las APIs de "MapQuest Traffic" y "Microsoft Bing Map Traffic", que incluyen información de accidentes y congestión. Luego, se integraron estos datos eliminando duplicados mediante un umbral heurístico basado en la distancia de Haversine (250 metros) y el tiempo de ocurrencia (10 minutos). Se utilizaron técnicas de geocodificación inversa,

datos meteorológicos, información temporal y anotaciones de puntos de interés para mejorar los datos en múltiples etapas.

3.2.3. Estructura de los datos

En US Accidents, cada registro de accidente incluye las siguientes columnas con información adicional [20]:

1. **ID**: Este es un identificador único del registro del accidente.
2. **Severity**: Muestra la gravedad del accidente, un número entre 1 y 4, donde 1 indica el menor impacto en el tráfico (es decir, un retraso corto como resultado del accidente) y 4 indica un impacto significativo en el tráfico (es decir, un retraso largo).
3. **Start_Time**: Muestra la hora de inicio del accidente en la zona horaria local.
4. **End_Time**: Muestra la hora de finalización del accidente en la zona horaria local. La hora de finalización se refiere a cuando se disipó el impacto del accidente en el flujo de tráfico.
5. **Start_Lat**: Muestra la latitud en coordenadas GPS del punto de inicio.
6. **Start_Lng**: Muestra la longitud en coordenadas GPS del punto de inicio.
7. **End_Lat**: Muestra la latitud en coordenadas GPS del punto final.
8. **End_Lng**: Muestra la longitud en coordenadas GPS del punto final.
9. **Distance(mi)**: La longitud del tramo de carretera afectado por el accidente.
10. **Description**: Muestra una descripción en lenguaje natural del accidente.
11. **Number**: Muestra el número de calle en el campo de dirección.
12. **Street**: Muestra el nombre de la calle en el campo de dirección.
13. **Side**: Muestra el lado relativo de la calle (Derecho/Izquierdo) en el campo de dirección.
14. **City**: Muestra la ciudad en el campo de dirección.
15. **County**: Muestra el condado en el campo de dirección.
16. **State**: Muestra el estado en el campo de dirección.
17. **Zipcode**: Muestra el código postal en el campo de dirección.
18. **Country**: Muestra el país en el campo de dirección.
19. **Timezone**: Muestra la zona horaria basada en la ubicación del accidente (este, central, etc.).
20. **Airport_Code**: Denota una estación meteorológica basada en el aeropuerto que es la más cercana a la ubicación del accidente.
21. **Weather_Timestamp**: Muestra la marca de tiempo del registro de observación meteorológica (en hora local).
22. **Temperature(F)**: Muestra la temperatura (en Fahrenheit).
23. **Wind_Chill(F)**: Muestra la sensación térmica (en Fahrenheit).
24. **Humidity(%)**: Muestra la humedad (en porcentaje).
25. **Pressure(in)**: Muestra la presión del aire (en pulgadas).

26. **Visibility(mi)**: Muestra la visibilidad (en millas).
27. **Wind_Direction**: Muestra la dirección del viento.
28. **Wind_Speed(mph)**: Muestra la velocidad del viento (en millas por hora).
29. **Precipitation(in)**: Muestra la cantidad de precipitación en pulgadas, si la hay.
30. **Weather_Condition**: Muestra la condición meteorológica (lluvia, nieve, tormenta, niebla, etc.).
31. **Amenity**: Una anotación POI que indica la presencia de una amenidad en una ubicación cercana.
32. **Bump**: Una anotación POI que indica la presencia de un tope o reductor de velocidad en una ubicación cercana.
33. **Crossing**: Una anotación POI que indica la presencia de un cruce en una ubicación cercana.
34. **Give_Way**: Una anotación POI que indica la presencia de una señal de ceda el paso en una ubicación cercana.
35. **Junction**: Una anotación POI que indica la presencia de una intersección en una ubicación cercana.
36. **No_Exit**: Una anotación POI que indica la presencia de una señal de no salida en una ubicación cercana.
37. **Railway**: Una anotación POI que indica la presencia de una vía de tren en una ubicación cercana.
38. **Roundabout**: Una anotación POI que indica la presencia de una rotonda en una ubicación cercana.
39. **Station**: Una anotación POI que indica la presencia de una estación en una ubicación cercana.
40. **Stop**: Una anotación POI que indica la presencia de una señal de alto en una ubicación cercana.
41. **Traffic_Calming**: Una anotación POI que indica la presencia de una medida de calmado de tráfico en una ubicación cercana.
42. **Traffic_Signal**: Una anotación POI que indica la presencia de un semáforo en una ubicación cercana.
43. **Turning_Loop**: Una anotación POI que indica la presencia de un bucle de giro en una ubicación cercana.
44. **Sunrise_Sunset**: Muestra el período del día (es decir, día o noche) basado en el amanecer/atardecer.
45. **Civil_Twilight**: Muestra el período del día (es decir, día o noche) basado en el crepúsculo civil.
46. **Nautical_Twilight**: Muestra el período del día (es decir, día o noche) basado en el crepúsculo náutico.
47. **Astronomical_Twilight**: Muestra el período del día (es decir, día o noche) basado en el crepúsculo astronómico.

3.2.4. Tratamiento inicial: identificación de factores relevantes

Del archivo CSV, "US_Accidents_March23.csv", que contiene los registros de los accidentes y se almacena en un dataframe de la librería pandas, se toman únicamente los registros del estado de California; se decide fijar un estado como lugar de interés, porque esto va a permitir una mayor estabilidad en la variables exógenas y, en definitiva, predicciones más realistas y de mayor calidad; además se seleccionan las columnas que van a ofrecer información temporal y que pueden influir en la variable objetivo en una predicción de series temporales y se eliminan aquellas filas que tienen datos nulos o NaN (que representan escasamente un 6% del total del conjunto de datos).

```
df = pd.read_csv('US_Accidents_March23.csv')

df = df[df['State'] == "CA"]

cols_relevant = [
    'Severity', 'Start_Time', 'End_Time', 'Distance(mi)',
    'Temperature(F)', 'Humidity(%)', 'Pressure(in)',
    'Visibility(mi)', 'Wind_Speed(mph)', 'Wind_Direction',
    'Precipitation(in)', 'Sunrise_Sunset'
]

df = df[cols_relevant]
df = df.dropna()
```

Figura 5 Código utilizado para la lectura del CSV y la selección de columnas. Fuente: Propia

A continuación, se realizan una serie de transformaciones sobre los datos para conseguir los formatos deseados, como, por ejemplo, las unidades en el sistema métrico, en vez del sistema imperial:

1. Distancia de millas (mi) a metros (m).
2. Temperatura de Fahrenheit (°F) a Celsius (°C).
3. Presión de pulgadas de mercurio (inHg) a hectopascales (hPa).
4. Visibilidad de millas (mi) a metros (m).
5. Velocidad del viento de millas por hora (mph) a kilómetros por hora (km/h).
6. Precipitación de pulgadas (in) a milímetros (mm).
7. Marcas temporales a objetos datetime, eliminando los milisegundos.
8. Cálculo de la duración del accidente en minutos.

```
df['Distance(m)'] = (df['Distance(mi)'] * 1.60934 * 1000).round(2)
df['Temperature(C)'] = ((df['Temperature(F)'] - 32) * 5.0/9.0).round(2)
df['Pressure(hPa)'] = (df['Pressure(in)'] * 33.8639).round(2)
df['Visibility(m)'] = (df['Visibility(mi)'] * 1.60934 * 1000).round(2)
df['Wind_Speed(km/h)'] = (df['Wind_Speed(mph)'] * 1.60934).round(2)
df['Precipitation(mm)'] = (df['Precipitation(in)'] * 25.4).round(2)
df['Start_Time'] = pd.to_datetime(df['Start_Time'].str.split('.').str[0], format='%Y-%m-%d %H:%M:%S')
df['End_Time'] = pd.to_datetime(df['End_Time'].str.split('.').str[0], format='%Y-%m-%d %H:%M:%S')
df['Duration(min)'] = ((df['End_Time'] - df['Start_Time']).dt.total_seconds() / 60).astype(np.int64)
```

Figura 6 Código utilizado para la transformación de las columnas necesarias del dataframe. Fuente: Propia

Finalmente, se seleccionan las columnas finales y se extrae el dataframe sobre el que se va a trabajar en el desarrollo del proyecto, el cual se compone de la siguientes columnas:

1. **Severity**: Muestra la gravedad del accidente, un número entre 1 y 4, donde 1 indica el menor impacto en el tráfico (es decir, un retraso corto como resultado del accidente) y 4 indica un impacto significativo en el tráfico (es decir, un retraso largo).
2. **Start_Time**: Muestra la hora de inicio del accidente en la zona horaria local.
3. **Duration(min)**: Muestra la duración del accidente en minutos.
4. **Distance(m)**: La longitud del tramo de carretera afectado por el accidente en metros.
5. **Temperature(C)**: Muestra la temperatura en grados Celsius.
6. **Humidity(%)**: Muestra la humedad en porcentaje.
7. **Pressure(hPa)**: Muestra la presión del aire en hectopascales.
8. **Visibility(m)**: Muestra la visibilidad en metros.
9. **Wind_Direction**: Muestra la dirección del viento.
10. **Wind_Speed(km/h)**: Muestra la velocidad del viento en kilómetros por hora.
11. **Precipitation(mm)**: Muestra la cantidad de precipitación en milímetros.
12. **Weather_Condition**: Muestra la condición meteorológica (lluvia, nieve, tormenta, niebla, etc.).
13. **Sunrise_Sunset**: Muestra el período del día (es decir, día o noche) basado en el amanecer/atardecer.

```
cols_final = [
    'Severity', 'Start_Time', 'Duration(min)', 'Distance(m)',
    'Temperature(C)', 'Humidity(%)', 'Pressure(hPa)',
    'Visibility(m)', 'Wind_Speed(km/h)', 'Wind_Direction',
    'Precipitation(mm)', 'Sunrise_Sunset'
]

df = df[cols_final]
```

Figura 7 Código utilizado para la selección de las columnas del dataframe final. Fuente: Propia

La información completa del dataframe es la siguiente:

```
Index: 1148104 entries, 958 to 7728393
Data columns (total 12 columns):
 #   Column             Non-Null Count    Dtype
---  ------             --------------    -----
 0   Severity           1148104 non-null  int64
 1   Start_Time         1148104 non-null  datetime64[ns]
 2   Duration(min)      1148104 non-null  int64
 3   Distance(m)        1148104 non-null  float64
 4   Temperature(C)     1148104 non-null  float64
 5   Humidity(%)        1148104 non-null  float64
 6   Pressure(hPa)      1148104 non-null  float64
 7   Visibility(m)      1148104 non-null  float64
 8   Wind_Speed(km/h)   1148104 non-null  float64
 9   Wind_Direction     1148104 non-null  object
 10  Precipitation(mm)  1148104 non-null  float64
 11  Sunrise_Sunset     1148104 non-null  object
dtypes: datetime64[ns](1), float64(7), int64(2), object(2)
```

Figura 8 Información completa del dataframe final. Fuente: Propia

Y algunos ejemplos de registros de accidentes serían los siguientes:

	Severity	Start_Time	Duration(min)	Distance(m)	Temperature(C)	Humidity(%)	Pressure(hPa)	Visibility(m)	Wind_Speed(km/h)	Wind_Direction	Precipitation(mm)	Sunrise_Sunset
4177589	2	2022-02-01 19:49:00	123	405.55	12.78	74.00	1004.06	16093.40	0.00	CALM	0.00	Night
5871659	2	2021-09-15 11:47:00	128	1894.19	21.11	63.00	1011.51	11265.38	14.48	W	0.00	Day
5734722	2	2021-09-18 11:12:30	205	5561.88	20.56	63.00	989.50	16093.40	9.66	VAR	0.00	Day
4314950	2	2022-08-18 16:35:27	174	9242.44	31.67	30.00	990.18	16093.40	20.92	W	0.00	Day
1318984	2	2020-10-06 14:05:37	59	0.00	28.89	28.00	1011.51	16093.40	0.00	CALM	0.00	Day

Figura 9 Ejemplos de filas del dataframe final. Fuente: Propia

3.3. Análisis descriptivo y exploratorio de los datos

Una vez extraídos los datos iniciales, se va a realizar un análisis descriptivo y exploratorio de los mismos para observar y entender mejor el contexto sobre el que se va a realizar este trabajo.

	Severity	Start_Time	Duration(min)	Distance(m)	Temperature(C)	Humidity(%)	Pressure(hPa)	Visibility(m)	Wind_Speed(km/h)	Precipitation(mm)
count	1148104.00	1148104	1148104.00	1148104.00	1148104.00	1148104.00	1148104.00	1148104.00	1148104.00	1148104.00
mean	2.09	2021-05-04 11:21:37.403329792	126.22	977.53	17.30	58.80	999.51	14497.04	9.85	0.10
min	1.00	2016-03-23 05:03:47	3.00	0.00	-22.22	1.00	565.87	0.00	0.00	0.00
25%	2.00	2020-05-14 03:51:30	46.00	0.00	12.22	39.00	995.94	16093.40	0.00	0.00
50%	2.00	2021-06-29 06:10:30	81.00	136.79	16.67	62.00	1008.81	16093.40	9.66	0.00
75%	2.00	2022-05-02 10:31:00	133.00	952.73	22.22	79.00	1013.89	16093.40	14.48	0.00
max	4.00	2023-03-31 23:25:30	525599.00	122116.71	52.78	100.00	1977.31	160934.00	1749.35	264.16
std	0.33	NaN	1499.21	2424.34	7.70	24.56	29.25	4340.85	8.80	0.73

Figura 10 Resumen estadístico de los datos. Fuente: Propia

En primer lugar, se realiza una descripción de las variables numéricas, extrayendo un resuman estadístico que va a permitir comprender sus características básicas:

A continuación, se va a generar un gráfico de barras para visualizar la distribución de la gravedad de los accidentes en un conjunto de datos.

```
plt.figure(figsize=(10, 6))
sns.countplot(x='Severity', data=df, hue='Severity', palette='viridis', legend=False)
plt.title('Distribución de la Severidad de los Accidentes')
plt.xlabel('Severidad')
plt.ylabel('Frecuencia')
plt.show()
```

Figura 11 Código utilizado para la creación del diagrama de barras de distribución de gravedad. Fuente: Propia

Como se puede observar en la gráfica, la gran mayoría de accidentes ocurridos tienen una gravedad de grado 2, es decir, conllevan un impacto medio-bajo en el tráfico, también se encuentran una gran cantidad, aunque en mucha menor medida, de accidentes de grado 3, que conllevan un retraso medio-alto en el tráfico. Por último, los accidentes de grado 1 y 4, con menor y mayor impacto respectivamente, son muy poco comunes en el estado de California.

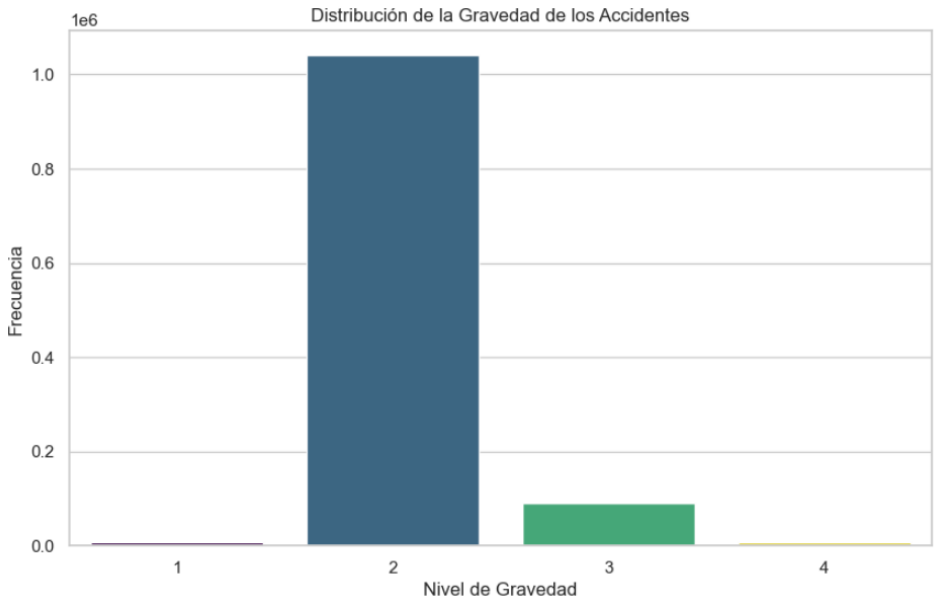

Figura 12 Diagrama de barras de distribución de gravedad de los accidentes. Fuente: Propia

El siguiente gráfico es muy similar al anterior, pero en él vamos a observar la distribución de los accidentes que tienen lugar por el día y por la noche.

```
plt.figure(figsize=(10, 6))
sns.countplot(x='Sunrise_Sunset', data=df, hue='Sunrise_Sunset', palette='viridis', dodge=False, legend=False)
plt.title('Distribución de los Accidentes por Período del Día')
plt.xlabel('Período del Día')
plt.ylabel('Frecuencia')
plt.show()
```

Figura 13 Código utilizado para la creación del diagrama de barras de distribución de periodo del día. Fuente: Propia

Como se muestra, ocurren el doble de accidentes por el día que por la noche.

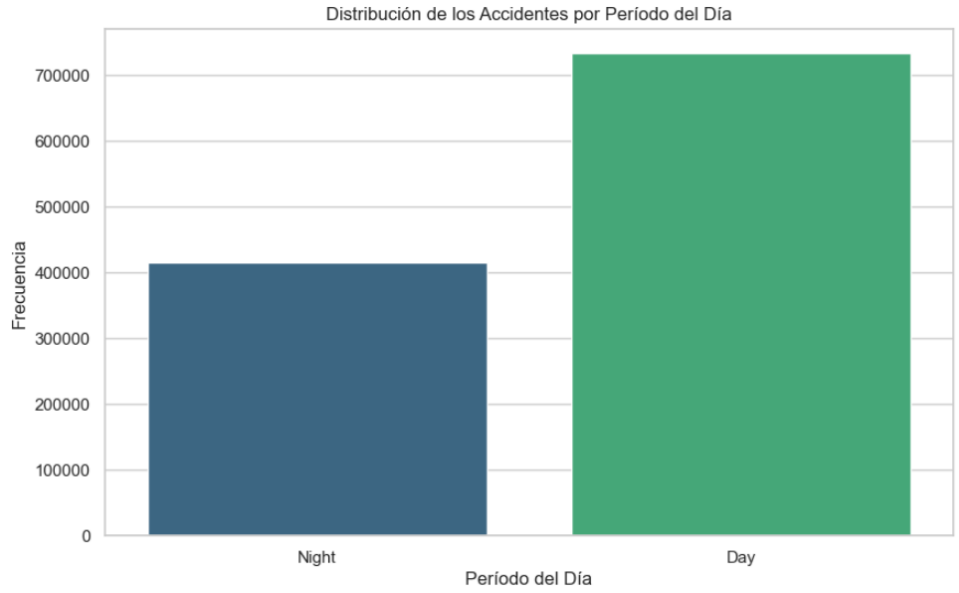

Figura 14 Diagrama de barras de distribución de los accidentes por periodo del día. Fuente: Propia

En siguiente lugar, se generan una serie de gráficos de caja para analizar la relación entre la gravedad de los accidentes y diversas variables numéricas como la distancia, temperatura, humedad, presión, visibilidad, velocidad del viento y precipitación.

```
numeric_vars = ['Distance(m)', 'Temperature(C)', 'Humidity(%)', 'Pressure(hPa)', 'Visibility(m)', 'Wind_Speed(km/h)', 'Precipitation(mm)']

plt.figure(figsize=(16, 12))
for i, var in enumerate(numeric_vars, 1):
    plt.subplot(4, 2, i)
    sns.boxplot(x='Severity', y=var, data=df, hue='Severity', palette='viridis', dodge=False, legend=False)
    plt.title(f'{var} por Severidad')
    plt.xlabel('Severidad')
    plt.ylabel(var)
plt.tight_layout()
plt.show()
```

Figura 15 Código utilizado para la creación de los gráficos de caja de las variables numéricas para cada nivel de gravedad del accidente. Fuente: Propia

En los siete boxplots que se han generado, excepto en la de humedad y temperatura, se observa un distribución bastante irregular y dispersa de los datos, con una gran cantidad de valores atípicos y alejada de la normal. En el caso de la humedad y la temperatura, sobre todo en la humedad, se aprecia un acercamiento a la normal, con menor cantidad de valores atípicos.

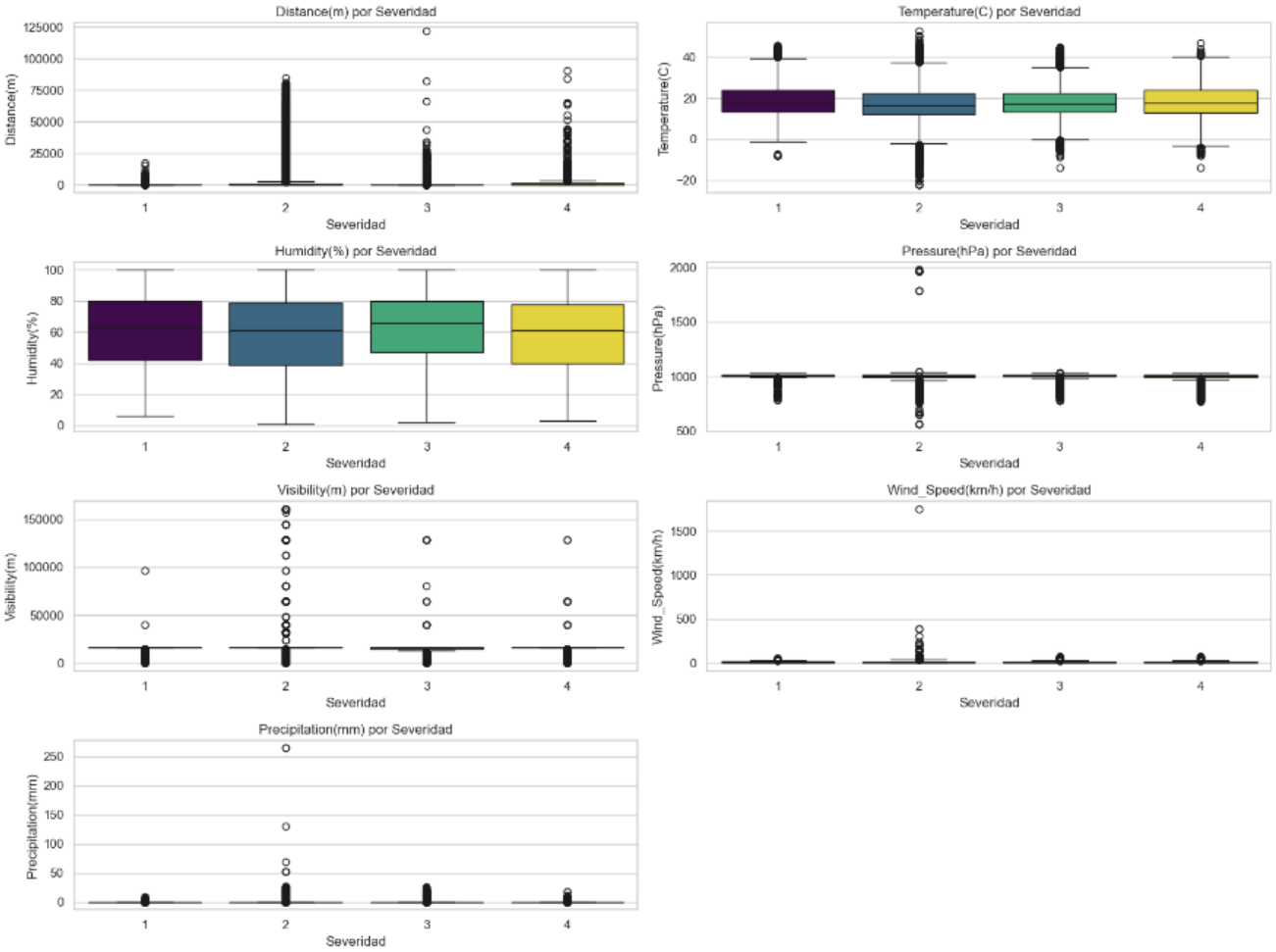

Figura 16 Gráficos de caja de las variables numéricas para cada nivel de gravedad del accidente. Fuente: Propia

La última gráfica es la más importante, ya que es la que más va a influir en la selección de variables para la predicción de las series temporales. En esta se va a generar una matriz de correlación entre las diferentes variables numéricas y meteorológicas, es decir, las potenciales variables exógenas que podríamos incorporar a los modelos, esto va a permitir apreciar claramente si existe alguna relación lineal fuerte entre dos variables.

El hecho de que exista una correlación fuerte entre dos variables puede crear una multicolinealidad en las series temporales, lo que supone un problema ya que afectaría negativamente a la precisión y estabilidad de los modelos. Por eso, es necesario eliminar algunas de estas variables para asegurar una estimación estable y precisa de los coeficientes, ya que en caso contrario se dificultaría la interpretación de los resultados y la confianza de las predicciones.

```python
exog_df = df[['Temperature(C)', 'Humidity(%)', 'Pressure(hPa)', 'Visibility(m)', 'Wind_Speed(km/h)', 'Precipitation(mm)']]

plt.figure(figsize=(12, 8))
correlation_matrix = exog_df.corr()
heatmap = sns.heatmap(correlation_matrix, annot=True, cmap='viridis', linewidths=0.5)
heatmap.set_xticklabels(heatmap.get_xticklabels(), rotation=30, ha='right')
plt.title('Matriz de Correlación entre Variables')
plt.show()
```

Figura 17 Código utilizado para la creación de la matriz de correlación. Fuente: Propia

Al generar la matriz de correlación, se observa con claridad como existe cierte relación lineal entre diferentes variables, sin embargo, llama la atención el caso de la humedad, ya que tiene una alta relación negativa con la temperatura y la visibilidad y una alta relación positiva con las precipitaciones. Por esta razón, a la hora de trabajar con los modelos de predicción de series temporales, será necesario eliminar la humedad de las variables exógenas para evitar que genere multicolinealidad y afecte negativamente.

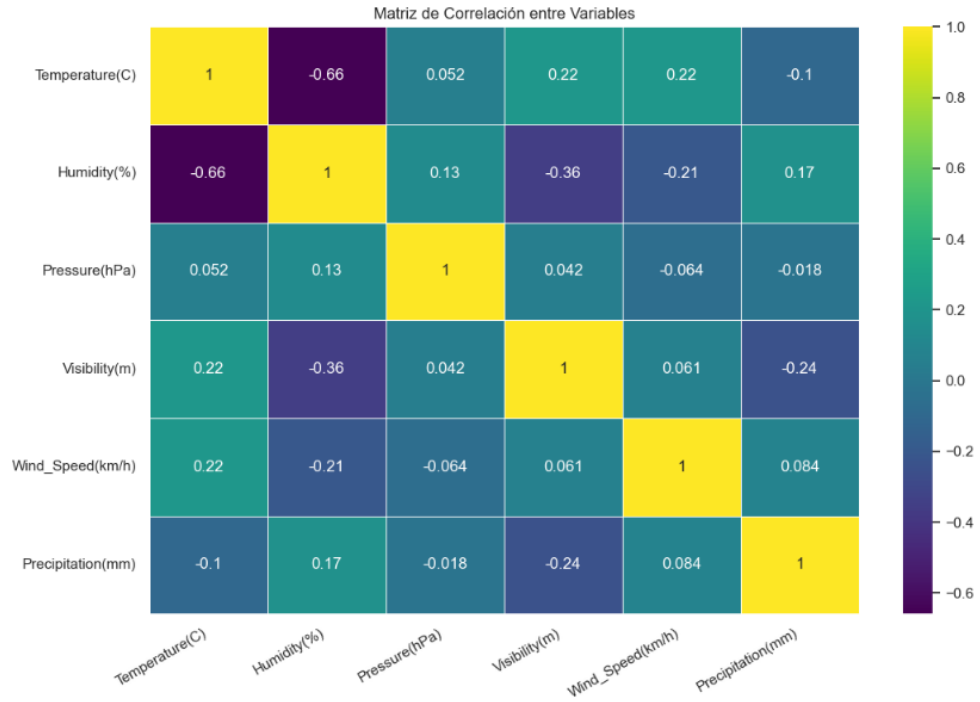

Figura 18 Matriz de correlación entre las variables meteorológicas. Fuente: Propia

3.4. Análisis de la serie temporal

Antes de realizar el análisis propio de las series temporales, es necesario realizar un segundo tratamiento de los datos para seleccionar correctamente las variables y asegurar una frecuencia estable de las ocurrencias que conforman el dataframe.

En primer lugar, se seleccionan las columnas relevantes para realizar las predicciones y las variables exógenas que no vayan a producir un problema de colinealidad, es decir, todas las variables meteorológicas numéricas, a excepción de la humedad.

```
cols_ts = [
    'Severity', 'Start_Time', 'Temperature(C)', 'Humidity(%)',
    'Pressure(hPa)', 'Visibility(m)', 'Wind_Speed(km/h)',
    'Precipitation(mm)'
]

df = df[cols_ts]
```

Figura 19 Código utilizado para crear el dataframe con las variables relevantes para la serie temporal. Fuente: Propia

A continuación, como los datos originales están formado por accidentes ocurridos en momentos exactos en el tiempo, se encuentra un problema en la distribución de los datos a lo largo de la serie temporal, ya que esta no es uniforme. Por eso, para asegurar una distribución uniforme de los datos se van a agrupar los accidentes por día, añadiendo una variable nueva "Accident_Count" con el recuento de los accidentes ocurridos cada día y calculando la media de las variables meteorológicas, esto es posible al contar con sucesos cercanos geográficamente, todos dentro del estado de California.

```
df['Accident_Count'] = 1

df = df.resample('D', on='Start_Time').agg({
    'Severity': 'mean',
    'Temperature(C)': 'mean',
    'Humidity(%)': 'mean',
    'Pressure(hPa)': 'mean',
    'Visibility(m)': 'mean',
    'Wind_Speed(km/h)': 'mean',
    'Precipitation(mm)': 'mean',
    'Accident_Count': 'sum'
})
```

Figura 20 Código utilizado para crear el dataframe de accidentes con frecuencia diaria y recuento de número de sucesos. Fuente: Propia

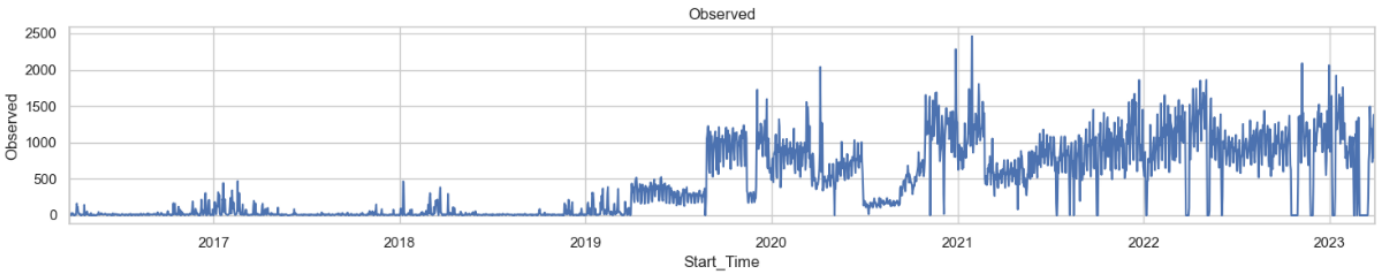

Figura 21 Gráfica con el recuento de accidentes diarios desde 2016 hasta 2023. Fuente: Propia

Finalmente, también se aprecia como el número de accidentes registrados antes de 2020 es significativamente menor que en fechas más actuales, problema derivado seguramente del propio registro de sucesos, por esta razón únicamente se van a tomar aquellos registros con fecha posterior al 1 de enero de 2020 para evitar que las predicciones se vean afectadas negativamente por ello.

Además, se encuentran algunos días esporádicos en los que no se registraron los accidentes por lo que, para no romper la frecuencia de sucesos, se han generado los

datos de esos días haciendo uso de interpolación lineal, rellenando al completo el dataframe.

```
df = df[df.index >= '2020-01-01']
df['Accident_Count'] = df['Accident_Count'].replace(0, np.nan)
df['Accident_Count'] = df['Accident_Count'].interpolate(method='linear').round().astype(int)
df = df.interpolate(method='linear')
```

Figura 22 Código utilizado para tomar los sucesos a partir de 2020 y rellenar los días vacíos con interpolación lineal. Fuente: Propia

Algunos ejemplos de los datos finales son los siguientes:

Start_Time	Severity	Temperature(C)	Humidity(%)	Pressure(hPa)	Visibility(m)	Wind_Speed(km/h)	Precipitation(mm)	Accident_Count
2022-08-13	2.06	26.33	46.07	999.60	15843.92	9.84	0.00	916
2020-09-07	2.27	29.28	44.07	992.95	12672.06	6.91	0.00	135
2020-08-30	2.47	21.38	61.93	999.07	13130.75	9.29	0.00	132
2021-01-06	2.07	10.82	72.02	1002.96	12320.98	5.26	0.01	861
2022-07-02	2.06	21.77	53.77	989.50	15679.95	14.41	0.00	760

Figura 23 Ejemplos de los datos de accidentes diarios finales. Fuente: Propia

Con los datos listos para ser utilizados de forma temporal, se va a realizar el análisis propio de las series temporales de las dos variables objetivo que se desean predecir en este trabajo: el número de accidentes diarios y la gravedad media diaria de estos.

3.4.1. Serie temporal del recuento diario de accidentes

Para realizar el estudio de la serie temporal del número de accidentes diarios registrados se ha utilizado una descomposición estacional con el modelo aditivo para extraer las diferentes componentes a partir de los datos.

En primer lugar, se grafican las observaciones de la variable.

```
decomposition = seasonal_decompose(df['Accident_Count'], model='additive')

fig, (ax1, ax2, ax3, ax4) = plt.subplots(4, 1, figsize=(15, 12))
decomposition.observed.plot(ax=ax1)
ax1.set_ylabel('Observaciones')
ax1.set_title('Observaciones')
```

Figura 24 Código utilizado para extraer las componentes de la serie temporal y graficar las observaciones. Fuente: Propia

Como se puede observar en el siguiente gráfico existen picos pronunciados en el número de accidentes, aunque, al haber eliminado los registros anteriores a 2020, se mantiene relativamente estable. Cabe destacar que se observan días con registros muy bajos de accidentes, problema que se consideran derivado de nuevo de la propia recogida de los datos, sin embargo, esto no debería influir significativamente en la posterior predicción al ser ocurrencias arbitrarias.

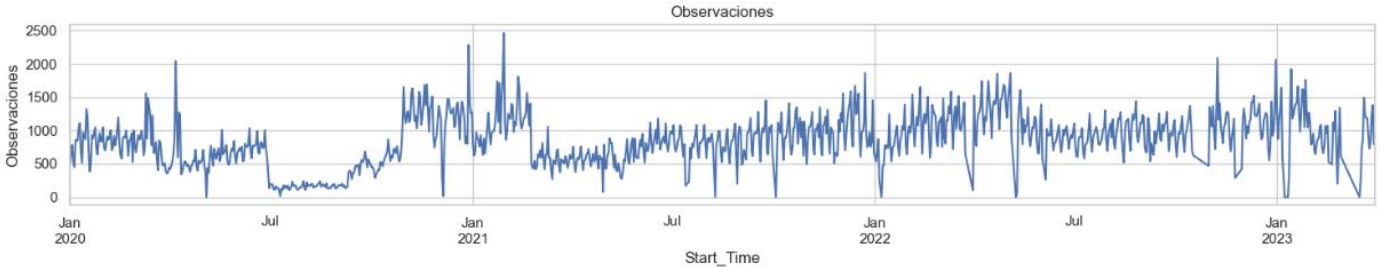

Figura 25 Gráfico de observaciones del recuento de accidentes diarios. Fuente: Propia

En segundo lugar, se grafica la tendencia.

```
decomposition.trend.plot(ax=ax2)
ax2.set_ylabel('Tendencia')
ax2.set_title('Tendencia')
```

Figura 26 Código utilizado para graficar la tendencia. Fuente: Propia

En el gráfico de tendencia se observa una versión suavizada del anterior gráfico, con sus crecimientos y decrecimientos, pero sin dibujar una clara tendencia general creciente o decreciente, hecho que tiene sentido por la propia naturaleza del caso de estudio y la franja de tiempo relativamente reducida en la que se registran los datos. Se puede intuir que la seria presenta estacionariedad en tendencia.

Figura 27 Gráfico de tendencia en el recuento diario de accidentes. Fuente: Propia

En tercer lugar, se grafica la componente estacional.

```
decomposition.seasonal.plot(ax=ax3)
ax3.set_ylabel('Estacionalidad')
ax3.set_title('Estacionalidad')
```

Figura 28 Código utilizado para graficar la estacionalidad. Fuente: Propia

De este gráfico cabe destacar la repetición de ciclos con pronunciados crecimientos y decrecimientos, derivados de los picos del gráfico de observaciones.

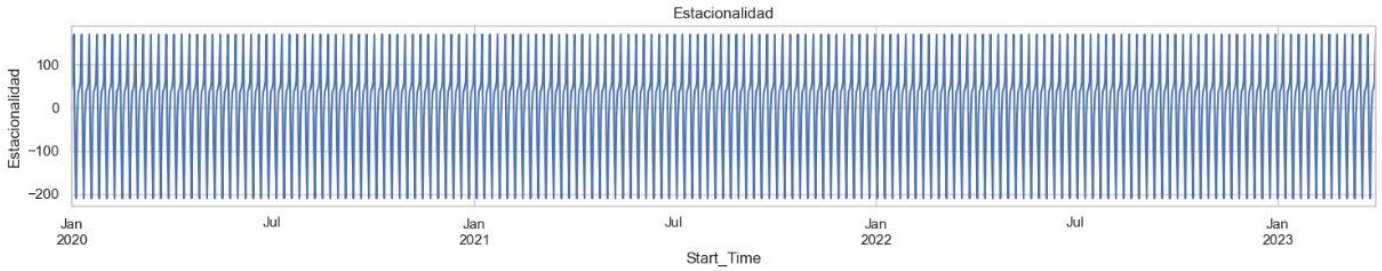

Figura 29 Gráfico de estacionalidad en el recuento diario de accidentes. Fuente: Propia

La última componente que se grafica es la componente residual o ruido.

```
decomposition.resid.plot(ax=ax4)
ax4.set_ylabel('Ruido')
ax4.set_title('Ruido')
```

Figura 30 Código utilizado para graficar el ruido. Fuente: Propia

En esta gráfica se observa, a través de sus fuertes picos, cómo han sido extraídos los casos en los que se han registrado una cantidad muy alta o baja de accidentes.

Sin embargo, el hecho de que estos casos extremos formen parte de este gráfico asegura una mayor fiabilidad de los modelos de predicción al haberse detectado, no solo los sucesos aleatorios, sino también los problemas derivados de la recogida de datos como elementos residuales.

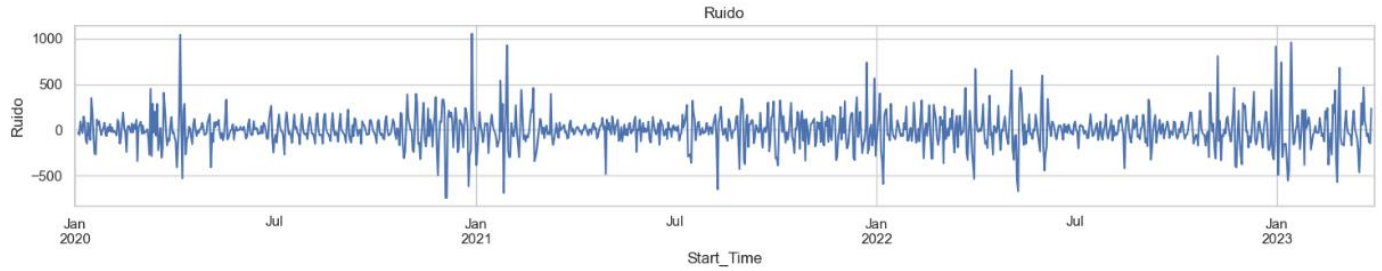

Figura 31 Gráfico del ruido en el recuento diario de accidentes. Fuente: Propia

A continuación, se van a calcular y graficar las funciones de autocorrelación simple (ACF) y autocorrelación parcial (PACF) a lo largo de 50 retardos.

```
fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(15, 8))
plot_acf(df['Accident_Count'], lags=50, ax=ax1)
plot_pacf(df['Accident_Count'], lags=50, ax=ax2)

ax1.set_title('Autocorrelación')
ax2.set_title('Autocorrelación Parcial')
```

Figura 32 Código utilizado para graficar el ACF y el PACF. Fuente: Propia

En el gráfico de la función de autocorrelación simple se observa un decaimiento gradual de los valores a medida que aumenta el número de retardos o lags, lo que determina que esta serie temporal cuenta con una dependencia temporal prolongada, es decir, los valores de la serie temporal están correlacionados con muchos de sus valores anteriores y por tanto no solo dependen de sus valores inmediatos anteriores, sino también de valores más lejanos en el pasado.

Este disminución gradual de los valores también indica la no estacionariedad de la serie temporal y por lo tanto sus propiedades estadísticas cambian con el tiempo.

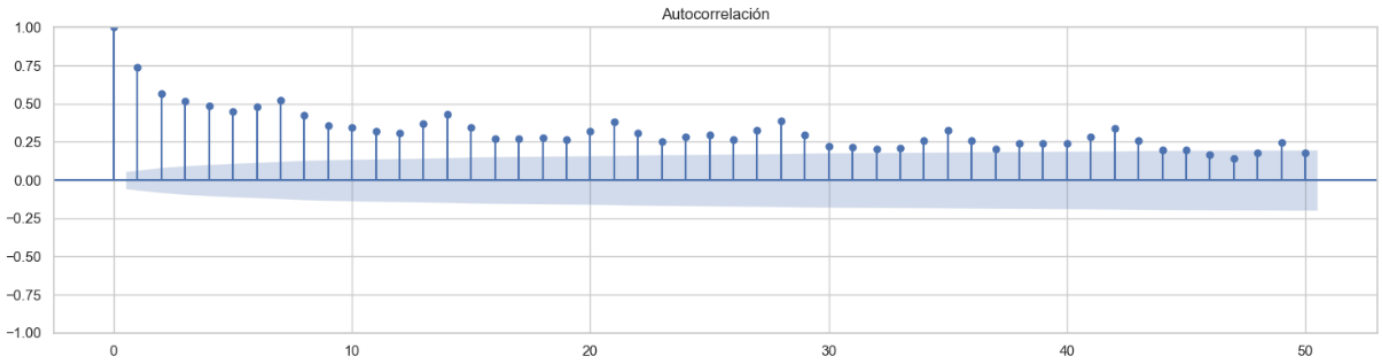

Figura 33 Gráfico de la función de autocorrelación simple del recuento diario de accidentes. Fuente: Propia

Por otra parte, en el gráfico de la función de autocorrelación parcial se pueden observar picos significativos en los primeros dos retardos (lag 1 y lag 2), lo que indica que hay una fuerte correlación parcial entre el valor actual de la serie temporal y los valores en los retardos 1 y 2.

Después de estos dos primeros retardos, los valores de la función disminuyen drásticamente y se mantienen muchos más cercanos a cero, indicando que no hay correlaciones parciales significativas en retardos más lejanos, esto significa que, una vez que se han considerado los valores en lag 1 y lag 2, los valores en retardos más altos no aportan información adicional significativa sobre el valor actual de la serie temporal.

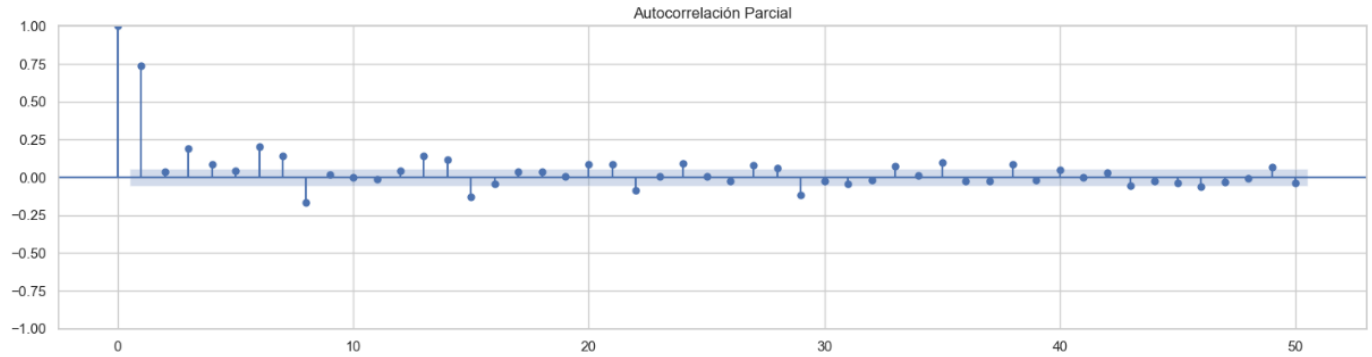

Figura 34 Gráfico de la función de autocorrelación parcial del recuento diario de accidentes. Fuente: Propia

3.4.2. Serie temporal de la gravedad media de los accidentes diarios

En el estudio de la serie temporal de la gravedad media de los accidentes diarios se va a utilizar el mismo código que en el análisis anterior, sustituyendo únicamente la llamada a la columna del dataframe de “Accident_Count” a “Severity”, por lo que únicamente se van a mostrar las gráficas generadas.

El primer gráfico es el de las observaciones diarias de la media de la gravedad de los accidentes ocurridos, como se había observado en el primer diagrama de barras del análisis de los datos, la mayoría de los accidentes tiene una gravedad de nivel 2 y, en menor medida, nivel 3, siendo el nivel 1 y el 4 muy infrecuentes.

Por esa razón, en este gráfico es compresible que los valores de las medias se mantengan de manera muy estable entorno al valor 2, con algo más de variación al comienzo de la función. También se encuentran algunos picos, sobre todo uno muy pronunciado al final del gráfico que se debe a la existencia de un único registro ese día con una gravedad de nivel 4.

Figura 35 Gráfico de observaciones de la gravedad media de los accidentes diarios. Fuente: Propia

El segundo gráfico es el de tendencia, que muestra un comportamiento similar al del anterior apartado con una versión más suavizada del gráfico de observaciones, bastante estable y mostrando una ligera tendencia descendente, obviando el pico de nivel 4.

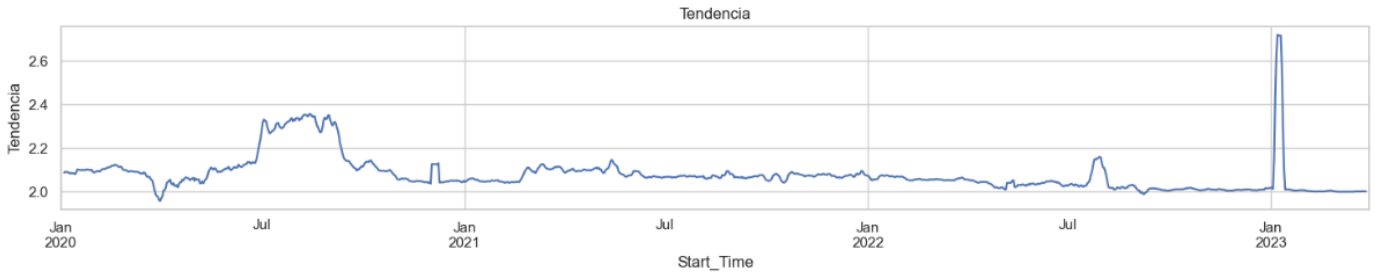

Figura 36 Gráfico de tendencia en la gravedad media de los accidentes diarios. Fuente: Propia

El tercer gráfico es el de estacionalidad, que muestra también ciclos cortos y pronunciados con fuertes subidas y bajadas.

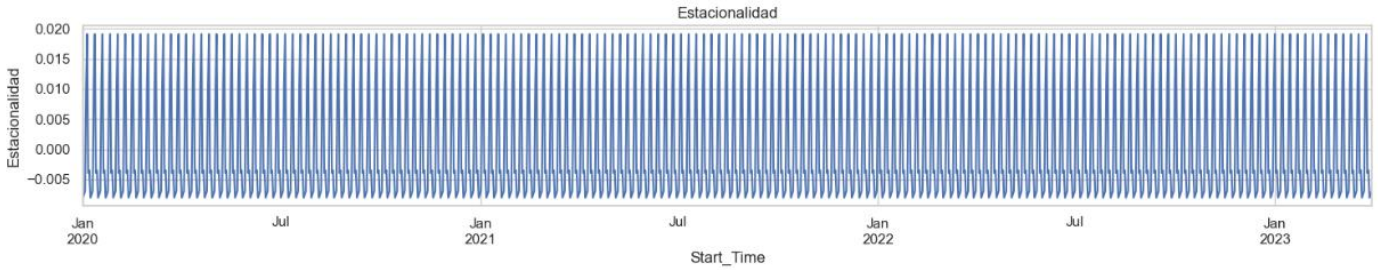

Figura 37 Gráfico de estacionalidad en la gravedad media de los accidentes diarios. Fuente: Propia

El gráfico de la última componente es el del ruido, el cual es menor que en el del número de accidentes al contar con un gráfico más estable y menos aleatorio, pero que se encarga de extraer los picos derivados de fallos en la recopilación de datos de los que se ha hablado anteriormente y que, por lo tanto, no van a influir de manera significativa en las predicciones.

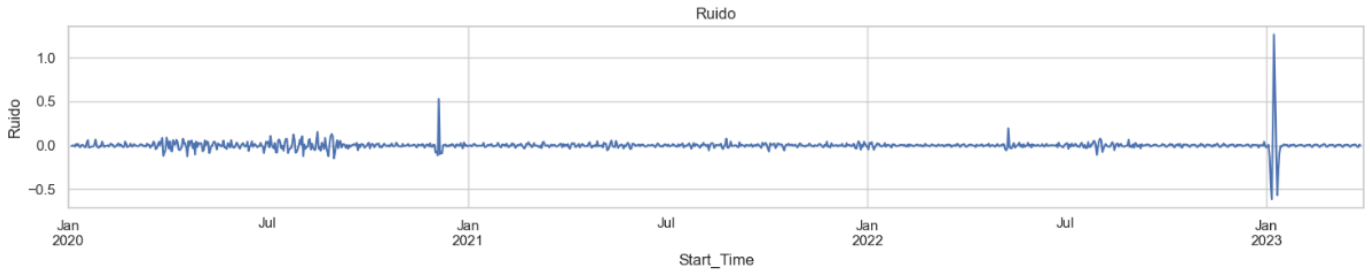

Figura 38 Gráfico del ruido en la gravedad media de los accidentes diarios. Fuente: Propia

Los últimos gráficos generados son los de la autocorrelación simple y parcial, los cuales tienen un comportamiento muy similar a los procedentes de la serie temporal de recuento de accidentes diarios.

En el gráfico de la función de correlación simple se observa un decaimiento gradual de los valores a medida que aumenta el número de retardos, lo que indica una dependencia temporal prolongada y la no estacionariedad de la serie temporal, ya que sus propiedades estadísticas cambian con el tiempo.

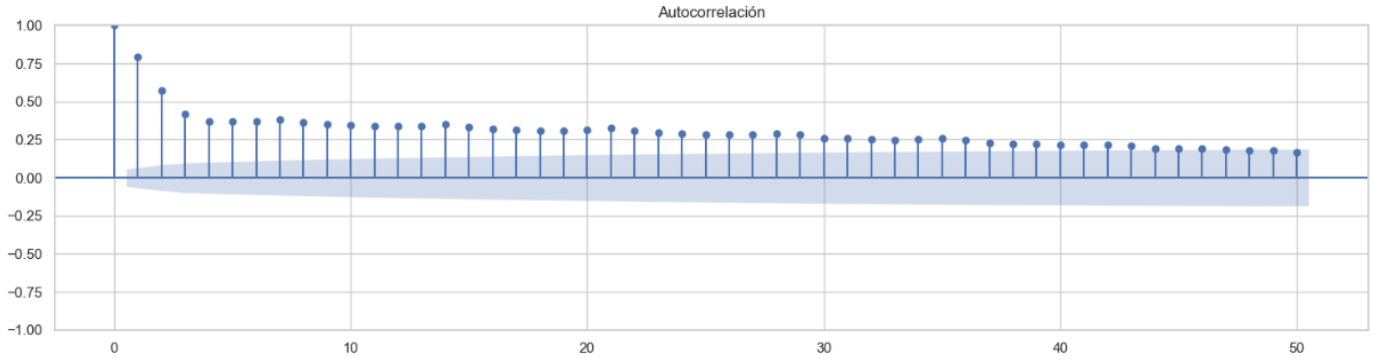

Figura 39 Gráfico de la función de autocorrelación simple de la gravedad media de los accidentes diarios. Fuente: Propia

En el gráfico de la función de autocorrelación parcial, hay picos significativos en los primeros dos retardos (lag 1 y lag 2), lo que muestra una fuerte correlación parcial con los valores en esos retardos. Después de estos, los valores disminuyen drásticamente, indicando que los retardos más altos no aportan información adicional significativa sobre el valor actual de la serie temporal.

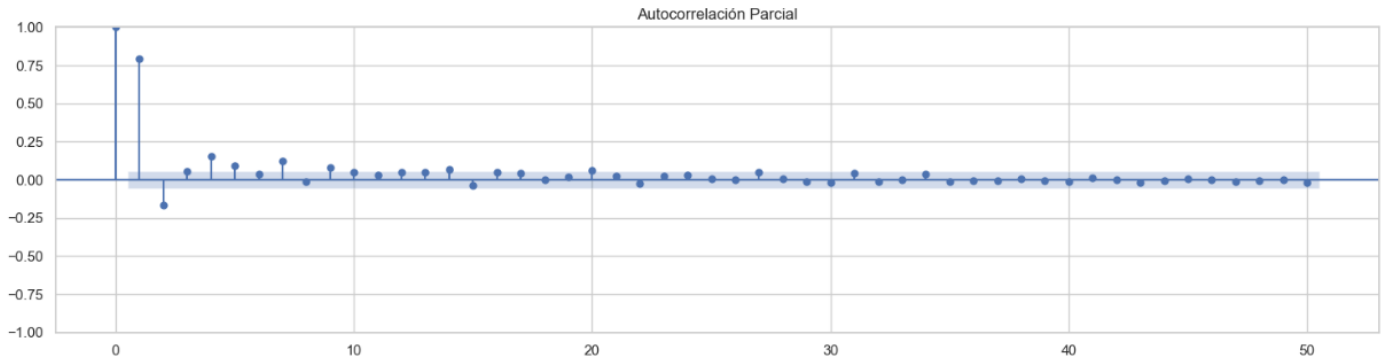

Figura 40 Gráfico de la función de autocorrelación parcial de la gravedad media de los accidentes diarios. Fuente: Propia

3.5. Predicción de las series temporales

Antes de comenzar a realizar las predicciones, hay que tener en cuenta que en todos los casos se ha hecho la misma división de los datos de 80% train y 20% test, es decir, para las pruebas de los modelos se va a realizar el entrenamiento con el primer 80% de los días y las pruebas con el 20% de los días más actuales.

Además, se ha generado una función para extraer las métricas de evaluación de cada uno de los modelos, en este caso el error absoluto medio (MAE), el error cuadrático medio (MSE), la raíz del error cuadrático medio (RMSE) y el coeficiente de determinación (R^2).

Esta función cual cumple dos funciones, en primer lugar, va a permitir probar con diferentes hiperparámetros en cada modelo para encontrar los mejores y, en segundo lugar, comparar todos los modelos finales para encontrar aquel que realice las predicciones más acertadas.

```
def evaluate_forecast(actual, forecast):
    mae = mean_absolute_error(actual, forecast)
    mse = mean_squared_error(actual, forecast)
    rmse = np.sqrt(mse)
    r2 = r2_score(actual, forecast)
    return mae, mse, rmse, r2
```

Figura 41 Código utilizado para calcular las métricas de evaluación de los modelos. Fuente: Propia

3.5.1 Predicción del recuento diario de accidentes

En primer lugar, se van a aplicar los diferentes modelos para predecir sobre la serie temporal de recuento diario de accidentes, para ello se ha generado un nuevo dataframe en el que se elimina la columna de gravedad, para que no intervenga como variable exógena en el entrenamiento y no afecte a las predicciones finales.

```
df_count = df.drop('Severity', axis=1)
```

Figura 42 Código utilizado para general el dataframe utilizado en la predicción del recuento diario de accidentes. Fuente: Propia

Enfoque Clásico – Medias Móviles

Para generar el primer modelo se va a utilizar el método de medias móviles, este, sin embargo, no acepta variables exógenas por lo que las predicciones se van a realizar utilizando únicamente los datos temporales de recuento de accidentes, sin tener en cuenta los datos meteorológicos.

En primer lugar, se divide el conjunto de datos en entrenamiento que contiene el 80% de los datos y un prueba con el 20% restante.

```
train_size = int(len(df_count) * 0.8)
train, test = df_count.iloc[:train_size], df_count.iloc[train_size:]
```

Figura 43 Bloque de código 1 del modelo de medias móviles. Fuente: Propia

Se inicializan variables para almacenar el mejor tamaño de ventana “best_window_size” y su puntuación correspondiente “best_score”. También se crea una lista vacía “metrics_results” para almacenar los resultados de las métricas para cada tamaño de ventana.

```
best_window_size = None
best_score = float('inf')
metrics_results = []
```

Figura 44 Bloque de código 2 del modelo de medias móviles. Fuente: Propia

A continuación, se crea un bucle que itera sobre diferentes tamaños de ventana (de 1 a 20) para calcular la media móvil en el conjunto de entrenamiento, generando así una predicción para el conjunto de prueba. Evalúa las predicciones usando las métricas MAE, MSE, RMSE y R^2 y guarda los resultados. Además, actualiza el mejor tamaño de ventana y su puntuación basándose en el RMSE.

```
for window_size in range(1, 21):
    rolling_mean = train['Accident_Count'].rolling(window=window_size).mean()
    forecast = rolling_mean[-len(test):].values
    mae, mse, rmse, r2 = evaluate_forecast(test['Accident_Count'], forecast)
    metrics_results.append((window_size, mae, mse, rmse, r2))

    if rmse < best_score:
        best_score = rmse
        best_window_size = window_size
```

Figura 45 Bloque de código 3 del modelo de medias móviles. Fuente: Propia

Una vez encontrado el mejor tamaño de ventana, en este caso 11, se extraen las métricas asociadas a dicho tamaño y se almacenan en variables separadas para su análisis posterior.

```
metrics_ma = [metrics for metrics in metrics_results if metrics[0] == best_window_size][0]
ma_mae, ma_mse, ma_rmse, ma_r2 = metrics_ma[1:5]
```

Figura 46 Bloque de código 4 del modelo de medias móviles. Fuente: Propia

En siguiente lugar, se recalcula la media móvil utilizando el mejor tamaño de ventana en el conjunto de entrenamiento y se genera una predicción para el conjunto de prueba. Luego, se crea una copia del conjunto de prueba para añadir la columna de predicciones y se visualizan los datos de entrenamiento, prueba y predicción en un gráfico.

```
rolling_mean = train['Accident_Count'].rolling(window=best_window_size).mean()
test = test.copy()
test.loc[:, 'Forecast'] = rolling_mean[-len(test):].values

plt.figure(figsize=(15, 5))
plt.plot(train.index, train['Accident_Count'], label='Train', color="royalblue")
plt.plot(test.index, test['Accident_Count'], label='Test', color="gold")
plt.plot(test.index, test['Forecast'], label='Forecast', color="limegreen")
plt.legend(loc='upper right')
plt.title(f'Predicción con Medias Móviles (Tamaño de Ventana = {best_window_size})')
plt.margins(x=0)
plt.show()
```

Figura 47 Bloque de código 5 del modelo de medias móviles. Fuente: Propia

Finalmente, se organiza toda la información de las métricas recopiladas en un dataframe para facilitar la revisión y comparación de los resultados obtenidos para cada tamaño de ventana.

```
metrics_df = pd.DataFrame(metrics_results, columns=['Window_Size', 'MAE', 'MSE', 'RMSE', 'R2'])
metrics_df = metrics_df.set_index('Window_Size')
metrics_df
```

Figura 48 Bloque de código 6 del modelo de medias móviles. Fuente: Propia

Un ejemplo de los resultados de la aplicación de cada tamaño de ventana es el siguiente.

Window_Size	MAE	MSE	RMSE	R2
1	363.56	232431.99	482.11	-0.55
2	360.05	215583.95	464.31	-0.44
3	362.34	211250.48	459.62	-0.41
4	355.97	202164.84	449.63	-0.35
5	344.78	191790.45	437.94	-0.28
6	336.34	182541.62	427.25	-0.22
7	323.52	170531.38	412.95	-0.14
8	310.31	157229.90	396.52	-0.05

Figura 49 Métricas resultantes de la prueba de tamaño de ventana para el modelo de medias móviles. Fuente: Propia

Las métricas de evaluación finales son las siguientes y en ellas se aprecian errores muy altos y un coeficiente de determinación cercano a 0, denotando el pobre ajuste de las predicciones a la serie original.

	Model	MAE	MSE	RMSE	R2
0	Medias Móviles	302.60	144174.29	379.70	0.04

Figura 50 Métricas de evaluación del modelo de medias móviles final. Fuente: Propia

Por esta razón, el gráfico final alcanzado muestra como las predicciones realizadas por el modelo son bastante desacertadas, ajustándose con dificultades a los datos de prueba.

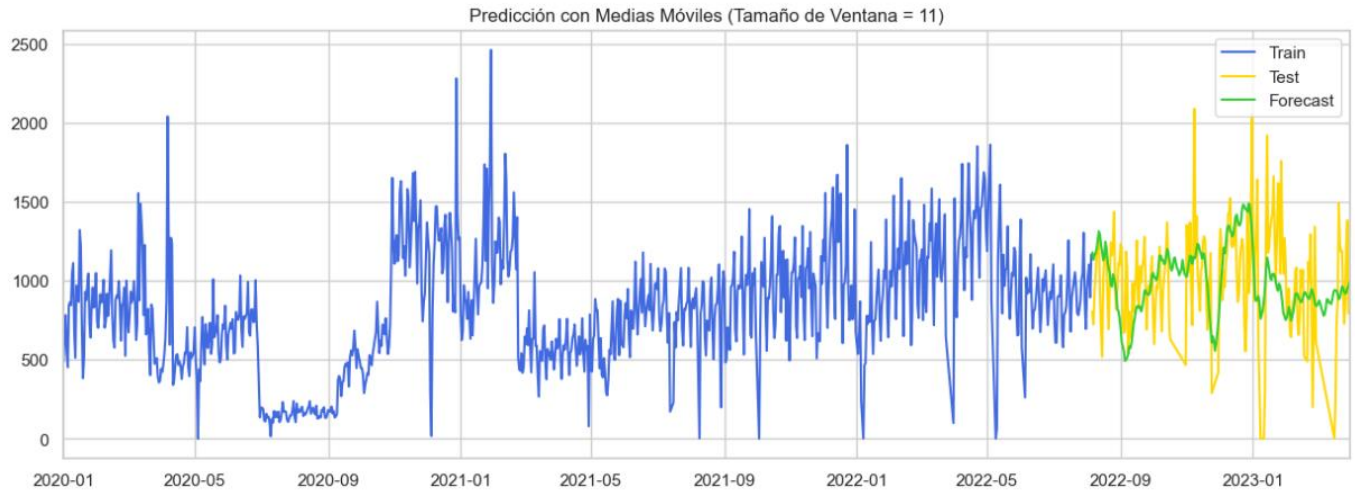

Figura 51 Gráfico de recuento de accidentes diarios generado con los datos originales y las predicciones del modelo de medias móviles. Fuente: Propia

SARIMAX

Para el segundo modelo, se va a entrenar un modelo SARIMAX, el cual ya tiene en cuenta las variables meteorológicas exógenas.

En primer lugar, se vuelven a dividir los datos en 80% entrenamiento y 20% prueba.

```
train_size = int(len(df_count) * 0.8)
train, test = df_count.iloc[:train_size], df_count.iloc[train_size:]
```

Figura 52 Bloque de código 1 del modelo SARIMAX. Fuente: Propia

Se definen los rangos de los parámetros p, d, y q, así como los parámetros estacionales p, d, q y el período estacional, en este caso semanal (7 días). Luego, se generan todas las combinaciones posibles de estos parámetros.

```
p = d = q = range(0, 3)
seasonal_p = seasonal_d = seasonal_q = range(0, 2)
seasonal_period = [7]

pdq = list(product(p, d, q))
seasonal_pdq = [(x[0], x[1], x[2], sp) for x in \
                list(product(seasonal_p, seasonal_d, seasonal_q)) for sp in seasonal_period]
```

Figura 53 Bloque de código 2 del modelo SARIMAX. Fuente: Propia

Se inicializan variables para almacenar la mejor combinación de parámetros “best_pdq” y “best_seasonal_pdq” y su puntuación correspondiente “best_score”. También se crea una lista vacía (metrics_results) para almacenar los resultados de las métricas para cada combinación de parámetros.

```
best_score = float('inf')
best_pdq = None
best_seasonal_pdq = None
metrics_results = []
```

Figura 54 Bloque de código 3 del modelo SARIMAX. Fuente: Propia

A continuación, se crea un bucle que itera sobre todas las combinaciones de parámetros definidos anteriormente, ajustando un modelo SARIMAX para cada combinación y evaluando su rendimiento en el conjunto de prueba con las métricas. Se guardan los resultados y se actualizan las mejores combinaciones de parámetros y su puntuación basada en el RMSE.

```
for param in pdq:
    for param_seasonal in seasonal_pdq:
        try:
            model = SARIMAX(train['Accident_Count'], order=param, seasonal_order=param_seasonal, \
                            enforce_stationarity=False, enforce_invertibility=False)
            results = model.fit(disp=False)
            forecast = results.get_forecast(steps=len(test)).predicted_mean
            mae, mse, rmse, r2 = evaluate_forecast(test['Accident_Count'], forecast)
            metrics_results.append((param, param_seasonal, mae, mse, rmse, r2))

            if rmse < best_score:
                best_score = rmse
                best_pdq = param
                best_seasonal_pdq = param_seasonal
        except:
            continue
```

Figura 55 Bloque de código 4 del modelo SARIMAX. Fuente: Propia

Una vez encontradas las mejores combinaciones de parámetros, en este caso (p, d, q) = (1, 2, 2) y (P, D, Q, m) = (1, 0, 1, 7), se extraen las métricas asociadas a dichas combinaciones y se almacenan en variables separadas para su análisis posterior.

```
metrics_sarimax = [metrics for metrics in metrics_results \
                   if metrics[0] == best_pdq and metrics[1] == best_seasonal_pdq][0]
sarimax_mae, sarimax_mse, sarimax_rmse, sarimax_r2 = metrics_sarimax[2:6]
```

Figura 56 Bloque de código 5 del modelo SARIMAX. Fuente: Propia

Se ajusta el modelo SARIMAX utilizando las mejores combinaciones de parámetros encontradas, se genera una predicción para el conjunto de prueba, se añade la columna de predicciones al conjunto de prueba y se visualizan los datos de entrenamiento, prueba y predicción en un gráfico.

```
best_model = SARIMAX(train['Accident_Count'], order=best_pdq, seasonal_order=best_seasonal_pdq, \
                     enforce_stationarity=False, enforce_invertibility=False)
sarimax_results = best_model.fit(disp=False)
test.loc[:, 'Forecast'] = sarimax_results.get_forecast(steps=len(test)).predicted_mean

plt.figure(figsize=(15, 5))
plt.plot(train.index, train['Accident_Count'], label='Train', color="royalblue")
plt.plot(test.index, test['Accident_Count'], label='Test', color="gold")
plt.plot(test.index, test['Forecast'], label='Forecast', color="limegreen")
plt.legend(loc='upper right')
plt.title(f'Predicción con SARIMAX ((p, d, q) = {best_pdq}, (P, D, Q, m) = {best_seasonal_pdq})')
plt.margins(x=0)
plt.show()
```

Figura 57 Bloque de código 6 del modelo SARIMAX. Fuente: Propia

Por último, se organiza toda la información de las métricas recopiladas en un dataframe para facilitar la revisión y comparación de los resultados obtenidos para cada combinación de parámetros.

```
metrics_df = pd.DataFrame(metrics_results, columns=['Order', 'Seasonal_Order', 'MAE', 'MSE', 'RMSE', 'R2'])
metrics_df = metrics_df.set_index(['Order', 'Seasonal_Order'])
metrics_df
```

Figura 58 Bloque de código 7 del modelo SARIMAX. Fuente: Propia

Un ejemplo de los resultados de la aplicación de cada combinación de parámetros es el siguiente.

Order	Seasonal_Order	MAE	MSE	RMSE	R2
(0, 0, 0)	(0, 0, 0, 7)	915.11	987023.11	993.49	-5.60
	(0, 0, 1, 7)	903.79	968238.68	983.99	-5.47
	(0, 1, 0, 7)	277.79	145459.93	381.39	0.03
	(0, 1, 1, 7)	276.53	141125.24	375.67	0.06
	(1, 0, 0, 7)	577.05	492520.87	701.80	-2.29

Figura 59 Métricas resultantes de la prueba de parámetros para el modelo SARIMAX. Fuente: Propia

Aunque ligeramente mejores que en el anterior modelo, las métricas de evaluación finales siguen siendo bastante malas, con un error muy alto y un R^2 cercano a 0.

	Model	MAE	MSE	RMSE	R2
1	SARIMAX	276.42	137427.11	370.71	0.08

Figura 60 Métricas de evaluación del modelo SARIMAX final. Fuente: Propia

El gráfico demuestra como el modelo SARIMAX únicamente ha extraído la componente estacional con una ligera tendencia decreciente y ha generado las predicciones en base a ella, dando como resultado un ajuste muy equivocado.

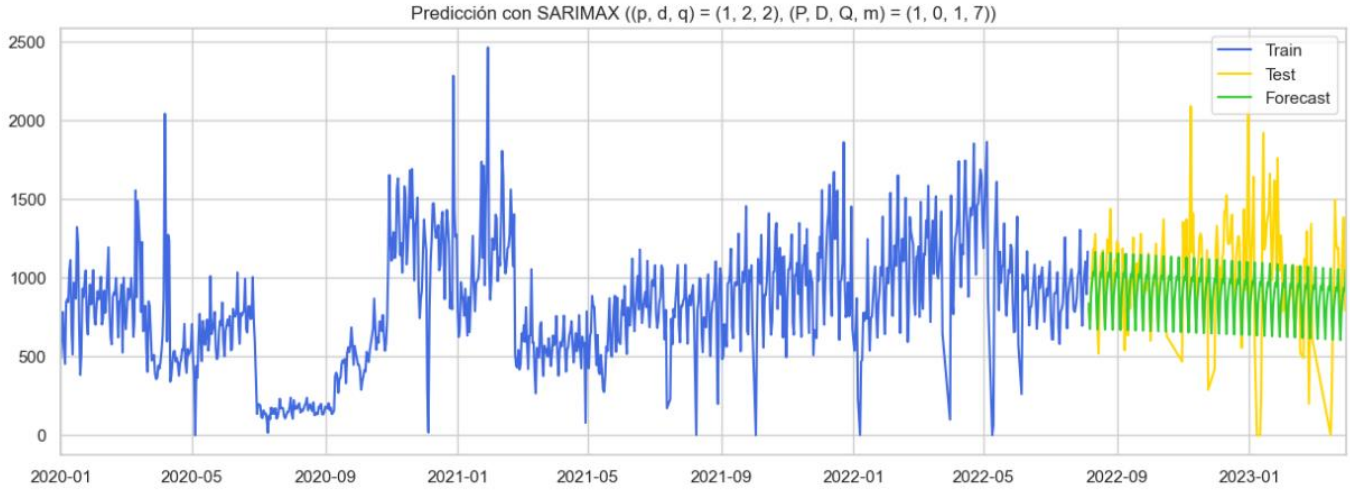

Figura 61 Gráfico de recuento de accidentes diarios generado con los datos originales y las predicciones del modelo SARIMAX. Fuente: Propia

Prophet

El siguiente modelo que se va a desarrollar es Prophet, desarrollado por Facebook.

Al comienzo, se preparan los datos para ser utilizados con Prophet. Se renombra la columna de fechas a “ds” y la columna de conteo de accidentes a “y”. Luego, se divide el conjunto de datos en entrenamiento y prueba.

```
df_prophet = df_count[['Accident_Count']].reset_index()
df_prophet.rename(columns={'Start_Time': 'ds', 'Accident_Count': 'y'}, inplace=True)

train_size = int(len(df_prophet) * 0.8)
train, test = df_prophet.iloc[:train_size], df_prophet.iloc[train_size:].copy()
```

Figura 62 Bloque de código 1 del modelo Prophet. Fuente: Propia

Se definen los valores de los parámetros “changepoint_prior_scale” y “seasonality_prior_scale” que serán evaluados. Además, se inicializan variables para almacenar la mejor combinación de parámetros “best_params” y su puntuación correspondiente “best_score”. También se crea una lista vacía “metrics_results” para almacenar los resultados de las métricas para cada combinación de parámetros.

```
changepoint_prior_scale_values = [0.01, 0.1, 0.5, 1.0]
seasonality_prior_scale_values = [0.01, 0.1, 1.0, 10.0]

best_score = float('inf')
best_params = None
metrics_results = []
```

Figura 63 Bloque de código 2 del modelo Prophet. Fuente: Propia

Después, se genera un bucle que itera sobre todas las combinaciones de los parámetros definidos anteriormente, ajustando un modelo Prophet para cada combinación y evaluando su rendimiento en el conjunto de prueba con las métricas. Se guardan los resultados y se actualizan las mejores combinaciones de parámetros y su puntuación basada en el RMSE.

```
for changepoint_prior_scale in changepoint_prior_scale_values:
    for seasonality_prior_scale in seasonality_prior_scale_values:
        model = Prophet(
            changepoint_prior_scale=changepoint_prior_scale,
            seasonality_prior_scale=seasonality_prior_scale
        )
        model.fit(train)

        future = model.make_future_dataframe(periods=len(test))
        forecast = model.predict(future)
        forecast_test = forecast['yhat'][-len(test):].values

        mae, mse, rmse, r2 = evaluate_forecast(test['y'], forecast_test)
        metrics_results.append((changepoint_prior_scale, seasonality_prior_scale, mae, mse, rmse, r2))

        if rmse < best_score:
            best_score = rmse
            best_params = (changepoint_prior_scale, seasonality_prior_scale)
```

Figura 64 Bloque de código 3 del modelo Prophet. Fuente: Propia

Una vez encontradas las mejores combinaciones de parámetros, en este caso "Changepoint Prior Scale" = 0,5 y "Seasonality Prior Scale" = 0,01, se extraen las métricas asociadas a dichas combinaciones y se almacenan en variables separadas para su análisis posterior.

```
best_metrics = [metrics for metrics in metrics_results if \
                metrics[0] == best_params[0] and metrics[1] == best_params[1]][0]
prophet_mae, prophet_mse, prophet_rmse, prophet_r2 = best_metrics[2:6]
```

Figura 65 Bloque de código 4 del modelo Prophet. Fuente: Propia

A continuación, se ajusta el modelo Prophet utilizando las mejores combinaciones de parámetros encontradas, se genera una predicción para el conjunto de prueba, se añade la columna de predicciones al conjunto de prueba y se visualizan los datos de entrenamiento, prueba y predicción en un gráfico.

```
model = Prophet(
    changepoint_prior_scale=best_params[0],
    seasonality_prior_scale=best_params[1]
)
model.fit(train)
future = model.make_future_dataframe(periods=len(test))
forecast = model.predict(future)
test['Forecast'] = forecast['yhat'][-len(test):].values

plt.figure(figsize=(15, 5))
plt.plot(train['ds'], train['y'], label='Train', color="royalblue")
plt.plot(test['ds'], test['y'], label='Test', color="gold")
plt.plot(test['ds'], test['Forecast'], label='Forecast', color="limegreen")
plt.legend(loc='upper right')
plt.title(f'Predicción con Prophet (Changepoint Prior Scale = {best_params[0]}, \
                    Seasonality Prior Scale = {best_params[1]})')
plt.margins(x=0)
plt.show()
```

Figura 66 Bloque de código 5 del modelo Prophet. Fuente: Propia

Para terminar, se organiza toda la información de las métricas recopiladas en un dataframe para facilitar la revisión y comparación de los resultados obtenidos para cada combinación de parámetros.

```
metrics_df = pd.DataFrame(metrics_results, columns=['Changepoint_Prior_Scale', 'Seasonality_Prior_Scale', \
                                                    'MAE', 'MSE', 'RMSE', 'R2'])
metrics_df = metrics_df.set_index(['Changepoint_Prior_Scale', 'Seasonality_Prior_Scale'])
metrics_df
```

Figura 67 Bloque de código 6 del modelo Prophet. Fuente: Propia

Un ejemplo de los resultados de la aplicación de cada combinación de parámetros es el siguiente.

Changepoint_Prior_Scale	Seasonality_Prior_Scale	MAE	MSE	RMSE	R2
0.01	**0.01**	356.75	218851.45	467.82	-0.46
	0.10	390.77	247566.39	497.56	-0.65
	1.00	390.05	246349.84	496.34	-0.65
	10.00	392.05	249253.05	499.25	-0.67
0.10	**0.01**	370.69	237108.34	486.94	-0.59
	0.10	625.39	577822.58	760.15	-2.86
	1.00	634.26	591919.94	769.36	-2.96
	10.00	629.50	584408.79	764.47	-2.91

Figura 68 Métricas resultantes de la prueba de parámetros para el modelo Prophet. Fuente: Propia

En este caso, las métricas de evaluación finales han resultado ser, de nuevo, muy desacertadas.

	Model	MAE	MSE	RMSE	R2
2	Prophet	292.37	144446.74	380.06	0.03

Figura 69 Métricas de evaluación del modelo Prophet final. Fuente: Propia

En el gráfico final se observa como el modelo Prophet ha actuado de manera similar a SARIMAX, extrayendo la componente estacional con ligeras tendencias crecientes y decrecientes y generando las predicciones en base a ello, resultando de nuevo en un ajuste erróneo.

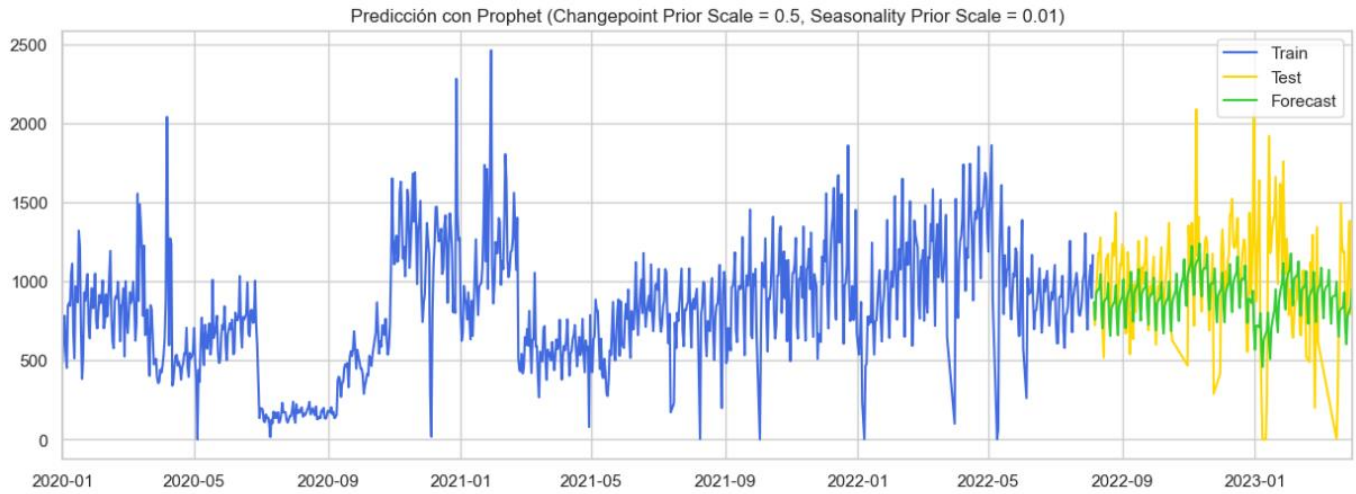

Figura 70 Gráfico de recuento de accidentes diarios generado con los datos originales y las predicciones del modelo Prophet. Fuente: Propia

LSTM

A continuación, se va a entrenar un modelos LSTM para realizar las predicciones de la serie temporal.

En primer lugar, se normalizan los datos de la columna “Accident_Count” usando la técnica de escalado MinMax, lo que transforma los datos para que estén en el rango de 0 a 1.

```
scaler = MinMaxScaler(feature_range=(0, 1))
scaled_data = scaler.fit_transform(df_count[['Accident_Count']])
```

Figura 71 Bloque de código 1 del modelo LSTM. Fuente: Propia

Se define una función para crear el conjunto de datos para el modelo LSTM, generando secuencias de “time_step” longitud. Luego, se utiliza esta función para crear los conjuntos de características “X” y etiquetas “y” a partir de los datos escalados.

```
def create_dataset_lstm(data, time_step=1):
    X, Y = [], []
    for i in range(len(data) - time_step - 1):
        a = data[i:(i + time_step), 0]
        X.append(a)
        Y.append(data[i + time_step, 0])
    return np.array(X), np.array(Y)

time_step = 10
X, y = create_dataset_lstm(scaled_data, time_step)
```

Figura 72 Bloque de código 2 del modelo LSTM. Fuente: Propia

Después, se realiza la división de los conjuntos de datos “X” e “y” en conjuntos de entrenamiento y prueba.

```
train_size = int(len(X) * 0.8)
X_train, X_test = X[:train_size], X[train_size:]
y_train, y_test = y[:train_size], y[train_size:]
```

Figura 73 Bloque de código 3 del modelo LSTM. Fuente: Propia

Se redimensionan los datos de entrenamiento y prueba para que sean compatibles con la entrada esperada por el modelo LSTM, agregando una dimensión extra.

```
X_train = X_train.reshape(X_train.shape[0], X_train.shape[1], 1)
X_test = X_test.reshape(X_test.shape[0], X_test.shape[1], 1)
```

Figura 74 Bloque de código 4 del modelo LSTM. Fuente: Propia

A continuación, se define una función para crear el modelo LSTM con una capa LSTM y una capa densa de salida. El modelo se compila utilizando el optimizador y la función de pérdida especificados.

```
def create_model_lstm(units=50, activation='relu', optimizer='adam'):
    model = Sequential()
    model.add(LSTM(units, activation=activation, input_shape=(time_step, 1)))
    model.add(Dense(1))
    model.compile(optimizer=optimizer, loss='mean_squared_error')
    return model
```

Figura 75 Bloque de código 5 del modelo LSTM. Fuente: Propia

Se configura una búsqueda en cuadrícula "GridSearchCV" para encontrar los mejores hiperparámetros del modelo LSTM, evaluando varias combinaciones de unidades, funciones de activación, optimizadores, tamaños de lote y épocas de entrenamiento.

```
model = KerasRegressor(model=create_model_lstm, verbose=0)

param_grid = {
    'model__units': [50, 100],
    'model__activation': ['relu', 'tanh'],
    'model__optimizer': ['adam', 'rmsprop'],
    'batch_size': [16, 32],
    'epochs': [10, 20]
}

grid = GridSearchCV(estimator=model, param_grid=param_grid, cv=3)
grid_result = grid.fit(X_train, y_train)
```

Figura 76 Bloque de código 6 del modelo LSTM. Fuente: Propia

En siguiente lugar, se entrena el mejor modelo encontrado por la búsqueda en cuadrícula, en este caso con parámetros "Batch Size" = 16, "Epochs" = 20, "Activation Model" = "tanh", "Optimizer" = "adam" y "Units" = 100 y se realizan predicciones sobre los conjuntos de entrenamiento y prueba. Luego, las predicciones y las etiquetas reales se transforman de nuevo a su escala original.

```
best_model = grid_result.best_estimator_

train_predict = best_model.predict(X_train)
test_predict = best_model.predict(X_test)

train_predict = scaler.inverse_transform(train_predict.reshape(-1, 1))
test_predict = scaler.inverse_transform(test_predict.reshape(-1, 1))
y_train = scaler.inverse_transform(y_train.reshape(-1, 1))
y_test = scaler.inverse_transform(y_test.reshape(-1, 1))
```

Figura 77 Bloque de código 7 del modelo LSTM. Fuente: Propia

Se utiliza la función para evaluar el rendimiento del modelo LSTM en el conjunto de prueba.

```
lstm_mae, lstm_mse, lstm_rmse, lstm_r2 = evaluate_forecast(y_test, test_predict)
```

Figura 78 Bloque de código 8 del modelo LSTM. Fuente: Propia

Por último, se preparan los índices de tiempo para las predicciones de entrenamiento y prueba. Luego, se visualizan los datos originales junto con las predicciones del modelo LSTM para los conjuntos de entrenamiento y prueba en un gráfico.

```
train_size = int(len(X) * 0.8)

train_index = df_count.index[time_step:train_size + time_step]
test_index = df_count.index[train_size + time_step:train_size + time_step + len(test_predict)]

plt.figure(figsize=(15, 5))
plt.plot(df_count.index, scaler.inverse_transform(scaled_data), label='Original Data', color="royalblue")
plt.plot(train_index, train_predict, label='Train Prediction', color="gold")
plt.plot(test_index, test_predict, label='Test Prediction', color="limegreen")
plt.legend(loc='upper right')
plt.title(f'Predicción con LSTM (Batch Size = {grid_result.best_params_["batch_size"]}, \
Epochs = {grid_result.best_params_["epochs"]}, \
Activation Model = {grid_result.best_params_["model__activation"]}, \
Optimizer = {grid_result.best_params_["model__optimizer"]}, \
Units = {grid_result.best_params_["model__units"]})')
plt.show()
```

Figura 79 Bloque de código 9 del modelo LSTM. Fuente: Propia

Con LSTM, las métricas de evaluación han resultado ser mucho mejores que aplicando los anteriores de modelos, con uno valores de error mucho menor y un coeficiente de determinación bastante más alejado de 0, denotando un ajuste más exacto a la serie original.

	Model	MAE	MSE	RMSE	R2
3	LSTM	223.32	87376.36	295.59	0.42

Figura 80 Métricas de evaluación del modelo LSTM final. Fuente: Propia

Como se observa en el gráfico, las predicciones se ajustan de manera mucho más precisa a la serie temporal, con una tendencia muy similar. Sin embargo, los picos no resultan tan pronunciados como en los datos originales, hecho que puede resultar incluso más beneficioso y realista al suavizar las inestabilidades causadas en el registro de los accidentes.

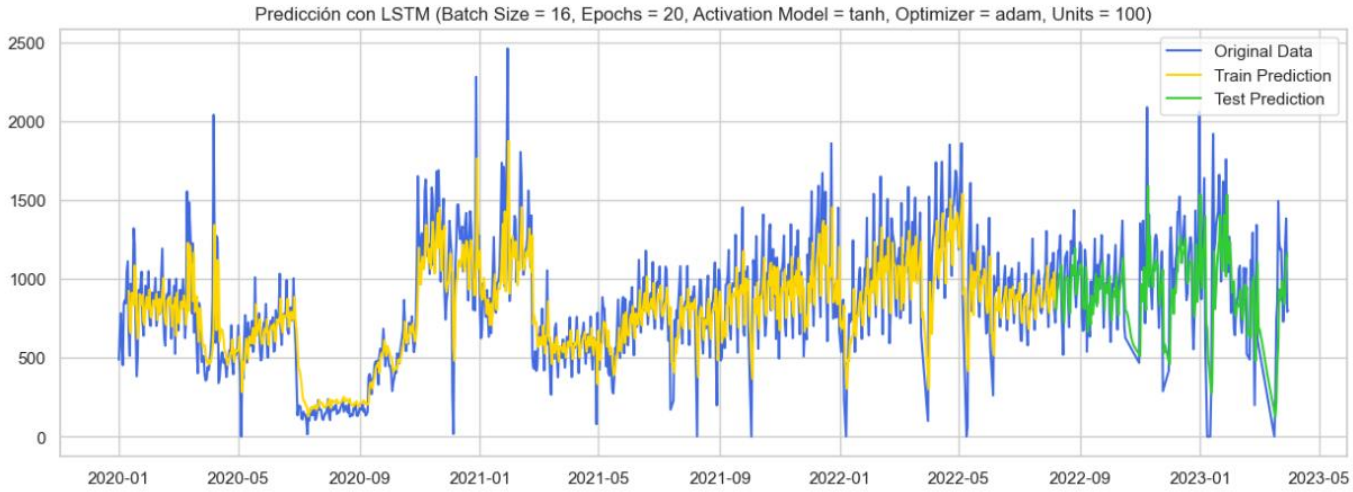

Figura 81 Gráfico de recuento de accidentes diarios generado con los datos originales y las predicciones del modelo LSTM. Fuente: Propia

Redes de Kolmogorov-Arnold (KAN)

El último modelo por entrenar es el basado en redes de Kolmogorov-Arnold (KAN), el más novedoso de todos.

En primer lugar, se realiza la división en conjunto de entrenamiento y prueba.

```
train_size = int(len(df_count) * 0.8)
train, test = df_count.iloc[:train_size], df_count.iloc[train_size:]
```

Figura 82 Bloque de código 1 del modelo KAN. Fuente: Propia

Se normalizan los datos utilizando el StandardScaler. Las características de entrada “X” se escalan para ambos conjuntos de entrenamiento y prueba, mientras que las etiquetas “y” se mantienen sin cambios.

```
scaler = StandardScaler()
X_train = scaler.fit_transform(train.drop('Accident_Count', axis=1).values)
y_train = train['Accident_Count'].values.reshape(-1, 1)
X_test = scaler.transform(test.drop('Accident_Count', axis=1).values)
y_test = test['Accident_Count'].values.reshape(-1, 1)
```

Figura 83 Bloque de código 2 del modelo KAN. Fuente: Propia

Se convierten los datos de entrada y las etiquetas de entrenamiento y prueba en tensores de PyTorch, que son necesarios para el modelo KAN.

```
train_input = torch.tensor(X_train, dtype=torch.float32)
train_label = torch.tensor(y_train, dtype=torch.float32)
test_input = torch.tensor(X_test, dtype=torch.float32)
test_label = torch.tensor(y_test, dtype=torch.float32)
```

Figura 84 Bloque de código 3 del modelo KAN. Fuente: Propia

Se define el modelo KAN con los parámetros especificados, “Grid” = 10 y “k” = 10. Luego, se prepara un diccionario con los datos de entrenamiento y prueba, y se entrena el modelo utilizando el optimizador LBFGS durante 50 pasos.

```
model = KAN(width=[X_train.shape[1], 1, 1], grid=10, k=10)

dataset = {'train_input': train_input, 'train_label': train_label, \
           'test_input': test_input, 'test_label': test_label}

results = model.train(dataset, opt="LBFGS", steps=50)
```

Figura 85 Bloque de código 4 del modelo KAN. Fuente: Propia

A continuación, se realizan predicciones sobre el conjunto de prueba utilizando el modelo entrenado. Las predicciones se convierten a un formato NumPy y se evalúan utilizando las métricas.

```
predictions = model(test_input).detach().numpy()
actual = y_test

kan_mae, kan_mse, kan_rmse, kan_r2 = evaluate_forecast(actual, predictions)
```

Figura 86 Bloque de código 5 del modelo KAN. Fuente: Propia

Se almacenan las pérdidas de entrenamiento y prueba registradas durante el proceso de entrenamiento del modelo.

```
train_losses = results['train_loss']
test_losses = results['test_loss']
```

Figura 87 Bloque de código 6 del modelo KAN. Fuente: Propia

Después, se crea un dataframe con las fechas del conjunto de prueba, las etiquetas reales y las predicciones del modelo, para facilitar la visualización y análisis de los resultados.

```
test_dates = test.index
df_forecast = pd.DataFrame({'Actual': actual.flatten(), 'Forecast': predictions.flatten()}, \
                           index=test_dates)
```

Figura 88 Bloque de código 7 del modelo KAN. Fuente: Propia

Finalmente, se visualizan los datos de entrenamiento, prueba y las predicciones del modelo KAN en un gráfico, permitiendo comparar el rendimiento del modelo con los datos reales.

```
plt.figure(figsize=(15, 5))
plt.plot(train.index, train['Accident_Count'], label='Train', color="royalblue")
plt.plot(test.index, test['Accident_Count'], label='Test', color="gold")
plt.plot(test.index, df_forecast['Forecast'], label='Forecast', color="limegreen")
plt.legend(loc='upper right')
plt.title('Predicción con KAN (Grid = 10, k = 10)')
plt.margins(x=0)
plt.show()
```

Figura 89 Bloque de código 8 del modelo KAN. Fuente: Propia

Las métricas de evaluación resultantes son las peores halladas en ningún momento con un coeficiente de determinación muy por debajo de 0, lo que demuestra que el modelo ajusta peor que la línea horizontal que representaría la media de los datos.

	Model	MAE	MSE	RMSE	R2
4	KAN	390.29	235615.38	485.40	-0.58

Figura 90 Métricas de evaluación del modelo KAN final. Fuente: Propia

Además, en el gráfico de la serie temporal final, se observa un ajuste muy desacertado de las predicciones en comparación con los datos reales.

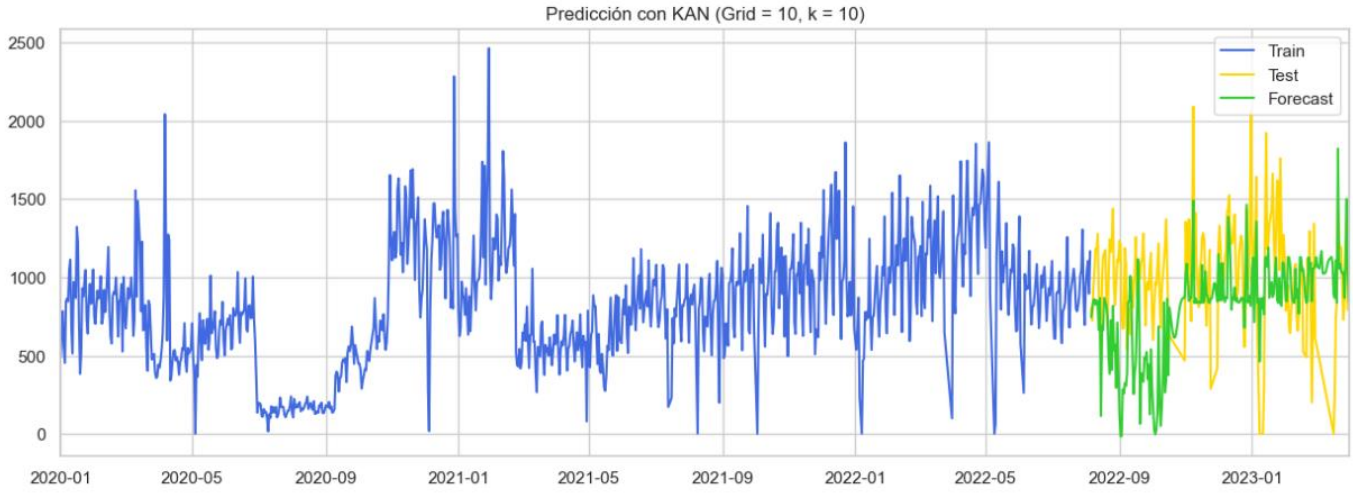

Figura 91 Gráfico de recuento de accidentes diarios generado con los datos originales y las predicciones del modelo KAN. Fuente: Propia

Evaluación de Modelos

Tras el entrenamiento de todos los modelos con la serie temporal de recuento de accidentes diarios, se va a realizar una comparación entre ellos para encontrar el que produce resultados más realistas y acertados y, para ello, se va a hacer uso de las métricas de evaluación y gráficos finales.

	Model	MAE	MSE	RMSE	R2
0	Medias Móviles	302.60	144174.29	379.70	0.04
1	SARIMAX	276.42	137427.11	370.71	0.08
2	Prophet	292.37	144446.74	380.06	0.03
3	LSTM	223.32	87376.36	295.59	0.42
4	KAN	390.29	235615.38	485.40	-0.58

Figura 92 Métricas de evaluación finales de todos los modelos a la hora de predecir la serie temporal de recuento de accidentes diarios. Fuente: Propia

Aunque los gráficos muestran un ajuste completamente mejor en el modelo LSTM en comparación con el resto, las métricas de evaluación respaldan la hipótesis al mostrar tanto errores más bajos como un coeficiente de determinación mucho más alto y, por esa razón, se puede determinar el modelo LSTM como el mejor a la hora de predecir la serie temporal del recuento de accidentes diarios a partir de los datos utilizados en este trabajo.

3.5.2 Predicción de la gravedad media de los accidentes diarios

En segundo lugar, se van a aplicar los mismos modelos para realizar la predicción de la serie temporal que recoge la gravedad media de los accidentes diarios por lo que, en primer lugar, se ha generado un nuevo dataframe en el que se elimina la columna de recuento de accidentes, para que no intervenga como variable exógena en el entrenamiento y no afecte a las predicciones finales.

```
df_severity = df.drop('Accident_Count', axis=1)
```

Figura 93 Código utilizado para general el dataframe utilizado en la predicción de la gravedad media de los accidentes diarios. Fuente: Propia

El código utilizado para realizar las predicciones en este apartado es el mismo que en el anterior, haciendo uso del dataframe “df_severity” en vez de “df_count” y sustituyendo las llamadas a la columna “Accident_Count” por la columna “Severity” ,por lo que a continuación solo se van a presentar las métricas y gráficos finales obtenidos del entrenamiento de los modelos.

Enfoque Clásico – Medias Móviles

Para el modelo de medias móviles, el mejor tamaño de venta encontrado ha sido 7 y la métricas de evaluación finales obtenidas muestran un ajuste muy pobre en el coeficiente de determinación, aunque los errores sean relativamente bajos. Esto se debe a que, al

encontrar una serie temporal históricamente tan estable, la mínima variación en los datos que se predicen produce un desajuste que se ve reflejado en el R^2.

	Model	MAE	MSE	RMSE	R2
0	Medias Móviles	0.07	0.03	0.18	-0.05

Figura 94 Métricas de evaluación del modelo de medias móviles final. Fuente: Propia

Sin embargo, analizando la serie temporal, se puede concluir en que, aunque el ajuste reflejado en las métricas parezca desacertado, los valor predichos son realistas al mantenerse alrededor del valor 2, el cual es el nivel más común de gravedad encontrado. Sin embargo, el modelo de medias móviles produce una serie de inconsistencias claras que dañan el resultado final.

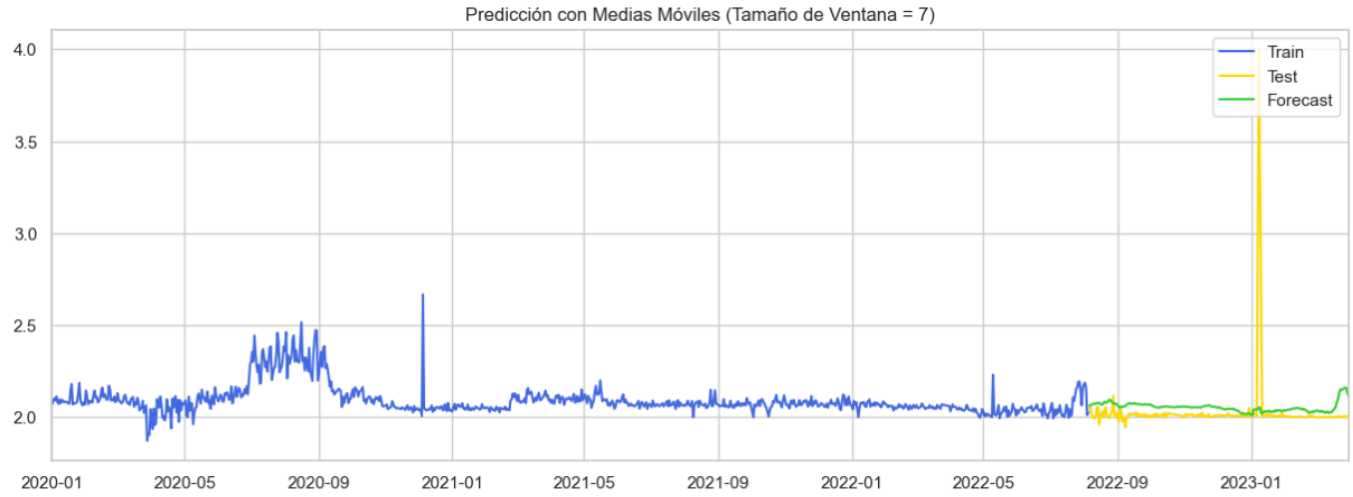

Figura 95 Gráfico de gravedad media de accidentes diarios generado con los datos originales y las predicciones del modelo de medias móviles. Fuente: Propia

SARIMAX

En el caso de SARIMAX, los mejores parámetros encontrados han sido (p, d, q) = (0, 1, 0) y (P, D, Q, m) = (0, 0, 0, 7) y las métricas muestran un resultado similar al del anterior modelos, por lo que el análisis recae en la observación del gráfico.

	Model	MAE	MSE	RMSE	R2
1	SARIMAX	0.04	0.03	0.17	-0.00

Figura 96 Métricas de evaluación del modelo SARIMAX final. Fuente: Propia

En el gráfico de la serie temporal se observa como SARIMAX ha generado valores prácticamente iguales a lo largo de la serie temporal, dando como resultado una recta, sin embargo, esta es ligeramente superior a los valores reales, entendible debido a que

los valores pasado de entrenamiento son ligeramente superiores, explicando así el desajuste encontrado. Pero se puede concluir que, al igual que en el caso del modelo de medias móviles, los resultados son realistas al mantenerse cercanos a 2.

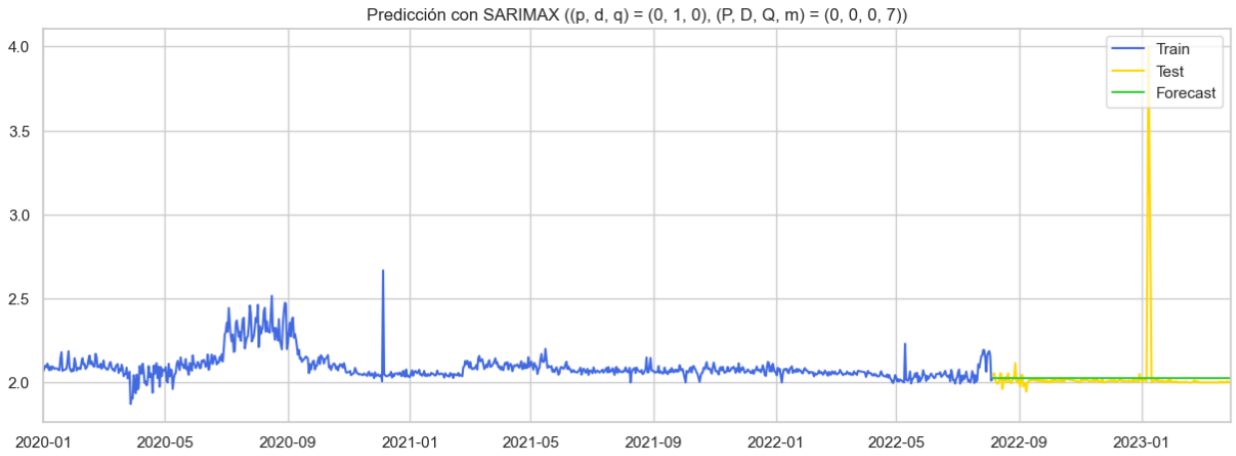

Figura 97 Gráfico de gravedad media de accidentes diarios generado con los datos originales y las predicciones del modelo SARIMAX. Fuente: Propia

Prophet

Para el modelo Prophet, los mejores parámetros son “Changepoint Prior Scale” = 1,0 y “Seasonality Prior Scale” = 10,0 y las métricas reflejan resultados similares a los dos casos anteriores.

	Model	MAE	MSE	RMSE	R2
2	Prophet	0.06	0.03	0.18	-0.04

Figura 98 Métricas de evaluación del modelo de Prophet final. Fuente: Propia

El gráfico refleja un comportamiento similar al resultante en la predicción de recuento diario de accidentes, en el que se ha extraído la componente estacional con ligeras tendencias crecientes y decrecientes y se han generado las predicciones en base a ello, resultando en el peor ajuste observable de todos los modelos.

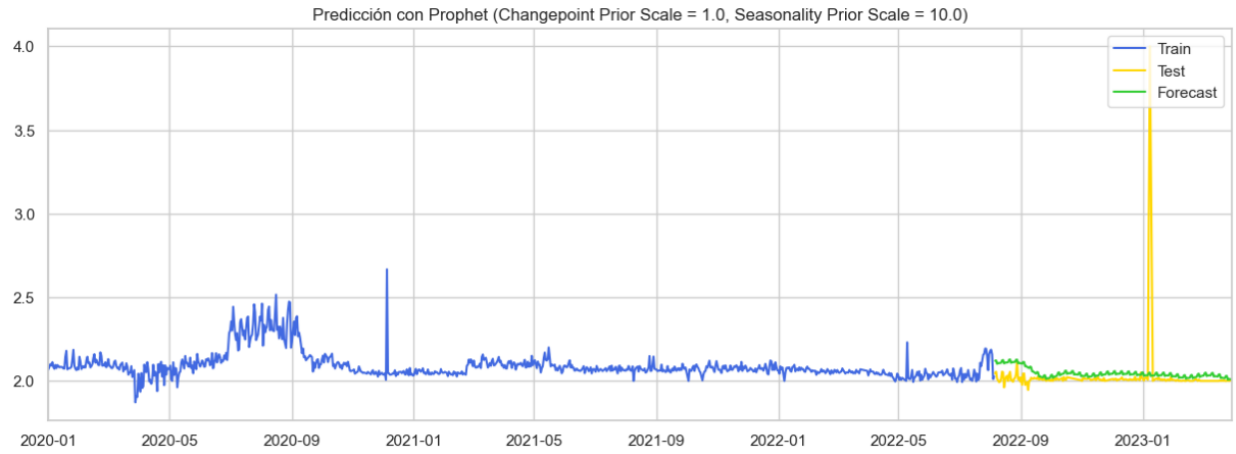

Figura 99 Gráfico de gravedad media de accidentes diarios generado con los datos originales y las predicciones del modelo Prophet. Fuente: Propia

LSTM

En el caso de LSTM, el mejor modelo resultado de los parámetros "Batch Size" = 32, "Epochs" = 20, "Activation Model" = "tanh", "Optimizer" = "adam" y "Units" = 100 y las métricas de evaluación reflejan mejores resultados, con un error ligeramente menor y un coeficiente de determinación algo superior, pero la diferencia significativa se aprecia en la observación de la serie temporal generada.

	Model	MAE	MSE	RMSE	R2
3	LSTM	0.04	0.03	0.17	0.09

Figura 100 Métricas de evaluación del modelo LSTM final. Fuente: Propia

Aunque el ajuste reflejado en el coeficiente de determinación siga siendo cercano a 0, fenómeno explicado anteriormente, se aprecia una serie temporal de predicciones mucho más cercana a los valores reales, manteniéndose constante en torno a 2 en la mayoría del tiempo, pero mostrando las variaciones que se encuentran en determinados momentos. Por esta razón, se puede concluir en que las predicciones del modelo LSTM son suficientemente acertadas para ser consideradas como realistas.

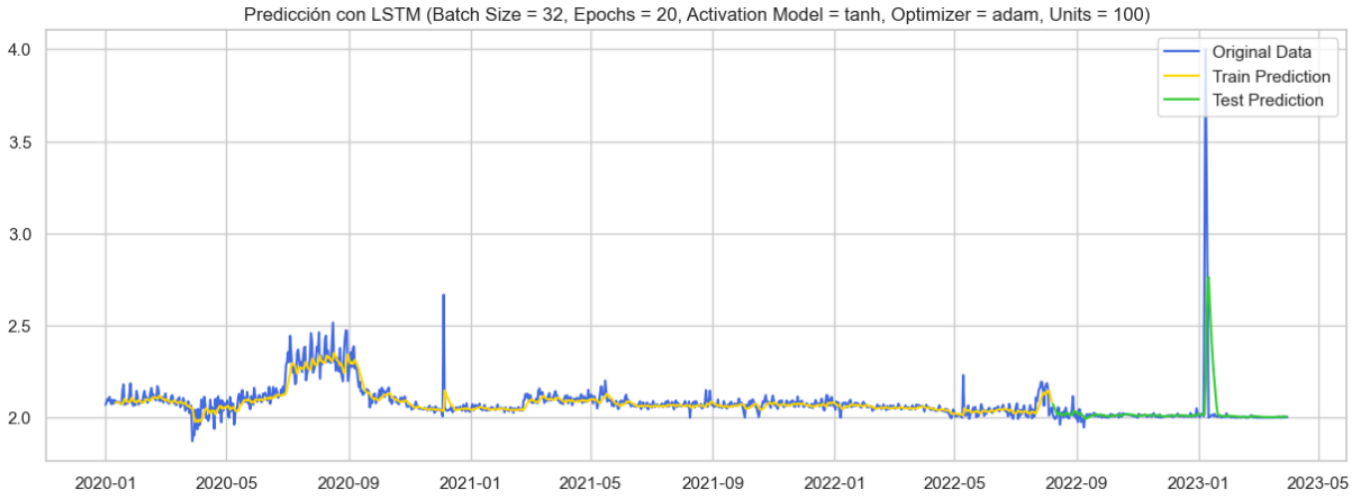

Figura 101 Gráfico de gravedad media de accidentes diarios generado con los datos originales y las predicciones del modelo LSTM. Fuente: Propia

Redes de Kolmogorov-Arnold (KAN)

El último modelo por entrenar es el basado en redes de Kolmogorov-Arnold (KAN), el cual se ha entrenado con los mismo parámetros que en el anterior apartado y produce igualmente métricas de evaluación muy pobres.

	Model	MAE	MSE	RMSE	R2
4	KAN	0.12	0.04	0.21	-0.45

Figura 102 Métricas de evaluación del modelo de KAN final. Fuente: Propia

De igual manera, la gráfica muestra un gran desajuste con una gran cantidad de inconsistencias y separación respecto del valor común 2 como nivel de gravedad medio.

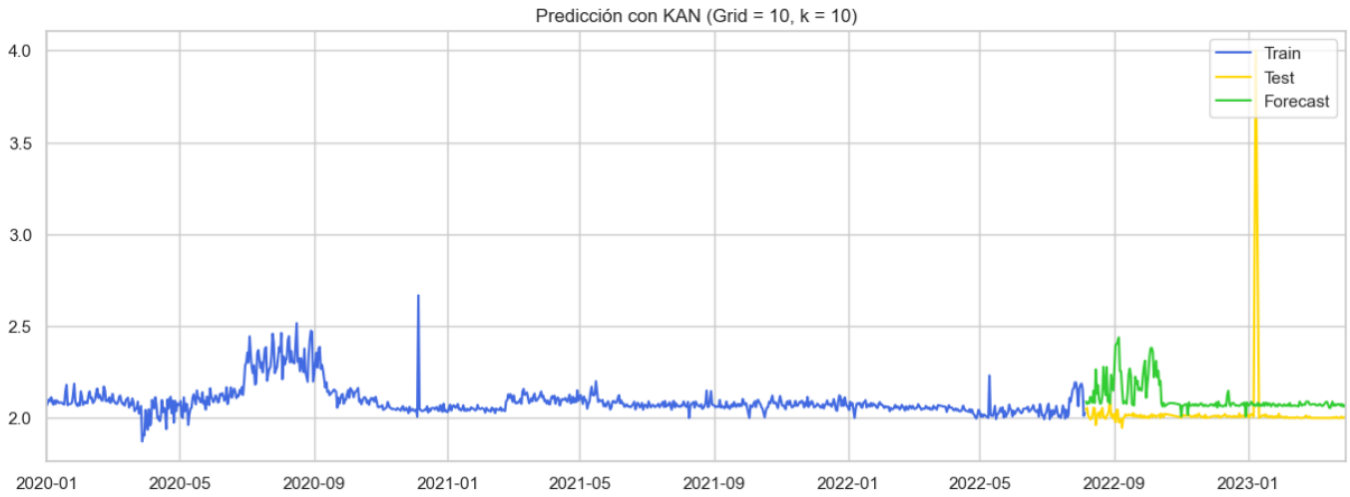

Figura 103 Gráfico de gravedad media de accidentes diarios generado con los datos originales y las predicciones del modelo KAN. Fuente: Propia

Evaluación de Modelos

Una vez finalizado el entrenamiento los modelos con la serie temporal de gravedad me día de accidentes diarios, se va a realizar una comparación entre ellos para encontrar el que produce resultados más realistas y acertados y, para ello, se va a hacer uso de las métricas de evaluación y gráficos finales.

	Model	MAE	MSE	RMSE	R2
0	Medias Móviles	0.07	0.03	0.18	-0.05
1	SARIMAX	0.04	0.03	0.17	-0.00
2	Prophet	0.06	0.03	0.18	-0.04
3	LSTM	0.04	0.03	0.17	0.09
4	KAN	0.12	0.04	0.21	-0.45

Figura 104 Métricas de evaluación finales de todos los modelos a la hora de predecir la serie temporal de gravedad media de accidentes diarios. Fuente: Propia

En este caso, ninguno de los resultados de las métricas resulta demasiado alentador y, aunque LSTM muestre los mejores resultados de nuevo, se debe observar su gráfica para determinarlo como el mejor.

A excepción de LSTM, las gráficas del resto de modelos, en su mayoría, muestran un ajuste decente a la media de nivel gravedad, ligeramente superior a 2, sin embargo, LSTM produce un ajuste mucho más preciso, consiguiendo generar variaciones respecto de esta media en los momentos correctos, permitiendo justificar su bajo coeficiente de correlación en los propios fallos existentes en el registro de los accidentes.

Por esta razón, se puede determinar que, desde un punto de vista realista, LSTM es capaz de generar los datos más acertados y consistentes, pudiéndolo considerar cómo el mejor modelo a la hora de predecir la serie temporal de la gravedad media de los accidentes diarios.

4. Conclusiones y mejoras futuras

4.1. Conclusiones

El objetivo de este trabajo consistía en realiza un estudio y comparación de diferentes técnicas de predicción de series temporales centrando la comparación en el análisis de predicciones, para, a continuación, poder utilizarlas sobre un conjunto de datos de accidentes de tráfico en California para encontrar cuál produce los mejores resultados en ese contexto. Se ha logrado completar satisfactoriamente este propósito, cumpliendo todas las fases planificadas y encontrando LSTM como mejor modelo de predicción de series temporales para el caso de estudio de este trabajo.

En un primer momento, se pretendía utilizar el conjunto de datos original para realizar las predicciones de series temporal, sin embargo, tras realizar una serie de pruebas se encontró que el hecho de que la distribución temporal no fuera uniforme, al ser registros específicos de accidentes, generaba predicciones muy pobres y, en el caso de algunos modelos, directamente no permitía realizar el entrenamiento con ellos.

Sin embargo, se logró solventar el problema modificando el conjunto de datos para contar con registros diarios, lo que permitió generar modelos que funcionaran correctamente y, aunque en su mayoría dieran resultados no deseados, se alcanzó un caso exitoso con el uso de LSTM.

También cabe destacar el mal rendimiento del modelo basado en redes de Kolmogorov-Arnold, como se ha podido apreciar en el estado de la cuestión, esta técnica es muy potente y novedosa, publicada el 30 de abril de 2024, sin embargo, a la hora de realizar su implementación en Python dio una gran cantidad de problemas y errores al tratar de realizar una correcta optimización de parámetros y se encontraron dificultades para encontrar un modelo funcional, aunque generara predicción muy desacertadas.

En definitiva, se puede afirmar con seguridad que el modelo KAN sería capaz de rendir mucho mejor con un conjunto de datos que no suponga tanta inestabilidad como el utilizado en este trabajo y con una correcta optimización de parámetros.

Por último, se menciona que, aunque se hayan alcanzado satisfactoriamente los objetivos del proyecto, a lo largo de su desarrollo se han planteado una serie de mejoras adicionales que podrían beneficiar en los resultados del estudio, las cuales se plantean a continuación.

4.2. Mejoras futuras

Las tres mejoras futuras que se han detectado durante el desarrollo de la aplicación y que podrían añadir mayor valor son las siguientes:

Mejora de los datos

Los mayores impedimentos que se han encontrado en este trabajo han sido derivados de la imposibilidad de comprobar la calidad de los datos.

Sin embargo, con una estudio mucho más profundo y extenso de los mismos e incluso la adición de otro conjunto de registros de accidentes para complementar, se podrían llegar a eliminar las inconsistencias y, con mayor certeza, lograr generar modelos de predicción de series temporales mucho más realistas y acertados.

Revisión del modelo de redes de Kolmogorov-Arnold

La predicción de series temporales con KAN es la más novedosa entre todas las técnicas y por esa razón resultaba la más interesante, sin embargo, su implementación ha resultado ser mucho más compleja de los esperado, por razones de tiempo y porque su desarrollo y análisis se encuentra más avanzado para otros usos como las ecuaciones en derivadas parciales y los resultados merecen un análisis detallado posterior corrigiendo fallos detectados en la optimización de parámetros, derivados muy probablemente de errores de programación en la librería original.

En definitiva, tras una serie de actualizaciones en la librería original que corrijan los errores y contando con un buen conjunto de datos se podrían alcanzar predicciones de series temporales muy efectivas utilizando el modelo basado en redes de Kolmogorov-Arnold.

Desarrollo de la técnica basada en modelos ocultos de Markov

Estos modelos cuentan con una base matemática profunda y compleja y lograr entenderlos completamente y aplicarlos a la predicción de series temporales requeriría de un estudio y desarrollo muy superiores a los que abarca este trabajo.

Sin embargo, contando con la posibilidad de seguir desarrollando este trabajo a futuro, se podría dedicar el tiempo necesario a implementar los modelos ocultos de Markov para predecir series temporales, añadiendo un gran valor al resultado final.

5. Referencias

5.1. Bibliografía

[1] “A comparative analysis of traditional and machine learning methods for traffic incident duration prediction” por S. Kumar, D. Toshniwal, y M. Parida.

[2] “Time series modeling of traffic accidents in Spain” por J. Abellán, G. López, y J. Pérez.

[3] “Effect of urbanization on the impact of traffic crashes” por G. Yannis, E. Papadimitriou, y K. Folla.

[4] “Introduction to Time Series and Forecasting” por P. J. Brockwell y R. A. Davis

[5] “Forecasting: theory and practice” por F. Petropoulos, D. Apiletti, V. Assimakopoulos, M. Z. Babai, D. K. Barrow, S. Ben Taieb, C. Bergmeir, R. J. Bessa, J. Bijak, J. E. Boylan, J. Browell, C. Carnevale, J. L. Castle, P. Cirillo, M. P. Clements, C. Cordeiro, F. L. C. Oliveira, S. De Baets, A. Dokumentov, J. Ellison, P. Fiszeder, P. H. Franses, D. T. Frazier, M. Gilliland, M. S. Gönül, P. Goodwin, L. Grossi, Y. Grushka-Cockayne, M. Guidolin, M. Guidolin, U. Gunter, X. Guo, R. Guseo, N. Harvey, D. F. Hendry, R. Hollyman, T. Januschowski, J. Jeon, V. R. R. Jose, Y. Kang, A. B. Koehler, S. Kolassa, N. Kourentzes, S. Leva, F. Li, K. Litsiou, S. Makridakis, G. M. Martin, A. B. Martinez, S. Meeran, T. Modis, K. Nikolopoulos, D. Önkal, A. Paccagnini, A. Panagiotelis, I. Panapakidis, J. M. Pavía, M. Pedio, D. J. Pedregal, P. Pinson, P. Ramos, D. E. Rapach, J. J. Reade, B. Rostami-Tabar, M. Rubaszek, G. Sermpinis, H. L. Shang, E. Spiliotis, A. A. Syntetos, P. D. Talagala, T. S. Talagala, L. Tashman, D. Thomakos, T. Thorarinsdottir, E. Todini, J. R. Trapero Arenas, X. Wang, R. L. Winkler, A. Yusupova y F. Ziel

[6] “How to Choose the Right Forecasting Technique” por J. C. Chambers, S. K. Mullick y D. D. Smith

[7] “Series Temporales, Modelo Clásico” por S. de la Fuente Fernández

[8] “Time Series Analysis and Its Applications” por R. H. Shumway y D. S. Stoffer

[9] “Application of Sarimax Model to Forecast Daily Sales in Food Retail Industry” por N. Sivanandam, D. Ahrens, M. Fernandes

[10] “Forecasting at Scale” por S. J. Taylor y B. Letham

[11] “Long Short-term Memory” por S. Hochreiter y J. Schmidhuber

[12] “KAN: Kolmogorov-Arnold Networks” por Z. Liu, Y. Wang, S. Vaidya, F. Ruehle, J. Halverson, M. Soljacic, T. Y. Hou y M. Tegmark

[13] “A Countrywide Traffic Accident Dataset” por S. Moosavi, M. H. Samavatian, S. Parthasarathy y R. Ramnath

5.2. Webgrafía

[14] Texas A&M Transportation Institute: "Urban Mobility Report", [en línea]
mobility.tamu.edu/umr/ a 22 de mayo de 2024

[15] Departamento de Vehículos Motorizados de California (DMV): "Annual Report of Fatal and Injury Motor Vehicle Traffic Collisions"
dmv.ca.gov/portal/ a 22 de mayo de 2024

[16] Administración Nacional de Seguridad del Tráfico en las Carreteras (NHTSA): "Distracted Driving"
nhtsa.gov/ a 22 de mayo de 2024

[17] Arellano, M. (2001): "Introducción al Análisis Clásico de Series de Tiempo", [en línea] 5campus.com, Estadística
5campus.com/leccion/seriest a 23 de mayo de 2024

[18] Mohamad (2016): “Análisis de Tendencias para Datos de Series de Tiempo”, [en línea]
support.numxl.com/hc/es/articles/115000147386-An%C3%A1lisis-de-Tendencias-para-Datos-de-Series-de-Tiempo a 24 de mayo de 2024

[19] J. Tesch: “TFG_MACO”, [en línea] GitHub
github.com/Playtesch/TFG_MACO a 30 de mayo de 2024

[20] S. Moosavi (2021): “US-Accidents: A Countrywide Traffic Accident Dataset”, [en línea]
smoosavi.org/datasets/us_accidents a 30 de mayo de 2024