IMF Business School

MÁSTER EN DATA SCIENCE Y BUSINESS ANALYTICS
MODALIDAD ONLINE

Estudio de la actividad sísmica global en 2023 mediante modelos de machine learning y redes neuronales

TFM elaborado por: Cristian Morillo Losada
Tutor/a de TFM: Abel Ángel Soriano Vázquez

- Benavente, 29 de Marzo de 2024 –

Contenido

1. Resumen

El presente estudio se centra en la aplicación de modelos de machine learning y deep learning para predecir la magnitud de los seísmos, utilizando datos recopilados por el USGS en el año 2023. El conjunto de datos contiene numerosas variables con datos registrados en todas partes del mundo. Los seísmos representan un riesgo significativo para la seguridad y la infraestructura, lo que resalta la importancia de desarrollar modelos predictivos precisos y eficaces.

Antes de desarrollar los modelos predictivos, se realizó un análisis exploratorio y detallado de los datos sísmicos, con el fin de comprender mejor la naturaleza de los datos. Entre estos análisis se incluyen gráficos y tablas sobre la distribución de distintas variables como la magnitud, profundidad o el tipo de seísmos, así como mapas geoespaciales con la distribución de los seísmos, análisis temporales, análisis sobre las estaciones detección sísmica y también sobre los errores que se cometen en las mediciones.

Tras esta exploración inicial, se realizó el estudio con los modelos de predicción. Los resultados revelaron que los modelos de regresión lineal y KNN no son adecuados para este tipo de problemas, mientras que los modelos de ensamblaje, como Random Forest y XGBoost fueron los que obtuvieron un mejor rendimiento, ligeramente por encima de los modelos de deep learning entre los que se encontraban un modelo de red neuronal simple y un modelo de red neuronal recurrente de tipo LSTM.

2. Introducción

2.1. Seísmos

Los seísmos son ampliamente conceptualizados como vibraciones de la superficie terrestre que pueden ocasionar, a consecuencia de la energía liberada en rocas, diversos movimientos de la propia superficie manifestados materialmente en forma de fisuras y deslizamientos del suelo, incendios o tsunamis (Abdul Salam et al., 2021; Abebe et al., 2023). Calificado como desastre natural, este fenómeno puede ocasionar efectos catastróficos de diversa índole; daños tanto humanos, financieros como estructurales (Abdul Salam et al., 2021; Patil et al., 2023), sobre todo aquellos que alcanzan magnitudes por encima de 4.5 según la escala de Richter (Abdul Salam et al., 2021).

La causa fundamental de los seísmos se encuentra en la actividad tectónica de la Tierra, debido al movimiento de las placas tectónicas. La teoría de la tectónica de placas postula que la superficie terrestre está dividida en varias placas que flotan sobre el manto terrestre y se desplazan a lo largo del tiempo. Estos desplazamientos de las placas dan lugar a diversos límites entre ellas, produciendo zonas donde convergen, divergen o se deslizan una respecto a la otra. En estos límites es donde se asocia la actividad sísmica (Andel et al., 2024) (Fig. 1).

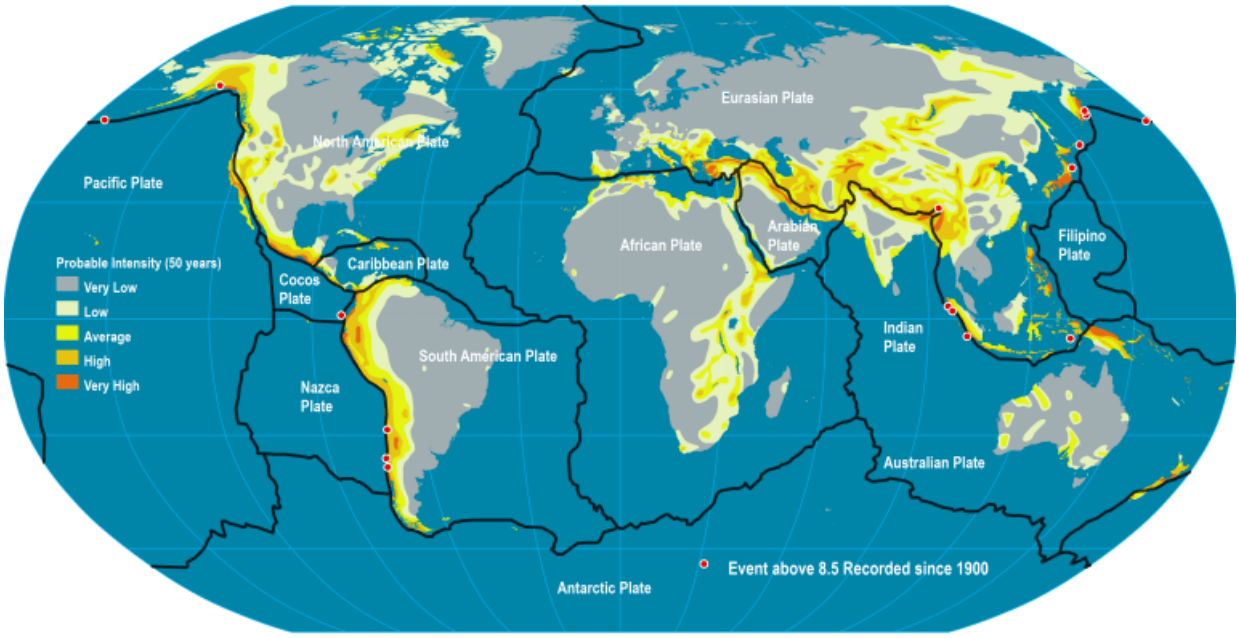


Figura 1- Mapa con los límites de placas y las zonas asociadas con la actividad sísmica. Extraído de https://transportgeography.org/contents/chapter9/transportation-and-disasters/global-plate-tectonics/

Uno de los límites más asociados a la actividad sísmica son las zonas de subducción. Estas zonas están caracterizadas por ser el punto de convergencia de dos placas tectónicas, una de ellas es una placa oceánica que al ser menos densa se introduce bajo la otra placa continental, generando una presión elevada en el límite de ambas, que cuando supera el estrés acumulado y la resistencia de las rocas produce una liberación de energía que produce los seísmos.

Las zonas de subducción no son los únicos procesos tectónicos que pueden desencadenar seísmos, también se producen en los límites de placas divergentes, donde las placas se separan, como en las dorsales oceánicas, y en los límites de placas transformantes, donde las placas se deslizan una respecto a la otra, como en las fallas terrestres. Además, en estas zonas se producen otros fenómenos geológicos como la actividad volcánica que también puede contribuir en la generación de los seísmos (U.S. Geological Survey, 2024; Andel et al., 2024) (Fig. 2).

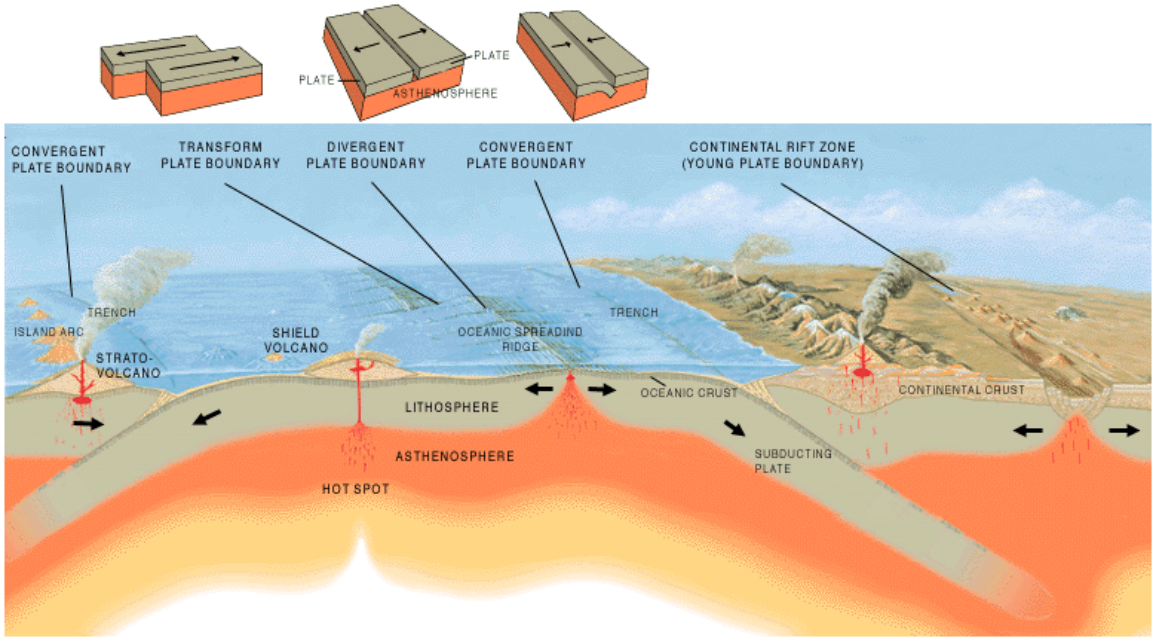


Figura 2- Esquema sobre el movimiento de las placas y la generación de zonas convergentes, divergentes y transformantes, con la actividad volcánica asociada. Extraída de https://pubs.usgs.gov/gip/earthq1/plate.html

Partiendo de esta problemática, existe un consenso entre la comunidad científica sobre la clave necesidad de construir modelos de detección cada vez más avanzados, a pesar de que algunos plantean la imposibilidad de predicción exacta de dicho fenómeno por su naturaleza no lineal (Asim et al., 2017; Patil et al., 2023). En este contexto, los modelos predictivos de *machine learning* y *deep learning* emergen como herramientas fundamentales para abordar estos problemas.

2.2. Machine learning y Deep Learning

El machine learning, o aprendizaje automático, es una disciplina de la inteligencia artificial que se centra en el desarrollo de algoritmos y modelos que pueden aprender de los datos y realizar tareas específicas sin intervención humana explícita (Géron, 2019). Este campo ha experimentado un crecimiento exponencial en las últimas décadas, impulsado por avances en la capacidad computacional, el acceso a grandes volúmenes de datos y el desarrollo de algoritmos cada vez más sofisticados (Jordan y Mitchell, 2015). La funcionalidad principal del machine learning es capacitar a un modelo para que aprenda patrones a partir de los datos disponibles, con el objetivo de hacer predicciones o tomar decisiones en nuevas situaciones. Esto se logra mediante la identificación de características relevantes en los datos y la construcción de un modelo matemático que pueda generalizar estos patrones para hacer predicciones sobre datos no vistos previamente (James et al., 2013).

Uno de los aspectos más destacados del machine learning es su capacidad para adaptarse y mejorar con el tiempo a medida que se le proporciona más información. Esto se logra a través de procesos de entrenamiento iterativos, donde el modelo ajusta sus parámetros en función de la retroalimentación proporcionada por los datos de entrenamiento (Hastie et al., 2009). Existen diferentes enfoques entre los que se incluyen el aprendizaje supervisado, donde el modelo se entrena en un conjunto de datos etiquetados; el aprendizaje no supervisado, donde el modelo encuentra patrones en datos no etiquetados; y el aprendizaje por refuerzo, donde el modelo aprende a través de la interacción con un entorno y la retroalimentación recibida (Fig. 3).

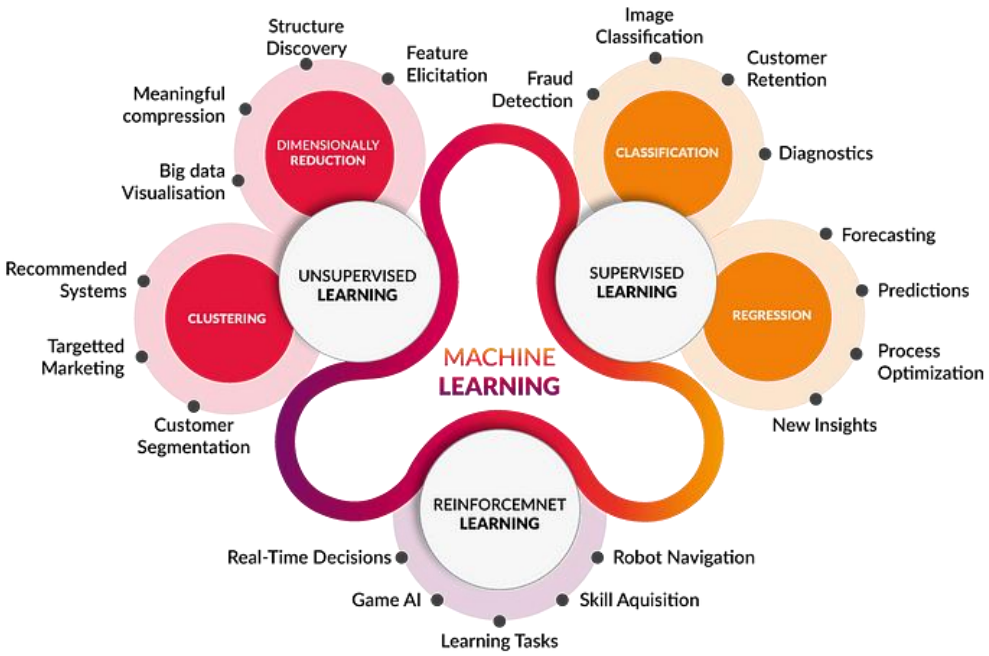


Figura 3- Esquema de los diferentes modelos de machine learning: aprendizaje supervisado (supervised learning), aprendizaje no supervisado (unsupervised learning) y aprendizaje por refuerzo (reinforcement learning). Extraído de https://medium.com/@yadavdeepika729/different-machine-learning-models-2387575e64cd

El deep learning, o aprendizaje profundo, es una subdisciplina del machine learning que se centra en el entrenamiento de redes neuronales artificiales con múltiples capas de procesamiento para aprender representaciones de datos con múltiples niveles de abstracción (Goodfellow et al., 2016). A diferencia de otros enfoques de machine learning, el deep learning permite que los modelos aprendan automáticamente características complejas y de alto nivel directamente de los datos, sin necesidad de extracción de características de forma manual (LeCun et al., 2015). Esto se logra mediante el uso de arquitecturas de redes neuronales profundas, que consisten en múltiples capas de unidades de procesamiento que se organizan de manera jerárquica para aprender representaciones cada vez más abstractas de los datos de entrada (Schmidhuber, 2015). Los componentes de estas redes suelen ser una capa de entrada con varios nodos que procesan y transmiten los datos a las capas ocultas, que procesan la información a distintos niveles, y una o varias capas de salida que emiten los datos de salida del modelo (Amazon Web Services, 2024) (Fig. 4).

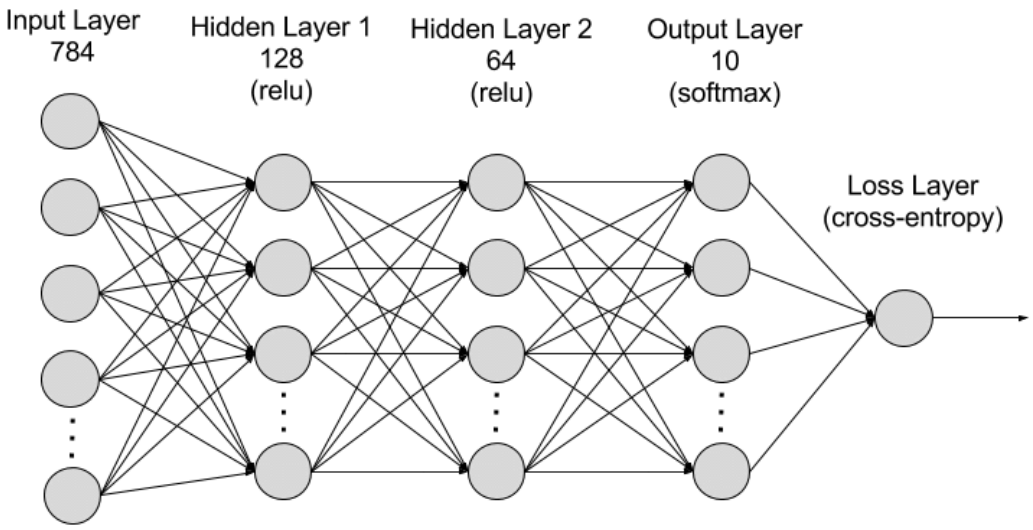


Figura 4- Arquitectura de una red neuronal. Extraído de https://aws.amazon.com/es/what-is/deep-learning/

El surgimiento del deep learning ha revolucionado numerosos campos, incluyendo el procesamiento de imágenes y video, el reconocimiento de voz, la traducción automática, la conducción autónoma y muchos otros. Su éxito se debe en parte a los avances en la capacidad computacional y el acceso a grandes conjuntos de datos etiquetados. Además, el desarrollo de algoritmos de optimización eficientes y técnicas de regularización ha contribuido a mejorar el rendimiento y la generalización de los modelos (Ioffe y Szegedy, 2015).

En el ámbito de la detección de seísmos, tanto los modelos de machine learning como de deep learning han demostrado ser herramientas poderosas en la predicción, ofreciendo la capacidad de analizar grandes volúmenes de datos sísmicos y extraer patrones complejos para predecir la ocurrencia y la magnitud de los seísmos con mayor

precisión y anticipación. Como se mencionó anteriormente, uno de los principales desafíos en la predicción de seísmos es la naturaleza altamente no lineal de los datos sísmicos, que pueden estar sujetos a una variedad de factores y perturbaciones. Estos modelos son capaces de adaptarse y aprender a medida que reciben nuevos datos. De este modo, pueden capturar relaciones complejas y no lineales entre las señales y los eventos sísmicos, lo que mejora la capacidad para predecir con precisión la ocurrencia y la magnitud de los seísmos.

En el estado actual de la investigación sobre predicción sísmica, se observa un enfoque creciente en la aplicación de modelos de aprendizaje automático para mejorar la precisión y la capacidad predictiva. De acuerdo con los modelos de machine learning, Novick y Last (2023) proponen un estudio exhaustivo que utiliza algunos algoritmos como el Random Forest para predecir la ocurrencia de seísmos, empleando indicadores sísmicos y características temporales. Por otro lado, Rashidi y Ghassemieh (2023) se centran en la predicción de la magnitud de seísmos inducidos por actividades humanas, como la inyección subterránea, utilizando técnicas como máquinas de vectores de soporte y el algoritmo AdaBoost. En contraste, Joshi et al. (2024) desarrollan el modelo SeisEML, que combina diferentes algoritmos de regresión para predecir la aceleración del suelo durante seísmos, destacando su aplicación en múltiples regiones sísmicamente activas.

En relación con el deep learning, Wang et al. (2023) proponen una técnica basada en redes neuronales profundas para predecir la magnitud de los seísmos utilizando datos de ondas P registradas por una sola estación. Su enfoque destaca por su rapidez y precisión en la estimación de magnitudes, lo que podría mejorar significativamente los sistemas de alerta temprana de seísmos. Por otro lado, Sadhukhan et al. (2023) utilizan técnicas de deep learning, incluyendo LSTM y Bi-LSTM, para predecir magnitudes sísmicas empleando indicadores sísmicos calculados a partir de varios catálogos de seísmos. Su estudio destaca por su capacidad para predecir magnitudes en diferentes regiones geográficas, lo que sugiere una aplicabilidad más amplia de las técnicas. En un enfoque diferente, Çekim et al. (2023) exploran el uso de métodos de deep learning como CNN y LSTM para prever la magnitud promedio de seísmos en una región específica. Su estudio destaca por su enfoque integral que combina métodos de deep learning con técnicas tradicionales de análisis de series temporales, lo que proporciona una visión más completa de la predicción de sismos.

Todos estos estudios muestran la eficacia y versatilidad de los modelos de aprendizaje automático en la predicción y evaluación del riesgo sísmico, ofreciendo resultados mejorados en términos de precisión y fiabilidad en comparación con enfoques tradicionales. Esta tendencia hacia modelos más avanzados refleja una evolución en la capacidad de los investigadores para abordar los desafíos asociados con la predicción y mitigación de eventos sísmicos.

2.1. Objetivos

Dada la relevancia e importancia de la predicción temprana de los seísmos según lo expuesto, en este trabajo se propone explorar diferentes modelos tanto de machine learning como de deep learning para para predecir la magnitud de los seísmos en base a los datos recopilados en 2023 por el USGS (U.S. Geological Survey, 2024). Los objetivos de este trabajo son:

- Explorar y analizar el conjunto de datos llevando a cabo distintos análisis sobre la magnitud y profundidad de los seísmos, su distribución geoespacial y temporal, análisis de las estaciones de detección sísmica y de los errores en la medición.

- Utilizar varios modelos de machine learning y deep learning para la predicción de la magnitud de los seísmos.

- Comparar los modelos utilizados para valorar cuáles obtienen mejores resultados predictivos.

3. Materiales y métodos

3.1. Conjunto de datos

Para la realización de este trabajo se utilizó un conjunto de datos que contiene información recopilada de estaciones de detección sísmica por todo el mundo con el objetivo de analizar la actividad sísmica y predecir la magnitud de los seísmos. El conjunto de datos se encuentra subido en la página web de Kaggle (Keser, M., 2024) desde donde se ha utilizado para realizar el análisis. Kaggle es una plataforma en línea que permite a cualquier persona poder acceder y utilizar distintos conjuntos de datos para realizar desafíos de análisis de datos, pudiendo compartir los resultados y colaborar en proyectos con más personas. Además, contiene los Kaggle Notebooks, unos notebooks basados en Jupyter (Proyecto Jupyter, 2024), una aplicación de código abierto que permite crear documentos en código de varios lenguajes de programación entre los que se incluye Python, que es el lenguaje que se utiliza en este proyecto. Con los Kaggle Notebooks se puede tener acceso a GPUs (Unidades de Procesamiento Gráfico) que aceleran significativamente el tiempo de ejecución de modelos como las redes neuronales, lo que es muy beneficioso para la realización de este trabajo.

La fuente de origen de los datos proviene de la página oficial del Servicio Geológico de los Estados Unidos (U.S. Geological Survey, 2024). Esta agencia científica fue fundada en 1879 y pertenece al gobierno federal de Estados Unidos. Además de aportar datos sobre los seísmos, también proporciona una variedad de datos sobre hidrología, cartografía y topografía, para ayudar a la toma de decisiones medioambientales, de recursos y de seguridad pública.

El conjunto de datos contiene múltiples variables que capturan información sobre los eventos sísmicos registrados durante el año 2023 en todas las partes del mundo, mostrando dónde ocurrieron y cómo fueron detectados. Los datos están almacenados en un archivo *.csv* accesible a través del siguiente código en Kaggle:

```
import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

Las variables que contiene el archivo son las siguientes:

- **Time**: fecha y hora a la que ocurrió el seísmo.
- **Latitude**: coordenadas geográficas que marcan la latitud.

- **Longitude**: coordenadas geográficas que marcan la longitud.
- **Depth**: profundidad del seísmo en kilómetros.
- **Mag**: magnitud del seísmo.
- **MagType**: tipo de medida de magnitud.
- **Nst**: número de estaciones sísmicas que reportaron el seísmo.
- **Gap**: la brecha entre la cobertura de diferentes estaciones sísmicas.
- **Dmin**: distancia mínima al epicentro del seísmo para la estación más cercana.
- **Rms**: raíz cuadrada media del espectro de amplitud del seísmo.
- **Net**: red que informa el seísmo.
- **Id**: identificador único para el evento del seísmo.
- **Updated**: fecha y hora que indica cuándo se actualizó por última vez la información del seísmo.
- **Place**: descripción de la ubicación del seísmo.
- **Type**: tipo de evento sísmico.
- **HorizontalError**: error horizontal en la determinación de la ubicación.
- **DepthError**: error en la determinación de la profundidad.
- **MagError**: error en la determinación de la magnitud.
- **MagNst**: número de estaciones sísmicas utilizadas para calcular la magnitud.
- **Status**: estado del evento sísmico.
- **LocationSource**: fuente que informa la ubicación del seísmo.
- **MagSource**: fuente que informa la magnitud del seísmo.

3.2. Preprocesamiento de los datos

El preprocesamiento de los datos es un paso previo importante en cualquier proyecto de análisis de datos. Consiste en preparar y limpiar los datos para que los análisis posteriores se lleven a cabo correctamente, evitando la falta de calidad e integridad de los datos que puede influir en los resultados finales.

Una de las primeras tareas es cargar los datos desde la fuente de origen. Como los datos se encuentran en un archivo *.csv* tenemos que utilizar la biblioteca Pandas (McKinney, W., 2010) para cargarlos. Mediante la función ***pd.read_csv()*** de Pandas, cargamos los datos desde el archivo *.csv* en un DataFrame, que es una estructura de datos tabular que nos permitirá manipular y analizar los datos más fácilmente.

Después de cargar los datos, el siguiente paso es explorar y entender como están estructurados los datos. Esto se logra utilizando algunas funciones y métodos de Pandas:

- Con la función ***head(10)*** mostramos las primeras diez filas del DataFrame, lo que nos proporciona una vista preliminar de los datos y sus columnas. Esto nos permite ver cómo son los valores que se almacenan en cada columna y ver que formato tienen.

- Con la función ***info()*** obtenemos información sobre el DataFrame, incluyendo la cantidad de valores no nulos que hay en cada variable, y el tipo de datos de cada una.

- Con la función ***describe(include='all')*** calculamos estadísticas descriptivas para cada columna del DataFrame: recuento, media, desviación, valores mínimos y máximos y los cuartiles. Esto proporciona una visión detallada que puede servir de ayuda a la hora de la limpieza de datos.

Tras esta exploración inicial de los datos, se empezó con la limpieza de estos para garantizar la calidad e integridad. Se siguieron los siguientes pasos:

- Se utilizaron las funciones ***isnull().sum()*** para identificar los valores nulos que existían en cada columna del DataFrame. Esto nos mostró que había valores nulos en las columnas *nst, gap, dmin, place, horizontalError, magError* y *magNst*.

- Se eliminaron las filas con valores nulos de las columnas específicas que detectamos con las funciones anteriores utilizando el método ***dropna()*** con el argumento '*subset*' donde especificamos las columnas donde se buscarán y eliminarán.

- Se verificó la presencia de valores duplicados utilizando la función ***duplicated()***, y se guardaron en una variable nombrada como 'duplicados' para imprimirla en pantalla y mostrar el número total de duplicados que era de 1788.

```
# Explorar total de valores duplicados
duplicados = df[df.duplicated()]
print("Número total duplicados: ", len(duplicados))
```

- Se eliminaron todas las filas duplicadas utilizando el método ***drop_duplicates()***. Esto garantizó que cada fila fuera única en el conjunto de datos.

- Tras haber explorado previamente las estadísticas descriptivas con la función ***describe()***, me di cuenta que en algunas columnas había valores que no podían ser 0, como las columnas *nst* y *mgNst*, ya que no puede haber eventos sísmicos donde el número de estaciones que los detecten sea 0, se trata de un error en la inserción de los datos. Por tanto, observando los valores mínimos que existen en estas columnas se detecta que se corresponden con 0. Se filtraron estas filas mediante indexación booleana con ***loc*** seleccionando las filas que son distintas de 0, el resto se descartan.

```
# Eliminar filas donde 'nst' y 'magNst' sean iguales a 0
df = df.loc[(df['nst'] != 0) & (df['magNst'] != 0)]
```

- Por último, se convirtió la columna *time* al tipo de dato '*datetime*' utilizando la función ***pd.to_datetime()*** con el fin de facilitar el uso de estos datos posteriormente.

3.3. Análisis Exploratorio de los Datos (EDA)

Después del preprocesamiento de los datos se realizó un análisis exploratorio de datos con el objetivo de comprender mejor la estructura, características y relaciones entre los conjuntos de datos. En este análisis se han empleado sobre todo métodos de visualización de datos que nos ayudan a entender mejor el problema. Los resultados de estos análisis se detallarán más adelante, pero conviene explicar los tipos de análisis que se han realizado.

3.3.1. Análisis de la magnitud, profundidad y el tipo de seísmos

Primero se comenzó explorando las características fundamentales de los seísmos, como son la magnitud y la profundidad, y también el tipo de seísmo que se registra como evento. Para ello se realizaron una serie de pasos:

- **Exploración del tipo de seísmo**: se exploraron las diferentes categorías de datos de la columna *type* para identificar qué tipo de seísmos son los que se analizan. Esto es importante para comprender la diversidad de los eventos sísmicos registrados.

  Los distintos tipos de seísmos que se engloban en la columna *type* son los siguientes:

  1. **Earthquake** (Seísmo): evento sísmico natural causado por la liberación de energía en la corteza terrestre debido al movimiento de placas tectónicas.

2. **Mining Explosion** (Explosión en minería): explosión causada por actividades mineras, como explosiones controladas para extraer minerales.

3. **Quarry Blast** (Explosión de cantera): explosión controlada en una cantera para extraer rocas u otros minerales.

4. **Ice Quake** (Temblor de hielo): evento sísmico causado por la fractura del hielo, que se puede asociar con la desintegración de icebergs o capas de hielo.

5. **Explosion** (Explosión): evento sísmico causado por una explosión, que puede ser de origen natural o humano.

6. **Volcanic Eruption** (Erupción volcánica): liberación de magma, ceniza y gases desde un volcán.

- **Conteo y visualización de los tipos de seísmos**: se creó un gráfico de barras utilizando la biblioteca de visualización Seaborn (Waskom, M., 2020), que proporciona una interfaz de alto nivel para crear gráficos más atractivos que con otras bibliotecas. Esta biblioteca también se utilizará para gráficos posteriores. Se utilizó ***sns.countplot()*** para contar y visualizar la frecuencia de cada tipo de seísmo, lo que nos proporciona una idea de la distribución de los diferentes tipos de seísmos que son registrados.

- **Histogramas de magnitud y profundidad**: se utilizó también Seaborn para crear histogramas con ***sns.histplot()*** para explorar la distribución de la magnitud y profundidad de los seísmos. Este análisis proporciona información sobre como se distribuyen los eventos sísmicos registrados en el conjunto de datos. Para representar una línea de tendencia y que sea más agradable la visualización de la distribución se utiliza el método de estimación de la densidad de kernel (KDE) dentro de la función de creación del histograma, aplicado como *kde = True*. Al tener un histograma en intervalos discretos, con el número de observaciones por intervalo, las barras aparecen escalonadas. El KDE suaviza la distribución alrededor de cada punto de manera que produce una línea continua que representa la densidad de probabilidad subyacente y mejora la visualización.

- **Diagrama de dispersión entre magnitud y profundidad**: se utilizó ***sns.scatterplot()*** para observar la relación entre la magnitud y la profundidad de manera gráfica.

3.3.2. Análisis geoespacial

El análisis geoespacial proporciona información sobre la distribución y localización geográfica de los seísmos registrados. Se utilizaron dos formas de representar los seísmos:

- **Gráfico de puntos sobre el mapa del mundo:** se utilizó la biblioteca Geopandas para cargar los límites del mapamundi y se generó un gráfico de puntos utilizando ***sns.scatterplot()*** para visualizar la distribución geográfica de los seísmos. La magnitud de los seísmos se representó mediante un gradiente de colores con el parámetro *hue*, para facilitar la identificación de las áreas geográficas con mayor intensidad sísmica.

  Geopandas (Wasserman, J., 2021) es una biblioteca útil para tareas relacionadas con los sistemas de información geográfica (SIG), análisis espacial y visualización de datos geoespaciales. Se utiliza la función ***gpd.read_file()*** para cargar los datos geoespaciales del mapamundi y dentro de ella se usa el parámetro ***gpd.datasets.get_path('naturalearth_lowres')*** para proporcionar la ruta al archivo *shapefile* del mapamundi incluido en los datasets de Geopandas.

  ```
  import geopandas as gpd

  # Cargar los límites mapamundi usando geopandas
  mapamundi = gpd.read_file(gpd.datasets.get_path('naturalearth_lowres'))
  ```

  Después con la función ***boundary.plot()*** se agregan los límites del mapamundi al gráfico de puntos creado con Seaborn que contiene las coordenadas de los seísmos, y esto permite visualizar las fronteras del mapamundi y los puntos con las localizaciones de los seísmos, con el gradiente de color según la magnitud del seísmo en ese punto.

- **Mapa interactivo con Folium:** se utilizó la biblioteca Folium para crear un mapa interactivo donde cada punto es un seísmo igual que en el anterior gráfico, pero al hacer click o pasar el ratón por encima en pantalla se muestra información relevante sobre el seísmo. En este trabajo se ha decidido mostrar información sobre la magnitud, profundidad y lugar del seísmo.

  Folium (Folium, 2024) es una biblioteca que permite crear mapas interactivos y visualizaciones geoespaciales utilizando Leaflet.js, una biblioteca de JavaScript de código abierto para mapas interactivos. Con la función ***folium.Map()*** se crea el mapa interactivo y con la función ***folium.Marker()*** se crean los marcadores de cada seísmo para mostrar la información.

3.3.3. Análisis temporal

Este análisis se centra en la frecuencia y distribución temporal de los seísmos. Se realizaron los siguientes gráficos:

- **Gráfico de frecuencia de los seísmos por mes:** se calculó la frecuencia de seísmos por mes utilizando las funciones ***resample('D', on = 'time').size***. Esto agrupa los datos de la columna *time* por día y calcula el tamaño de cada grupo, es decir, el número de registros por día. Después se genera un gráfico de líneas para visualizar la variación durante todo el año 2023.

- **Descomposición de serie temporal:** se utilizó la biblioteca Statsmodels (Statsmodels, 2024), para descomponer la serie temporal en los componentes de tendencia, estacionalidad y residuos. Mediante la función ***seasonal_decompose()*** se consigue descomponer la serie. En este caso asumimos un modelo aditivo mediante el parámetro ***model = 'additive'***, que asume que la serie temporal es la suma de la tendencia, estacionalidad y los residuos.

```
from statsmodels.tsa.seasonal import seasonal_decompose

# Descomposición de la serie temporal
series = seasonal_decompose(frecuencia_mensual, model='additive', period=12)
```

  Los componentes de la serie permiten entender la estructura y patrones subyacentes. La tendencia muestra la dirección en la que se mueven los datos a largo plazo. La estacionalidad muestra los patrones recurrentes o periódicos de la serie temporal. Los residuos son la parte de la serie temporal que no se explican por los otros dos componentes y representan la variabilidad aleatoria.

3.3.4. Análisis de las estaciones de detección sísmica

Con las variables que contiene el conjunto de datos, también se pudo hacer un análisis de las estaciones que detectan y registran los seísmos. Para ello se hizo lo siguiente:

- **Distribución de las estaciones de detección por ubicación y magnitud:** las columnas *locationSource* y *magSource* contienen códigos que representan las fuentes que registraron la ubicación y magnitud de los seísmos, respectivamente. Estos códigos son estándares utilizados por distintas organizaciones y redes sísmicas de todo el planeta para identificar la fuente de información. Los códigos que encontramos en este conjunto de datos son:

  - **us:** United States Geological Survey (USGS)

  - **pr:** Puerto Rico Seismic Network

- **nc:** Northern California Seismic System
- **tx:** University of Texas at Austin
- **ci:** Caltech/USGS Southern California Seismic Network
- **uu:** University of Utah Seismograph Stations
- **se:** European-Mediterranean Seismological Centre (EMSC)
- **nm:** New Madrid Seismic Zone
- **uw:** University of Washington, Pacific Northwest Seismic Network

Se crearon gráficos de barras utilizando ***sns.countplot()*** para visualizar la distribución del número de estaciones de detección sísmica según las fuentes que informan de la ubicación y la magnitud del seísmo.

- **Tablas con el número de seísmos detectados**: se crearon varias tablas para comprobar los seísmos con el mayor número de fuentes que informaron sobre la localización y magnitud del evento sísmico, y también para ver los que tienen el menor número de fuentes informando del evento.

- **Gráfico dispersión entre la magnitud y las variables Nst y magNst**: se utilizaron gráficos de dispersión con ***sns.scatterplot()*** para ver la relación entre la magnitud de los seísmos y el número de estaciones sísmicas que detectaron los seísmos, tanto su localización como su magnitud.

3.3.5. Análisis de los errores de medición

En este análisis se exploraron los errores asociados a las mediciones de los seísmos, incluyendo errores horizontales, de profundidad y de magnitud. Los pasos que se siguieron fueron:

- **Estadísticas descriptivas de los errores**: se mostraron estadísticas descriptivas utilizando ***.describe()*** para conocer la media, desviación estándar, mínimo y máximo, así como los cuantiles de los errores.

- **Histograma de los errores:** se crearon histogramas utilizando ***sns.histplot()*** para visualizar la distribución de los errores horizontales, de profundidad y de magnitud. Estos histogramas son iguales a los realizados en el análisis de la magnitud y profundidad utilizando el método de estimación de la densidad de kernel (KDE) para mostrar la línea de tendencia.

- **Matriz de correlación de errores:** se calculó la matriz de correlación entre los errores utilizando ***sns.heatmap().***

3.4. Modelado

En esta sección se explicarán los diferentes modelos de machine learning y deep learning que se han utilizado para predecir la magnitud de los seísmos.

3.3.1. Modelo de regresión lineal

Los modelos de regresión lineal son uno de los modelos más simples y utilizados para predecir variables continuas. En el contexto de la predicción de la magnitud de los seísmos, estos modelos buscan establecer una relación lineal entre un conjunto de variables predictoras y la variable respuesta. La ecuación general de estos modelos tiene la forma de:

$$y = \beta_0 + \beta_1 x + \varepsilon$$

donde:

y es la variable respuesta (dependiente),

x es la variable predictora (independiente),

β_0 es la ordenada en el origen,

β_1 es la pendiente de la línea de regresión,

ε es el error que muestra la variabilidad no explicada por el modelo.

El objetivo es encontrar la mejor línea recta que se ajuste a los datos observados, estimando los parámetros β_0 y β_1 mediante el método de los mínimos cuadrados. Este método encuentra los valores de los coeficientes que minimizan la suma de los cuadrados de los residuos, es decir, la diferencia entre los valores observados y los predichos por el modelo (James et al., 2013).

Para predecir la magnitud de los seísmos en este modelo y para el resto de los modelos que se emplearán en este trabajo, se han utilizado como variables predictoras las columnas *time, latitude, longitude* y *depth*, y como variable respuesta la columna *mag.*

Para poder utilizar la variable *time* previamente se ha tenido que transformar en variable numérica. Para ello se ha creado una serie que almacena los valores numéricos resultantes de la conversión de la columna *time*. La conversión se realiza utilizando el método ***.astype(int)*** que convierte los valores en números enteros:

```
time_numeric = pd.Series(df['time'].astype(int))

df['time_numeric'] = time_numeric
```

Antes de entrenar el modelo, es necesario definir las variables independientes (*x*) y la dependiente (*y*) y dividir los datos en conjuntos de entrenamiento y test. Esta preparación ya quedará hecha para todos los modelos utilizados en adelante. Definimos las variables y después dividimos los datos, importamos el modelo de regresión lineal y la métrica que se utilizará para evaluar los modelos mediante las siguientes bibliotecas de Scikit-Learn:

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
```

Scikit-Learn, también conocida como sklearn, es una biblioteca de aprendizaje automático que se usa en programación con Python (Scikit-Learn, 2024). Esta biblioteca proporciona numerosos algoritmos de aprendizaje automático, además de herramientas para el preprocesamiento de datos y la evaluación de los modelos.

La división de los datos en los conjuntos de entrenamiento y prueba se realiza con la función ***train_test_split*** donde asignamos un 80% del conjunto de datos para entrenamiento y un 20% para test. El parámetro ***random_state*** se utiliza para garantizar la reproducibilidad de los datos:

```
# Dividir los datos en conjuntos de entrenamiento y prueba (80% entrenamiento, 20% prueba)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Después de la división de los datos, se crea el modelo de regresión lineal utilizando ***LinearRegressión()***. El modelo se entrena utilizando los datos de entrenamiento mediante el método ***fit()***. El modelo ajusta los coeficientes para minimizar la suma de los cuadrados de las diferencias entre los valores predichos y los valores reales:

```
# Entrenar un modelo de regresión lineal
modelo_reg_lineal = LinearRegression()
modelo_reg_lineal.fit(X_train, y_train)
```

A continuación, se realizan las predicciones utilizando el conjunto de prueba (test) mediante el método ***predict()***:

```
# Realizar predicciones en el conjunto de prueba
predicciones_reg_lineal = modelo_reg_lineal.predict(X_test)
```

Finalmente, se calcula el **error cuadrático medio** (**MSE**) para evaluar el rendimiento del modelo de regresión lineal. Esta métrica de evaluación se utilizará para comparar todos los modelos ya que tiene una fácil interpretación, y se calcula como la media o promedio de los errores al cuadrado entre las predicciones del modelo y los valores reales. Cuánto más alto sea el valor de esta métrica peor será el modelo, ya que indicará que las predicciones del modelo están lejos de los valores reales promedio. Por el contrario, valores bajos indican que el modelo tiene mayor capacidad para predecir los valores de la variable objetivo.

3.3.2. Modelo KNN (K-Nearest Neighbors)

Los modelos KNN (K-Nearest Neighbors) son algoritmos de aprendizaje supervisado que se utilizan para problemas de clasificación y regresión. Estos modelos se basan en categorizar puntos cercanos en el espacio con valores o etiquetas similares. En el caso de la predicción de la magnitud de seísmos, se utilizan modelos KNN de regresión que estiman el valor de la magnitud basándose en el promedio de los valores de los *K* vecinos más cercanos en el espacio.

A la hora de implementar el modelo, se debe elegir el valor de *K*. Un valor bajo de este parámetro puede resultar en un modelo con menor sesgo al capturar patrones más complejos y detallados, pero también puede haber sobreajuste, mientras que valores altos proporciona más estabilidad al modelo, pero también aumenta el sesgo. La elección de este parámetro es importante para el rendimiento del modelo. Después se calcula la distancia para cada valor del conjunto de entrenamiento. Las distancias más comunes son la distancia euclidiana y la de Manhattan. Una vez se ha calculado la distancia, se identifican los *K* vecinos más cercanos y se estima el promedio de los valores (Javatpoint, 2024).

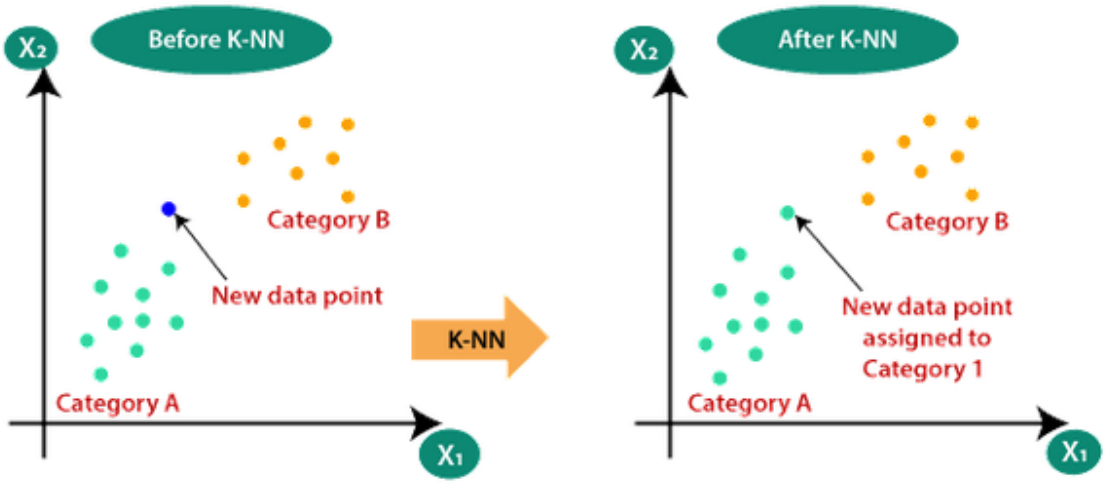


Figura 5- Diagrama sobre la aplicación del modelo KNN en un conjunto de datos. Extraído de https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning

Como se comentó previamente, la división de los datos en conjuntos de entrenamiento y test ya se ha realizado cuando se implementó el modelo de regresión lineal. A partir de aquí, lo que se hace es importar el estimador ***KNeighborsRegressor()*** para crear el modelo y definir los parámetros para el número de vecinos con ***n_neighbors***. Los siguientes pasos son iguales a lo visto en el anterior modelo, se entrena el modelo con el conjunto de entrenamiento a través de la función ***.fit()*** y se realizan las predicciones con el conjunto de prueba (test), para después calcular el MSE del modelo.

La diferencia en este proceso con respecto al anterior modelo, a parte del estimador que se usa para crear el modelo, es la creación de un objeto ***GridSearchCV*** para buscar el mejor valor de *K* mediante validación cruzada. Esto realiza una búsqueda exhaustiva de los mejores hiperparámetros para obtener el modelo con la mejor métrica de MSE.

3.3.3. Modelo de Random Forest

Los modelos de Random Forest son unos algoritmos de aprendizaje supervisado que se basan en la idea de construir árboles de decisión durante el entrenamiento y combinar las predicciones de todos para obtener una predicción global más precisa. Cada árbol de decisión individual se entrena con muestras aleatorias de los datos del conjunto de entrenamiento y se realizan predicciones independientes. Con las predicciones de cada árbol, se hace un promedio para obtener la predicción final (Chaya, 2020).

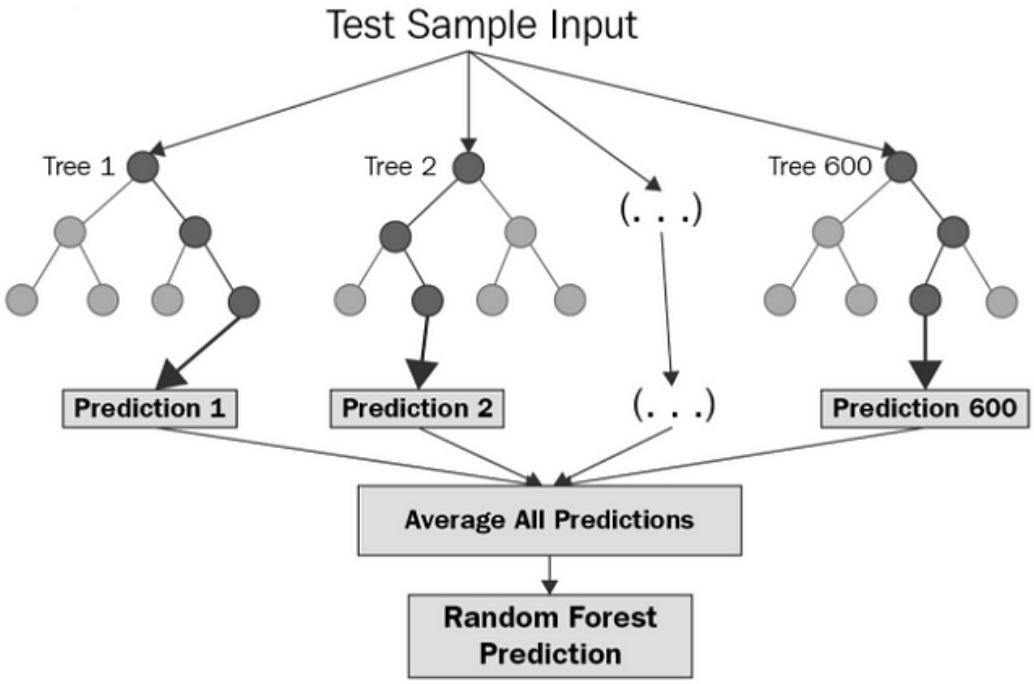


Figura 6- Diagrama sobre la estructura de un modelo de Random Forest con los diferentes árboles de decisión. Extraído de https://levelup.gitconnected.com/random-forest-regression-209c0f354c84

Al igual que se comentó anteriormente, el conjunto de datos ya se dividió previamente en entrenamiento y test. Por tanto, para implementar este modelo lo que hay que hacer es importar la clase necesaria de Scikit-learn para crear el modelo y definir el rango máximo de valores del parámetro profundidad del árbol, que controla la profundidad máxima que alcanza cada árbol en la búsqueda. Este parámetro es ***max_depth*.** Se importa el objeto ***RandomForestRegressor*** para crear el modelo y se utiliza ***GridSearchCV*** para realizar la búsqueda de los mejores hiperparámetros del modelo. En concreto, se busca el mejor valor de ***max_depth***. A continuación, se entrena el modelo con los datos de entrenamiento utilizando el método ***.fit()*** y posteriormente se elige el mejor estimador con el método ***best_estimator_*.** Se utiliza este estimador para probar el conjunto de prueba (test) y calcular el MSE.

3.3.4. Modelo XGBoost

El algoritmo que da nombre a estos modelos, *XGBoost* (*Extreme Gradient Boosting*), es una técnica de ensamblado llamada *boosting* que construye árboles de decisión de manera secuencial y en cada iteración el nuevo modelo intenta corregir los errores del modelo anterior, lo que permite mejorar gradualmente la precisión del modelo final (Chen et al., 2016).

Para crear y optimizar el modelo de regresión se importa la biblioteca ***xgboost***, ***GridSearchCV*** para la búsqueda de los mejores hiperparámetros y ***mean_squared_error*** para el cálculo del MSE. Los hiperparámetros que vamos a probar para optimizar el modelo son:

- ***max_depth***: profundidad máxima de cada árbol de decisión que se usa en el modelo.

- ***learning_rate***: la tasa de aprendizaje que controla la contribución de cada árbol al modelo final.

- ***n_estimators***: el número de árboles a utilizar en el modelo.

A continuación se crea un objeto que contiene el modelo con ***XGBRegressor()***, y se crea otro objeto con ***GridSearchCV*** que busca los mejores hiperparámetros para el modelo utilizando validación cruzada. Después se ajusta el modelo con el método ***fit()*** pasando los datos de entrenamiento, y se obtiene el mejor modelo seleccionando el estimador óptimo utilizando ***best_estimator_***. Por último, se realizan las predicciones sobre el conjunto de prueba con el método .***predict()*** y se calcula el error cuadrático medio (MSE) comparando las predicciones con los valores reales.

Como podemos observar, para todos los modelos de regresión utilizados hasta el momento se han seguido los mismos pasos:

a. Importación de las bibliotecas necesarias.

b. Creación del modelo.

c. Optimización de los hiperparámetros del modelo.

d. Entrenamiento del modelo.

e. Selección del mejor estimador.

f. Predicciones con el conjunto de prueba (test).

g. Cálculo del MSE.

Las diferencias son el tipo de modelo que se usan, y la configuración de los parámetros asociados al modelo, estableciendo a su vez distintos valores.

3.3.5. Modelo de red neuronal

Como se comentó en la introducción, los modelos de redes neuronales profundas son una poderosa herramienta para crear modelos predictivos. Estos modelos están diseñados para imitar la forma de procesar y transmitir la información por las neuronas biológicas, permitiendo crear un sistema capaz de aprender y tomar decisiones. La red neuronal está compuesta por capas de neuronas interconectadas que realizan una operación matemática con los datos de entrada y transmiten la información a través de conexiones ponderadas a la capa siguiente, así sucesivamente hasta llegar a la capa de salida de la red.

Para usar un modelo de red neuronal para la predicción de la magnitud de los seísmos se ha decidido implementarlo utilizando TensorFlow y Keras. TensorFlow es una plataforma de código abierto desarrollada por Google para implementar modelos de aprendizaje automático y redes neuronales (TensorFlow, 2024), mientras que Keras es una biblioteca que proporciona una interfaz fácil de usar para construir los modelos de redes neuronales (Keras, 2024). Para implementar el modelo se importan ambas de la siguiente manera:

```
import tensorflow as tf
from tensorflow import keras
```

Antes de empezar con la definición de la arquitectura del modelo, los datos de entrada, tanto los de entrenamiento como los de prueba (test) se normalizan para mejorar su rendimiento en los modelos. Esto se hace utilizando ***StandardScaler*** de Scikit-learn:

```
from sklearn.preprocessing import StandardScaler

# Escalar los datos para mejorar el rendimiento de la red neuronal
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

A continuación, se define la arquitectura de la red neuronal utilizando la API Sequential de Keras. La función ***.Sequential()*** permite crear un modelo vacío al que se le van agregando diferentes capas de forma secuencial. Cada capa procesa la salida de la capa anterior. En el modelo que se implementa en este trabajo, se ha optado por utilizar tres capas densas (*fully connected*):

- La primera capa densa contiene 64 neuronas, y se utiliza la función de activación ReLU (Rectified Linear Unit). La función de activación ReLU es una función no lineal que ayuda a aprender representaciones complejas de los datos.

- La segunda capa densa contiene 32 neuronas y también utiliza la función de activación ReLU.

- La tercera y última capa contiene una sola neurona, y es la capa de salida del modelo. Como estamos tratando un problema de predicción de la magnitud de los seísmos se espera una única salida numérica.

```
# Definir la arquitectura de la red neuronal
model = keras.Sequential([
    keras.layers.Dense(64, activation='relu', input_shape=[X_train.shape[1]]),
    keras.layers.Dense(32, activation='relu'),
    keras.layers.Dense(1)
])
```

Después de definir la arquitectura de la red neuronal, se compila utilizando la función ***.compile()***. Con esta función se configura el proceso de entrenamiento del modelo. En este caso, hemos usado el parámetro *Adam* como optimizador y el MSE como la medida de la función de pérdida. El optimizador es el algoritmo que ajusta los pesos de las neuronas durante el entrenamiento y ajusta automáticamente la tasa de aprendizaje y la función de pérdida es la medida de discrepancia entre las predicciones que hace el modelo y los valores reales durante el entrenamiento.

```
# Compilar el modelo
model.compile(optimizer='adam', loss='mean_squared_error')
```

A continuación ya se puede entrenar el modelo, y para ello se utiliza como en el resto de modelos anteriores la función ***.fit()***. Para el entrenamiento se utilizan los datos escalados (*X_train*_scaled) y las etiquetas correspondientes (*y_train*), que son los valores reales que el modelo intentará predecir. El parámetro ***epochs*** especifica el número de épocas, es decir, el número de veces que el modelo pasará por todo el conjunto de datos de

entrenamiento. El parámetro ***batch_size*** indica el número de muestras que se utilizan para actualizar los pesos. Al ejecutar el código, el modelo será entrenado durante el número de épocas que se especifique y en cada época el modelo ajustará sus pesos para minimizar la función de pérdida, en este caso el MSE.

Por último, se utiliza el conjunto de prueba para determinar el rendimiento del modelo. Como en los modelos anteriores, se utiliza la función ***.predict()*** y se calcula el MSE del modelo.

3.3.6. Modelo de red neuronal recurrente (RNN) – LSTM

Las redes neuronales recurrentes (RNNs) están teniendo relevancia a la hora de modelar datos secuenciales, como el procesamiento del habla o el lenguaje. A diferencia de las redes neuronales convencionales, las RNNs procesan la secuencia de entrada elemento por elemento, manteniendo en las capas ocultas un vector que contiene implícitamente información sobre el historial de todos los elementos anteriores de la secuencia. Es decir, la salida no solo depende de la entrada, también de la información anterior. A pesar de la capacidad que tienen estas redes, las RNNs tienen dificultad para aprender y retener información a largo plazo, por lo que se han desarrollado otro tipo de modelos como las redes de memoria a corto y largo plazo (LSTM). Esta variante es capaz de aprender dependencias temporales a largo plazo en datos secuenciales y mantener o actualizar el estado de la memoria interna durante un período de tiempo largo (Lecun et al., 2015).

Es por ello, que el último modelo que se implementará en este trabajó será un modelo LSTM que estará definido de manera similar al anterior, pero variando la primera capa densa por una capa LSTM. Las tres capas que contendrán son las siguientes:

- Capa LSTM: es la capa principal y en este caso se utilizarán 64 unidades LSTM y la función de activación ReLU. Esta capa también recibe la forma de entrada de los datos de entrenamiento, en este caso *(X_train_scaled[1], 1)*, que indica que por cada paso de tiempo se representa una observación y se tiene una sola característica por observación.

- Capa intermedia: es una capa densa con 32 neuronas o unidades y también utiliza la función ReLU.

- Capa de salida: es la capa densa final con una sola unidad, como en el anterior modelo.

```python
from keras.models import Sequential
from keras.layers import LSTM, Dense

# Definir la arquitectura del modelo LSTM
modelo_lstm = Sequential()
modelo_lstm.add(LSTM(units=64, activation='relu', input_shape=(X_train_scaled.shape[1], 1)))
modelo_lstm.add(Dense(units=32, activation='relu'))
modelo_lstm.add(Dense(units=1))
```

El resto del código es igual que el anterior modelo, se compila el modelo utilizando el optimizador Adam, se entrena el modelo con ***.fit()*** y se hacen las predicciones y se calcula el MSE. La diferencia en estos pasos es que se ajustan los datos de entrenamiento y también los de test para tener la forma de 'número de muestras, número de pasos de tiempo, número de características' que es el formato necesario para el modelo LSTM.

4. Resultados

En este apartado se analizarán los resultados obtenidos en el análisis exploratorio de los datos (EDA) y los resultados de los modelos de machine learning y de deep learning utilizados para predecir la magnitud de los seísmos.

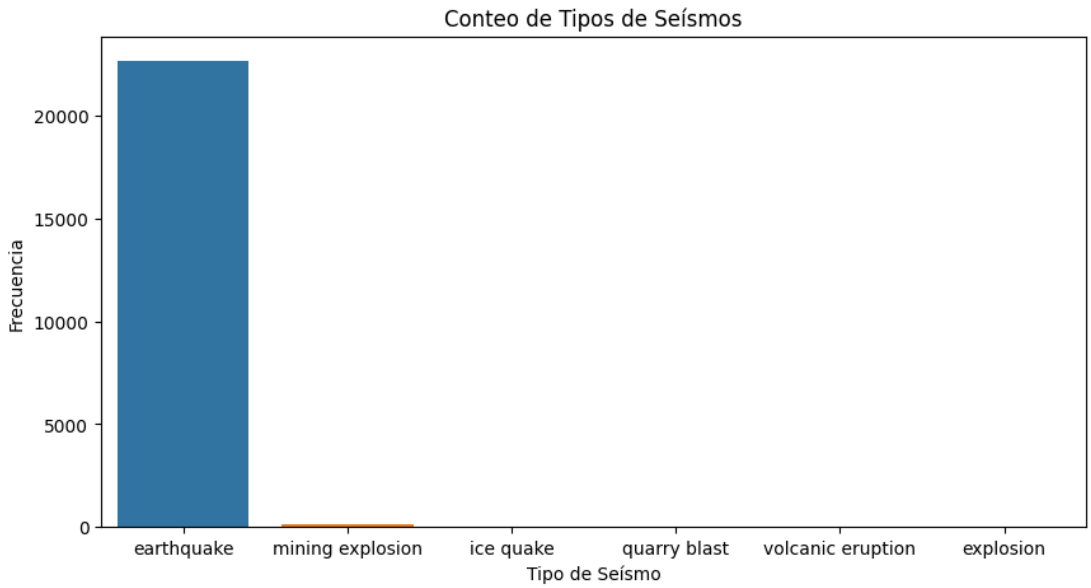
4.1. Resultados del Análisis Exploratorio de los Datos (EDA)

En primer lugar, se examinarán las gráficas y demás resultados obtenidos en el análisis preliminar de los datos con el fin de conocer mejor las variables que componen este conjunto de datos sobre información sísmica.

4.1.1. Resultados del análisis de la magnitud, profundidad y tipo de seísmos

Como se explicó en el apartado de materiales y métodos, comenzamos analizando el tipo de seísmos. Se utilizó la función ***.unique()*** para averiguar en el conjunto de datos los diferentes tipos de seísmo (los tipos de seísmos están explicados en el apartado 3.3.1). Es importante saber el tipo de seísmo que produce el evento, ya que no es lo mismo un seísmo causado de forma natural que uno causado por actividad humana. El primero se produce de manera aleatoria, mientras que el segundo puede conocerse cuando va a ocurrir. Por tanto, lo primero que se quiso averiguar es la frecuencia de cada tipo de seísmo en nuestro conjunto de datos, mostrado en el siguiente gráfico de barras:

La casi totalidad de los eventos sísmicos registrados en este conjunto de datos se corresponden con seísmos naturales (*earthquake*), causados por la liberación de energía



en la corteza terrestre. Un pequeño porcentaje se corresponden a seísmos causados por una explosión minera (*mining* explosión) y del resto de seísmo no se puede apreciar si existen eventos mirando únicamente la gráfica. Para saber el número exacto de eventos de cada tipo de seísmo se realizó también una tabla mostrando las frecuencias:

	Type	Frecuencia
0	earthquake	22691
1	mining explosion	159
2	volcanic eruption	13
3	ice quake	1
4	quarry blast	1
5	explosion	1

Esta tabla nos muestra que existen 22.691 eventos sísmicos causados por seísmos naturales y 159 causados por una explosión en minería. Del resto era imposible apreciar si existían registros tan solo con observar el gráfico de frecuencias; vemos que existen 13 registros de seísmos causados por una erupción volcánica y tan solo un registro del resto de tipos de seísmos.

Entre los objetivos de este trabajo está el de encontrar los modelos con mejor capacidad predictiva, por ello se ha optado por prescindir de todos los tipos de seísmos que no sean producidos de manera natural (tipo *earthquake*) al tener poca incidencia en la totalidad de los datos y como en el caso de los producidos por una explosión en minería, tampoco tienen el componente de aleatoriedad. Antes de eliminarlos para continuar los análisis, se ha decidido explorar que datos contienen y sacar algunas conclusiones.

Entre los eventos de tipo *'mining explosion'* tenemos los siguientes registros:

```
df.loc[df['type']=='mining explosion']
```

	time	latitude	longitude	depth	mag	magType	nst	gap	dmin	rms	...	updated	place	type
115	2023-01-02 19:02:17.679000+00:00	43.8095	-105.2956	0.0	3.3	ml	14.0	83.0	0.962	0.49	...	2023-03-11T22:51:33.040Z	17 km ENE of Wright, Wyoming	mining explosion
245	2023-01-04 18:04:26.544000+00:00	47.5257	-92.5739	0.0	2.8	ml	9.0	106.0	1.155	1.00	...	2023-03-11T22:51:50.040Z	0 km SSE of Parkville, Minnesota	mining explosion
246	2023-01-04 18:59:38.592000+00:00	43.6984	-105.3634	0.0	3.4	ml	18.0	71.0	1.048	0.82	...	2023-03-11T22:51:39.040Z	11 km ESE of Wright, Wyoming	mining explosion
324	2023-01-06 18:25:44.843000+00:00	47.5429	-92.7329	0.0	3.0	ml	25.0	79.0	0.928	1.00	...	2023-03-11T22:51:43.040Z	3 km N of Kinney, Minnesota	mining explosion
335	2023-01-06 21:34:44.291000+00:00	43.7374	-105.2726	0.0	3.4	ml	22.0	71.0	0.972	0.72	...	2023-03-11T22:51:43.040Z	Wyoming	mining explosion
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
15567	2023-10-16 21:00:29.905000+00:00	45.0540	-106.6866	0.0	2.9	ml	19.0	104.0	1.667	0.91	...	2023-12-26T21:24:26.040Z	32 km SSW of Birney, Montana	mining explosion
15611	2023-10-17 21:12:56.104000+00:00	43.6162	-105.1993	0.0	3.1	ml	26.0	98.0	0.981	1.22	...	2023-12-26T21:24:47.040Z	27 km ESE of Wright, Wyoming	mining explosion
15661	2023-10-18 19:04:01.715000+00:00	45.0737	-106.7599	0.0	3.0	ml	50.0	67.0	1.658	0.86	...	2023-12-26T21:24:34.040Z	33 km SW of Birney, Montana	mining explosion
15666	2023-10-18 20:57:19.050000+00:00	43.7789	-105.3012	0.0	3.2	ml	29.0	60.0	0.976	0.54	...	2023-12-26T21:24:35.040Z	15 km E of Wright, Wyoming	mining explosion
15761	2023-10-20 17:50:58.724000+00:00	44.0455	-105.3857	0.0	3.2	ml	28.0	55.0	0.976	0.75	...	2023-12-26T21:24:48.040Z	21 km SSE of Antelope Valley-Crestview, Wyoming	mining explosion

159 rows × 22 columns

Si nos fijamos en la columna *place*, tenemos varios registros con localización en Wright, Wyoming. Buscando información sobre esta localización, en este lugar existe una mina de carbón, por lo que los datos son congruentes con el tipo de seísmo.

Del mismo modo, para los eventos de tipo '*volcanic eruption*':

```
df.loc[df['type']=='volcanic eruption']
```

	time	latitude	longitude	depth	mag	magType	nst	gap	dmin	rms	...	updated	place	type
13860	2023-09-14 15:48:26.701000+00:00	15.4692	146.1768	10.000	4.5	mb	13.0	194.0	0.439	0.78	...	2023-11-18T22:29:31.040Z	53 km ENE of Saipan, Northern Mariana Islands	volcanic eruption
13883	2023-09-15 04:26:31.134000+00:00	15.4105	145.4988	10.000	4.2	mb	13.0	102.0	0.332	0.60	...	2023-11-18T22:29:31.040Z	35 km NW of Saipan, Northern Mariana Islands	volcanic eruption
13890	2023-09-15 07:25:40.633000+00:00	15.4528	145.9570	10.289	4.4	mb	19.0	120.0	2.132	0.73	...	2023-11-18T22:29:31.040Z	34 km NE of Saipan, Northern Mariana Islands	volcanic eruption
13891	2023-09-15 07:28:50.217000+00:00	15.4575	145.7466	10.405	4.5	mb	19.0	121.0	0.227	0.64	...	2023-11-18T22:29:31.040Z	27 km N of Saipan, Northern Mariana Islands	volcanic eruption
13892	2023-09-15 07:32:33.601000+00:00	15.6131	145.8842	9.440	4.5	mb	19.0	82.0	0.387	0.52	...	2023-11-18T22:29:31.040Z	46 km NNE of Saipan, Northern Mariana Islands	volcanic eruption
24089	2023-09-14 15:48:26.701000+00:00	15.4692	146.1768	10.000	4.5	mb	13.0	194.0	0.439	0.78	...	2023-11-18T22:29:31.040Z	53 km ENE of Saipan, Northern Mariana Islands	volcanic eruption
24100	2023-09-15 04:26:47.195000+00:00	15.4943	145.8433	4.874	4.7	mb	56.0	139.0	0.264	0.51	...	2023-11-18T22:29:30.040Z	32 km NNE of Saipan, Northern Mariana Islands	volcanic eruption
24101	2023-09-15 04:28:47.562000+00:00	15.4540	145.7736	9.921	4.7	mb	25.0	152.0	2.051	0.99	...	2023-11-18T22:29:31.040Z	26 km N of Saipan, Northern Mariana Islands	volcanic eruption
24102	2023-09-15 06:19:55.523000+00:00	15.5082	145.5699	9.966	4.6	mb	15.0	95.0	2.025	0.43	...	2023-11-18T22:29:31.040Z	38 km NNW of Saipan, Northern Mariana Islands	volcanic eruption
24103	2023-09-15 06:22:28.475000+00:00	15.5257	145.6122	8.736	4.6	mb	23.0	120.0	0.338	0.66	...	2023-11-18T22:29:31.040Z	37 km NNW of Saipan, Northern Mariana Islands	volcanic eruption
24104	2023-09-15 07:28:50.217000+00:00	15.4575	145.7466	10.405	4.5	mb	19.0	121.0	0.227	0.64	...	2023-11-18T22:29:31.040Z	27 km N of Saipan, Northern Mariana Islands	volcanic eruption
24105	2023-09-15 07:32:33.601000+00:00	15.6131	145.8842	9.440	4.5	mb	19.0	82.0	0.387	0.52	...	2023-11-18T22:29:31.040Z	46 km NNE of Saipan, Northern Mariana Islands	volcanic eruption
24106	2023-09-15 08:00:48.677000+00:00	15.5466	145.5517	10.000	4.8	mb	33.0	112.0	0.388	0.73	...	2023-11-18T22:29:31.040Z	42 km NNW of Saipan, Northern Mariana Islands	volcanic eruption

13 rows × 22 columns

Tenemos todos los eventos localizados en las Islas Marianas del Norte, unas islas de origen volcánico, con volcanes activos actualmente. La magnitud de estos seísmos también es mayor que los registros de los seísmos de tipo '*mining explosion*', lo que también es congruente dado que las explosiones volcánicas deberían ser más fuertes. Para el resto de los tipos de seísmos, solo existe un registro:

```
df.loc[df['type']=='ice quake']
```

	time	latitude	longitude	depth	mag	magType	nst	gap	dmin	rms	...	updated	place	type	horiz
9836	2023-06-27 16:31:12.169000+00:00	58.4586	-133.1326	1.0	2.8	ml	10.0	182.0	0.753	0.46	...	2023-09-08T14:18:33.040Z	77 km ENE of Juneau, Alaska	ice quake	

1 rows × 22 columns

```
df.loc[df['type']=='quarry blast']
```

	time	latitude	longitude	depth	mag	magType	nst	gap	dmin	rms	...	updated	place	type	h
11531	2023-07-29 21:29:00.690000+00:00	49.667833	-114.801167	-2.0	3.07	ml	27.0	234.0	0.2872	0.26	...	2023-07-30T13:47:01.540Z	British Columbia, Canada	quarry blast	

1 rows × 22 columns

```
df.loc[df['type']=='explosion']
```

	time	latitude	longitude	depth	mag	magType	nst	gap	dmin	rms	...	updated	place	type
18778	2023-12-14 20:37:21.790000+00:00	44.261667	-120.885333	-1.6	2.77	ml	17.0	148.0	0.1846	0.11	...	2023-12-29T17:58:16.015Z	5 km SW of Prineville, Oregon	explosion

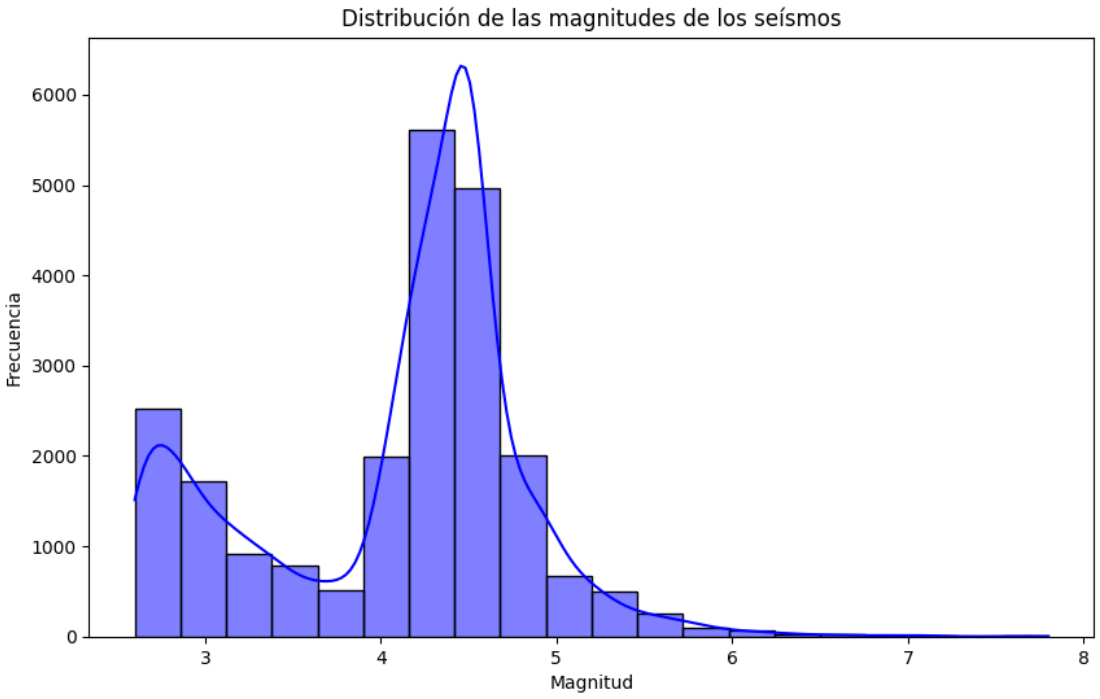
1 rows × 22 columns

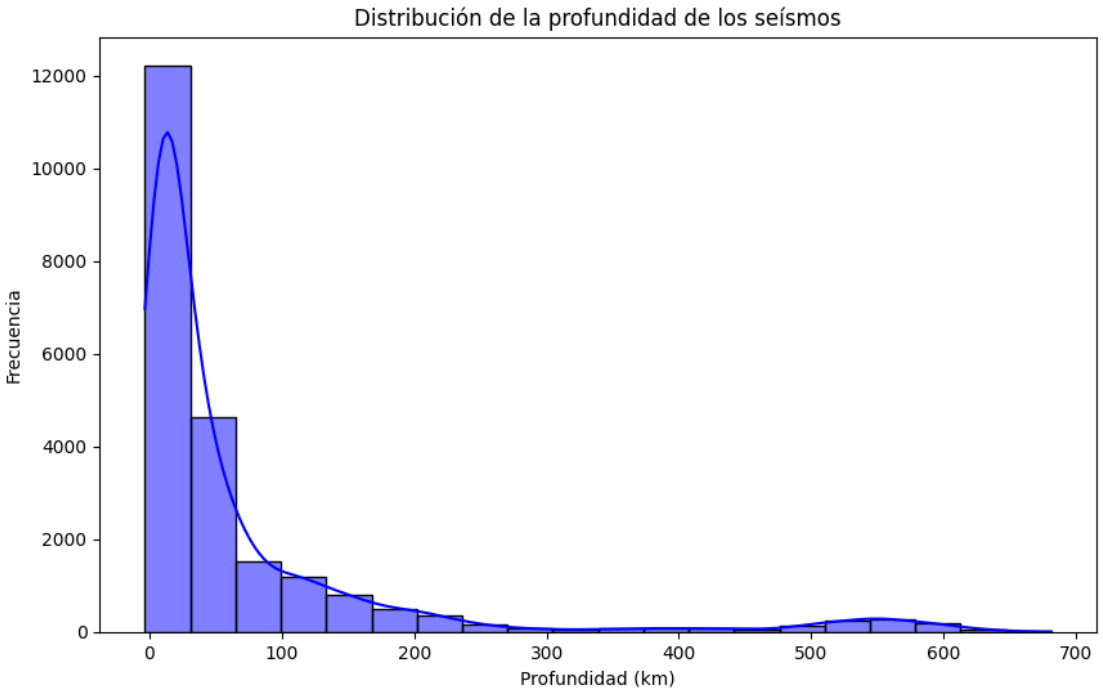
Podemos observar que la localización del evento registrado como temblor de hielo (‘*ice quake*’) ocurrió en Alaska, una zona donde existen numerosos glaciares que ocupan un extenso territorio. Los otros dos registros de tipo ‘*quarry blast*’ y ‘*explosion*’ no se puede saber con certeza la causa pero tienen acción humana.

Explorado el tipo de seísmos de los eventos registrados, nos quedamos únicamente con los ‘*earthquake*’ como se ha explicado:

```
df = df.loc[df['type'] == 'earthquake']
```

Lo siguiente que se quiso explorar, fue la distribución de las magnitudes y la profundidad de los seísmos. Esto se muestra en los siguientes histogramas:

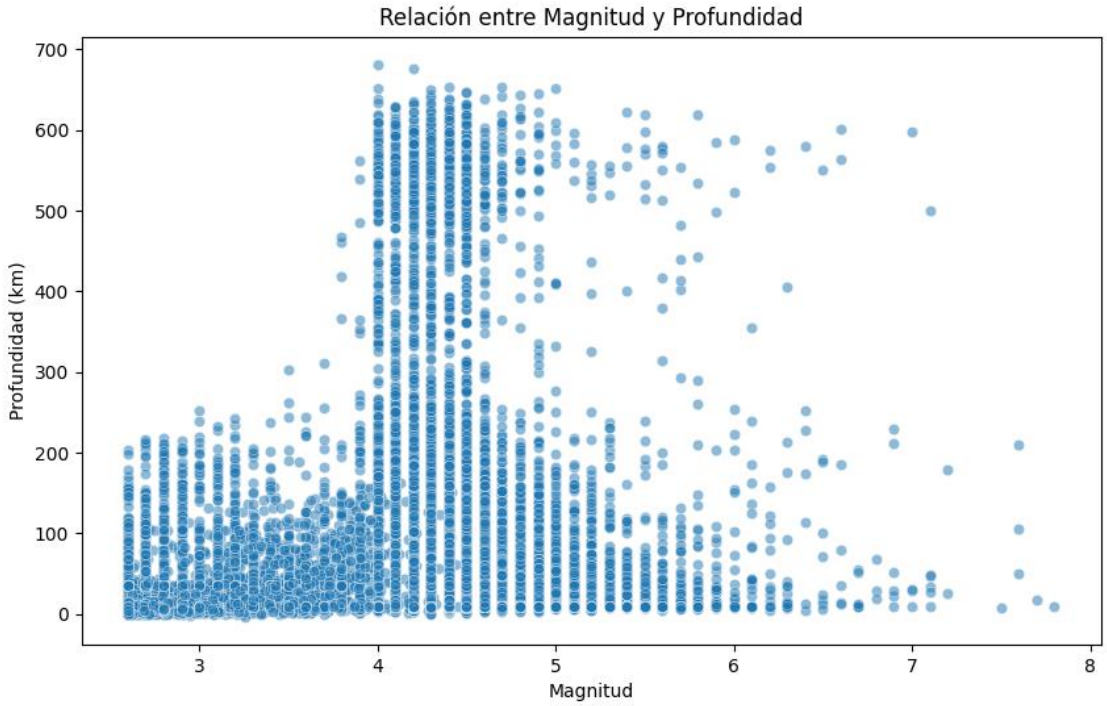




En el gráfico de la distribución de las magnitudes, tenemos que la mayoría de los eventos sísmicos ocurrieron con magnitudes entre 4 y 5. También tenemos otro pico en magnitudes inferiores a 3, mientras que los seísmos con magnitudes por encima de 5-6 fueron poco frecuentes. Hay que tener en cuenta que los eventos sísmicos de gran intensidad son los que mayores destrozos ocasionan, pero afortunadamente no son tan frecuentes, y la mayoría tienen una intensidad intermedia.

En el gráfico de la distribución de la profundidad de los seísmos, la mayoría ocurren a menos de 100 km de la superficie terrestre. Esto puede deberse al grosor de la corteza terrestre, que puede variar entre 5 km en la corteza terrestre submarina hasta los 75 km en la corteza terrestre del Himalaya.

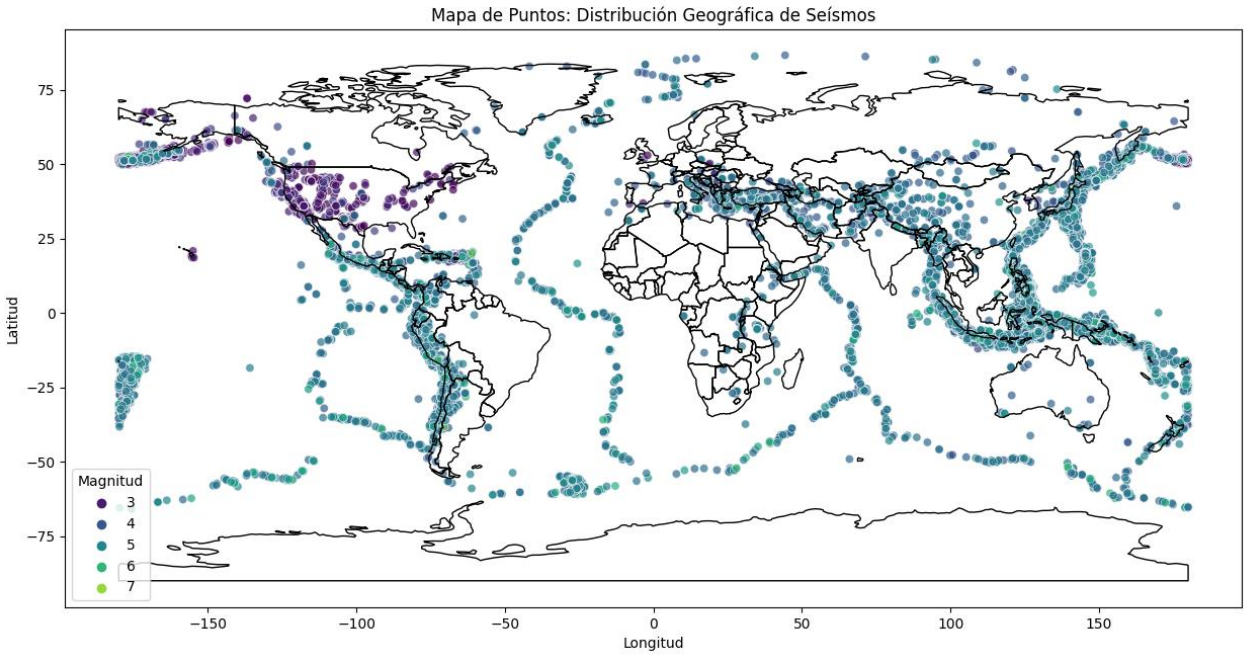
Para ver la relación entre las dos variables, se creó un diagrama de dispersión:



Este gráfico nos está indicando que los seísmos con magnitudes entre 4 y 5, que eran los mayoritarios, se producen desde zonas poco profundas hasta zonas muy profundas de hasta casi 700 km. Mientras que la mayoría de los seísmos con magnitudes por debajo de 4 y por encima de 5 suelen producirse en zonas poco profundas, de menos de 200 km. Cabe destacar que no existen eventos sísmicos con profundidades por encima de los 300 km y magnitudes inferiores a 3-3.5, posiblemente debido a la intensidad tan baja no se llegan a detectar.

4.1.2. Resultados del análisis geoespacial

Con el fin de conocer la distribución geográfica de los eventos sísmicos se realizaron dos gráficos que utilizan el mapa geográfico de la Tierra como fondo donde se sitúan los puntos de ocurrencias de los seísmos, mediante las coordenadas geográficas conseguidas con las variables *latitude* y *longitude*. En el punto 3.3.2. se explicaron las bibliotecas utilizadas para cargar los límites del mapamundi y como se mostrarían los datos. El primer mapa utilizado para mostrar las localizaciones es el siguiente:



Para entender los resultados de estos gráficos hay que tener conocimiento sobre la tectónica de placas. Si comparamos el gráfico de la distribución de los seísmos con la imagen que muestra las placas terrestres, se puede apreciar una correlación entre la distribución de los seísmos por los límites entre las placas. Esto se debe a que en estas zonas ocurren fenómenos de subducción entre placas, fallas o zonas de ensanchamiento como las dorsales oceánicas que dan como resultado el desplazamiento de la corteza por encima del manto ocasionando los fenómenos sismológicos.

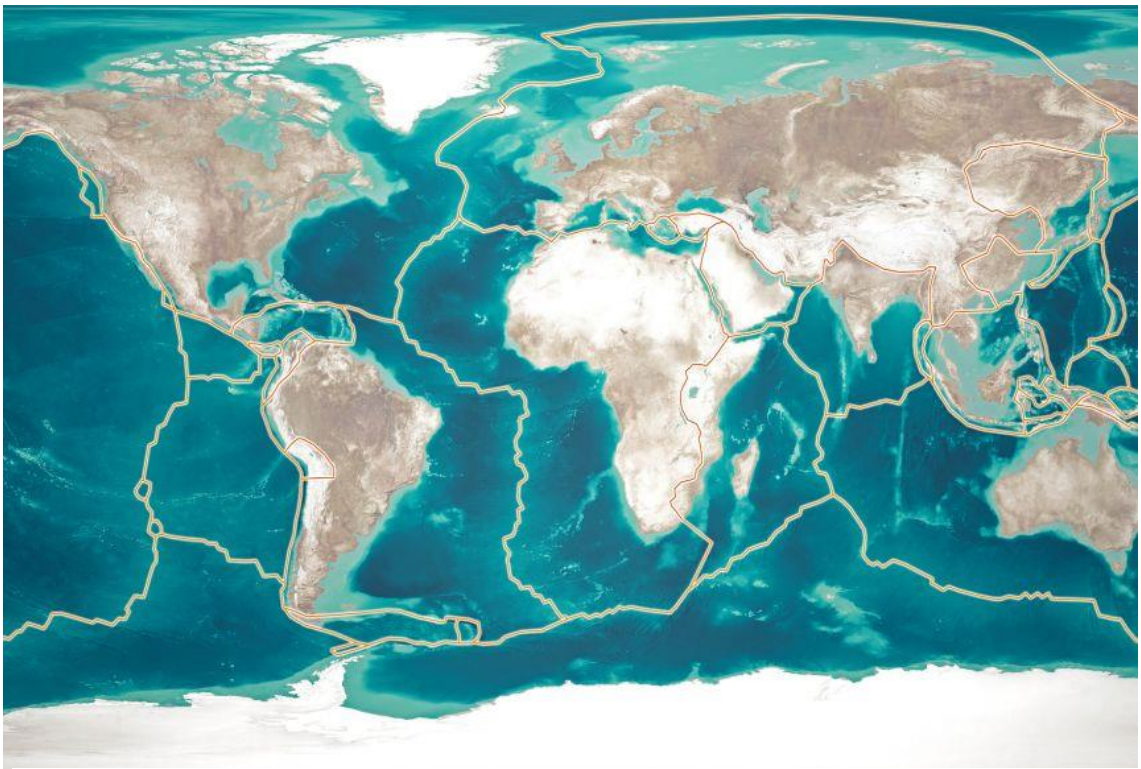
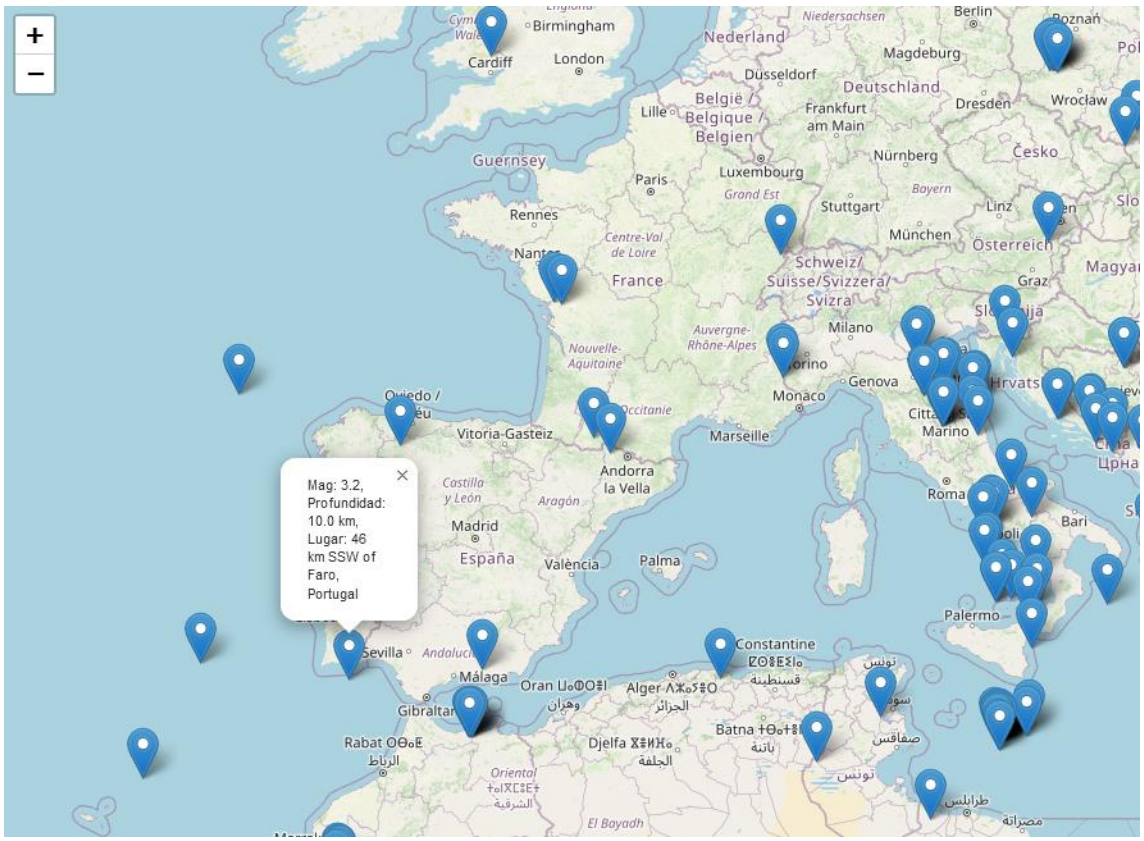


Figura 7- Límites de las placas que forman la corteza terrestre. Extraído de: https://education.nationalgeographic.org/resource/plate-boundaries/

Existe una gran concentración de seísmos por la zona cercana a Japón y las islas Filipinas, y también en cordilleras importantes como la del Himalaya. Estas zonas tienen una fuerte influencia de choque de placas lo que produce frecuentes eventos sísmicos. La predicción de los seísmos en estas zonas es crucial para prevenir a la ciudadanía.

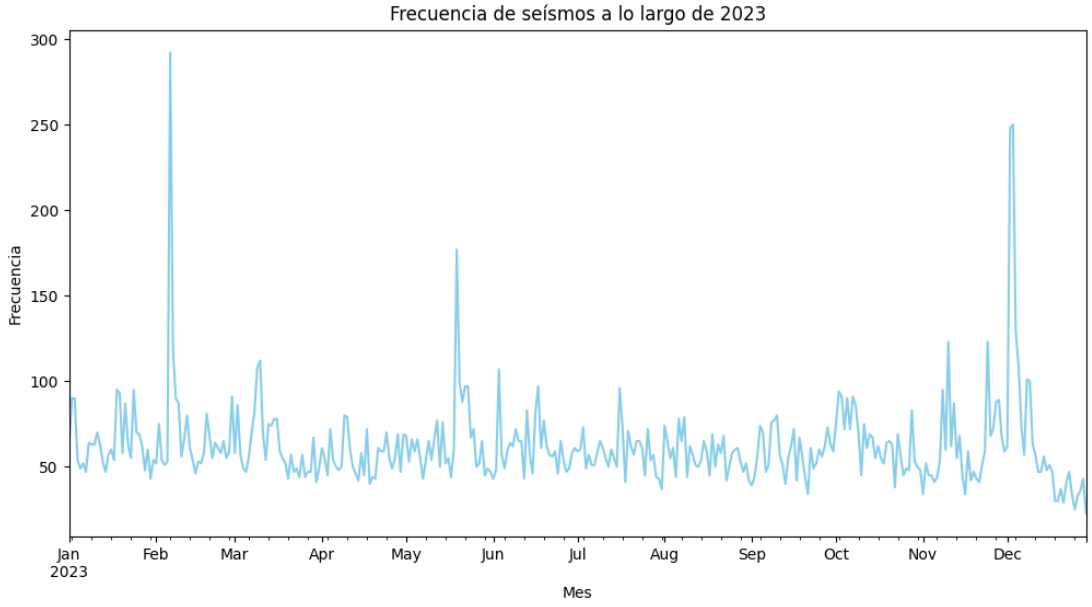
El siguiente mapa muestra los resultados de manera similar al anterior, pero utilizando la librería Folium, que permite tener una interacción en el mapa. Se ha querido mostrar ya que proporciona información útil de cada evento. En este caso se ha decidido mostrar información sobre la magnitud, profundidad y el lugar donde ocurrió el seísmo. En el ejemplo mostrado, aparece la información de un seísmo ocurrido cerca de Faro, Portugal, con una magnitud de 3.2 y una profundidad de 10 km.



Estos mapas ayudan a explorar las zonas donde se producen más seísmos, y dan una idea general acerca de la distribución y las zonas donde se requiere una especial atención a estos fenómenos.

4.1.3. Resultados del análisis temporal

Hasta ahora hemos visto como se distribuyen los seísmos según su magnitud, profundidad, tipo de seísmo, y su localización geográfica, pero no hemos analizado su distribución temporal. El conjunto de datos contiene información únicamente del año 2023. Lo primero que hacemos es mostrar la frecuencia de los seísmos a lo largo de este período de tiempo:



Se pueden distinguir tres picos en el gráfico, siendo el más acusado uno a principios de febrero, con una frecuencia de casi 300 seísmos en ese espacio corto de tiempo. Después observamos otro pico de casi 200 seísmos en un período corto a mitad de mayo, y otro pico a principios de diciembre donde se alcanzaron 250 seísmos.

Podemos observar también que la frecuencia media de seísmos a lo largo del año está entre 50 y 100, es decir, en períodos cortos como días suele haber una frecuencia de seísmos de entre 50 y 100.

A continuación, se realizó una descomposición de series temporales mediante el método de descomposición estacional. Este método separa la serie temporal en cuatro componentes:

- Serie temporal original: es el gráfico de frecuencias que hemos analizado arriba, con ningún tipo de transformación.

- Gráfico de tendencia: muestra la dirección general en la que se mueven los datos a lo largo del tiempo.

- Gráfico de estacionalidad: muestra la componente estacional, es decir, representa patrones repetitivos o cíclicos que ocurren en intervalos regulares de tiempo, como estaciones, meses, días…

- Gráfico de residuos: representa las fluctuaciones aleatorias o errores que no pueden explicarse ni por la tendencia ni por la estacionalidad.

El gráfico de tendencia no muestra una tendencia clara, aparecen zonas con aumento que se corresponde con los picos donde había más frecuencia de seísmos, pero podemos decir que la tendencia es más o menos estable a lo largo del año, con estas fases altas más impredecibles.

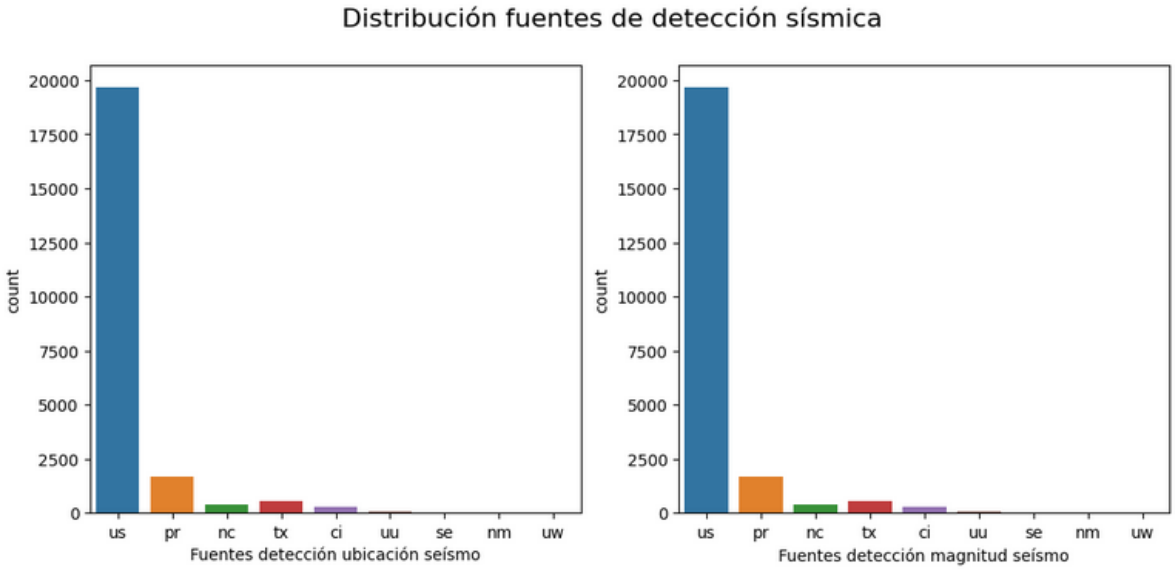
En cuanto a la estacionalidad, vemos que hay un componente estacional corto, de un período de días, como se comentó en la gráfica de frecuencias, la mayoría de los días se tienen entre 50 y 100 seísmos.

Por último, el gráfico de residuos muestra que están distribuidos de manera uniforme en torno a cero, lo que indica que el modelo de descomposición es adecuado y se ha capturado la estructura temporal subyacente.

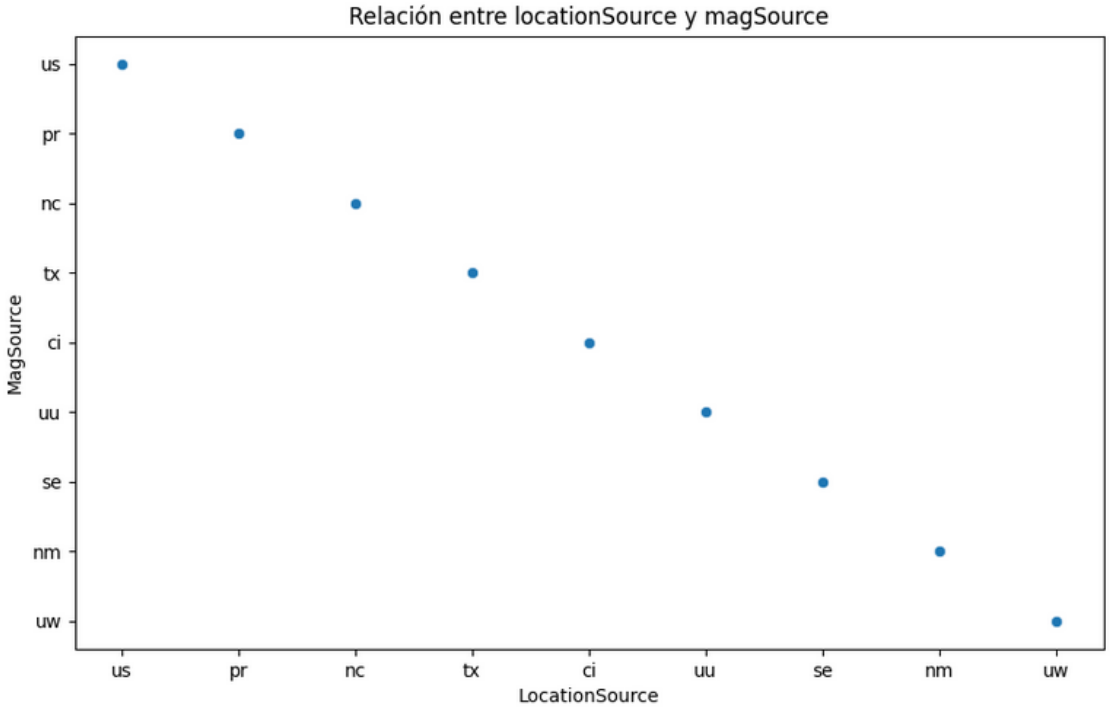
4.1.4. Resultados del análisis de las estaciones de detección sísmica

En primer lugar, se analizó la distribución de las fuentes que se usaron en la detección de los seísmos. En el apartado 3.3.4. se explicaron los códigos utilizados por las distintas organizaciones y redes sísmicas de todo el planeta para diferenciar e identificar las fuentes que informan de los seísmos.

En el conjunto de datos existen dos variables que muestran las fuentes en función de si informan sobre la localización o la magnitud del seísmo, que son *locationSource* y *magSource* respectivamente. Ambas variables contienen las mismas categorías, por lo que se decidió explorar la frecuencia de cada categoría. Los dos gráficos siguientes muestran la distribución de las fuentes de detección sísmica de cada variable:



Se puede observar que ambas variables contienen la misma frecuencia de datos, es decir, las fuentes que reportan sobre la magnitud de los seísmos también reportan sobre la localización de los seísmos. También se hizo un gráfico de dispersión para ver la correlación entre las dos variables y efectivamente, tienen una correlación lineal:



A continuación, se pasó a analizar el número de estaciones sísmicas que detectaron la localización y la magnitud de los seísmos. Esto se corresponde con las variables *nst* y *magNst*. En este caso, por cada evento sísmico puede haber varias estaciones que detecten tanto la localización como la magnitud. Lo que se hizo fue mostrar mediante las siguientes tablas los seísmos que fueron detectados por más estaciones y los que fueron detectados por menos. Como se explicó en el apartado 3.2 de limpieza de datos, se eliminaron las filas que contenían valores 0 en estas variables, ya que no puede haber seísmos que no hayan sido detectados por ninguna estación.

La tabla con los eventos sísmicos que fueron detectados por más estaciones de detección sísmica sobre la localización del seísmo fueron los siguientes:

	date	depth	mag	nst	magNst	place
24678	2023-10-11	8.0	6.3	423.0	49.0	24 km NNW of Herāt, Afghanistan
23104	2023-07-16	25.0	7.2	410.0	156.0	106 km S of Sand Point, Alaska
26553	2023-12-23	29.0	5.8	410.0	47.0	115 km SSE of Vilyuchinsk, Russia
24943	2023-10-26	10.0	5.9	388.0	99.0	143 km E of Ust'-Kamchatsk Staryy, Russia
21762	2023-05-10	210.0	7.6	380.0	32.0	82 km WNW of Hihifo, Tonga

Vemos que el evento 24.678 que ocurrió en Afganistán fue detectado por 423 estaciones sísmicas, y a su vez también hubo 49 estaciones que detectaron la magnitud de este seísmo que fue de 6.3.

Los eventos sísmicos que fueron detectados por menos estaciones sobre su localización fueron:

	date	depth	mag	nst	magNst	place
6498	2023-04-27	4.85	2.74	3.0	2.0	Puerto Rico region
13288	2023-09-03	7.68	3.34	3.0	3.0	64 km NE of Cruz Bay, U.S. Virgin Islands
9974	2023-06-30	90.00	3.84	3.0	2.0	Puerto Rico region
19186	2023-12-28	80.33	3.27	3.0	4.0	66 km E of Cruz Bay, U.S. Virgin Islands
4326	2023-03-15	5.72	2.67	3.0	2.0	Puerto Rico region

En este caso tan solo hubo 3 estaciones de detección que localizaron a estos eventos sísmicos. Si nos fijamos se corresponden con seísmos ocurridos en torno a las islas de Puerto Rico y de las islas Vírgenes. Quizás el número de estaciones de detección en estas zonas es menor a otras zonas que se encuentren en superficie continental.

La siguiente tabla muestra los seísmos cuya magnitud fue detectada por más estaciones de detección sísmica:

	date	depth	mag	nst	magNst	place
25190	2023-11-08	77.731	5.6	126.0	884.0	Kuril Islands
21138	2023-04-05	560.074	5.1	128.0	814.0	156 km SSW of Tarauacá, Brazil
24731	2023-10-13	168.306	5.1	214.0	796.0	30 km WSW of Imabetsu, Japan
20901	2023-03-20	215.062	4.9	135.0	678.0	111 km NNW of Nikolski, Alaska
21546	2023-04-28	587.560	6.0	113.0	676.0	south of the Fiji Islands

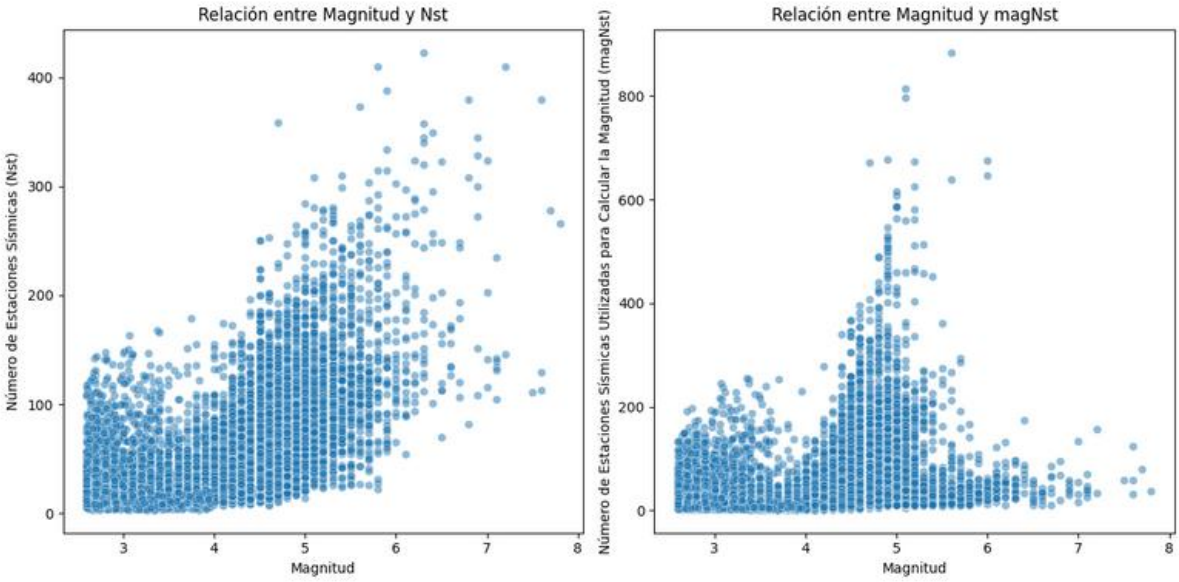
En este caso tenemos seísmos como el que ocurrió en las islas Kuril cuya magnitud fue detectada por 884 estaciones, mientras que su localización la detectaron 126 estaciones. Esto puede indicar que en esta zona existen varias estaciones con detectores de magnitud pero que no están capacitadas para saber la localización del seísmo.

Por último, los seísmos cuya magnitud fue detectada por menos estaciones sísmicas fueron:

	date	depth	mag	nst	magNst	place
13608	2023-09-09	86.355	3.9	9.0	1.0	92 km S of La Tirana, Chile
2972	2023-02-20	99.369	4.1	16.0	1.0	108 km S of Calama, Chile
10753	2023-07-15	10.000	4.1	27.0	1.0	North Atlantic Ocean
2146	2023-02-06	10.000	4.2	19.0	1.0	10 km SSW of Çelikhan, Turkey
18926	2023-12-19	10.000	3.7	10.0	1.0	7 km WNW of San Marcos, Costa Rica

Tan solo hizo falta una estación de detección para saber la magnitud de estos seísmos, pero se usaron varias para saber su localización.

Para comprender un poco más sobre las estaciones de detección, se realizaron diagramas de dispersión para ver la relación entre las estaciones y la magnitud de los seísmos:



En el gráfico de la izquierda, se puede intuir una tendencia creciente entre el número de estaciones que detectan la localización de los seísmos y la magnitud de estos. Es decir, para los seísmos que tienen una magnitud mayor, hay más estaciones de detección sísmica que informan de la localización del seísmo.

En el gráfico de la derecha, no observamos una tendencia clara, pero se aprecia un pico en torno a los seísmos de magnitud 5, que son los que más estaciones de detección pueden alcanzar para informar sobre la magnitud de estos. Tenemos que para seísmos en torno a magnitudes de 3 se necesitan como mucho 200 estaciones de detección para informar sobre la magnitud, mientras que para seísmos en torno a magnitudes de 5 se necesitan hasta 500 estaciones de detección para informar.

4.1.5. Resultados del análisis de los errores en la medición

El último análisis exploratorio que se llevó a cabo fue el de las variables asociadas a los errores de medición: *horizontalError, depthError* y *magError*.

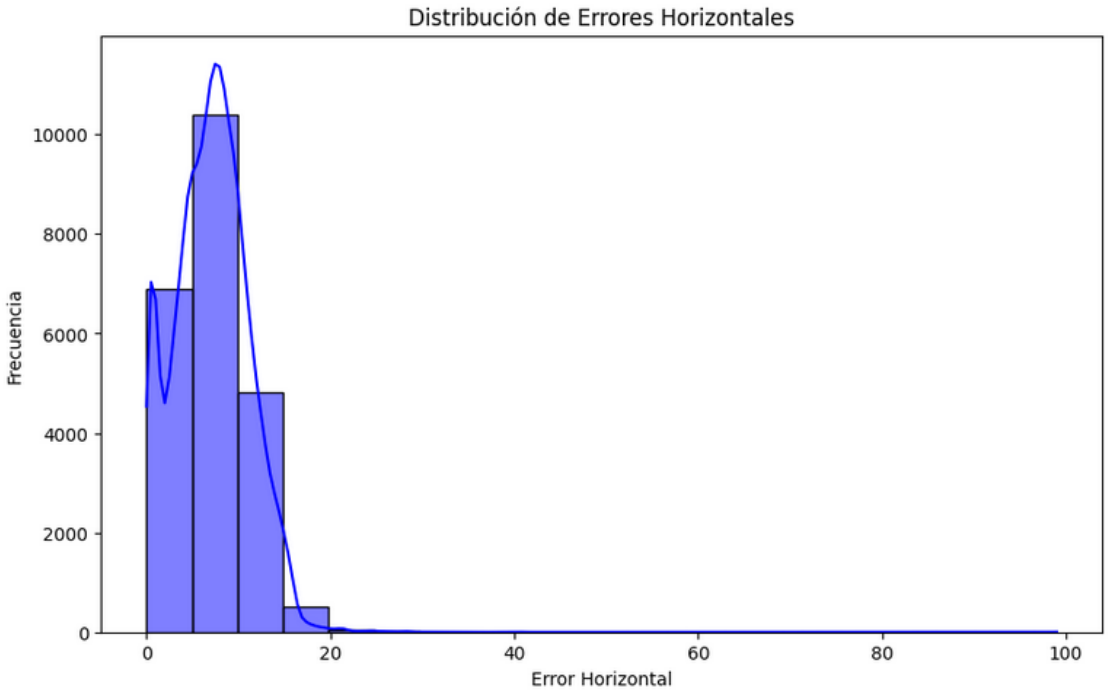
Lo primero que se hizo fue sacar estadísticas descriptivas:

```
# Estadísticas descriptivas de las columnas relacionadas con errores
errores = df[['horizontalError', 'depthError', 'magError']].describe()
print(errores.round(3))
```

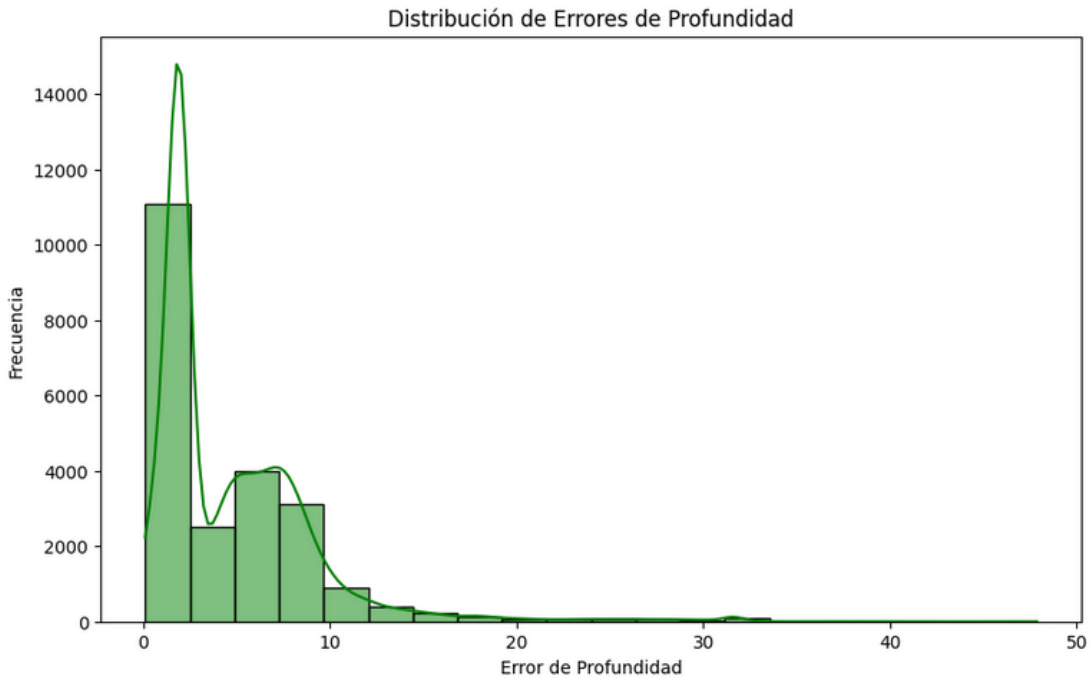
```
       horizontalError  depthError   magError
count        22691.000   22691.000  22691.000
mean             7.118       4.791      0.119
std              4.004       4.464      0.057
min              0.000       0.100      0.000
25%              4.300       1.881      0.080
50%              7.150       2.961      0.109
75%              9.750       6.962      0.148
max             99.000      47.878      0.770
```

Podemos observar que la variable del error horizontal es la que mayor media tiene, por tanto, este tipo de error es el que más se ha cometido a la hora de la detección, seguido del error en la medición de la profundidad. El error que tiene una media muy baja es el error cometido al medir la magnitud, por lo que los sistemas de detección son más precisos a la hora de medir la magnitud que la profundidad. También el error de magnitud tiene una desviación típica muy baja comparada con la que tienen los otros tipos de error.

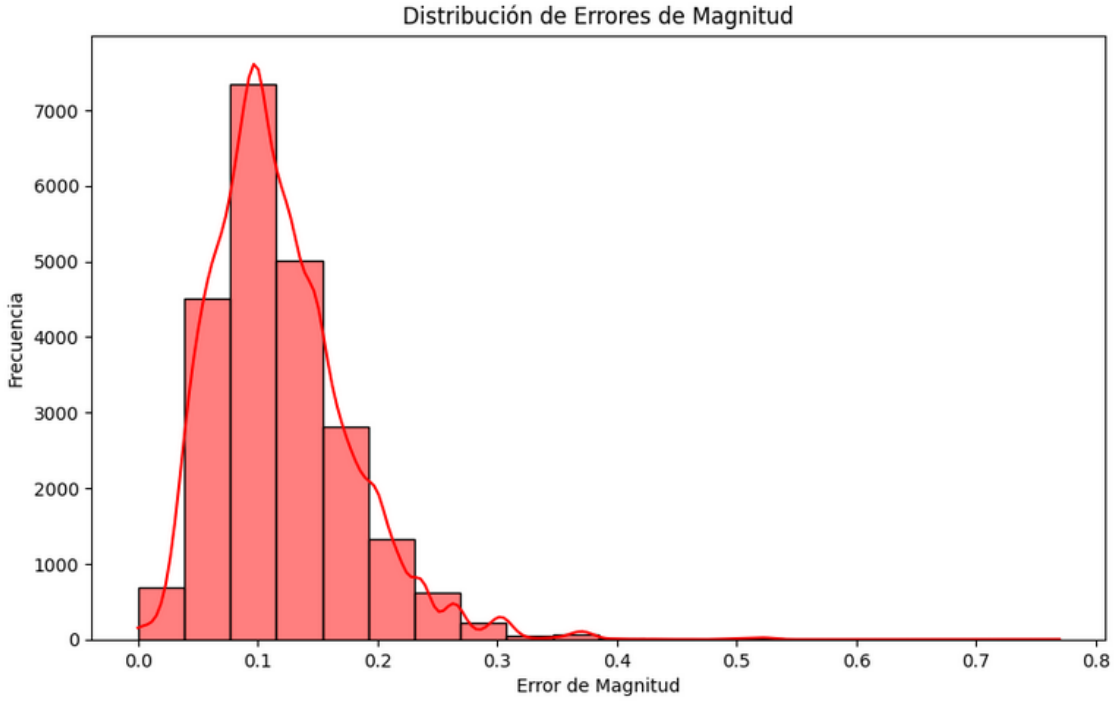
Esto se puede apreciar también en los gráficos de distribución de los errores:



En el gráfico de la distribución de los errores horizontales, vemos que la frecuencia de errores es muy alta, por tanto, este tipo de medición podría no ser muy fiable.



El gráfico de distribución de los errores en la medición de la profundidad también arroja frecuencias altas cuando el error tiene valores elevados, pero el pico se sitúa en errores más cercanos a 0, por tanto esta medición es más precisa que la anterior.

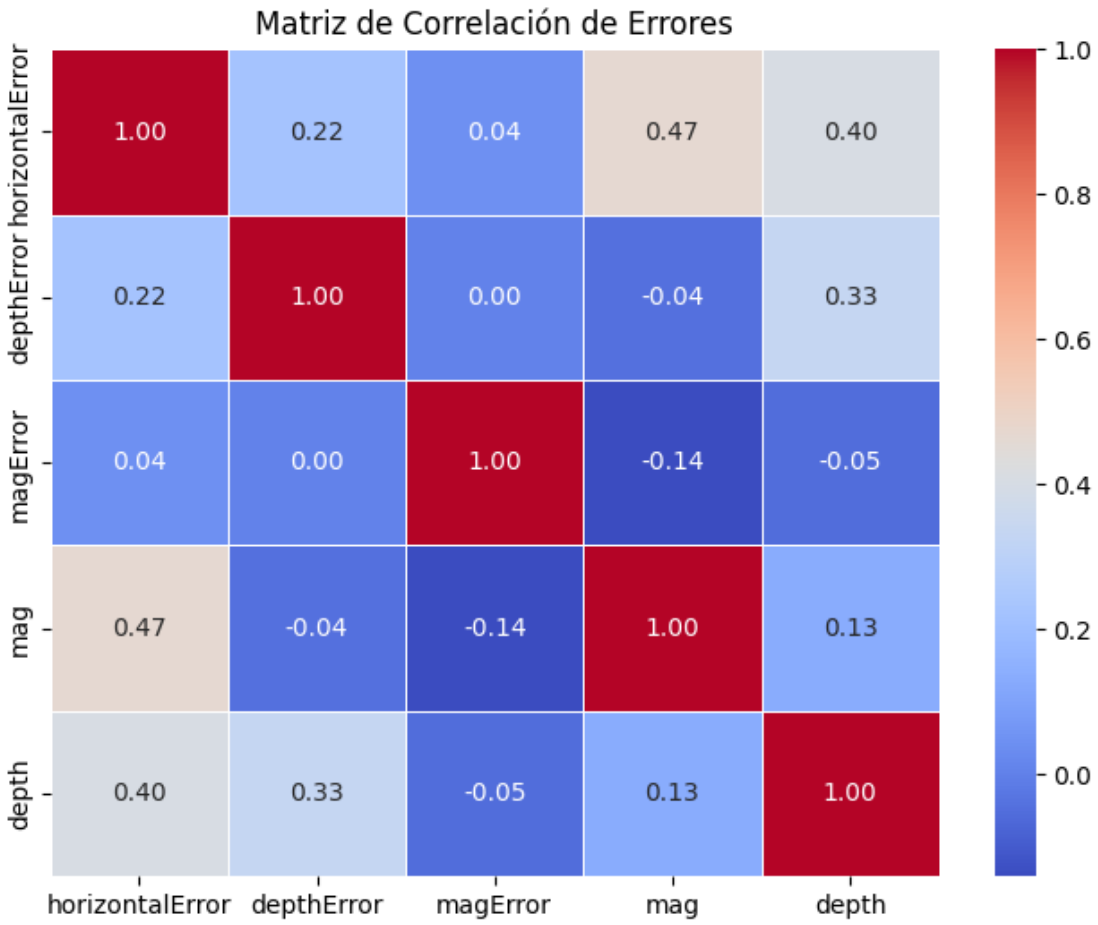


Por último, el gráfico de distribución de los errores en la medición de la magnitud muestra que este tipo de medición es la más fiable de todas, ya que la mayoría de los datos tienen valores en torno a 0.1, lo cual es un valor muy bajo y bastante fiable si tenemos en cuenta los anteriores.

Para terminar con este análisis exploratorio, se hizo una matriz de correlación de las variables de medición y las variables de los errores asociadas a las mediciones.

```
# Matriz de correlación
matriz_correlacion = df[['horizontalError', 'depthError', 'magError', 'mag', 'depth']].corr()

# Visualización de la matriz de correlación
plt.figure(figsize=(8, 6))
sns.heatmap(matriz_correlacion, annot=True, cmap='coolwarm', fmt=".2f", linewidths=0.5)
plt.title('Matriz de Correlación de Errores')
plt.show()
```



De esta matriz podemos sacar varias conclusiones:

- El error en la medición de la magnitud (*magError*) es el que mejores resultados ha dado anteriormente, siendo por tanto la magnitud de los seísmos la variable más fiable en cuanto a medición. El coeficiente de correlación entre la magnitud y el error asociado es de -0.14. Esto quiere decir que tiene correlación negativa, sugiriendo que, al aumentar la magnitud de los seísmos, el error en la medición es menor. Por tanto, cuanto más intenso son los seísmos y mayor es su magnitud, menor es el error asociado.

- El error en la medición de la profundidad (*depthError*) no parecía del todo fiable, sin embargo, el coeficiente de correlación del error y la variable profundidad (*depth*) es de 0.33, lo que indica una correlación directa. Esto sugiere que a medida que aumenta la profundidad también aumenta el error, pero de una manera moderada, ya que el coeficiente no es muy alto.

- El error en la medición horizontal no tiene una variable directa asociada, pero si lo comparamos con la magnitud y la profundidad vemos que los coeficientes de correlación son 0.47 y 0.40, respectivamente. Por tanto, este tipo de error es bastante alto y poco fiable.

4.2. Resultados del modelado

En esta sección se tratará de explicar los resultados que arrojan los diferentes modelos de machine learning y deep learning que se han utilizado para predecir la magnitud de los seísmos.

4.2.1. Resultados del modelo de regresión lineal

La definición y creación del modelo de regresión lineal se explica en el apartado 3.3.1 de este trabajo. Al ser el primer modelo, se dio una descripción completa y detallada de los pasos a seguir, además de no necesitar probar ningún parámetro, por lo que en este apartado únicamente se mostrará el código completo utilizado y el resultado del MSE. El código es el siguiente:

```python
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

# Dividir los datos en características (X) y la variable objetivo (y)
X = df[['time_numeric', 'latitude', 'longitude', 'depth']]  # Características
y = df['mag']  # Variable objetivo

# Dividir los datos en conjuntos de entrenamiento y prueba (80% entrenamiento, 20% prueba)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Entrenar un modelo de regresión lineal
modelo_reg_lineal = LinearRegression()
modelo_reg_lineal.fit(X_train, y_train)

# Realizar predicciones en el conjunto de prueba
predicciones_reg_lineal = modelo_reg_lineal.predict(X_test)

# Calcular el error cuadrático medio (MSE) en el conjunto de prueba
mse_reg_lineal = mean_squared_error(y_test, predicciones_reg_lineal)
print("Error Cuadrático Medio (Regresión Lineal):", mse_reg_lineal)
```

```
Error Cuadrático Medio (Regresión Lineal): 0.3252362286788169
```

Como se puede apreciar en la salida del código, el error cuadrático medio (MSE) después del entrenamiento y test fue de **0.32**. Este valor es algo elevado y cómo veremos a continuación habrá modelos que tengan un error más bajo.

Los modelos de regresión lineal asumen una relación lineal entre las variables predictoras y la variable objetivo, pero en este caso no estamos tratando con un problema lineal y la eficacia del modelo se queda corta.

4.2.2. Resultados del modelo KNN (K-Nearest Neighbors)

El modelo KNN fue explicado en el apartado 3.3.2 de este trabajo. Se explicó la importancia de elegir correctamente el valor de *K*, utilizando *GridSearchCV*, que realiza una validación cruzada.

En este trabajo, se optó por realizar una búsqueda del valor *K* entre 1 y 21. Se creó el objeto *parámetros_knn* donde se estableció el rango de valores a buscar. El código completo implementado para este modelo es el siguiente:

```python
from sklearn.neighbors import KNeighborsRegressor
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import mean_squared_error

# Definir los parámetros a probar
parametros_knn = {'n_neighbors': range(1, 21)}  # desde 1 hasta 20 inclusive

# Crear el modelo KNN
knn_model = KNeighborsRegressor()

# Crear el objeto GridSearchCV
grid_search_knn = GridSearchCV(knn_model, parametros_knn, cv=5, scoring='neg_mean_squared_error')

# Entrenar el modelo
grid_search_knn.fit(X_train, y_train)

# Obtener el mejor modelo
mejor_modelo_knn = grid_search_knn.best_estimator_

# Hacer predicciones con el mejor modelo
predicciones_mejor_knn = mejor_modelo_knn.predict(X_test)

# Calcular el MSE del mejor modelo
mse_mejor_knn = mean_squared_error(y_test, predicciones_mejor_knn)
print("Mejor modelo KNN - MSE:", mse_mejor_knn)
```

```
Mejor modelo KNN - MSE: 0.5623137435435428
```

La salida muestra un MSE de **0.56**. Esto es un error bastante alto, incluso peor que el modelo de regresión lineal. Pese a realizar una búsqueda del modelo con los mejores hiperparámetros, este tipo de modelo no ha funcionado bien para la predicción de la magnitud.

En este caso este tipo de modelos busca calcular distancias entre puntos, puede ser que el tipo de datos no sea el adecuado y afecte negativamente al rendimiento del modelo.

4.2.3. Resultados del modelo de Random Forest

El modelo de Random Forest se explicó en el apartado 3.3.3 de este trabajo. En este caso, también se explicó el funcionamiento del código, pero faltó concretar que se definió un rango de valores para el parámetro de profundidad de los árboles de decisión, de manera similar al anterior, se realiza una búsqueda del mejor modelo con la mejor configuración de parámetros. El código completo es el siguiente:

```python
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import GridSearchCV

# Definir el rango de valores para el parámetro de profundidad
param_grid = {'max_depth': range(1, 11)}

# Inicializar el modelo de Bosque Aleatorio
rf_regressor = RandomForestRegressor(random_state=42)

# Inicializar GridSearchCV
grid_search = GridSearchCV(estimator=rf_regressor, param_grid=param_grid, cv=5, scoring='neg_mean_squared_error')

# Entrenar GridSearchCV
grid_search.fit(X_train, y_train)

# Obtener el mejor estimador
best_rf_regressor = grid_search.best_estimator_
print("Mejor estimador de Bosque Aleatorio:", best_rf_regressor)

# Realizar predicciones sobre el conjunto de prueba utilizando el mejor estimador
y_pred = best_rf_regressor.predict(X_test)

# Calcular el error cuadrático medio (MSE)
mse_rf = mean_squared_error(y_test, y_pred)
print("Error cuadrático medio (MSE):", mse_rf)
```

```
Mejor estimador de Bosque Aleatorio: RandomForestRegressor(max_depth=10, random_state=42)
Error cuadrático medio (MSE): 0.14090278963560668
```

Se utilizó un rango de valores para el parámetro de profundidad de entre 1 y 11. El resultado obtenido mediante el MSE fue de **0.1409**. Esto arroja un error bastante más bajo que los anteriores y por lo tanto un rendimiento superior de este modelo.

El mejor rendimiento obtenido con este modelo puede deberse a que es capaz de capturar relaciones no lineales y adaptarse mejor a la complejidad de los datos, además de tener más robustez frente a características irrelevantes, debido a que la construcción de múltiples árboles de decisión que luego se promedian sus predicciones hace que se obtenga un modelo más generalizado.

4.2.4. Resultados del modelo XGBoost

Como se comentó en el apartado 3.3.4, los hiperparámetros que se van a optimizar para seleccionar el mejor modelo son *max_depth, learning_rate* y *n_estimators*. Se van a definir los valores que queremos usar en un objeto que llamaremos *parametros_xgb*. De esta manera se hará una búsqueda entre diferentes modelos que usarán profundidades de árbol distintas, el número de árboles también será distinto y la tasa de aprendizaje. El código completo es el siguiente:

```python
import xgboost as xgb
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import mean_squared_error

# Definir los parámetros a probar
parametros_xgb = {
    'max_depth': [3, 7],  # Profundidad máxima del árbol
    'learning_rate': [0.01, 0.1, 0.3],  # Tasa de aprendizaje
    'n_estimators': [100, 200, 300]  # Número de árboles a utilizar
}

# Crear el modelo XGBoost
xgb_model = xgb.XGBRegressor()

# Crear el objeto GridSearchCV para buscar los mejores hiperparámetros
grid_search_xgb = GridSearchCV(xgb_model, parametros_xgb, cv=5, scoring='neg_mean_squared_error')

# Entrenar el modelo
grid_search_xgb.fit(X_train, y_train)

# Obtener el mejor modelo XGBoost
mejor_modelo_xgb = grid_search_xgb.best_estimator_
print("Mejor modelo XGBoost:", mejor_modelo_xgb)

# Realizar predicciones sobre el conjunto de prueba utilizando el mejor modelo
y_pred_xgb = mejor_modelo_xgb.predict(X_test)
```

```python
# Calcular el error cuadrático medio (MSE) del mejor modelo
mse_xgb = mean_squared_error(y_test, y_pred_xgb)
print("Error cuadrático medio (MSE) del mejor modelo XGBoost:", mse_xgb)
```

```
Mejor modelo XGBoost: XGBRegressor(base_score=None, booster=None, callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=None, device=None, early_stopping_rounds=None,
             enable_categorical=False, eval_metric=None, feature_types=None,
             gamma=None, grow_policy=None, importance_type=None,
             interaction_constraints=None, learning_rate=0.1, max_bin=None,
             max_cat_threshold=None, max_cat_to_onehot=None,
             max_delta_step=None, max_depth=7, max_leaves=None,
             min_child_weight=None, missing=nan, monotone_constraints=None,
             multi_strategy=None, n_estimators=100, n_jobs=None,
             num_parallel_tree=None, random_state=None, ...)
Error cuadrático medio (MSE) del mejor modelo XGBoost: 0.14109659489370033
```

El MSE del mejor modelo XGBoost seleccionado tras ajustar los mejores hiperparámetros es de **0.1411**. Este modelo arroja unos resultados casi iguales, por lo que su rendimiento es también muy bueno.

La eficacia de los modelos de Random Forest y XGBoost puede radicar en que utilizan métodos de ensamblado, combinando múltiples modelos de aprendizaje para mejorar el rendimiento predictivo final.

4.2.5. Resultados del modelo de red neuronal

El modelo de red neuronal que se implementa para predecir la magnitud de los seísmos es un modelo simple que se construye utilizando TensorFlow y Keras como se comentó en el apartado 3.3.5 de este trabajo.

La arquitectura ya fue explicada, junto con la función de compilación y el proceso de entrenamiento y predicción. Sin embargo, no se comentó el número de épocas que se han utilizado ni el tamaño del lote (*batch*). Para entrenar el modelo se creó el objeto *history_nn* y se utilizó la función ***.fit()***. Uno de los parámetros de esta función para las redes neuronales es ***epochs***, que es el número de épocas, que en este caso se estableció en 100. Esto significa que el modelo realizará 100 pasadas completas a través del conjunto de datos de entrenamiento para ajustar los pesos y mejorar el rendimiento. El otro parámetro que se ajusta es ***batch_size***, que se corresponde con el tamaño del lote, que en este caso se estableció en 32. Esto quiere decir que se utilizan 32 ejemplos de entrenamiento en cada iteración del algoritmo de optimización para actualizar los pesos.

El código completo del modelo de red neuronal implementado es el siguiente:

```
import tensorflow as tf
from tensorflow import keras
from sklearn.metrics import mean_squared_error

from sklearn.preprocessing import StandardScaler

# Escalar los datos para mejorar el rendimiento de la red neuronal
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Definir la arquitectura de la red neuronal
model = keras.Sequential([
    keras.layers.Dense(64, activation='relu', input_shape=[X_train.shape[1]]),
    keras.layers.Dense(32, activation='relu'),
    keras.layers.Dense(1)
])

# Compilar el modelo
model.compile(optimizer='adam', loss='mean_squared_error')

# Resumen arquitectura modelo
model.summary()

# Entrenar el modelo
history_nn = model.fit(X_train_scaled, y_train, epochs=100, batch_size=32, validation_split=0.2)

# Evaluar el modelo en el conjunto de prueba
y_pred_nn = model.predict(X_test_scaled)
mse_nn = mean_squared_error(y_test, y_pred_nn)
print("Error cuadrático medio (MSE) de la red neuronal:", mse_nn)
```

Hay que mencionar también que con la función ***.summary()*** obtenemos un resumen de la arquitectura del modelo que en este caso es:

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 64)                320

 dense_1 (Dense)             (None, 32)                2080

 dense_2 (Dense)             (None, 1)                 33

=================================================================
Total params: 2433 (9.50 KB)
Trainable params: 2433 (9.50 KB)
Non-trainable params: 0 (0.00 Byte)
```

Se pueden apreciar las tres capas densas que tiene el modelo, con 64, 32 y 1 unidades respectivamente, y la columna *Param* que indica el número de parámetros entrenables en cada capa.

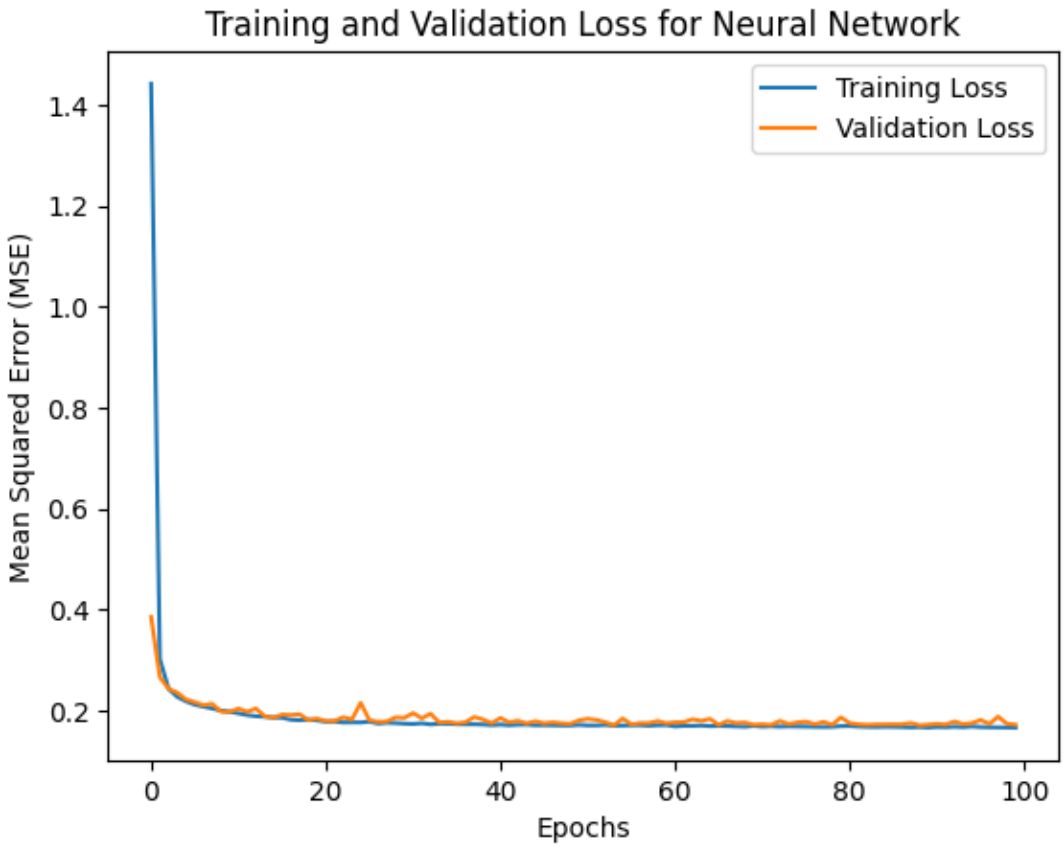
Al ejecutar el código, se muestra en la salida el proceso de entrenamiento durante todas las épocas que hemos establecido. Se indica el número de lotes y el progreso del entrenamiento, junto con el valor de pérdida (*loss*) en el conjunto de entrenamiento y el valor de pérdida en el conjunto de validación (*val_loss*). En nuestro ejemplo llegaría a 100 *epochs*.

```
Epoch 1/100
454/454 [==============================] - 4s 4ms/step - loss: 1.4418 - val_loss: 0.3852
Epoch 2/100
454/454 [==============================] - 1s 3ms/step - loss: 0.3018 - val_loss: 0.2658
Epoch 3/100
454/454 [==============================] - 1s 3ms/step - loss: 0.2429 - val_loss: 0.2430
Epoch 4/100
454/454 [==============================] - 1s 3ms/step - loss: 0.2268 - val_loss: 0.2351
Epoch 5/100
454/454 [==============================] - 1s 3ms/step - loss: 0.2180 - val_loss: 0.2217
Epoch 6/100
454/454 [==============================] - 1s 3ms/step - loss: 0.2114 - val_loss: 0.2164
Epoch 7/100
454/454 [==============================] - 1s 3ms/step - loss: 0.2074 - val_loss: 0.2102
Epoch 8/100
454/454 [==============================] - 1s 3ms/step - loss: 0.2039 - val_loss: 0.2131
Epoch 9/100
454/454 [==============================] - 1s 3ms/step - loss: 0.1997 - val_loss: 0.1976
Epoch 10/100
454/454 [==============================] - 1s 3ms/step - loss: 0.1975 - val_loss: 0.1968
```

```
Epoch 99/100
454/454 [==============================] - 1s 3ms/step - loss: 0.1662 - val_loss: 0.1736
Epoch 100/100
454/454 [==============================] - 1s 3ms/step - loss: 0.1658 - val_loss: 0.1718
142/142 [==============================] - 0s 2ms/step
Error cuadrático medio (MSE) de la red neuronal: 0.1650671515219932
```

El error cuadrático medio (MSE) que arroja este modelo que hemos implementado es de **0.1650**. Esto indica un buen rendimiento del modelo, aunque es dos décimas más alto que los anteriores modelos de machine learning.

También hemos representado la evolución de la pérdida de MSE durante el entrenamiento y validación de la red neuronal a lo largo de las épocas:



La rápida disminución de la pérdida de entrenamiento y validación al comienzo sugiere que el modelo aprende rápidamente los datos de entrenamiento y validación y mejora significativamente el rendimiento. Después se mantiene estable durante el resto de *epochs* lo que indica que no mejora su rendimiento, pero la similitud de las curvas de entrenamiento y validación sugieren que el modelo generaliza bien los datos nuevos y que no está sobreajustado.

4.2.6. Resultados del modelo de red neuronal recurrente (RNN) - LSTM

Al igual que el modelo anterior, este modelo de red neuronal recurrente se implementa utilizando TensorFlow y Keras, y la diferencia con el anterior radica en la configuración de las capas donde se añaden unidades LSTM. La arquitectura de este tipo de redes neuronales está explicada en el apartado 3.3.6 de este trabajo. El optimizador y los parámetros de entrenamiento son iguales al anterior modelo. El código es el siguiente:

```
from keras.models import Sequential
from keras.layers import LSTM, Dense

# Definir la arquitectura del modelo LSTM
modelo_lstm = Sequential()
modelo_lstm.add(LSTM(units=64, activation='relu', input_shape=(X_train_scaled.shape[1], 1)))
modelo_lstm.add(Dense(units=32, activation='relu'))
modelo_lstm.add(Dense(units=1))

# Compilar el modelo
modelo_lstm.compile(optimizer='adam', loss='mean_squared_error')

# Resumen arquitectura modelo
modelo_lstm.summary()

# Entrenar el modelo
history_lstm = modelo_lstm.fit(X_train_scaled.reshape((X_train_scaled.shape[0], X_train_scaled.shape[1], 1)),
                               y_train, epochs=100, batch_size=32, validation_split=0.2)

# Calcular las predicciones en el conjunto de prueba
predicciones_lstm = modelo_lstm.predict(X_test_scaled.reshape((X_test_scaled.shape[0], X_test_scaled.shape[1], 1)))

# Calcular el error cuadrático medio (MSE)
mse_lstm = mean_squared_error(y_test, predicciones_lstm)
print("Error cuadrático medio (MSE) del modelo LSTM:", mse_lstm)
```

El resumen de la arquitectura de este modelo es la siguiente:

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 64)                16896

 dense_3 (Dense)             (None, 32)                2080

 dense_4 (Dense)             (None, 1)                 33

=================================================================
Total params: 19009 (74.25 KB)
Trainable params: 19009 (74.25 KB)
Non-trainable params: 0 (0.00 Byte)
```

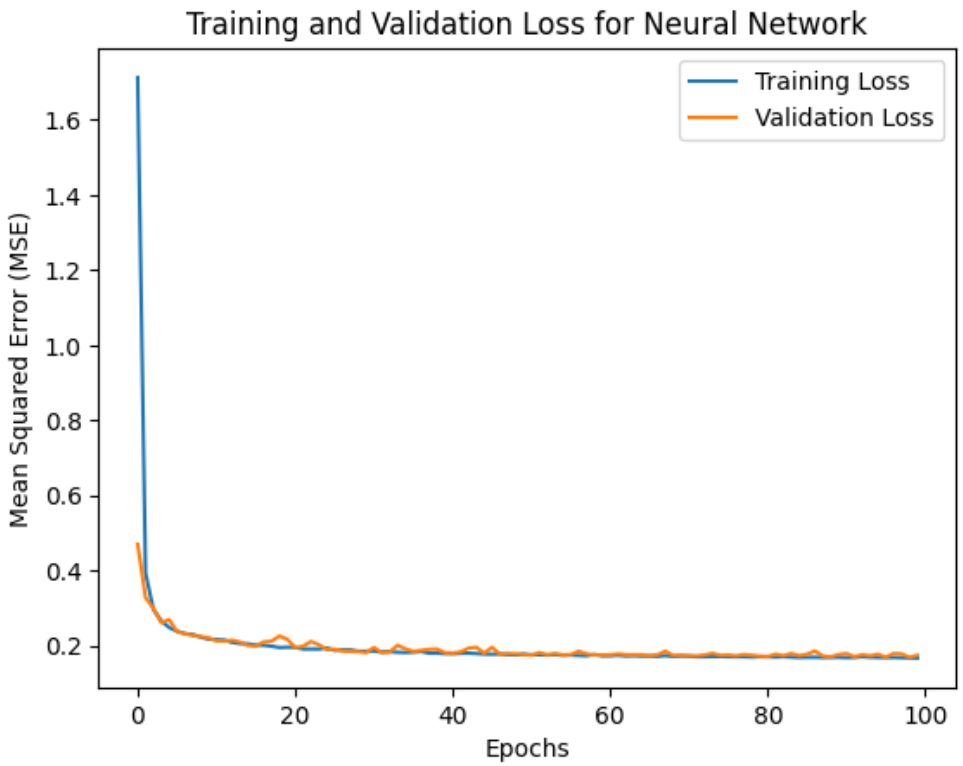
En este caso vemos que la primera capa es una capa LSTM y los parámetros entrenables en cada capa también difiere.

La salida tras ejecutar el código nos da lo siguiente (se omiten las primeras líneas):

```
Epoch 99/100
454/454 [==============================] - 4s 10ms/step - loss: 0.1674 - val_loss: 0.1689
Epoch 100/100
454/454 [==============================] - 4s 9ms/step - loss: 0.1669 - val_loss: 0.1747
142/142 [==============================] - 0s 2ms/step
Error cuadrático medio (MSE) del modelo LSTM: 0.16663126805134185
```

Por tanto, con este modelo tenemos un MSE de **0.1666**. Este valor es casi igual al obtenido con el modelo de red neuronal simple. La evolución de la pérdida de MSE durante el entrenamiento y validación es la siguiente:

```python
# Grafica de la pérdida (MSE) en entrenamiento y test para la red neuronal
plt.plot(history_lstm.history['loss'], label='Training Loss')
plt.plot(history_lstm.history['val_loss'], label='Validation Loss')
plt.title('Training and Validation Loss for Neural Network')
plt.xlabel('Epochs')
plt.ylabel('Mean Squared Error (MSE)')
plt.legend()
plt.show()
```



Podemos comprobar que es muy similar a la del modelo anterior, con una rápida pérdida al comienzo del entrenamiento y validación, que luego se mantiene estable sin diferencias, por lo que el modelo se adecua correctamente y sin sobreajuste.

4.2.7. Comparación resultados modelos

A modo de resumen y para ver mejor los resultados obtenidos sobre el rendimiento de los modelos usando el MSE como medida de comparación, se creó una tabla con el MSE de cada modelo:

```
# Crear un DataFrame con los resultados de MSE de cada modelo
resultados_mse = {
    'Modelo': ['Regresión Lineal', 'KNN', 'Random Forest', 'XGBoost', 'Red Neuronal', 'LSTM'],
    'MSE': [mse_reg_lineal, mse_mejor_knn, mse_rf, mse_xgb, mse_nn, mse_lstm]
}

df_resultados = pd.DataFrame(resultados_mse)

# Mostrar la tabla comparativa
print("Tabla comparativa de MSE de cada modelo:")
print(df_resultados)
```

```
Tabla comparativa de MSE de cada modelo:
             Modelo       MSE
0  Regresión Lineal  0.325236
1               KNN  0.562314
2     Random Forest  0.140903
3           XGBoost  0.141097
4      Red Neuronal  0.165067
5              LSTM  0.166631
```

Como se ha venido comentando al analizar cada modelo, los resultados del uso de modelos de machine learning y deep learning para la predicción de la magnitud de los seísmos muestran que los mejores modelos han sido los que utilizan técnicas de ensamblaje: Random Forest y XGBoost, con un MSE de 0.140903 y 0.141097 respectivamente. Los siguientes modelos en ofrecer resultados similares, pero con un MSE un poco más alto han sido los modelos de redes neuronales, en torno a un MSE de 0.16. Los modelos que han demostrado peores resultados en este tipo de problema han sido el modelo de regresión lineal, con un MSE de 0.325236 y el modelo KNN con un MSE de 0.562314.

5. Conclusiones

En este trabajo se ha abordado la problemática de la predicción de la magnitud de los seísmos utilizando modelos de machine learning y deep learning. Se han explorado diversas técnicas y algoritmos para analizar y predecir la ocurrencia de los seísmos, aprovechando los datos recopilados por el USGS en 2023. A lo largo del estudio, se han evaluado diferentes enfoques y se han comparado los resultados obtenidos, con el objetivo de identificar los modelos más eficaces para la predicción de la magnitud sísmica. A continuación, se resumirán los hallazgos más relevantes y se discutirán las implicaciones de este trabajo en el campo de la predicción de seísmos.

El análisis exploratorio inicial reveló que la mayoría de los eventos sísmicos en el conjunto de datos fueron seísmos naturales, lo que nos hizo prescindir del resto para este estudio. La distribución de estos eventos en función de la magnitud alcanzada nos mostró que la mayoría tenían magnitudes intermedias entre 4 y 5, además de tener una profundidad menor a 100 km desde la superficie. Esto nos da una idea de la naturaleza de los seísmos, asociado al funcionamiento de la tectónica de placas como se explicó en la introducción.

También se encontró una correlación entre la distribución geográfica de los seísmos y los límites de las placas tectónicas, con concentraciones significativas cerca de zonas de subducción y cordilleras importantes. Esto nos advierte de poner especial atención en aquellas zonas donde la peligrosidad puede ser mayor y se necesitan tener buenos sistemas de detección.

En cuanto a la distribución temporal, se identificaron tres picos de actividad sísmica durante el año, y un promedio de entre 50 y 100 seísmos por día, con una tendencia estable a lo largo de todo el año. Si bien es cierto que los seísmos de alta magnitud son bastante impredecibles, la mayoría de los seísmos ocurren de forma continuada a lo largo del tiempo.

El análisis de las estaciones de detección sísmica indicó una correlación entre el número de estaciones que detectan los seísmos y su magnitud, aunque hubo variaciones en la cantidad de estaciones que detectan la magnitud y la localización de los seísmos. Aquellos seísmos con mayor magnitud suelen ser detectados por mayor número de estaciones sísmicas.

Por último, en este análisis exploratorio, se pasó a analizar los errores en la medición, donde se concluyó que la medición de la magnitud era la más precisa, al tener errores asociados a cada evento muy bajos. Sin embargo, otras medidas como la profundidad o la medición horizontal tenían errores asociados más bajos, por lo que habría que mejorar en las técnicas de medición de estas últimas.

Tras analizar los datos sísmicos y se evaluaron varios modelos de machine learning y deep learning. Según los resultados obtenidos se puede concluir que los modelos de ensamblaje, como Random Forest y XGBoost, superaron significativamente a los modelos lineales y de vecinos más cercanos (KNN) en la predicción de la magnitud de los seísmos. Esta superioridad puede atribuirse a la capacidad de los modelos de ensamblaje para capturar relaciones no lineales y adaptarse a la complejidad inherente de los datos sísmicos. Además, estos modelos ofrecen una mayor robustez frente a características irrelevantes, lo que puede ser crucial en un conjunto de datos con múltiples variables y factores influyentes en la magnitud de los seísmos.

Por otro lado, aunque los modelos de redes neuronales también demostraron ser competitivos, con un rendimiento cercano a los modelos de ensamblaje, su entrenamiento y ajuste de hiperparámetros puede ser más complejo en comparación con los modelos de ensamblaje. Es por ello, que en un futuro como mejora de lo planteado en este trabajo se pudieran mejorar los modelos de redes neuronales probando diferentes combinaciones de hiperparámetros o explorando arquitecturas más complejas y profundas.

6. Referencias

Abebe, E., Kebede, H., Kevin, M., y Demissie, Z. (2023). Earthquakes magnitude prediction using deep learning for the Horn of Africa. *Soil Dynamics and Earthquake Engineering*, *170.*

Abdul Salam, M., Ibrahim, L., y Abdelminaam, D. S. (2021). Earthquake Prediction using Hybrid Machine Learning Techniques. *International Journal of Advanced Computer Science and Applications (IJACSA)*, *12*(5).

Amazon Web Services. What is Deep Learning?. Recuperado el 15 de Marzo de 2024. https://aws.amazon.com/es/what-is/deep-learning/

Andel, T. H. y Murphy, . J. B. Plate tectonics. *Encyclopedia Britannica*. Recuperado el 15 de Marzo de 2024. https://www.britannica.com/science/plate-tectonics

Asim, K. M., Martínez-Álvarez, F., Basit, A. y Iqbal, T. (2017). Earthquake magnitude prediction in Hindukush region using machine learning techniques. *Natural Hazards, 85*, 471–486.

Çekim, H.Ö., Karakavak, H.N., Özel, G. y Tekin, S. (2023). Earthquake magnitude prediction in Turkey: A comparative study of deep learning methods, ARIMA and singular spectrum analysis. *Environmental Earth Sciences, 82*(387).

Chaya (2020). Random Forest Regression. Level Up Coding. Recuperado el 16 de Marzo de 2024. https://levelup.gitconnected.com/random-forest-regression-209c0f354c84

Chen, T. y Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. En *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 785-794).

Folium. Recuperado el 15 de Marzo de 2024. https://python-visualization.github.io/folium/

Géron, A. (2019). *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems*. Sebastopol: O'Reilly Media.

Goodfellow, I., Bengio, Y. y Courville, A. (2016). *Deep learning*. MIT Press.

Hastie, T., Tibshirani, R. y Friedman, J. (2009). *The elements of statistical learning: Data mining, inference, and prediction.* Nueva York: Springer.

Ioffe, S. y Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. Arxiv.

James, G., Witten, D., Hastie, T. y Tibshirani, R. (2013). *An introduction to statistical learning: With applications in R.* Nueva York: Springer.

Javatpoint. K-Nearest Neighbor Algorithm for Machine Learning. Recuperado el 16 de Marzo de 2024.
https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning

Jordan, M.I. y Mitchell, T.M. (2015). Machine learning: Trends, perspectives, and prospects. *Science*, *349*(6245), 255-260.

Joshi, A., Raman, B., Mohan, C.K. y Cenkeramaddi, L.R. (2024). Application of a new machine learning model to improve earthquake ground motion predictions. *Natural Hazards, 120*, 729-753.

Keras. Recuperado el 15 de Marzo de 2024.
https://keras.io/

Keser, M. Earthquakes 2023 Global. Recuperado el 7 de Febrero de 2024.
https://www.kaggle.com/datasets/mustafakeser4/earthquakes-2023-global/data

Lecun, Y., Bengio, Y. y Hinton, G. (2015). Deep learning. Nature, 521, 436-444.

McKinney, W. (2010). Data Structures for Statistical Computing in Python. En S. van der Walt & J. Millman (Eds.), *Proceedings of the 9th Python in Science Conference* (pp. 51-56). Austin, TX. Recuperado el 15 de Marzo de 2024.
https://pandas.pydata.org/

Novick, D. y Last, M. (2023). Using machine learning models for earthquake magnitude prediction in California, Japan, and Israel. En S. Dolev, E. Gudes y P. Paillier (Eds.), *Cyber security, cryptology, and machine learning* (pp. 151-169). Springer, Cham.

Patil, A. S., & Panhalkar, S. S. (2023). Remote sensing and GIS-based landslide susceptibility mapping using LNRF method in part of Western Ghats of India. *Quaternary Science Advances*, *11.*

Proyecto Jupyter. Recuperado el 8 de Febrero de 2024.
https://jupyter.org/

Rashidi, J.N. y Ghassemieh, M. (2023). Predicting the magnitude of injection-induced earthquakes using machine learning techniques. *Natural Hazards, 118*, 545-570.

Sadhukhan, B., Chakraborty, S. y Mukherjee, S. (2023). Predicting the magnitude of an impending earthquake using deep learning techniques. *Earth Science Informatics, 16*, 803-823.

Scikit-learn: Machine Learning in Python. Scikit-learn.org. Recuperado el 15 de Marzo de 2024.
https://scikit-learn.org/stable/index.html

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural Networks, 61*, 85-117.

Statsmodels. Recuperado el 15 de Marzo de 2024.
https://www.statsmodels.org/stable/generated/statsmodels.tsa.seasonal.seasonal_decompose.html

TensorFlow. Recuperado el 15 de Marzo de 2024.
https://www.tensorflow.org/

U.S. Geological Survey. Recuperado el 7 de Febrero de 2024.
https://www.usgs.gov/

Wang, Y., Li, X., Wang, Z. y Liu, J. (2023). Deep learning for magnitude prediction in earthquake early warning. *Gondwana Research, 123*, 164-173.

Waskom, M. (2020). Seaborn: Statistical data visualization. Recuperado el 15 de Marzo de 2024.
https://seaborn.pydata.org/

Wasserman, J., Sosulski, K. y Oshan, T. (2021). Geopandas: Python Tools for Geographic Data. Recuperado el 15 de Marzo de 2024.
https://geopandas.org/en/stable/