Predicción de volatilidad

Felipe de Jesús Damián Rodríguez

A01707246

Tecnológico de Monterrey

Octubre 2025

1 Introducción

El objetivo de este proyecto es construir un modelo capaz de anticipar periodos de alta o baja volatilidad en el mercado financiero utilizando datos históricos del índice S & P500. La motivación surge de la importancia que tiene la volatilidad en la gestión de riesgos, la toma de decisiones de inversión y el diseño de estrategias automatizadas. La clasificación temprana de estos escenarios nos permite ajustar posiciones y reducir perdidas potenciales. Para esto implemente un flujo completo en donde limpiamos el dataset, generamos retornos logarítmicos, construimos las etiquetas basadas en la volatilidad futura, análisis exploratorio, un modelo baseline y finalmente una red neuronal diseñada para series de tiempo. El proyecto busca comparar el desempeño del modelo profundo frente al baseline y asi evaluar su utilidad práctica en la predicción de los retornos de riesgo.

2 Plantamiento del problema

Como mencione anateriormente el objetivo es predecir la volatilidad futura del índice S & P 500 si será alta (1) o baja (0) utilizando únicamente la información histórica de retornos logarítmicos. Esto lo formule como un problema de clasificación binaria supervisada aplicada a series de tiempo.

Los resultados esperados es un modelo que clasifique de manera confiable escenarios de volatilidad futura, que presente una mejora respecto al modelo baseline y un desempeño consistente en el conjunto de validación.

La entrada del modelo consiste en una secuencia continua de 60 días consecutivos de retornos logarítmicos del índice S&P 500. Cada ejemplo está formado por el vector $X = (log_{ret_{t-59}}, ..., log_{ret_t})$, que representa cómo ha cambiado el preció del índice durante los últimos 60 días antes del punto donde queremos predecir.

Usar retornos logarítmicos hace que los cambios diarios del precio sean más fáciles de comparar entre sí, porque expresan las variaciones en proporción al nivel del precio. [1]

La salida es una etiqueta binaria que clasifica la volatlidad futura del mercado durante los próximos 10 días. se asigna el valor $y = 1$ cuando la volatilidad futura la cual es medida como la desviación estándar de los retornos en ese ventana es mayor que la mediana de toda la serie. En caso contrario, se asigna $y = 0$.

Esto nos permite transformar un indicador que es continuo de variablidad en un problema de clasificación binaria, facilitando el análisis predictivo del comportamiento del mercado.

3 Dataset

En el proyecto utilice una serie histórica del S&P 500, obtenida mediante la librería yfinance, que proporciona datos financieros a través de la API Yahoo Finance. Se descargaron precios diarios desde al año 2000 hasta 2025, lo que genera un conjunto de datos de más de $6,500$ observaciones.

El dataset incluye las siguientes variables:

- Open: precio de apertura diario
- High: Preció máximo del día
- Low: precio mínimo del día
- Close: precio de cierre ajustado
- Volume: volumen de operación

A partir de estos datos se contruyó la variables principal del proyecto la cual es la de los retornos logarítmicos diarios, definidos como:

$log_{ret_t} = ln(\frac{P_t}{P_{t-1}})$

El uso de retornos logarítmicos es estándar en el análisis de series financieras porque estabiliza la varianza y facilita la comparación entre días con precios muy distintos. [1]

Posteriormente, el dataset fue preprocesado eliminando valores faltantes y generando las etiquetas para el modelo. Para ello, se calculó la **volatilidad futura** como la desviación estándar de los próximos 10 días. Con esta información cree una etiqueta binaria.

- 1: volatilidad futura mayor a la mediana de toda la serie
- 0: volatilidad futura menor o igual

Finalmente los datos los dividimos en 80 % para entrenamiento y 20 % para validación.Esto nos garantiza que el modelo aprenda patrones de comportamiento histórico sin sobreajuste y que su desempeño se evalúe en un conjunto independiente y no visto.

4 Análisis exploratorio

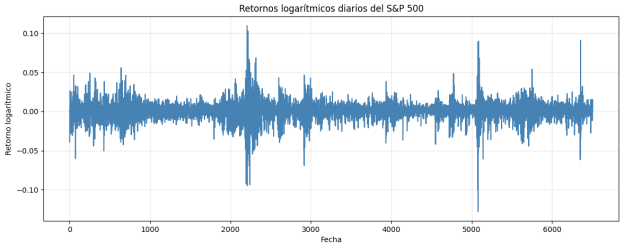


Figura 1: Desviación estándar móvil de 10 días del S&P 500.

La Figura 1 muestra cómo ha cambiado la volatilidad del S&P500 a lo largo del tiempo, medida como la desviación estándar de los retornos en ventanas de 10 días. A partir de ella se pueden extraer varios hallazgos.

1. Se observan periodos porlongados de baja volatilidad y de repente picos abruptos. Esto confirma la presencia de clustering de volatilidad, un fenómeno típico de los mercados financieros. [2]

2. Los picos más altos coinciden con eventos financieros conocidos como el periodo de 2008-2009 donde ocurrió una crisis financiera global y en 2020 en donde paso el crash por COVID-19. Estos episosdios muestran aumentos extremos y repentinos de volatilidad, validando que nuestra métrica captura correctamente los shocks del mercado.[3]

3. La mayor parte de la serie se mantiene en valores relativamente bajos por debajo de 0.02, lo cual es consistente con un mercado el cual es generalmente estable con episodios ocasionales de turbulencia.

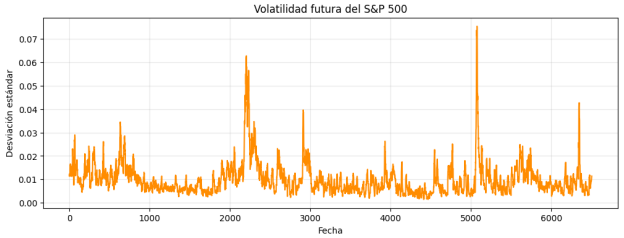


Figura 2: Serie de retornos Logarítmicos del S&P500

Un aspecto relevante de la Figura 2 es que, aunque la mayoría de los retornos se mantinen cerca de 0, hay días con movimientos inesperados que sobresalen del comportamiento normal.

Estos saltos muestran que el mercado puede volverse inestable de un día a otro, incluso sin que haya existido una tendencia previa evidente. Esto refuerza la idea de que el riesgo no es constante y que el mercado puede cambiar de estado rapidamente.

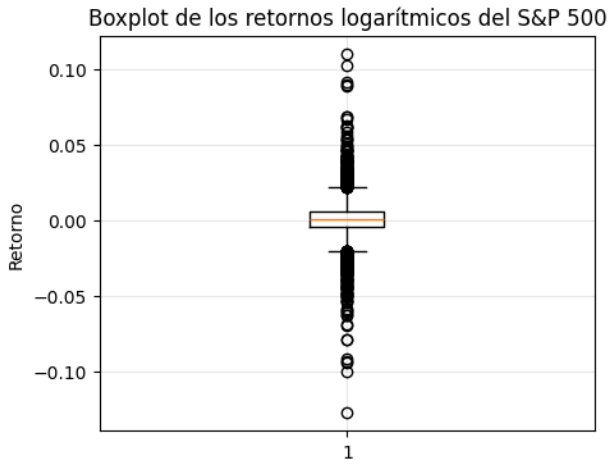


Figura 3: Boxplot de retornos Logarítmicos

El boxplot refueza la idea de que aunque la mayoría de los retornos están muy cerca de cero, existen muchos valores extremos en ambos lados. Esto confirma que el S&P 500 presentad días con movimientos inusuales que elevan la volatilidad y que no sigue una distribución que sea perfectamente simétrica.

Un punto importante que se observa al revisar tanto los retornos como la volatilidad futura es el fenómeno conocido como clustering de volatilidad. En la serie se forman bloques de días muy tranquilos seguidos de bloques de alta turbulencia, en lugar de ocurrir de manera aleatoria. Esto confirma el comportamiento típico de mercados financieros. [2]

5 Métrica: Accuracy

Para evaluar el desempeño del modelo se utilizará el accuracy como métrica principal. Debido a que la tarea consiste en clasificar si la volatilidad futura será alto o baja, el accuracy permite medir de manera directa qué proporción de predicciones coincide con la etiqueta real. Por lo tanto considero es adecuada porque el dataset quedó balanceado al construir las etiquetas, lo que evita los sesgos del accuracy en problemas desbalanceados.

Además el acurracy es una métrica estándar en problemas de predicción financiera cuando el objetivo es evaluar la capacidad del modelo para identificar correctamente escenarios binarios de mercado.

Antes de entrenar el modelo fije un desempeño objetivo en un rango de 65 % a 70 % el cual nos sirve como un criterio mínimo para considerar al modelo útil. [1]

6 Baseline

Como punto de referencia implementé un modelo baseline sencilllo basado en regresión logística. Este modelo utiliza únicamente información del pasado inmediato: los retornos de los últimos cuatro días y la volatilidad histórica de los último cinco días. Su objetivo no es predecir el valor numérico de la desviación estándar futura, sino clasificar si la volatilidad de los próximos diez días estará por encima o por debajo de la mediana histórica.

Para contruir las variables predictoras incluí los retornos logarítmicos de 1 a 4 días previos (ret_1,ret_2,ret_3,ret_4), junto con la desviación estándar móvil de cinco días (vol_5). Los retornos recientes permiten capturar señales de corto plazo asociadas al fenómeno de clustering de volatilidad, mientras que la volatilidad resume el nivel actual de variabilidad del mercado. Esta combiandción representa un enfoque típico en modelos simplees de predicción financiera.

La regresión logística toma esta cinco características como entrada y estima la probabilidad de que la volatilidad futura sea alta. El modelo aprende una combinación lineal de las variables para separar los casos de alta y baja volatilidad, produciendo así una predicción binaria.

Para evaluar su desempeño utilicé una partición temporal de 80 % para entrenamiento y 20 % para validación. Tras el entrenamiento, el modelo alcanzó una exactitud aproximada del 64 % en el conjunto de validación. Este desempeño es consistente con lo que suele observarse al emplear predictores lineales simples en la clasificación de la volatilidad financiera.

En resumen, este baseline proporciona un punto de comparación razonable y es claramente inferior a lo que se espera en el modelo final basado en redes neuronales.

7 Modelo Final: red neuronal CNN 1D

Para el modelo final utilicé una red neuronal convolucional 1D diseñada para trabajar con series de tiempo, cuyo objetivo es identificar patrones locales dentro de la ventana de 60 retornos logarítmicos. Este tipo de arquitectura es apropiada para capturar micro-tendencias, cambios abruptos y periodos de acumulación de volatilidad que suelen anticipar episodios turbulentos en los mercados financieros.

7.1. Arquitectura del modelo

La arquitectura final está inspirada en configuraciones utilizadas en la literatura para predicción financiera con CNNs [1], así como en arquitecturas de redes convolucionales 1D aplicadas a series temporales [4]. El modelo que entrené tiene la siguiente estructura:

- **Conv1D ($1 \rightarrow 64$, kernel $= 5$, ReLU)**
  Captura patrones locales básicos dentro de la serie temporal.
- **MaxPool1D (stride $= 2$)**
  Reduce la longitud de la secuencia y extrae características relevantes.
- **Conv1D ($64 \rightarrow 128$, kernel $= 7$, ReLU)**
  Kernel más amplio para captar patrones temporales de mayor alcance.
- **MaxPool1D (stride $= 2$)**
  Reduce nuevamente la dimensionalidad y ayuda a evitar sobreajuste.
- **Dropout (p $= 0.3$)**
  Regularización adicional para mejorar la generalización.
- **AdaptiveAvgPool1D(1)**
  Colapsa la secuencia a un único vector representativo.
- **Capa totalmente conectada ($128 \rightarrow 2$)**
  Produce la clasificación final (volatilidad alta o baja).

Esta arquitectura mantiene un equilibrio entre profundidas y simplicidad, lo cual es importante para evitar sobreajuste dado el tamaño del dataset. Además, es un enfoque común para modelar series financieras y un comportamiento no lineal.

7.2. Entrenamiento del modelo

Usé una ventana fija de 60 días y entrené la red con un máximo de 100 épocas bajo la siguiente configuración:

- Optimizador: Adam
- Learning rate: $1e-3$
- Tamaño del batch: 64

- Función de pérdida: CrossEntropyLoss
- Métrica principal: Accuracy
- Regularización (Dropout + Weight Decay)
- Callback: EarlyStopping(monitor = valid_loss, paciencia = 5)
- Split: 80 % entrenamiento, 20 % validación sin mezclar temporalmente.

Implementar weight decay y dropout juntos ayudó a que el modelo aprendiera de forma más estable y sin sobreajustarse. Gracias a esto, la red pudo generalizar mejor y la precisión final aumentó de manera consistente. En pocas palabras, usar estas dos regularizaciones al mismo tiempo hizo que el modelo funcionara mejor.

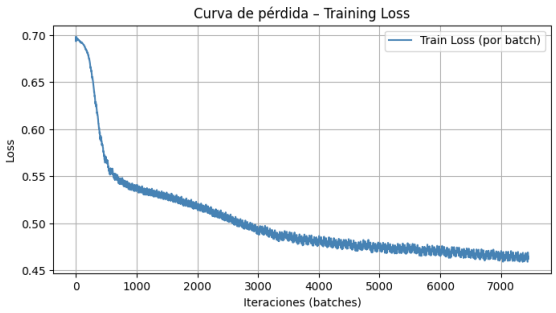
El entrenamiento se realizó utilizando la API de fastai [5]. Aunque definí un límite de 100 épocas, observé que el modelo tendía a converger de manera consistente entre las épocas de 60 y 90, después de ese punto, las mejoras en la pérdida de validación eran muy pequeñas. El EarlyStoppingCallback no siempre se activó debido a las fluctuaciones del valid_loss, que en algunos casos seguía disminuyendo ligeramente sin cumplir los criterios estrictos de paciencia.

7.3. Resultados

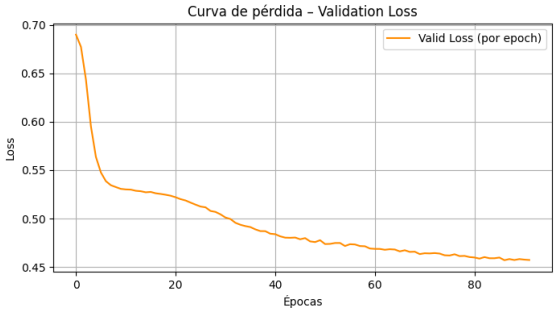
El modelo final alcanzó un accuracy de 79 %, lo que representa una mejora sustancial respecto al baseline basado en regresión logística, que obtuvo alrededor del 64 %. Esta diferencia confirma que la arquitectura CNN1D es capaz de capturar patrones no lineales y dependencias temporales dentro de la ventana de 60 días que un modelo lineal no puede modelar adecuadamente.

Las curvas de pérdida permiten analizar el comportamiento del entrenamiento. En la gráfica de training loss se observa una disminución continua a lo largo de los batches, lo cual indica que la red ajusta sus parámetros de manera estable y sin comportamientos erráticos. Por otra parte, la curva de validation loss muestra una caída pronunciada en las primeras épocas, seguida de mejoras graduales que se extienden hasta aproximadamente la época 60. A partir de ese punto, la pérdida de validación continúa disminuyendo, pero de forma marginal, lo que sugiere que el modelo está convergiendo.

Aunque se utilizó un callback de early stopping, en el entrenamiento presesntado si llega a activarse y detuvo el proceso en la época 91, momento en el que el validation loss dejó de



(a) Evolución del training loss por batch durante el entrenamiento del modelo CNN1D



(b) Comportamiento del validation loss por época para el modelo CNN1D

Figura 4: Curvas de pérdida del modelo CNN1D: entrenamiento y validación

mejorar y la exactitud se estabilizó. Aun así, decidí mantener un límite máximo de 100 épocas como tope general, ya que en ejecuciones previas observé que, después de ese punto, el modelo rara vez obtenía mejoras significativas e incluso podía comenzar a degradar su rendimiento en validación. De este modo, el corte automático en la época 91 reflejó adecuadamente el punto de convergencia del modelo, mientras que el límite de 100 épocas sirve como medida preventiva para evitar sobreentrenamiento en casos donde las fluctuaciones del validation loss impiden que el early stopping se active de inmediato.

De manera consistente, observé que el modelo suele alcanzar valores de exactitud entre 77 % y 80 %, dependiendo de las pequeñas variaciones resultantes del proceso de entrenamiento. Además, la mayoría de las ejecuciones requieren al menos unas 60 épocas para superar de forma estable el umbral del 78 %, lo que sufiere que la red necesita una fase de entrenamiento relativamente prolongada para aprender patrones temporales relevantes y consolidar su capacidad predictiva.

7.4. Ajuste de hiperparámetros

Durante el desarrollo del modelo se realizó un ajuste iterativo de hiperparámetros con el objetivo de mejorar la estabilidad del entrenamiento y la capacidad predictiva. Las principales ajustes que se realizaron fueron los siguientes:

- Número de filtros: se probaron configuraciones con 32, 64 y 128 filtros. La arquitectura final (64→128) ofreció el mejor equilibrio entre capacidad de representación y estabilidad del entrenamiento.
- Tamaño de kernel: se evaluaron kernels de 3,5 y 7. Los kernels más amplios (5 y 7) capturaron patrones temporales de mayor escala y redujeron la pérdida de validación

- Dropout: se añadió una con $p = 0{,}3$, lo que ayudó a reducir fluctuaciones en la validación y mejorar la generalización.
- Learning rate: se probaron valores como $1e-2$, $1e-3$ y $5e-4$. El valor final elegido fue $1e-3$, ya que ofreció un entrenamiento más estable.
- Early stopping: el callback se activó en la época 91 en el entrenamiento final, indicando que la validación había dejado de mejorar de forma significativa.

En conjunto, estos ajustes permitieron obtener un modelo más robusto y estable, con mejor capacidad de generalización. Como resultado, el modelo alcanza niveles de exactitud que suelen ubicarse entre 77 % y 80 % en la mayoría de las ejecuciones, llegando al 79 % en el entrenamiento presentado en este reporte.

8 Limitaciones y trabajo futuro

Aunque el modelo final obtuvo un desempeño notable frente al baseline, presenta varias limitaciones importantes. La arquitectura está basa únicamente en retornos logarítmicos y en una ventana fija de 60 días, por lo que no incorpora información adicional del mercado ni variables macroeconómicas que podrían influir en la volatilidad. Esto reduce la capacidad del modelo para capturar dinámicas más complejas o eventos externos que afectan el comportamiento del índice.

Otra limitación es que, aunque se realizaron ajustes de hiperparámetros, estos no constituyen una búsqueda exhaustiva. Se evaluaron variaciones en kernels, filtros, regularización y número de épocas, pero técnicas como random search o bayesian optimization podrían revelar configuraciones más eficientes.

En cuanto al trabajo futuro, sería valioso explorar arquitecturas más avanzadas para series temporales, como Temporal Convolutional Networks (TCN), LSTM o GRU. También podriamos usar más características como indicadores técnicos, volumen o variables macroeconómicas. Otra línea relevante es utilizar esquemas de validación más robustos, como walk-forward validation, que reflejan mejor la dinámica real de los mercados financieros. Finalmente, una búsqueda de hiperparámetros mediante random search o bayesian optimization podría mejorar aún más el desempeño predictivo del modelo.

9 Conclusiones

En este proyecto se desarrolló un modelo capaz de clasificar periodos de alta y baja volatilidad en el índice S&P 500 utilizando ventanas de 60 días de retornos logarítmicos. El modelo baseline, basado en regresión logística, obtuvo accuracy cercana al 64 %, lo que proporcionó un punto de referencia inicial. La arquitectura CNN1D permitió capturar patrones no lineales y dependencias temporales presentes en la serie, lo que llevó auna mejora considerable del desempeño. Después de ajustar filtros, tamaños de kernel, dropout, regularización y tasa de aprendizaje, el modelo final alcanzó un accuracy de validación de 79 %, con valores que en distintas ejecuciones suelen ubicarse entre 77 % y 80 %. Esto evidencia que las redes convolucionales son una herramienta efectiva para modelar dinámicas de volatilidad en series financieras.

El desarrollo del proyecto también permitió identificar retos importantes, como la sensibilidad del modelo a la selección de hiperparámetros, la dependencia de una ventana fija de entrenamiento y la falta de información adicional del mercado. Aun así, los resultados muestran que una CNN1D bien configurada puede generalizar de manera estable y superar significativamente a un modelo lineal. Este trabajo sienta una base sólida para futuras líneas de investigación, incluyendo arquitecturas más complejas, integración de variables adicionales y esquemas de validación más robusto que permitan evaluar el desempeño predictivo en distintos escenarios de mercado.

Declaración de uso de herramientas de IA

Durante la elaboración de este proyecto se utilizó ChatGPT (modelo GPT-5.1, OpenAI) exclusivamente como herramienta de apoyo para aclarar dudas conceptuales, resolver problemas específicos de código, y comprender mejor ciertas implementaciones relacionadas con redes neuronales y series de tiempo. Todas las decisiones metodológicas, el diseño del modelo, la experimentación y la interpretación de resultados fueron realizadas de manera independiente por el autor.

De igual manera, en el notebook ejecutable de Google Colab y en los archivos de código del proyecto se incluyen comentarios que documentan algunos de los prompts utilizados durante el desarrollo. Estos comentarios reflejan únicamente apoyo en tareas de depuración de código y explicación de conceptos, sin intervenir en la lógica central del modelo ni en las decisiones de implementación.

Referencias

[1] Ruey S. Tsay. *Analysis of Financial Time Series*. Wiley, Hoboken, NJ, 3rd edition, 2010.

[2] Gerard Sánchez. Retornos, riesgo y volatilidad (parte ii), September 18 2020. URL `https://gsnchez.com/blog/article/Retornos-riesgo-y-volatilidad-parte-ii`. Último acceso: 26 November 2025.

[3] Vito Tanzi. La crisis financiera y económica de 2008-2009: efectos fiscales y monetarios, 2010. URL `https://asip.org.ar/la-crisis-financiera-y-economica-de-2008-2009-efectos-fiscales-y-monetarios/`. Conferencia pronunciada en el XXXVII Seminario Internacional de Presupuesto Público, Madrid, 5–8 julio 2010. Último acceso: 26 November 2025.

[4] Serkan Kiranyaz, Turker Ince, and Moncef Gabbouj. Convolutional neural networks for patient-specific ecg classification. *IEEE Transactions on Biomedical Engineering*, 63(3): 664–675, 2015.

[5] Jeremy Howard and Sylvain Gugger. *Deep Learning for Coders with fastai and PyTorch: AI Applications Without a PhD*. O'Reilly Media, 2020. ISBN 9781492045526.

Apéndice

Repositorio del proyecto

El código completo del proyecto, incluyendo los notebooks, el dataset procesado y la implementación de los modelos (baseline y CNN1D), se encuentra disponible en el siguiente repositorio público de GitHub:
`https://github.com/FelipeDamianR/-volatility-prediction-cnn`

Notebook ejecutable en Google Colab

Para reproducir de manera integrada todo el flujo del proyecto —carga de datos, análisis exploratorio, baseline y entrenamiento del modelo final— se proporciona el siguiente notebook ejecutable en Google Colab, donde se encuentra el código completo en un solo archivo:

https://colab.research.google.com/drive/1W7JjVjdbFoV-JySLRRAk3EcExu8tG2W8?usp=sharing

Dataset limpio utilizado

El proyecto emplea una versión procesada del conjunto de datos original, la cual incluye precios del S&P 500, retornos logarítmicos, etiquetas de volatilidad futura y todas las variables derivadas necesarias para los modelos. Este archivo es utilizado por todos los notebooks posteriores a la fase de limpieza. De igual manera se puede descargar atraves del repositorio git o no hace falta descargarlo dentro del notebook ejecutable de Google Colab: https://docs.google.com/spreadsheets/d/1XLAPZH7pLloQtIcLsNv3fag_ODqQA10X8RWazQ_SZzo/edit?usp=sharing