Modelo Predictivo para el Concurso Hull Tactical Market Prediction

Autor: Elias Sebastian Gill Quintana

Tutor: Diego Pedro Pinto Roa

Ingeniería en Informática

Facultad Politécnica
Universidad
Nacional de Asunción

1. Contexto y Motivación

1. Contexto y Motivación

Contexto del problema

El presente trabajo tiene como objetivo analizar y diseñar una propuesta de solución al desafío "*Hull Tactical Market Prediction*", organizado en la plataforma ***Kaggle***. Este problema pertenece al ámbito de la predicción financiera y plantea el reto de anticipar los rendimientos diarios del índice S&P 500 bajo una restricción de volatilidad.

1. Contexto y Motivación

Relevancia

La predicción de rendimientos financieros continúa siendo un tema central en el estudio de la eficiencia de los mercados. El avance de las técnicas de aprendizaje automático permite reconsiderar la “Hipótesis del mercado eficiente” en contextos donde los patrones no lineales o las interacciones entre múltiples variables podrían ofrecer señales útiles para la toma de decisiones.

La relevancia del problema radica en la posibilidad de construir estrategias de inversión más eficientes en riesgo y rendimiento, aplicando técnicas de predicción avanzadas sobre datos públicos y privados del mercado.

2. Descripción del problema

2. Descripción del problema

El objetivo del concurso consiste en desarrollar un modelo predictivo que, para cada día de negociación, determine la exposición óptima al mercado de valores (S&P 500) mediante una señal de asignación diaria comprendida en el intervalo [0, 2].

Interpretación de la señal de asignación (position):

- 0.0 → Posición completamente neutral (ninguna exposición al mercado).
- 1.0 → Exposición equivalente al benchmark (100 % del capital invertido en el índice).
- 2.0 → Máxima exposición permitida (200% del capital; 100 % propio + 100 % apalancado).

La señal se construye a partir de una estimación del exceso de rendimiento esperado del mercado respecto de la tasa libre de riesgo (market forward excess return).

El ranking se evalúa mediante el **Sharpe ratio ajustado**.

2. Descripción del problema

El dataset exhibe **heterocedasticidad** (varianza no constante)**, no estacionariedad** (comportamiento inestable) **y dependencia temporal compleja**, características que invalidan el uso de modelos lineales simples y requieren enfoques más sofisticados.

Adicionalmente, la naturaleza **secuencial y dependiente del tiempo** de los datos financieros impide utilizar métodos de muestreo aleatorio convencionales, ya que estos destruirían la estructura temporal y las relaciones no lineales entre las observaciones.

3. Estado del arte

3. Estado del arte

Limitaciones de los modelos clásicos

Aunque esenciales, los modelos clásicos encuentran barreras
al enfrentarse a la complejidad de las series de tiempo financieras,
especialmente la no linealidad y la alta dimensionalidad de los datos.

ARIMA y VAR (modelos clásicos)
- Asumen linealidad en las relaciones entre variables
- Escala limitada ante cientos de features
- Baja capacidad para adaptarse a eventos imprevistos y cambios de régimen

Redes Neuronales Recurrentes (LSTM, GRU)
- Requieren grandes volúmenes de datos y alto poder computacional
- Alta propensión al sobreajuste (overfitting) en el período de entrenamiento
- Dificultad para generalizar en entornos de alta ruido y baja señal

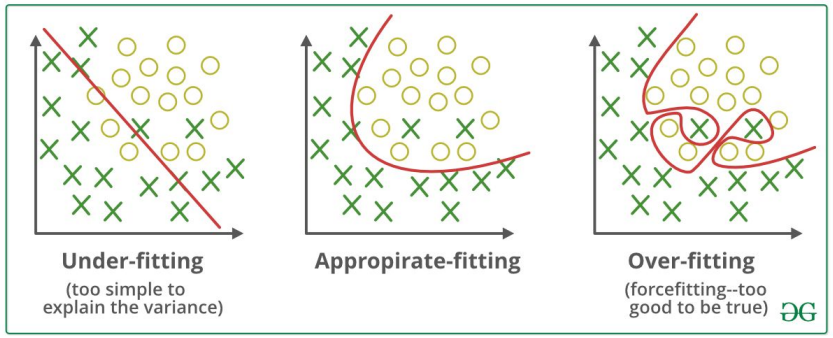
Los enfoques clásicos resultan insuficientes para maximizar el Sharpe ajustado
en un entorno con restricción de volatilidad.

3. Estado del arte

Problemas de sobreajuste y dependencia de un solo tipo de dato

Sobreajuste (Overfitting)
Ocurre cuando el modelo aprende demasiado bien los datos de entrenamiento, perdiendo capacidad de generalizar. Común en modelos complejos como redes neuronales profundas.

Dependencia de una sola fuente de información
Muchos enfoques se limitan exclusivamente a los precios históricos del S&P 500 o a un subconjunto reducido de features, ignorando variables externas disponibles.



4. Conceptos previos

4. Conceptos previos

Introducción a los sistemas ensamblados

Son un conjunto de écnicas que combina las predicciones de múltiples modelos base (diversos o complementarios) para obtener un único modelo final más robusto, preciso y estable.

Objetivo principal

Reducir el error de generalización mediante la agregación de la “sabiduría colectiva” de varios aprendices.

Tipos principales de ensamblados:

- Bagging: ej. Random Forest
- Boosting: ej. LightGBM, XGBoost
- Stacking: Meta-modelo aprende la mejor combinación de predictores base

4. Conceptos previos

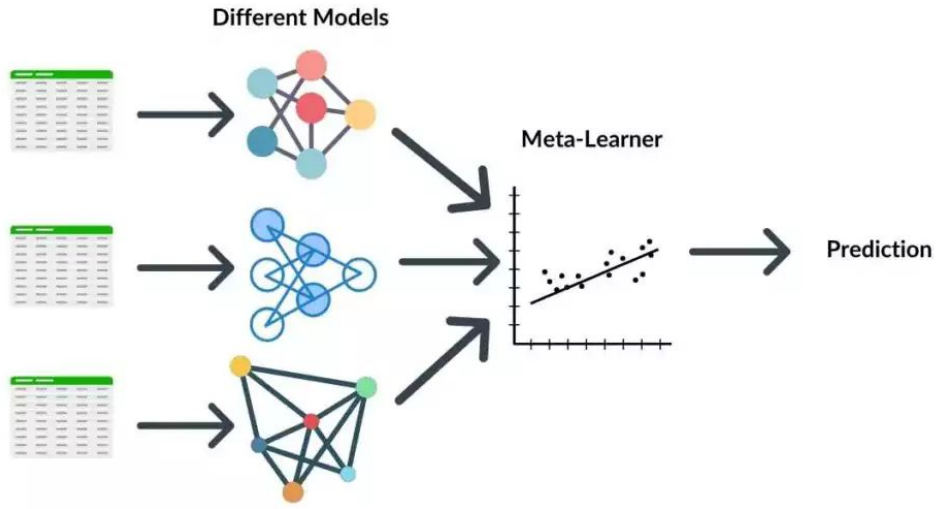
Stacking o apilado

Stacking (Ensamblado por apilado)

Método que combina predicciones de varios modelos base mediante un meta-modelo. Este meta modelo aprende la forma óptima de mezclar las predicciones de los demás para dar con una predicción final.

El uso el stacking permite suplir las carencias de los modelos individuales, reduciendo simultáneamente sesgo y varianza.

Mejora la generalización sin incrementar excesivamente la complejidad y permite integrar algoritmos heterogéneos en una única arquitectura robusta.



4. Conceptos previos

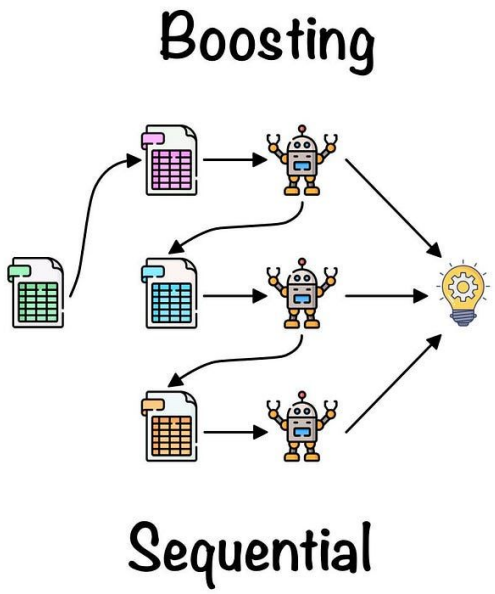
Boosting o ensamblado secuencial

Boosting (Ensamblado secuencial)

El boosting construye una secuencia de modelos débiles (generalmente árboles de decisión) donde cada uno se enfoca en corregir los errores cometidos por los anteriores. En lugar de entrenar modelos independientes, el algoritmo asigna mayor peso a las observaciones mal predichas en cada iteración, obligando al siguiente modelo a priorizarlas.

Esta corrección progresiva reduce principalmente el sesgo y genera un predictor final a partir de aprendices simples.

Su principal ventaja es la alta precisión en problemas complejos con interacciones no lineales, además de incorporar regularización nativa que controla la varianza y previene el sobreajuste.



4. Conceptos previos

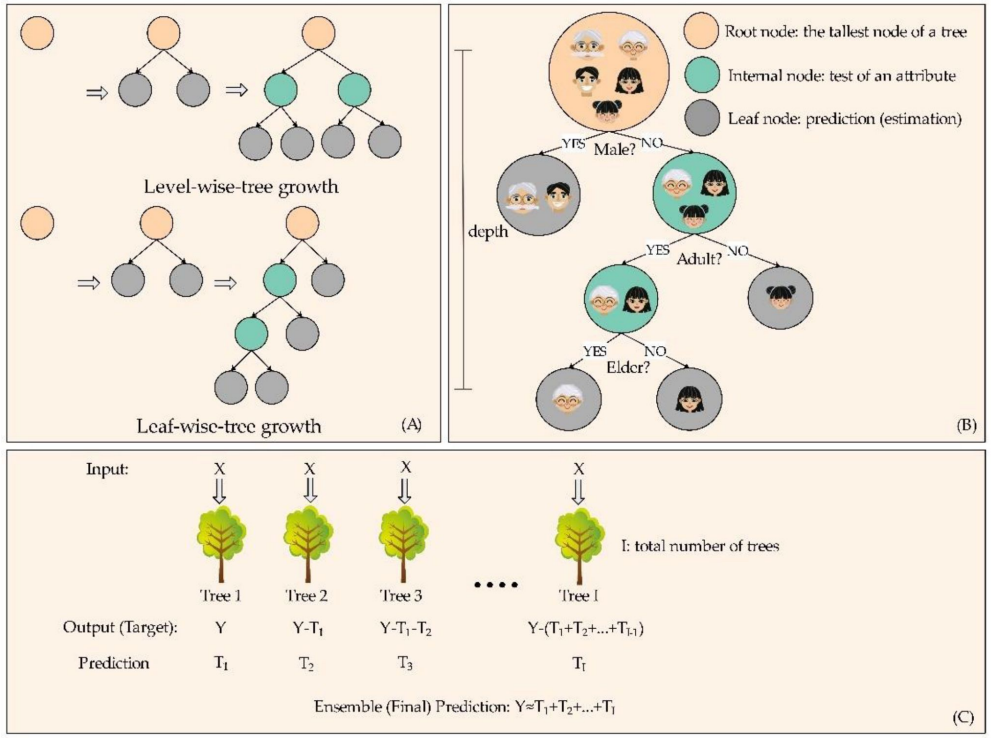
LightGBM

LightGBM es un sistema de **boosting basado en árboles de decisión** desarrolado por Microsoft.

Construye los árboles de forma aditiva, corrigiendo los errores de los árboles previos, y crece los árboles **hacia las hojas**.

Sus **ventajas** incluyen:

- Rapidez en entrenamiento y predicción.
- Bajo consumo de memoria.
- Manejo automático de valores faltantes y categóricos.
- Alto desempeño en regresión y clasificación.

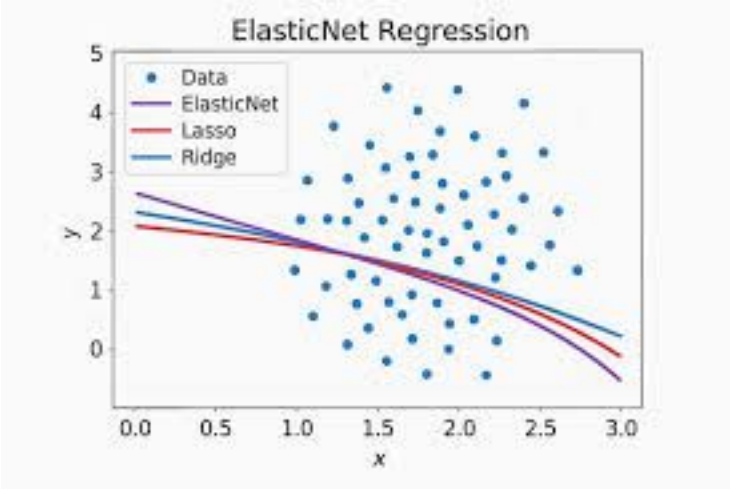


4. Conceptos previos

ElasticNet

ElasticNet es un modelo de regresión lineal que combina **regularización L1 (Lasso)** y **L2 (Ridge)**. Permite ajustar los coeficientes de las variables para reducir sobreajuste, eliminar variables irrelevantes y mantener estabilidad cuando hay correlación entre predictores.

Su principal ventaja es la flexibilidad: logra un balance entre selección de variables y penalización de magnitudes grandes, mejorando la generalización del modelo frente a datos ruidosos o con multicolinealidad. Es útil en problemas donde se requiere interpretabilidad y control sobre la complejidad del modelo.



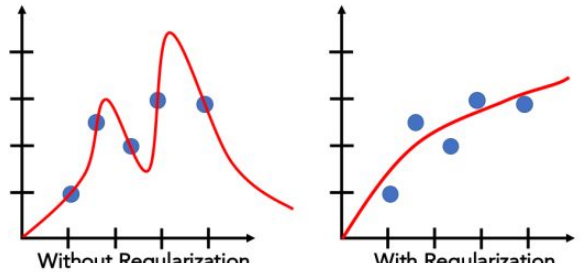
$$\frac{\sum_{i=1}^{n}(y_i - x_i^J \hat{\beta})^2}{2n} + \lambda \left(\frac{1-\alpha}{2} \sum_{j=1}^{m} \hat{\beta}_j^2 + \alpha \sum_{j=1}^{m} |\hat{\beta}_j| \right)$$

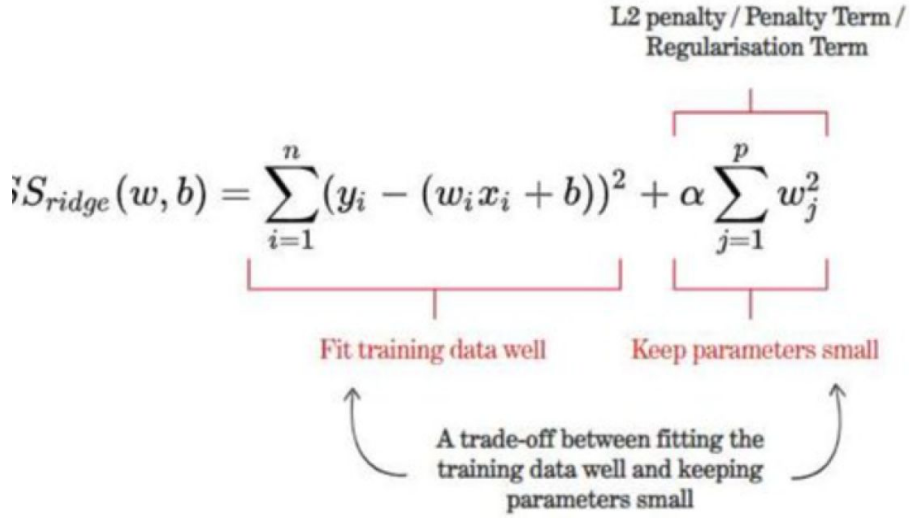
4. Conceptos previos

Ridge

Ridge es un modelo de regresión lineal con **regularización L2**, que penaliza los coeficientes grandes para reducir sobreajuste. Mantiene todas las variables en el modelo, pero ajusta sus magnitudes para mejorar la estabilidad y la generalización frente a datos ruidosos o altamente correlacionados.

Su ventaja principal es que produce predicciones más robustas que la regresión lineal estándar, especialmente cuando hay multicolinealidad, sin eliminar variables, lo que lo hace útil como meta learner en esquemas de stacking.





5. Modelo Propuesto

5. Modelo propuesto

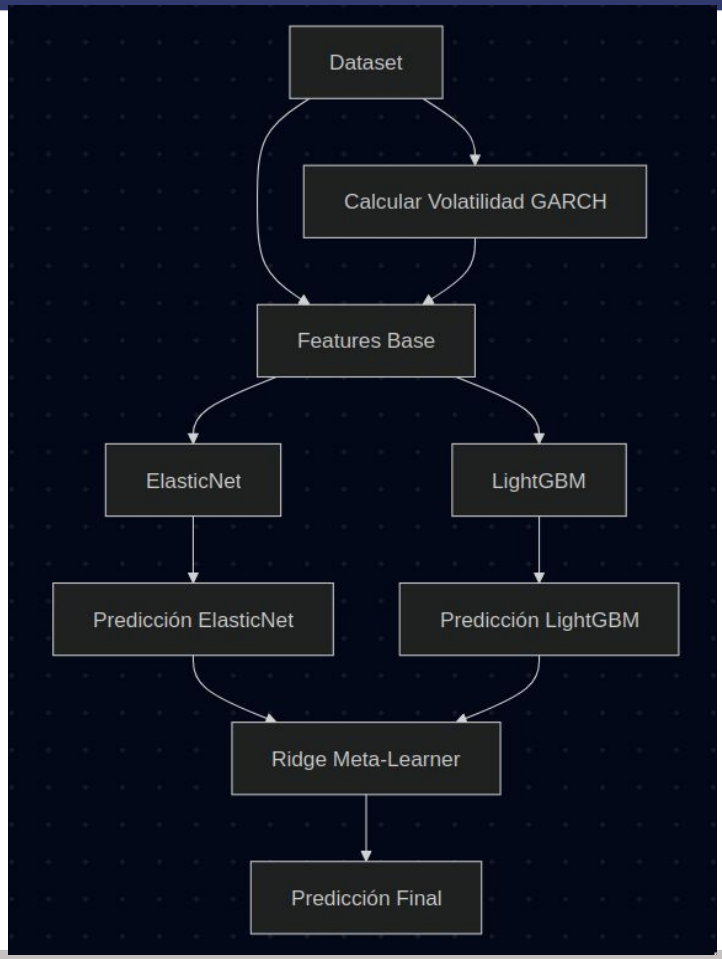
Modelo del stack

El modelo propuesto consiste en un sistema ensamblado basado en Stacking, utilizando los siguientes modelos regresivos:

- **ElasticNet** (penalización L1 + L2)
- **LightGBM** (gradient boost basado en arboles de decision)

Como Meta-learner se utilizará el modelo de regresión lineal **Ridge**.

Los modelos base recibirán como feature un valor de volatilidad calculado mediante el modelo estadístico **GARCH**.



5. Modelo propuesto

Criterios de selección de modelos

Se eligió un esquema de **stacking** porque permite integrar modelos con sesgos y capacidades complementarias, aprovechando la fortaleza individual de cada uno. Frente a otros métodos de ensamblado, el stacking ofrece mayor flexibilidad y robustez.

LightGBM aporta potencia para modelar no linealidades y efectos de interacción. Se prefirió LightGBM sobre XGBoost por su mejor rendimiento y potencial de mayor exactitud, aunque con el riesgo de sobreajuste, que deberá monitorearse.

ElasticNet introduce regularización y robustez ante multicolinealidad.

El meta learner **Ridge** actúa como capa final, estabilizando las predicciones y equilibrando el sesgo y la varianza del conjunto.

Este enfoque logra una representación más completa de la estructura subyacente de los datos sin recurrir a arquitecturas más pesadas como redes neuronales, que implicaría mayor complejidad computacional y riesgo de sobreajuste dada la naturaleza del dataset.

5. Modelo propuesto

Modificaciones sobre el modelo original

La propuesta original de este modelo sufrió dos cambios sustanciales: la eliminación de ARIMA como parte de los modelos base y la suspensión del uso de la volatilidad GARCH como tercera característica en el meta-modelo Ridge.

En el caso de ARIMA, según algunas pruebas realizadas, aunque aporta cierta mejora en las predicciones, el incremento en el costo de entrenamiento no justificaba su implementación.

Por otro lado, incorporar las predicciones de GARCH directamente en el meta-modelo añade demasiado ruido, lo que afecta negativamente la calidad de la inferencia.

6. Metodología de entrenamiento

6. Metodología de entrenamiento

Preprocesamiento de los datos

Antes del entrenamiento, los datos de train.csv se limpian manejando valores faltantes mediante imputación con **forward-fill**.

La volatilidad **GARCH** se calcula a partir de columnas V* y se integra como feature clave en todos los modelos.

$$r_t = \ln\left(\frac{P_t}{P_{t-1}}\right)$$

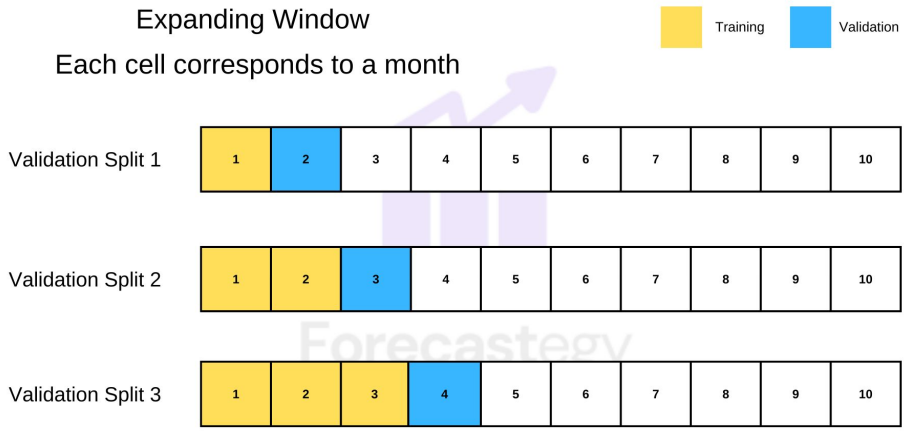
6. Metodología de entrenamiento

Entrenamiento de Modelos Base con Sliding Window

Los modelos base se entrenan usando un esquema de **TimeSeries Split con expanding window** para recorrer el dataset completo.

Es decir, se divide la serie en varios cortes ordenados en el tiempo. En cada corte, el modelo entrena con todos los datos disponibles hasta ese punto y valida en el bloque inmediatamente posterior. La ventana de entrenamiento crece con cada iteración.

GARCH se incorpora como parámetro en cada base learner para modelar volatilidad dinámica.



6. Metodología de entrenamiento

Entrenamiento del Meta-Learner

Una vez entrenados los modelos base, se vuelve a utilizar el dataset completo para el entrenamiento del meta-modelo.

El meta-learner Ridge solo recibe como features las predicciones de LightGBM y ElasticNet, aprendiendo a ponderarlas.

Para las pruebas se utiliza el dataset de test proporcionado por el desafío de la plataforma Kaggle.

Se implementa un backtesting para evaluar el desempeño diario de la estrategia de inversión, asignando fondos (de 0 a 2) en función de las predicciones generadas usando datos hasta el día previo.

7. Ajuste de Hiperparametros

7. Ajuste de hiperparametros

HiperParametrización de LightGBM

LightGBM entrena usando un conjunto explícito de parametros de entrenamiento, los cuales definen la estructura del modelo y su comportamiento durante el boosting.

El entrenamiento utiliza:

- "*objective=regression*" y "*metric=rmse*" como criterios de optimización.
- *"learning_rate=0.05"* para controlar la magnitud de cada actualización,
- *"num_leaves=100" y "max_depth=10"* para fijar la complejidad máxima de los árboles, y "*min_data_in_leaf=30"* para evitar divisiones demasiado pequeñas.
- El muestreo se regula con "*feature_fraction=0.8"*, "*bagging_fraction=0.8"* y "*bagging_freq=5"*.

Con estos valores la librería construye árboles secuenciales hasta un máximo de 2500 iteraciones, pero el proceso se detiene de forma automática mediante ***early-stopping*** cuando el error deja de mejorar, lo que determina la profundidad efectiva del boosting **sin necesidad** de una búsqueda **explícita** de hiperparámetros.

7. Ajuste de hiperparametros

HiperParametrización de ElasticNet

ElasticNetCV (la libreria utilizada) ajusta automáticamente sus hiperparámetros mediante validación cruzada interna. El modelo prueba distintas combinaciones de regularización total (alpha) y mezcla entre L1 y L2 (l1_ratio), entrenando y evaluando cada una para elegir la que logra el menor error.

En esta configuración se exploran alphas en un rango logarítmico amplio y l1_ratio en valores prefijados. Con ello, el procedimiento identifica de forma automática el nivel óptimo de penalización y la proporción adecuada entre ambas regularizaciones.

Parámetros utilizados en el entrenamiento:

- l1_ratio = [0.1, 0.5, 0.9, 1.0]
- alphas = logspace(-5, 1, 20)
- cv = 5
- max_iter = 50000
- n_jobs = -1
- random_state = 42

7. Ajuste de hiperparametros

HiperParametrización de Ridge (meta-modelo)

RidgeCV utiliza el mismo enfoque de validación cruzada interna que ElasticNetCV, pero solo ajusta el nivel de regularización L2 (alpha).

El modelo prueba distintos valores de alpha, entrena una versión para cada uno y selecciona automáticamente el que ofrece el menor error promedio.

El entrenamiento de meta-modelo se realiza sobre todos los datos de entrenamiento, pero los features que recibe son únicamente 2: La predicción del modelo LightGBM y la predicción de ElasticNet.

Parámetros utilizados en el entrenamiento:

- alphas = logspace(-3, 3, 13)

8. Métricas de desempeño y evaluación

8. Métricas de desempeño y evaluación

Métrica	Fórmula	Explicación
Raíz del Error Cuadrático Medio **(RMSE)**	$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$	Mantiene las unidades originales de la variable objetivo, facilitando la interpretación directa del error promedio en la predicción. Penaliza fuertemente los errores grandes.
Error Absoluto Medio **(MAE)**	$\text{MAE} = \frac{1}{n}\sum_{j=1}^{n}\lvert y_j - \hat{y}_j \rvert$	Calcula la diferencia absoluta entre cada valor real y su valor predicho, luego promedia todas esas diferencias.
Sharpe Ajustado (métrica oficial del desafío)	$S = \frac{R_p - R_f}{\sigma_p}$	Rendimiento excesivo dividido por volatilidad, penalizando estrategias con >120% volatilidad del mercado (métrica oficial Kaggle).

9. Resultados del modelo

9. Resultados del modelo

Resultados del entrenamiento

Al finalizar la fase de entrenamiento, disponemos de la información detallada sobre los parámetros y configuraciones obtenidos para cada uno de los modelos:

ElasticNet

- Alpha: 0.0616
- L1 ratio: 0.1
- Iteraciones: 48

LightGBM

- Learning rate: 0.03
- Boosting rounds: 5000
- Max depth: 10
- Número de hojas: 128

Meta modelo Ridge

- Alpha: 3.16
- Coeficientes:
  a. ElasticNet = 0.51,
  b. LightGBM = 0.15

9. Resultados del modelo

Mediciones sobre el modelo

Luego de la fase de entrenamiento realizamos un backtesting sobre un histórico extenso de 8990 días (el dataset de testing) para validar su desempeño.

¿Cómo se realizó la evaluación?

- Se usaron los datos del dataset proveído por la competición para simular la estrategia día a día, calculando la posición óptima en el mercado según las predicciones.
- La posición calculada se multiplica por el retorno real del mercado para obtener el retorno efectivo de la estrategia.
- A partir de estos retornos, se calcularon las métricas previamente definidas para medir su desempeño.

9. Resultados del modelo

Mediciones sobre el modelo

El **Sharpe** y **Sharpe ajustado** indican que el modelo logra una buena rentabilidad incluso al considerar penalizaciones por volatilidad. El **MAE** bajo implica que las predicciones del modelo están muy cercanas a los valores reales en promedio, y un **RMSE** bajo muestra que los errores grandes son poco frecuentes.

En conjunto, estos resultados reflejan que el modelo es preciso y estable para predecir el comportamiento esperado.

MÉTRICA	**RESULTADO**
Sharpe Ratio	1.8586
Sharpe Ajustado	1.8586
MAE	0.004514
RMSE	0.006055

10. Comparación con otros modelos

10. Comparación con otros modelos

Modelos a comparar

Se evaluaron también dos aproximaciones adicionales para comparar el rendimiento de nuestro modelo con otras arquitecturas.

El primero corresponde a un modelo simple basado en ARIMA.

El segundo es una réplica de una solución presentada por el usuario **Samoilov Mikhail** en Kaggle, que combina XGBoost, ElasticNet, Ridge y un modelo “GARCH” (simulado, no real) con asignación manual de pesos, muy similar a la arquitectura que proponemos.

10. Comparación con otros modelos

Resultados

COMPARACIÓN FINAL			
Métrica	MI MODELO	HULL TACTICAL	ARIMA(5,0,1) BENCHMARK
Retorno anualizado	15.33%	20.75%	~4-7%
Volatilidad anual	8.25%	23.49%	~18-25%
Sharpe Ajustado	1.8586	0.8590	~0.22-0.55
Vol vs límite 120%	0.410x	1.167x	~0.8-1.3x
Penalización	0.0%	2.8%	~0-10%
MAE	0.004514	0.007198	~0.009-0.012
RMSE	0.006055	0.010284	~0.012-0.015
Días evaluados	8990	8990	8990

10. Comparación con otros modelos

Conclusión

La comparación final muestra que nuestro modelo logra un balance superior entre retorno y riesgo en relación con los otros enfoques evaluados.

Aunque el modelo Hull Tactical presenta un retorno anualizado más alto, también exhibe una volatilidad considerablemente mayor y una penalización por exceso de riesgo, reflejando menor eficiencia en la gestión del riesgo.

En contraste, nuestro modelo alcanza un Sharpe Ratio casi el doble, con una volatilidad mucho más controlada y sin penalizaciones, lo que indica una estrategia más consistente y estable. Además, los errores frente al benchmark (MAE y RMSE) son menores, confirmando una mayor precisión en las predicciones a lo largo de un período amplio de evaluación.

11. Trabajos futuros

11. Trabajos futuros

Para trabajos futuros, sería valioso considerar las siguientes áreas de mejora:

- Optimizar la selección y ajuste de hiperparámetros para maximizar el desempeño
- Experimentar con diferentes configuraciones y arquitecturas, incluyendo cambiar el meta-modelo para mejorar la combinación de predicciones
- Explorar técnicas más avanzadas de selección y reducción de variables para simplificar el modelo sin perder precisión
- Evaluar nuevas estrategias de entrenamiento y limpieza de datos que puedan mejorar tanto la eficiencia como la calidad de las predicciones.

12. Conclusión

12. Conclusión

Este trabajo presenta una estrategia robusta de predicción financiera que combina modelos avanzados como ElasticNet, LightGBM y un meta-modelo Ridge para optimizar la asignación de posiciones en el mercado.

A través de una cuidadosa ingeniería de características, validación temporal y ajuste de hiperparámetros, se logró un modelo eficiente que equilibra rendimiento y estabilidad, superando tanto modelos simples como soluciones más complejas en métricas clave como el Sharpe ratio y la volatilidad ajustada.

Los resultados demuestran que una combinación bien diseñada de técnicas puede ofrecer una solución práctica y efectiva para la predicción de retornos en mercados financieros.

Gracias por su atención

Librerías utilizadas

Librería	Propósito
pandas	Manejo y Preprocesamiento de Datos
numpy	Modelado, Ensemble y Métricas
lightgbm	Gradient Boosting
statsmodels	Modelos estadísticos (ARIMA)
pmdarima	Auto-selección ARIMA
arch	Modelado GARCH
optuna	Visualización de resultados
seaborn	Visualización avanzada

Bibliografía y referencias

Libros:

- Bishop, C. M. (2006). Pattern Recognition and Machine Learning. Springer.
- Geron, A. (2019). Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow (2nd ed.). O'Reilly Media.
- Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.
- Kubat, M. (2017). An Introduction to Machine Learning (2nd ed.). Springer.
- Ng, A. (n.d.). Machine Learning Yearning. [Manuscript no publicado].

Competencia de Kaggle:

- Blair Hull, Petra Bakosova, Laurent Lanteigne, Aishvi Shah, Euan C Sinclair, Petri Fast, Will Raj, Harold Janecek, Sohier Dane, and Addison Howard. Hull Tactical - Market Prediction. https://kaggle.com/competitions/hull-tactical-market-prediction, 2025. Kaggle.
- Samoilov Mikhail. (n.d.). Hull Tactical Data Analysis. Kaggle. https://www.kaggle.com/code/samoilovmikhail/hull-tactical-data-analysis
- Imaad Mahmood. (n. d). Hull Market Prediction. Kaggle. https://www.kaggle.com/code/imaadmahmood/hull-market-prediction

Artículos en Línea y Blogs:

- Chesa, S. (n.d.). Stacking Ensembles: Combining XGBoost, LightGBM, and CatBoost to Improve Model Performance. Medium. https://medium.com/@stevechesa/stacking-ensembles-combining-xgboost-lightgbm-and-catboost-to-improve-model-performance-d4247d092c2e
- InsightBig. (n.d.). GARCH vs ML Models vs ANNs: Which One for Volatility Prediction. https://www.insightbig.com/post/garch-vs-ml-models-vs-anns-which-one-for-volatility-prediction