Trading Algorítmico

Allison Quintero

Pedro Villegas

Jesús Samuel Benjumea

Hassan Amid Chedraui

Dilger Felipe Becerra

Universidad EAFIT
Maestría y Línea de Énfasis en Ciencias de Datos y Analítica
Edwin Nelson Montoya Munera

2 de Junio de 2025

Introducción

En los mercados financieros modernos, caracterizados por su alta volatilidad e inmediatez, el trading a corto plazo se ha consolidado como una estrategia popular entre inversionistas que buscan beneficios rápidos a partir de fluctuaciones de precios en tiempos relativamente cortos. Sin embargo, esta modalidad de inversión conlleva desafíos significativos, particularmente relacionados con la toma de decisiones bajo presión, la sobrecarga informativa y la influencia de emociones y sesgos cognitivos.

Sesgos como el de confirmación, anclaje, sobreconfianza o aversión a la pérdida distorsionan la percepción racional del trader, afectando negativamente la ejecución de operaciones en momentos clave de entrada o salida del mercado. Estas decisiones, si se toman de manera impulsiva o emocional, pueden comprometer la rentabilidad individual e incluso contribuir a una mayor ineficiencia del mercado.

Este proyecto integrador, desarrollado por el equipo Quant Traders, propone una solución tecnológica orientada a mitigar el impacto de estos sesgos mediante el diseño de un sistema automatizado de apoyo a la toma de decisiones de inversión en tiempo semi real. La propuesta se fundamenta en principios de las finanzas conductuales, análisis técnico, ciencia de datos y la teoría de eficiencia de los mercados, con el objetivo de generar señales de trading objetivas, oportunas y respaldadas por datos empíricos.

El desarrollo de este sistema sigue la metodología CRISP-DM, abordando etapas que van desde la comprensión del problema y la exploración de los datos, hasta la implementación de modelos estadísticos y de aprendizaje no supervisado para la detección de patrones en precios e indicadores financieros. Se utilizan técnicas de álgebra lineal, estadística inferencial y análisis de series temporales, utilizando los conocimientos adquiridos en diferentes cursos del programa de maestría en ciencias de datos.

Como resultado, se busca no solo optimizar el desempeño individual del trader, sino también contribuir a la eficiencia global del mercado financiero, reduciendo la incidencia de comportamientos irracionales. De este modo, se fortalece la transparencia, la estabilidad y la calidad de la participación en los mercados de capital.

Marco Teórico

El presente proyecto utiliza diferentes conceptos como la hipótesis de eficiencia de los mercados, el análisis técnico y los modelos computacionales de ciencia de datos aplicados a la toma de decisiones de inversión. Su objetivo central es mitigar el impacto de los sesgos cognitivos en el trading a corto plazo mediante el uso de modelos automatizados de apoyo, basados en información empírica y procesamiento en tiempo semi real.

Finanzas Conductuales y Sesgos Cognitivos:

A diferencia de la teoría económica clásica, que asume agentes perfectamente racionales, las finanzas conductuales explican cómo las emociones, intuiciones y atajos mentales influyen en las decisiones económicas, incluso en contextos de alta presión como el trading intradía. Kahneman y Tversky (1979), en su teoría de la perspectiva, demostraron que los individuos evalúan las ganancias y pérdidas de forma asimétrica, lo que genera decisiones inconsistentes con la teoría de la utilidad esperada.

En el contexto del trading, varios sesgos cognitivos se manifiestan recurrentemente:

***Sesgo de confirmación*:** búsqueda selectiva de información que refuerza creencias previas.

Sesgo de anclaje: tendencia a fijarse en un valor inicial como punto de referencia, incluso si es irrelevante.

Sobreconfianza: sobreestimación de la capacidad para predecir el mercado, lo cual lleva a asumir riesgos excesivos.

Aversión a la pérdida: propensión a mantener posiciones perdedoras demasiado tiempo, mientras se liquidan posiciones ganadoras rápidamente por miedo a perder las ganancias.

Estos sesgos afectan la calidad del timing de las operaciones y son difíciles de detectar por parte del propio operador, ya que se activan de forma inconsciente ante la presión y el dinamismo del mercado.

Hipótesis de los Mercados Eficientes y Oportunidades de Corrección:

La Hipótesis de los Mercados Eficientes (EMH), formalizada por Eugene Fama (1970), sostiene que los precios de los activos financieros reflejan toda la información disponible en el mercado, haciendo imposible obtener rendimientos superiores de forma sistemática. No obstante, se ha demostrado que esta hipótesis no siempre se cumple, especialmente en el corto plazo, donde la irracionalidad, los eventos inesperados y los comportamientos masivos no informados generan ineficiencias temporales.

Estas ineficiencias abren la posibilidad de desarrollar sistemas de apoyo a decisiones que ayuden a los traders a actuar de forma más racional y fundamentada. La automatización de la toma de decisiones basada en datos puede contribuir a acercar el comportamiento del mercado a un estado de mayor eficiencia, eliminando ruido emocional y reduciendo la volatilidad artificial.

Análisis Técnico y Señales de Trading:

El análisis técnico es una metodología de estudio del mercado que se basa en el comportamiento pasado de los precios y los volúmenes de negociación para predecir movimientos futuros. Parte del supuesto de que el precio lo descuenta todo, y que existen patrones recurrentes de comportamiento que pueden ser detectados mediante indicadores técnicos. Entre los más comunes se encuentran:

Medias móviles simples y exponenciales (SMA, EMA): suavizan la serie de precios para identificar tendencias.

Índice de Fuerza Relativa (RSI): mide la velocidad y el cambio de los movimientos de precio para detectar sobrecompra o sobreventa.

MACD (Moving Average Convergence Divergence): combinas medias móviles para identificar cambios en la fuerza y dirección de una tendencia.

Bandas de Bollinger: muestran niveles de volatilidad mediante desviaciones estándar alrededor de una media móvil.

Si bien estos indicadores son ampliamente utilizados, su interpretación puede ser subjetiva y propensa a sesgos. Por ello, el presente proyecto busca automatizar su cálculo e

interpretación a través de un sistema que procese datos en tiempo semi real y sugiera señales de compra o venta de forma objetiva.

Estadística Aplicada a la Analítica Financiera:

La estadística provee las bases para modelar la incertidumbre, inferir relaciones y validar hipótesis en contextos donde los datos son ruidosos y variables, como en los mercados financieros. En este proyecto se utilizan herramientas estadísticas tanto en la exploración como en la validación de modelos predictivos:

Distribuciones de probabilidad: para modelar los retornos y la volatilidad.

Regresión lineal y no lineal: para encontrar relaciones entre precios y variables técnicas o fundamentales.

Inferencia estadística y pruebas de hipótesis: para evaluar si ciertas reglas de trading son estadísticamente significativas.

Modelos GARCH (Generalized Autoregressive Conditional Heteroskedasticity): utilizados para modelar la volatilidad condicional, muy relevante en activos financieros.

Todo esto nos permite distinguir patrones reales de ruido aleatorio, y sustenta la confiabilidad del sistema automatizado de señales que este proyecto propone.

Álgebra Lineal y Técnicas de Reducción de Dimensionalidad:

El álgebra lineal es un componente clave en la manipulación de grandes conjuntos de datos financieros, especialmente cuando se trabaja con múltiples activos, variables técnicas y ratios fundamentales. Algunas de las aplicaciones en este proyecto incluyen:

Matrices de covarianza y correlación: para evaluar relaciones entre activos e indicadores.

Distancias métricas (Euclidiana, Manhattan, etc.): para identificar similitudes entre comportamientos de activos o contextos de mercado.

Descomposición en valores singulares (SVD) y análisis de componentes principales (PCA): para la reducción de dimensionalidad y extracción de patrones ocultos en los datos.

Proyecciones ortogonales y mínimos cuadrados: para ajustar modelos predictivos y realizar inferencias lineales sobre precios o retornos.

Estas herramientas permiten construir representaciones simplificadas y eficientes de los mercados, que luego pueden alimentar algoritmos de clasificación, clustering o predicción.

Ciencia de Datos y Metodología CRISP-DM:

La ciencia de datos es el eje metodológico y técnico del presente trabajo, permitiendo integrar herramientas de distintas disciplinas para construir un sistema automatizado de señales de trading. La metodología elegida, CRISP-DM (Cross Industry Standard Process for Data Mining), facilita este proceso a través de seis etapas estructuradas:

Comprensión del negocio: se identifican los sesgos en el comportamiento de los traders como problema clave.

Comprensión de los datos: se exploran fuentes confiables como Yahoo Finance, Finviz y Bloomberg para construir una base robusta.

Preparación de los datos: se generan variables técnicas, se limpian series de tiempo y se normalizan datos para modelado.

Modelado: se aplican técnicas como K-means, PCA, regresiones y sistemas de reglas de trading.

Evaluación: se valida la eficacia de las señales generadas en términos estadísticos y financieros.

Despliegue: se sintetiza la solución mediante dashboards, reportes o simuladores con backtesting.

La implementación automatizada no solo reduce el impacto de los sesgos humanos, sino que también democratiza el acceso a herramientas de toma de decisiones sofisticadas, generando un entorno de inversión más transparente, objetivo y eficiente.

Desarrollo metodológico

Entendimiento del problema, pregunta de negocio o hipótesis:

En el trading a corto plazo las decisiones se tienen que tomar muy rápido, a veces en cuestión de segundos. Esto hace que los inversionistas estén en un ambiente con mucha presión y exceso de información. Por eso, no siempre se toman decisiones racionales, sino que pueden verse afectados por sesgos mentales o emocionales.

Algunos ejemplos de estos sesgos son: el de confirmación, donde uno solo busca info que refuerce lo que ya piensa, ignorando señales contrarias; el de anclaje, que hace que uno se quede pegado a un precio de referencia y no se ajuste a los cambios del mercado; el de sobreconfianza, que lleva a creer que se puede predecir el mercado mejor de lo que en verdad se puede; y el de aversión a la pérdida, que puede hacer que se mantengan posiciones perdedoras con la esperanza de que repunten, o se vendan ganadoras muy rápido por miedo a perder lo ganado.

Estos sesgos impactan el momento en el que uno entra o sale del mercado, lo que es super importante en operaciones de corto plazo. Si se entra tarde, se puede perder la subida, y si se sale antes de tiempo, se gana menos. También si uno se queda demasiado tiempo en una baja, las pérdidas se agrandan.

Debido a lo anterior, se plantea la siguiente pregunta problema:

¿Es posible diseñar un modelo algorítmico en el que utilizando datos financieros objetivos en tiempo casi real, reduzca la influencia de los sesgos cognitivos y mejore el timing en las decisiones de trading?

Con base en la anterior pregunta se formula la siguiente hipótesis:

El uso de indicadores técnicos integrados en modelos automatizados permite tomar decisiones de trading más eficientes que las basadas en intuición, al disminuir la influencia de los sesgos cognitivos.

Análisis Exploratorio de Datos:

Entendimiento de los datos:

Para este proyecto usamos datos históricos por minuto de varias acciones, entre ellas Apple, Tesla, Microsoft, Meta, Google, Amazon y Nvidia. La idea era tener un dataset amplio y con frecuencia alta para poder captar los movimientos del mercado.

Cuando empezamos a revisar los datos, nos dimos cuenta de que antes de agosto del 2022 había varios errores y registros mal cargados. Entonces, decidimos eliminar todo lo anterior a esa fecha. Si bien se perdió un poco de tamaño en el dataset, preferimos trabajar con datos limpios y confiables, ya que eso es clave para que el modelo funcione bien.

Preparación de los datos:

Primero se realizó una limpieza general del conjunto de datos. Esto incluyó convertir la columna de tiempo al formato timestamp, lo cual facilita el manejo de series temporales y permite hacer análisis ordenados por fecha. También se eliminaron registros que contenían valores nulos, duplicados o inconsistentes, ya que podrían afectar la calidad del análisis y el rendimiento del modelo.

Posteriormente, se generaron nuevas columnas con ratios financieros calculados a través de UDF utilizando librerías en Python. Esto permitió enriquecer el conjunto de datos con métricas relevantes para el análisis técnico y la construcción del modelo.

Entre los indicadores calculados se encuentran varios de los más utilizados en trading, como la media móvil simple (SMA) de 20 periodos, útil para observar tendencias; el RSI, que mide el momentum y ayuda a identificar zonas de sobrecompra o sobreventa; y las Bandas de Bollinger, que reflejan la volatilidad del precio en relación con su media móvil.

Finalmente, se construyó una variable llamada target, que indica si en cierto momento se presenta una oportunidad de compra. Esta toma el valor de 1 cuando se espera un aumento

positivo en el precio, y 0 en caso contrario. Su función principal es servir como referencia para entrenar el modelo de predicción a partir de los indicadores técnicos definidos.

Análisis descriptivo e insights importantes (correlación, causa-efecto, etc.):

Con los datos ya limpios y con los indicadores técnicos calculados, se llevó a cabo un análisis exploratorio por cada uno de los activos financieros incluidos en el estudio. El objetivo era detectar relaciones relevantes entre variables que pudieran aportar información útil para el modelado posterior. En términos generales, las bases de datos estaban bien estructuradas: no se presentaron valores nulos y las variables binarias como is_above_sma, rsi_overbought o target mostraban una codificación consistente entre cero y uno.

Uno de los patrones más evidentes fue la alta correlación entre el precio de cierre (close) y la media móvil simple de 20 periodos (sma20), con coeficientes mayores a 0.999 en todos los casos. También se encontró una relación muy fuerte entre las bandas de Bollinger (bbhigh y bblow) y el precio, lo cual tiene sentido ya que estos indicadores están directamente construidos a partir de la misma media móvil. Igualmente, se observó una fuerte correlación entre bbhigh y bblow, lo que confirma su comportamiento simétrico frente al precio.

En algunos activos, como GOOGL y MSFT, se identificó una correlación moderadamente alta entre el RSI (rsi20) y el retorno diario (return_1d), con coeficientes cercanos a 0.72 utilizando el método de Spearman. Esto indica que el RSI podría tener cierto valor predictivo en estos casos y servir como variable para anticipar movimientos alcistas o bajistas en el corto plazo.

Además, al revisar las distribuciones de las variables, se observó que tanto los retornos horarios como los diarios (return_1h, return_1d) presentan una distribución centrada en cero, pero con colas largas, lo cual es característico en series de tiempo financieras. En el caso de la volatilidad (volatility_10), se encontró una distribución asimétrica, sobre todo en acciones como NVDA y TSLA, donde fueron frecuentes los picos de variabilidad.

Durante este análisis, también se notó que algunas de estas distribuciones tienden a cambiar con el tiempo. Este comportamiento indica la posible presencia de data drift, lo cual puede poner en riesgo la capacidad del modelo para generalizar en nuevos periodos. Por esta razón, se sugiere implementar validación temporal (por ejemplo, time series cross-

validation) y un esquema de reentrenamiento periódico que permita mantener la precisión del modelo ante condiciones cambiantes del mercado.

Por último, se generaron gráficos de dispersión entre variables como sma20 y close, donde se confirmó visualmente que existe una relación prácticamente lineal, lo cual refuerza el valor de estos indicadores como insumos para el modelo, aunque no sean suficientes por sí solos.

¿Qué objetivo teníamos?:

El objetivo fue desarrollar un sistema de machine learning que permitiera generar **señales de compra y venta** en activos financieros, específicamente:

- **Señal 1**: entrar en posición (comprar).
- **Señal 0**: salir de la posición (vender o mantenerse fuera).

El propósito final es lograr que un modelo aprenda patrones de mercado y genere decisiones que **superen al rendimiento de una estrategia Buy & Hold**, al menos en consistencia o reducción de drawdown.

¿Qué datos usamos para entrenar?:

Los datos provienen de **archivos históricos por activo (ticker)**, cada uno conteniendo:

- timestamp: marca temporal.
- close: precio de cierre.
- volume: volumen negociado.

A estos datos les agregamos indicadores técnicos como features:

- sma20: media móvil simple de 20 periodos.
- rsi20: índice de fuerza relativa de 20 periodos.
- bbhigh / bblow: bandas de Bollinger (alta y baja).

- return_1h y return_1d: retornos pasados.
- volatility_10: volatilidad de corto plazo.
- volume_change: variación porcentual del volumen.
- Condiciones booleanas como:
    - is_above_sma
    - rsi_overbought
    - rsi_oversold
    - close_near_bbhigh

Target:

Se definió un target binario:

target = ((close.shift(-1) / close - 1) > 0.003).astype(int)

1 si el cierre siguiente sube más de 0.3%, 0 en caso contrario.
Esto fuerza al modelo a aprender **señales rentables**, no solo si sube o baja.

Modelos utilizados:

Entrenamos tres modelos principales, todos de clasificación binaria:

a. Regresión Logística (LogisticRegression)

- Modelo base y rápido.
- Interpretable y fácil de calibrar.

b. Árbol de Decisión (DecisionTreeClassifier)

- Permite aprendizaje de reglas claras (if-then).
- Capaz de modelar relaciones no lineales.

c. Random Forest (RandomForestClassifier)

- Ensamble de múltiples árboles (bagging).

- Mejora generalización y reduce overfitting.

Todos fueron entrenados con los **mismos datos escalados** (StandardScaler) y separados en un 80/20 split temporal.

Hiperparámetros utilizados:

Modelo	**Hiperparámetros Clave**
Regresión Logística	max_iter=1000
Árbol de Decisión	max_depth=5, random_state=42
Random Forest	n_estimators=100, max_depth=7

Métricas utilizadas:

Las métricas fueron calculadas sobre los datos de validación (20%) y también usadas en el backtesting final:

Métricas de clasificación.

- **Accuracy**: porcentaje de predicciones correctas.
- **Precision**: cuántas señales de compra fueron correctas.
- **Recall**: cuántas subidas reales fueron detectadas.
- **F1-score**: balance entre precision y recall.
- **AUC-ROC**: qué tan bien separa las clases 0 y 1.
- **Hamming Loss**: tasa de errores (cuanto menor, mejor).
- **Jaccard Score**: coincidencia entre predicho y real.
- **Matthews Correlation Coefficient (MCC)**: calidad general de clasificación binaria.

Métricas del Backtesting.

- Retorno total del portafolio modelo.

- Retorno Buy & Hold.
- Número de operaciones realizadas.
- Valor final del portafolio (con capital real y comisiones).

¿Qué resultados buscábamos?

El objetivo era:

- Que el modelo diera **señales precisas de entrada** (evitando ruido).
- Que **evitará drawdowns** grandes.
- Que el portafolio simulado basado en señales **superara o igualara a Buy & Hold**, especialmente en estabilidad o reducción de pérdidas.

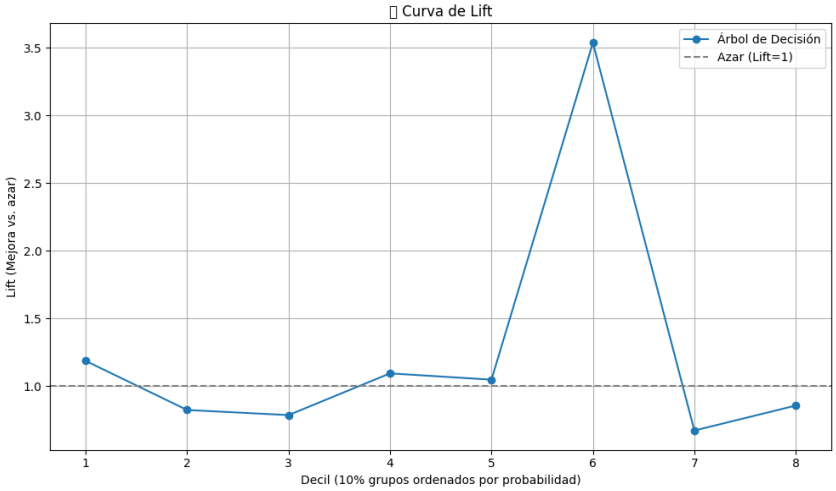
Resultados de las métricas:

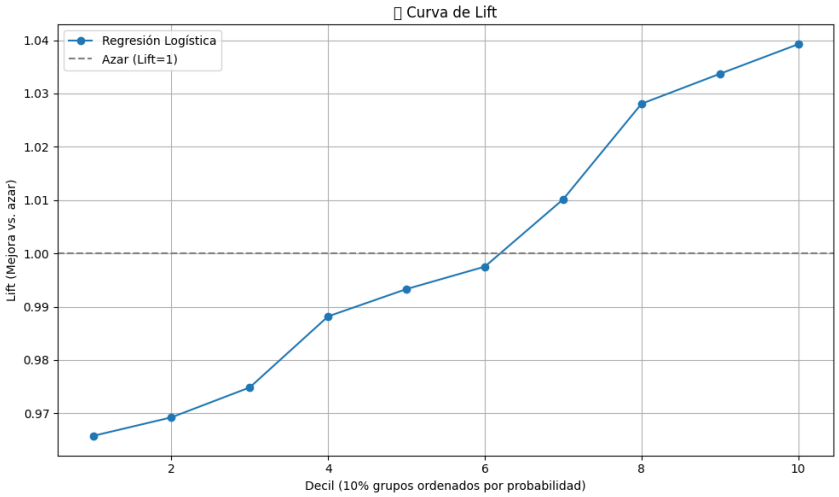
Modelo	**Accuracy**	**Precision**	**Recall**	**F1**	**AUC**	**MCC**
Regresión Log.	0.5268	0.4977	0.0386	0.0716	0.5164	0.0096
Árbol Decisión	0.5282	0.5061	0.1072	0.1769	0.5330	0.0221
Random Forest	0.5284	0.5102	0.0759	0.1322	0.5345	0.0205

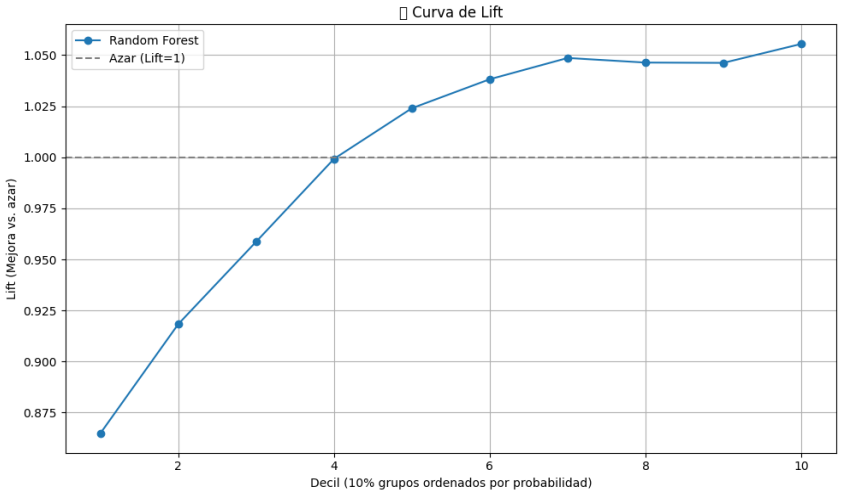
Curva de Lift:

La curva de lift muestra cuántas veces mejor que el azar es tu modelo al identificar positivos (target = 1) cuando clasificas los ejemplos por probabilidad descendente.

- Cuando el target es desbalanceado (pocas compras).
- Cuando te iporta capturar los mejores casos primero (top N señales más fuertes).
- Para comparar modelos más allá de accuracy o AUC.


Curva de Lift
Árbol de Decisión
Azar (Lift=1)
Lift (Mejora vs. azar)
Decil (10% grupos ordenados por probabilidad)


Curva de Lift
Regresión Logística
Azar (Lift=1)
Lift (Mejora vs. azar)
Decil (10% grupos ordenados por probabilidad)


Curva de Lift
Random Forest
Azar (Lift=1)
Lift (Mejora vs. azar)
Decil (10% grupos ordenados por probabilidad)

Resultados Backtesting:

Como se mencionó anteriormente el backtesting se realizó sobre un día específico del mercado, teniendo en cuenta que esperamos usar el modelo de forma intradía, operando a través de los minutos del mercado y que vaya intentando predecir la forma en que este se comportara en base a los indicadores del RSI y SMA.

Durante las pruebas pudimos apreciar que el modelo logra empatar al mercado y ante escenarios bajistas de este como en la gráfica, logra mantener una rentabilidad mayor a la del buy and hold que es la estrategia elegida como comparativa. Siendo así que el modelo logró reducir la pérdida ante el mercado en un punto porcentual (el capital inicial del buy and hold paso del 100% al 97% mientras que el del modelo paso del 100% al 98%). Y aunque esto parezca poco, al manejar capital de miles de millones un uno por ciento puede ser una diferencia entre tu despido y tu ascenso.

Conclusiones sobre los resultados de los modelos y el backtesting:

El modelo logró mantener un mejor control del riesgo, saliendo del mercado en los momentos más volátiles.

Aunque el retorno acumulado final es similar, la estrategia basada en ML muestra menor exposición y volatilidad, lo cual es favorable para gestión de riesgo.

La curva sugiere que el modelo no busca maximizar retorno constantemente, sino proteger capital y capturar subidas selectivamente.

Accuracy cercano al 52% y AUC ROC apenas por encima de 0.5 indican que los modelos están ligeramente por encima del azar, pero sin verdadera capacidad predictiva sólida.

Precision (~50%) sugiere que cuando el modelo predice una compra, lo hace con una eficacia moderada.

Sin embargo, el recall muy bajo (< 11%) implica que detecta muy pocas oportunidades reales de subida. F1 Scores bajos indican un mal balance entre acertar las compras y encontrarlas.

MCC cercano a 0 confirma que el modelo no tiene correlación fuerte entre predicciones y realidad, algo común en datasets desbalanceadas. Jaccard < 10% sugiere que las predicciones coinciden poco con la realidad. Hamming Loss alto (~47%) indica muchas predicciones incorrectas, tanto falsos positivos como negativos.

En los primeros deciles (top 10-20% de señales con mayor probabilidad), algunos modelos muestran un lift mayor a 2x, lo cual es positivo: concentran muchos de los aciertos al principio.

Esto valida su uso para generar top-N señales diarias, incluso si el rendimiento general del modelo es bajo. A medida que bajamos en los percentiles, el lift cae rápidamente hacia 1 (valor azar), mostrando que fuera del top-N, el modelo pierde poder predictivo. Esto sugiere que el modelo no es confiable como clasificador global, pero sí es útil como clasificador por ranking.

Como conclusión general podemos decir que los modelos lineales no son la mejor alternativa para este tipo de predicción en el mercado de valores, pero con un modelo y mayor cantidad de empresas en los datos es posible realizar un modelo más fiel que logre ajustar no solo en momentos de volatilidad del mercado sino también en momentos de calma.

Ingeniería de datos y Cloud Service

La solución de este proyecto dentro de un aspecto de ingeniería de datos y tecnologías de nube se divide en dos componentes, una arquitectura batch para el desarrollo, entrenamiento y procesamiento de los datos históricos naturalmente estructurados así como para el entrenamiento de los diferentes modelos de machine learning implementados, así como una arquitectura de streaming para el despliegue a tiempo real del proyecto.

Para el desarrollo bajo una arquitectura batch se realiza la ingesta, transformación y análisis de los datos crudos, así como el entrenamiento y validación de los diferentes modelos de machine learning tenidos en cuenta.

Los datos crudos representando precios de diferentes acciones por minuto durante un periodo histórico provenientes de una API gratuita (Alpaca) con un límite de uso de 200 llamadas por minuto (no excede nuestra necesidad) bajo una naturaleza estructurada y un

procesamiento en modo batch ya que se trata de agrupar datos con el objetivo de entrenar un modelo.

Para la ingesta inicial de datos (crudos) para el entrenamiento de los modelos se puede utilizar diferentes tecnologías dentro de la nube, para el caso de AWS es posible usar herramientas para la descarga de los datos como AWS Lambda para ejecutar funciones de consumo de la API o bajo una descarga previa de manera local almacenando manualmente en S3 localizados en la zona Raw dentro del datalake, ya que los datos para esta etapa no son necesario de manera continua.

Para el procesamiento de los datos, donde se generan actividades de limpieza, validación, transformación y cálculo de nuevas métricas en un conjunto de datos con millones de registros se pueden utilizar tecnologías como Amazon EMR con motores como Apache Spark para el procesamiento distribuido y paralelo de grandes volúmenes de datos. Además, se pueden utilizar herramientas de catalogación manual (dado la composición de los datos crudos) como Hive SQL o Spark SQL como separación de los datos por acción, sector económico, etc de acuerdo a la evaluación que se pretenda hacer y con Spark DF crear los ratios y métricas de análisis financiero técnico necesarias para los modelos y finalizar creando vistas analiticas sobre los datos transformados. Los resultados se almacenan en la zona Trusted dentro del datalake del S3 correspondiente.

Con estos datos ya es posible pasar a la fase de modelado y entrenamiento del modelo de clasificación, en este caso usamos Spark MLlib escalando el entrenamiento de forma distribuida sobre los datos almacenados entrenando modelos de una manera mucho más eficiente que otras herramientas de uso local como scikit-learn, y guardando este dentro de la zona Refined siendo este el repositorio de modelos dentro del S3 para su futura implementación.

Para el despliegue del proyecto en tiempo real buscando operar intradía con resolución por minuto generando un flujo de datos de streaming es necesario tener en cuenta diferentes tecnologías de nube. La captura de datos puede hacerse desde AWS Lambda programada para que se ejecute cada minuto a través de EventBridge y consulte los precios de una acción, también es posible usar Amazon Kinesis Data Stream (pero requiere de mayores costos) redirigiendo los automáticamente hacia destinos como S3 o Lambda.

Para el procesamiento en tiempo real inicialmente se plantea la utilización del mismo AWS Lambda para los cálculos de las métricas técnicas financieras necesarias, y que después del respectivo procesamiento los datos donde después se dirigirán hacia otra función lambda

conectada mediante el módulo de streaming de Apache Spark Structured Streaming en Amazon para terminar aplicando un modelo Spark MLlib previamente entrenado haciendo un llamado al repositorio de modelos ubicado en S3 para terminar conectándose a un broker (Alpaca) que establezca la posición de compra o venta de acuerdo al resultado proporcionado.

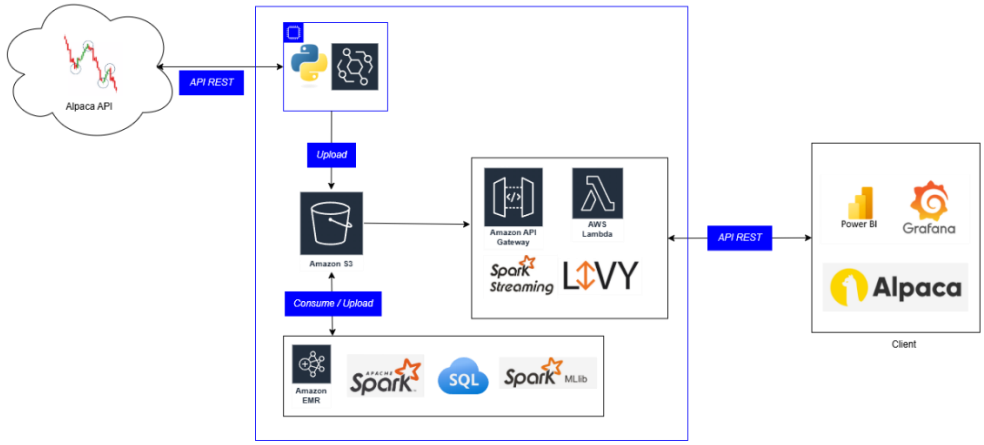
Para registrar las decisiones se utilizarán herramientas como Amazon Glue y Athena, almacenando cada predicción, minuto, variables, S3 para el almacenamiento y consulta de los logs y trazabilidad de estos, o alternativas para datos en series de tiempo como Amazon Timestream para registrar el comportamiento del activo que estamos evaluando así como otras herramientas como índices, volatilidades, niveles de riesgo, etc intentando generar la mayor cantidad de información relevante tanto para el modelo como para los inversionistas que están trabajando con esta herramienta.

El ciclo de vida de MLOps para el proyecto se estructura en dos grandes fases: una fase modelada, orientada al entrenamiento y mantenimiento del modelo, y una fase producción, enfocada en el uso en despliegue del modelo para toma de decisiones intradía. Ambas fases están diseñadas para operar de manera automatizada, monitorizada y con alta capacidad de respuesta ante cambios en los datos o el rendimiento del modelo.

El ciclo comienza con la automatización de los modelos y sus procesos de desarrollo y entrenamiento por medio de tecnologías como Amazon CodeBuild donde se ejecuten scripts desde S3 realizando el procesamiento y entrenamiento de los modelos desde Apache Spark desde Amazon EMR incluyendo su face respectiva de evaluación interna tanto de los procesos en los datos como de los resultados del modelo. Después de la validación el modelo se guarda junto a sus metadatos y scripts correspondientes (modelo de Spark MLlib) almacenados en S3 documentando versiones, fechas de entrenamiento, métricas de validación y parámetros.

Para el despliegue automatizado del modelo, dado que los modelos ed Spark MLlib requieren de un motor Spark para su ejecución, se debe de hacer el despliegue por medio de API Rest sobre Apache Livy donde Amazon Lambda enviará datos cada minuto a Spark cargando el modelo dentro de esa instancia para después analizar los datos que van llegando minuto a minuto. Para el despliegue es posible utilizar Amazon API Gateway, pasar a Amazon Lambda para conectarnos a Livy, implementar los modelos de Spark MLlib para terminar almacenando datos en el S3 para futura auditoría, visualización y análisis periódico.

Diagrama de arquitectura:



Ambiente de procesamiento.

Para desarrollar el proyecto de trading algorítmico, tenemos inicialmente un entorno local que es para experimentar y validar cada etapa del flujo de datos y del modelado. Se usó Python para lo dicho, y además bibliotecas como pandas y NumPy para cargar y limpiar los datos (precios históricos) de las 7 Magníficas. Se organizó la data y los notebooks de codigo para ser ejecutados en orden. Esto nos ayudó a comprobar cada script y mirar como cambian los resultados cada vez que se cambia un indicador técnico, por ejemplo.

Luego se empleó un entorno en la nube con Apache Spark sobre un clúster Amazon EMR. Finalmente, esta parte sirve para manejar volúmenes más grandes de datos y acelerar la generación de “features” para las estrategias de trading. En primer lugar, se limpian los datos como por ejemplo la transformación del formato de la fecha (timestamps), eliminación de registros inconsistentes y el cálculo de SMA20 o el RSI), todo esto en DataFrames de Spark. Luego, Spark MLlib ensambla vectores de características y escala las variables para entrenar los modelos mencionados previamente, aprovechando la capacidad de trabajo en paralelo de EMR. Finalmente, esto sirve mucho, al crecer el proyecto y sumar más símbolos (datos), los tiempos de procesamiento sigan siendo manejables. Además, se utiliza Amazon S3 como DataLake, organizando en zonas RAW los precios descargados de Alpaca, en TRUSTED los datos enriquecidos con indicadores técnicos y en REFINED los artefactos de modelos y los conjuntos listos para producción.

Aplicaciones.

Los datos procesados y los modelos entrenados adquieren valor como para aplicaciones centradas en la consulta, la visualización y la predicción en tiempo real. Por un lado, se usan herramientas de BI y notebooks para explorar cómo se comportan las señales de compra y venta generadas por el modelo. Con Amazon Athena y Glue Data Catalog, se hacen consultas SQL sobre la zona TRUSTED en S3, lo que me permite construir gráficos de indicadores técnicos, curvas ROC de clasificación o curvas de capital acumulado en mis backtests. Estas visualizaciones ayudan a determinar si las estrategias funcionan mejor que un simple “buy & hold” y guían en la elección de hiper parámetros y ventanas temporales.

Por otro lado, la capacidad de emitir señales en tiempo real es para que el proyecto sea realmente útil. Cada minuto, una función de AWS Lambda recopila los precios más recientes desde Alpaca, calcula rápidamente los indicadores críticos (como SMA y RSI) y envía esos datos a un servicio de Spark Structured Streaming o a un endpoint de Apache Livy. Allí, el modelo entrenado en MLlib recibe las variables y dice si se compra o no, cosa que se almacena en Timestream para un análisis histórico o incluso envío directamente al broker para ejecutar operaciones automáticas. Lo ya mencionado asegura que la estrategia de trading no dependa de la máquina local, sino que funcione de forma práctica y escalable.

Finalmente, los notebooks y las instrucciones de ejecución están en GitHub. Cualquiera puede clonar el proyecto, revisar el pipeline completo y adaptar los scripts para probar distintos activos o distintos horizontes temporales. Así, todo lo que está relacionado con visualizaciones, predicción y el almacenamiento en S3 se vinculan directamente con cada fase de la metodología de trading algorítmico, asegurando que los avances en la preparación de datos y en el entrenamiento de modelos se traduzcan en decisiones de inversión concretas y medibles en la práctica.

Conclusión:

Este proyecto ha demostrado de manera práctica cómo se puede construir un sistema de trading algorítmico desde la concepción de la idea hasta la preparación para un despliegue en tiempo real. Más allá de los resultados específicos de los modelos, que muestran un desempeño inicial moderado, pero con potencial de

mejora, el valor fundamental radica en el proceso de aprendizaje y la integración de conocimientos de diversas áreas.

La relevancia de este proyecto en un contexto real es significativa. Se abordó un problema común en el trading: la influencia de los sesgos cognitivos (como el exceso de confianza o el miedo a perder) que pueden llevar a tomar decisiones poco óptimas. La solución propuesta, un modelo algorítmico basado en datos financieros objetivos y análisis técnico, busca precisamente ofrecer una herramienta que apoye la toma de decisiones de compra de manera más sistemática y menos emocional. Aunque los modelos actuales obtuvieron una capacidad de predicción que supera ligeramente el azar, establecen una base sólida. En el trading, incluso un pequeño porcentaje de acierto por encima del 50%, combinado con una gestión de riesgo adecuada, puede ser el inicio de una estrategia rentable.

La arquitectura, que contempla desde la ingesta de datos históricos para entrenamiento hasta un flujo para operar con datos en tiempo real usando servicios en la nube como AWS, muestra un camino claro hacia cómo un proyecto de esta naturaleza podría implementarse en un entorno productivo para tomar decisiones intradías.

Conclusiones sobre aportes de cada materia para el proyecto

- **Álgebra en Ciencias de Datos:** Aunque a veces no es tan visible, el álgebra lineal es fundamental. Se utilizó implícitamente en el manejo de los datos, que son esencialmente tablas (matrices) con múltiples variables (vectores) como el precio de cierre, volumen, SMA20, RSI20, etc. Los modelos de Machine Learning, como la Regresión Logística, se basan en operaciones algebraicas para calcular los coeficientes (pesos) que determinan la importancia de cada variable. Además, la transformación y escalado de características, un paso importante antes de entrenar los modelos, también se apoya en conceptos algebraicos para asegurar que todas las variables contribuyan de forma equitativa.

- **Estadística en Analítica*:*** La estadística fue crucial en múltiples etapas. Se usó para el análisis exploratorio de datos, entendiendo la distribución de las variables, sus correlaciones y para identificar datos atípicos o inconsistencias. Muchos de los indicadores técnicos calculados (como las Medias Móviles Simples, el RSI y las Bandas de Bollinger) son inherentemente estadísticos. Finalmente, la evaluación del rendimiento de los modelos se realizó mediante métricas estadísticas.

- **Almacenamiento y Recuperación de los Datos:** Esta materia fue esencial para diseñar cómo manejar la gran cantidad de datos históricos y futuros. Se implementó una estructura de Data Lake en Amazon S3, dividida en zonas RAW (datos crudos de Alpaca), TRUSTED (datos limpios y enriquecidos con indicadores técnicos) y REFINED (modelos entrenados y datos listos para el consumo). Se utilizaron herramientas como Amazon EMR con Spark para el procesamiento distribuido de grandes volúmenes de datos y el cálculo eficiente de características, y se planificó el uso de AWS Lambda, EventBridge y Amazon Timestream para la ingesta y almacenamiento de datos en tiempo real. La catalogación de datos con Hive SQL o Glue Data Catalog permite que los datos sean fácilmente consultables y utilizables.

Referencias bibliográficas

- Thaler, R. H. (2005). *Advances in Behavioral Finance* (Vol. 2). Princeton University Press.
- Montier, J. (2007). *Behavioural Investing: A Practitioner's Guide to Applying Behavioural Finance*. Wiley.
- Malkiel, B. G. (2003). *The Efficient Market Hypothesis and Its Critics*. Journal of Economic Perspectives, 17(1), 59–82.
- Pring, M. J. (2002). *Technical Analysis Explained* (4th ed.). McGraw-Hill.
- Tsay, R. S. (2010). *Analysis of Financial Time Series* (3rd ed.). Wiley.
- Ruppert, D. (2010). *Statistics and Data Analysis for Financial Engineering*. Springer.
- Strang, G. (2016). *Introduction to Linear Algebra* (5th ed.). Wellesley-Cambridge Press.
- Shreve, S. E. (2004). *Stochastic Calculus for Finance II: Continuous-Time Models*. Springer.
- Chapman, P., Clinton, J., Kerber, R., Khabaza, T., Reinartz, T., Shearer, C., & Wirth, R. (2000). *CRISP-DM 1.0: Step-by-step data mining guide*. SPSS Inc.
- Provost, F., & Fawcett, T. (2013). *Data Science for Business: What You Need to Know About Data Mining and Data-Analytic Thinking*. O'Reilly Media.
- Yahoo Finance. (s.f.). https://finance.yahoo.com
- Finviz. (s.f.). https://finviz.com
- Bloomberg L.P. (s.f.). https://www.bloomberg.com

- Amazon Web Services. (n.d.). Servicios de computación sin servidor: AWS Lambda. Retrieved from https://aws.amazon.com/
- Amazon Web Services. (n.d.). AWS CodeBuild. Retrieved from https://aws.amazon.com/es/codebuild/
- Alpaca. (n.d.). Developer-first API for stock, options, crypto trading. Retrieved from https://alpaca.markets/
- Achelis, S. B. (2001). *Technical Analysis from A to Z*. McGraw-Hill.
- Bailey, D. H., Borwein, J. M., López de Prado, M., & Zhu, Q. J. (2014). The probability of backtest overfitting. *Journal of Computational Finance, 20*(4), 39–69. https://doi.org/10.2139/ssrn.2326253
- Chicco, D., & Jurman, G. (2020). The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. *BMC Genomics, 21*, 6. https://doi.org/10.1186/s12864-019-6413-7
- Dixon, M. F., Halperin, I., & Bilokon, P. (2020). *Machine Learning in Finance: From Theory to Practice*. Springer. https://doi.org/10.1007/978-3-030-41029-3
- Fernandez, O., & Fernandez, O. (2025). Aprende Qué Es Apache Livy. *Aprender BIG DATA*. Retrieved from https://aprenderbigdata.com/apache-livy/
- Japkowicz, N., & Stephen, S. (2002). The class imbalance problem: A systematic study. *Intelligent Data Analysis, 6*(5), 429–449. https://doi.org/10.3233/IDA-2002-6504
- Krauss, C., Do, X. A., & Huck, N. (2017). Deep neural networks, gradient-boosted trees, random forests: Statistical arbitrage on the S&P 500. *European Journal of Operational Research, 259*(2), 689–702. https://doi.org/10.1016/j.ejor.2016.10.031
- Ling, C. X., & Li, C. (1998). Data mining for direct marketing: Problems and solutions. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining* (pp. 73–79). AAAI Press. https://dl.acm.org/doi/10.5555/3000292.3000303
- López de Prado, M. (2018). *Advances in Financial Machine Learning*. Wiley.
- Neely, C. J., Rapach, D. E., Tu, J., & Zhou, G. (2014). Forecasting the equity risk premium: The role of technical indicators. *Management Science, 60*(7), 1772–1791. https://doi.org/10.1287/mnsc.2013.1825
- Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., … & Duchesnay, É. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research, 12*, 2825–2830. Retrieved from http://jmlr.org/papers/v12/pedregosa11a.html

- Sokolova, M., & Lapalme, G. (2009). A systematic analysis of performance measures for classification tasks. *Information Processing & Management, 45*(4), 427–437. https://doi.org/10.1016/j.ipm.2009.03.002
- Verbeke, W., Martens, D., Mues, C., & Baesens, B. (2011). Building comprehensible customer churn prediction models with advanced rule induction techniques. *Expert Systems with Applications, 38*(3), 2354–2364. https://doi.org/10.1016/j.eswa.2010.08.090
- Amazon Web Services. (s.f.). *Servicios de computación sin servidor: AWS Lambda*. https://aws.amazon.com/
- Alpaca. (s.f.). *Developer-first API for stock, options, crypto trading*. https://alpaca.markets/
- Fernandez, O. (2025). *Aprende qué es Apache Livy*. Aprender Big Data. https://aprenderbigdata.com/apache-livy/
- Amazon Web Services. (s.f.). *AWS CodeBuild: Servidor de compilación administrado*. https://aws.amazon.com/es/codebuild