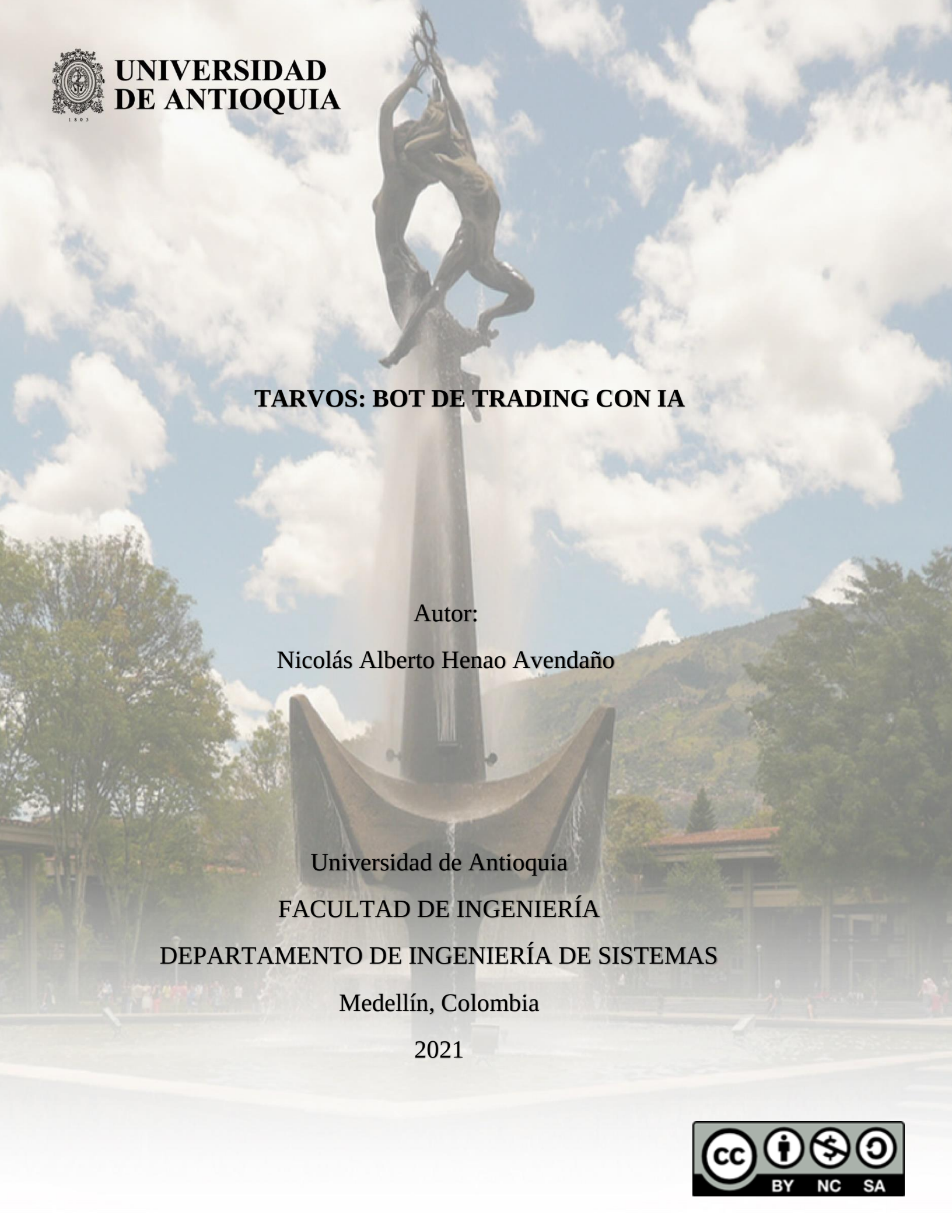

UNIVERSIDAD DE ANTIOQUIA

TARVOS: BOT DE TRADING CON IA

Autor:

Nicolás Alberto Henao Avendaño

Universidad de Antioquia

FACULTAD DE INGENIERÍA

DEPARTAMENTO DE INGENIERÍA DE SISTEMAS

Medellín, Colombia

2021

Tarvos: bot de trading con IA

Autor:

Nicolás Alberto Henao Avendaño

Informe práctica empresarial como requisito parcial para optar al título de:

Ingeniero de sistemas

Asesores:

César Augusto Arias Chica

Jaime Humberto Fonseca Espinal

Universidad de Antioquia

FACULTAD DE INGENIERÍA

DEPARTAMENTO DE INGENIERÍA DE SISTEMAS

Medellín, Colombia

2021.

1. Resumen

Muchas personas en el mundo del mercado financiero tienen como su finalidad la búsqueda de métodos que permitan distintas predicciones sobre los comportamientos de los mercados, ya sea teniendo en cuenta factores externos como internos. Esta búsqueda tiene detrás una gran retribución, ya que el poder entender mejor los comportamientos de los mercados permite tener un mayor control sobre las inversiones realizadas.
Este proyecto se enfocó principalmente en la extracción y transformación de datos de varios mercados, y la posterior búsqueda de un modelo de aprendizaje de máquina que permitiera operar sobre distintos mercados financieros con la finalidad de poder generar portafolios basado en los rendimientos que ofrecieran los distintos mercados.
Además se tuvo la finalidad de adaptar estos modelos a plataformas ampliamente utilizadas para inversiones en los mercados financieros.

2. Introducción

La extracción de información relevante en las series de tiempo financieras siempre ha sido tema de debate entre sus analistas. Traders fundamentales, técnicos o arbitradores tienden a definir su estrategia de análisis como la más óptima, sin embargo esta discrepancia da cuenta principalmente de la riqueza dinámica del sistema, que lleva a las diferentes percepciones desde diferentes saberes. Los sistemas dinámicos financieros han sido estudiados como parte de análisis no lineal de series de tiempo, desde una perspectiva puramente técnica; o tomando teorías de las ciencias de la decisión y tendencias macroeconómicas, todo esto con el fin de estimar el signo de pendiente más probable en un mercado particular. [1]
Las tecnologías usadas para tal fin son muy variadas. Metodologías manuales como: la detección de patrones heurísticos en las velas (representación propia de la serie de tiempo de un mercado que marca puntos importantes de lapsos definidos)[2] y el estudio de indicadores[3] y noticias macroeconómicas[4]; o también se ha usado de manera automática: el análisis no lineal de series de tiempo[5], aprendizaje de máquina[6], teoría de caos[7], simulación basada en agentes[8], entre muchas otras.

En este trabajo exploramos algunas metodologías de aprendizaje de máquina para predecir el comportamiento de las acciones con el fin de optimizar portafolios de inversión. Además implementamos un sistema que permite adaptar un modelo predictivo a plataformas de *trading* por medio de una interfaz.

3. Objetivos

3.1. Objetivo general

Desarrollar un sistema automático de creación de portafolios con inteligencia artificial para el mercado de acciones.

3.2. Objetivos específicos

- Dominar el lenguaje NinjaScript para el desarrollo de sistemas automáticos de *trading*.
- Adquirir habilidad en el uso del simulador la plataforma de *trading* de NinjaTrader para la extracción de datos.
- Evaluar por medio de un análisis exploratorio entre los distintos modelos de aprendizaje de máquina tradicional y aprendizaje profundo la conveniencia de uno de ellos.
- Desarrollar habilidades en el manejo de la librería de aprendizaje profundo TensorFlow.
- Programar los flujos de preprocesamiento de los datos del mercado: extracción, caracterización, limpieza (eliminar datos atípicos, redundantes y errados), transformación, balanceo, normalización y extracción de características de los datos.
- Programar y entrenar un modelo de inteligencia artificial que determine momentos aptos para realizar operaciones en el mercado.
- Programar el cálculo de las métricas financieras: tasa de retorno, volatilidad, tasa de Sharpe, tasa de acierto, *drawdown* máximo y duración de *drawdown.*

4. Marco Teórico

El mercado financiero está conformado por todos los mercados donde se transen valores, ya sean acciones, divisas u otros.[9] Dentro del mercado financiero la actividad de compraventa de acciones o divisas se conoce como *trading*[10]. Para el mejor desarrollo de esta actividad es común que se realice análisis técnico sobre los mercados. Edwards, Magee y Bassetti definen el análisis técnico como el estudio de las actividades propias del mercado, en busca de realizar una predicción sobre la tendencia que probablemente tome en el futuro.[3]

Dentro de la actividad del *trading*, cada persona realiza numerosas operaciones, siguiendo una estrategia propia de cada uno. El rendimiento de estas estrategias se ve comúnmente medido por ciertas métricas financieras. Entre estas se encuentran: la tasa de retorno, volatilidad, tasa de Sharpe, tasa de acierto, *drawdown* máximo, duración de *drawdown* y factor de recuperación.

Se debe entender por tasa de retorno la métrica de rendimiento de una inversión que expresa la ganancia total de una inversión, en término porcentual sobre la inversión inicial.[11]

La volatilidad es la medida estadística de dispersión, en este caso será sobre los retornos de las inversiones realizadas, calculándose como la desviación estándar de estos.[12]

La tasa de Sharpe es una métrica de rendimiento de inversiones que se basa en el retorno histórico y la volatilidad de este.[13]

La tasa de acierto se considera como la cantidad de inversiones que resultan en ganancia sobre la cantidad total de inversiones realizadas.

El *drawdown* máximo es la mayor pérdida desde un pico máximo hasta un punto mínimo de un proceso aleatorio. En el estudio actual se calculará sobre un periodo dado de inversiones. Su duración está dada por el tiempo entre la ocurrencia del pico y el punto mínimo.[14]

5. Metodología

Previo al inicio del proyecto fueron decididas las tecnologías a utilizarse en una breve reunión, en este caso consistieron del lenguaje de programación MQL5 [15] que es el lenguaje propio de la plataforma de comercio del mercado financiero Meta Trader 5 y el lenguaje NinjaScript [16] de la plataforma NinjaTrader; también el lenguaje de programación Python [17], lenguaje que cuenta con variedad de librerías de aprendizaje de máquina y aprendizaje profundo, que entre ellas se escogió TensorFlow[18] para la exploración del aprendizaje profundo en el proyecto.
Además de esto, se tomó la decisión de llevar un proceso de desarrollo por medio de ciclos semanales de desarrollo, que se dividieron en distintas etapas que serán descritas a continuación.

5.1. Extracción de datos

Esta etapa inició con una primera tarea enfocada en la familiarización con las plataformas de *trading*, para esto se exploró el uso de Meta Trader 5 y NinjaTrader, los mercados disponibles en estas plataformas y los formatos de los datos. Además se relizó el desarrollo de algunos sistemas de *trading* automatizado con estrategias simples para aprender y adaptarse a los lenguajes de programación MQL5 y NinjaScript, esta etapa fue apoyada del conocimiento y desarrollo previamente existente de productos para estas plataformas por parte de Moonblock.

Al tener ya un conocimiento base de las plataformas se procedió a realizar una selección previa de los mercados que iban a ser utilizados para una primera exploración. Para esto se seleccionaron de la plataforma Meta Trader 5 las acciones más populares, incluyendo Apple, Amazon, Microsoft, entre otras. En NinjaTrader los mercados de acciones requerían una licencia adicional para el acceso a sus datos, la adquisición de esta licencia fue considerada por un tiempo pero últimamente descartada porque salía del presupuesto disponible para el proyecto. Por esto mismo se decidió utilizar datos de los índices más populares en NinjaScript, tales como S&P500, Dow Jones y Nasdaq.

Con los mercados previamente seleccionados, se realizó la extracción de los datos por medio de las herramientas que las plataformas prestaban para este fin. Se revisó posteriormente la información y los datos extraídos de la plataforma NinjaScript eran incongruentes con los datos que se mostraban en la plataforma, por esto fue necesario desarrollar adicionalmente un sistema que permitiera extraer los datos históricos tal como se muestran en la plataforma.

5.2. Transformación de datos

En esta etapa se tomaron varios caminos pensando a futuro en el proceso que llevaría el proyecto, buscando maneras de adaptar los datos a los distintos modelos disponibles tanto de aprendizaje de máquina tradicional como aprendizaje profundo. Con esto se crearon varios conjuntos de datos que fueron transformados de la manera descrita a continuación.

5.2.1. Pristine

Este conjunto fue creado utilizando varias estrategias de *trading* ya existentes, en este caso se probaron varias estrategias individuales del método conocido como "Pristine", que fue diseñado por Greg Capra y Oliver Velez. En este caso se contaba ya con sistemas de *trading* automáticos para las dos plataformas usadas que aplicaban estas estrategias, con ayuda de estos se extrajeron los datos de cada una de las operaciones que la estrategia indicaba.
Para esto también se aplicó una optimización de cada estrategia para cada uno de los mercados previamente seleccionados. Esta optimización resultaba imposible en algunas estrategias para ciertos mercados, por esto se decidió al final utilizar solo la estrategia que mejores resultados obtuvo a lo largo de los distintos mercados, conocida como *Buy Setup*.

5.2.2. Dirección de cada dato

En este conjunto se agregó a cada uno de los datos del mercado un etiquetado considerando la dirección del siguiente dato. Este proceso se realizó con un script en Python que se encargaba de agregar las etiquetas.

5.2.3. Estrategia básica

Para este conjunto la estrategia usada fue abrir una operación en cada ocurrencia del mercado y dejarla abierta hasta obtener una pérdida o ganancia indicada. Debido a que esta estrategia resulta muy poco eficiente en sus resultados, se obtenían conjuntos de datos con desbalances de clases demasiado marcados, por esto también se decidió realizar una optimización de los parámetros de pérdida y ganancia de la estrategia, buscando tener datos mejor balanceados. Inicialmente estas operaciones se intentaron simular directamente en las plataformas de *trading*, pero NinjaTrader no permitía mantener más de una operación abierta al tiempo, por esto se hizo entonces una búsqueda de una librería para Python que permitiera simular las operaciones con los datos extraídos de las plataformas, pero las más populares que fueron probadas solo permitían utilizar datos diarios; por esto fue necesario desarrollar un script adicional en Python que procesara los datos del mercado y simulara las operaciones.

Adicional a la estrategia con la que se extrajo y etiquetó cada uno de los conjuntos de datos, también se tuvo en cuenta el modelo en el que podrían ser usados.
Para esto, por un lado se agregó a cada conjunto de datos una serie de indicadores técnicos que permitirían que cada dato tuviera en sí información temporal del mercado. Estos fueron extraídos directamente de las plataformas de *trading* y en algunos conjuntos de datos fue necesario cruzar la información luego de ser procesados los datos.

Por otro lado, considerando el uso de modelos de aprendizaje de máquina profundos, se decidió incluir la información temporal por medio de estrategias de codificación de series de tiempo como imágenes, tales como RP, MTF y GAF.

5.3. Evaluación de modelos

Para esta etapa se tomaron varias decisiones previas, primero para los dos tipos de modelos se iba a seleccionar un subconjunto de prueba que fuera continuo en el tiempo, por ello se seleccionó el último mes de cada conjunto de datos. A los datos restantes que serían los subconjuntos de entrenamiento y validación, se les aplicó un submuestreo para balancear las clases de operaciones ganadoras y perdedoras.
Por parte de los modelos de aprendizaje de máquina tradicionales, se decidió incluir un proceso de estandarización de variables, además de una selección de variables considerando que se habían extraído demasiados indicadores técnicos. También para estos se consideró realizar un entrenamiento de validación cruzada, utilizando el método StratifiedKFold, que permitiría mantener las clases balanceadas en cada entrenamiento y validación.

Dentro de esta etapa fueron utilizados distintos modelos para el aprendizaje de máquina tradicional, tales como regresión lineal, máquinas de soporte vectorial y modelos basados en árboles de decisión (AdaBoost y ExtraTrees). Por parte del aprendizaje profundo se definió una estructura base de una red neuronal convolucional que fue modificada durante el proceso de exploración.
Teniendo en cuenta el campo de aplicación de estos modelos, las métricas que se seleccionaron para el entrenamiento de los modelos fueron la precisión y el AUROC. La precisión permitiría saber qué efectividad tendrían las operaciones seleccionadas por el modelo, ya que los verdaderos positivos son las operaciones ganadoras. El AUROC permitiría conocer más a fondo sobre la relación entre la cantidad de falsos positivos y verdaderos positivos en distintos umbrales.

Durante este proceso fueron tomadas numerosas decisiones que serán explicadas para cada conjunto de datos con el que se contaba.

5.3.1. Pristine

Con estos conjuntos de datos se obtuvo muy poca cantidad de datos, obteniendo menos de 100 datos en algunos mercados. En la plataforma NinjaTrader, por la naturaleza de los mercados seleccionados, estas estrategias solían tener aún menor frecuencia y eficiencia, por esto se descartaron completamente los datos de esta plataforma. A pesar que el conjunto de datos más numeroso entre los restantes fue uno de aproximadamente 1800 datos, con este se decidió que solamente podría llegar a ser útil explorar los modelos de aprendizaje de máquina

tradicionales, ya que el aprendizaje profundo logra mostrar su eficiencia con cantidades mayores de datos.
Luego de los entrenamientos, ExtraTrees tuvo los mejores resultados.

5.3.2. Dirección de cada dato

Para estos datos se obtuvo resultados un poco mejores a los obtenidos con los datos anteriores, los mejores modelos fueron otra vez ExtraTrees y el modelo de aprendizaje profundo. Con los datos de mercado de Meta Trader 5 el modelo de aprendizaje profundo no logró tener tan buenos resultados como con los que procedían de NinjaScript.

5.3.3. Estrategia básica

Los resultados obtenidos con estos datos fueron similares a los de los datos anteriores.

5.4. Rendimiento financiero

Para poder conocer el rendimiento real que tendrían los modelos en el mercado era necesario poder calificarlos en términos de medidas financieras. Habiendo hecho una búsqueda previa de librerías que permitieran simular operaciones en Python, se descartó esta posibilidad debido a que todas trabajaban con datos diarios. Se procedió entonces a desarrollar un algoritmo que pudiera simular las operaciones que los modelos indicaran y así obtener de estas una serie de métricas financieras, tales como retorno, retorno anual, volatilidad anual, tasa de Sharpe, drawdown máximo y máxima duración de drawdown.

A pesar que los modelos para los datos de la estrategia básica como los de los de dirección de cada dato obtuvieron resultados similares, en esta etapa los resultados fueron totalmente distintos y entre estos solamente obtuvieron buenos resultados los datos de la estrategia básica. Además de esto, los resultados del modelo de aprendizaje profundo fueron mejores. Al contar con pocos datos de la estrategia Pristine, se obtuvo buenos resultados pero con muy pocas muestras en su conjunto de prueba.

5.5. Conexión del modelo

En esta etapa se requería hacer una exploración adicional , tanto en las plataformas de *trading* como en los lenguajes de programación correspondientes. En esta exploración se descubrió que a pesar que existían algunos métodos que permitían realizar la conexión al modelo por medio de sockets o peticiones web, estos métodos se encontraban restringidos por las plataformas, esto hizo que se descartara la posibilidad de probar los modelos que lograron un buen resultado en los mercados que se encontraban en esta plataforma.
Por otro lado, NinjaScript contaba con la posibilidad de realizar peticiones web. Con esto en mente se desarrolló entonces una API con ayuda de la librería Flask, que incluiría el preprocesamiento de los datos y la predicción del modelo. También se escribió el código en NinjaScript que permitiría hacer el llamado desde la plataforma enviando los datos del mercado.

5.6. Prueba real

Para la prueba real sería necesario tener tanto la plataforma de NinjaTrader como la API corriendo continuamente por un tiempo, en este caso se decidió que fuera por al menos 15 días. Para esto fue necesario alquilar un servidor con Windows, que luego de una búsqueda de los servicios disponibles, el de VPSServer fue seleccionado. La API con el modelo fue montada en un servidor personal de uno de los socios de la empresa que tenía disponibilidad para la duración de la prueba.

6. Resultados y análisis

En la primera etapa del proyecto, la cantidad de datos obtenidos varió dependiendo del mercado, se extrajeron datos de año y medio en cada mercado, esto resultó en conjuntos que rondaban desde los 190000 hasta los 240000 datos en los mercados de la plataforma MetaTrader, por parte de los tres índices en NinjaTrader, la cantidad de datos de estos no varió tanto, resultando en conjuntos de datos de aproximadamente 164000 datos. Además de esto se obtuvo también el código que permitía extraer los datos de mercado tal como se mostraban en la plataforma NinjaTrader. La cantidad de datos fue mayor en la plataforma MetaTrader ya que esta genera información incluso cuando los mercados ya se encuentran cerrados, la plataforma NinjaTrader mostraba ser más estricta con los horarios del mercado.

En la etapa de transformación de datos, se tuvo resultados variados, para los datos con la estrategia Pristine aplicada se pueden ver los resultados obtenidos para los mercados seleccionados en MetaTrader en la Tabla 1. Para obtener estos se modificó el código de la estrategia en MQL5 que permitía la extracción de estos datos agregando los indicadores técnicos. Además fue necesario traducir la lógica de este código para la plataforma NinjaTrader, ya que no se contaba con esta estrategia en la plataforma. De los datos provenientes de MetaTrader, fueron seleccionados solamente los datos obtenidos de Apple, ya que contaban con la mayor cantidad de datos y un buen balance de sus datos.
En cuanto a los datos de los mercados de NinjaTrader, la cantidad de operaciones obtenidas no lograban superar las 100, por lo que directamente fueron descartados para la exploración de modelos.

Tabla 1: Conjuntos de datos extraídos aplicando la estrategia Pristine.

	Apple	Amazon	Catterpillar	Chevron	MasterCard	Microsoft	Tesla	Wallmart
# datos	1834	1033	556	141	1013	582	982	492
% ganados	57,47%	53,24%	38,13%	21,28%	45,51%	61,17%	33,81%	38,41%

Para los datos a los que se les aplicó las otras dos estrategias se obtuvo resultados cercanos a los tamaños originales de los datos extraídos, ya que las estrategias eran aplicadas para cada uno de los datos obtenidos, solamente se excluían las operaciones que salieran del horario de los mercados. Los balances de las clases resultantes de la estrategia simple fueron mucho menores a la estrategia Pristine, obteniendo porcentajes de acierto que oscilaban entre el 19% y el 26%. Por otro lado, la estrategia de etiquetar cada dato con la dirección del dato siguiente obtuvo mejores balances oscilando entre 45% y 57% de porcentaje de acierto. Para aplicar estas dos estrategias fueron escritos varios códigos en Python, primero uno que permitía aplicar el etiquetado de cada dato con la dirección del siguiente, también uno que permitía simular la estrategia simple que se realizó con los datos, además este último incluía un proceso de optimización para los parámetros de la estrategia, que buscaría los mejores resultados en cuanto a el balance de clases de los datos resultantes. También fue necesario tener un código en Python que permitiera cruzar los datos obtenidos luego de aplicar las estrategias con los indicadores que fueron extraídos de las plataformas y otro código en Python que permitiera realizar la codificación de la serie de tiempo como imágenes para cada dato en los conjuntos.

Para la evaluación de los modelos se contó con dos códigos que fueron desarrollados en Python que sirvieron para replicar el flujo de trabajo para los entrenamientos y pruebas de cada modelo con los distintos conjuntos de datos. Los mejores resultados para cada modelo y datos de cada plataforma se pueden ver en la Tabla 2. En esta etapa los mejores resultados fueron obtenidos por el modelo de aprendizaje profundo con los datos extraídos de NinjaTrader, a pesar de esto el mismo modelo no logró resultados similares en los mercados de MetaTrader, esto debido a que uno de los datos más importantes para este modelo fue el volumen del mercado, información para la cual NinjaTrader ofrecía una mayor fiabilidad.

La etapa del rendimiento financiero fue en parte simultánea a la etapa de evaluación de los modelos, por lo que fue posible integrar parte de las métricas del mercado requeridas al flujo de trabajo del entrenamiento y evaluación de los modelos, y así poder medir los resultados dentro del mercado de los mejores modelos que iban siendo obtenidos durante el proceso. En esta etapa los resultados contrastaron en parte con los resultados dados según las métricas de los modelos, ya que las métricas del mercado dependían de otras variables internas del mercado. En comparación entre los datos a los que se les aplicó la segunda y la tercera estrategia, contaban con una gran diferencia, la cual era que en los que se etiquetaba la dirección del siguiente dato, esta dirección no indicaba un movimiento de magnitud constante, por esto las operaciones fluctuaban en sus ganancias y pérdidas, resultando en que a pesar de que los modelos tenían un buen rendimiento, las operaciones seleccionadas por los modelos no representaban una estrategia rentable en el mercado.

Tabla 2: Mejores resultados de cada modelo en los datos de prueba

		AUROC	Precisión	Estrategia
Regresión Lineal	MetaTrader	0,5	0,5	2
	NinjaScript	0,51	0,52	2
SVM	MetaTrader	0,53	0,55	3
	NinjaScript	0,51	0,54	2
AdaBoost	MetaTrader	0,61	0,67	1
	NinjaScript	0,63	0,65	2
ExtraTrees	MetaTrader	0,6	0,59	1
	NinjaScript	0,63	0,58	2
CNN	MetaTrader	0,52	0,56	2
	NinjaScript	0,69	0,64	3

El proceso llevado a cabo permitió lograr como resultado principal un sistema de trading automatizado con inteligencia artificial, que permite operar desde una plataforma ampliamente utilizada. Esto permitiría utilizar las operaciones realizadas por este sistema para la creación de portafolios con base en los rendimientos obtenidos.
Además de esto, del proceso también resultaron numerosos códigos que compondrían un flujo de trabajo para futuras exploraciones. También se exploró la posibilidad de conectar servicios externos a la plataforma de NinjaTrader y esto también queda como base para futuros usos.

Los resultados que fueron obtenidos de la prueba real llevada a cabo pueden ser observados en la Tabla 3.
El servicio de VPSServer presentó una caída durante la prueba, que fue solamente de unas pocas horas, por lo que no se podía atribuir a esto el resultado. Además por los tiempos de respuesta de la API, se perdían algunos instantes en que el mercado fluctuaba, que también resultaba en un cambio mínimo pero bastante importante para tener en cuenta a futuro. Por el rendimiento mostrado en esta prueba se decidió que el modelo no prestaba aún los rendimientos requeridos para considerar sus predicciones parte de una estrategia de trading rentable.

Tabla 3: Comparativa resultados en prueba real

	Datos de prueba (1 mes)	Datos reales (15 días)
Tasa de acierto (Precisión)	64,29%	25%
Retorno	4,1%	-0,3%
Volatilidad	0,14%	0,13%
Tasa de Sharpe	29,28	-2,31
Máximo drawdown	0,7%	0,5%
Duración de drawdown	5 días	4 días

7. Conclusiones

- En cuanto a los modelos de aprendizaje de máquina tradicional, se infirió que los ExtraTrees tuvieron buen desempeño porque se adaptaban fácilmente a la naturaleza estocástica de los datos, debido a que conceptualmente también tienen un gran componente aleatorio dentro de su funcionamiento.

- A pesar que fue un camino que no se decidió tomar, los modelos LSTM serían un buen punto para continuar la exploración de modelos debido a que estos resultan más apropiados para problemas relacionados a series de tiempo.

- Por los resultados obtenidos de la prueba real, se infiere que el modelo es dependiente del estado mercado, por esto es necesario tener en cuenta a futuro el considerar variables adicionales del mercado tales como la estacionalidad.

- Debido a rápidas fluctuaciones del mercado, la latencia resulta ser de mucha importancia, por esto el modelo y la plataforma de trading deberían estar en el mismo servidor o al menos en la misma red para reducir lo máximo posible los tiempos de respuesta y la influencia que estos puedan tener.

- Fue posible ver que dentro del aprendizaje de máquina las métricas de eficiencia del modelo no son lo único que importa y métricas específicas de la aplicación que se le está dando pueden afectar el rendimiento final del modelo.

- A pesar que fue posible explorar los modelos sobre acciones de la plataforma MetaTrader, los datos de esta plataforma no daban la fiabilidad suficiente para el modelo que mejores resultados otorgó. En una futura exploración también aportaría al proceso contar con la inversión de adquirir los datos de acciones de la plataforma NinjaScript, donde la fiabilidad de los datos fue mejor.

- En el campo profesional, resulta esencial la adaptación a ambientes multidisciplinarios donde puedan expandirse los conocimientos que estén tanto dentro como fuera del campo del ingeniero de sistemas. Esto con la finalidad de nutrir de distintos puntos de vista las soluciones dadas a los problemas planteados.

- En un ambiente laboral el proceso de desarrollo se puede ver más como un proceso de adaptación a las oportunidades que ofrezca la empresa; tanto en factores de conocimientos, recursos físicos y recursos monetarios, y los limitantes que puedan presentarse en estos mismos aspectos.

- A pesar que la metodología interna de la empresa podía no ser perfecta y conllevar a pequeños retrasos dentro del proyecto, el acompañamiento por parte de la Universidad reforzaba este proceso haciendo más presente las dificultades y los caminos para solucionarlas, permitiendo cumplir con los objetivos y cronograma planteados al iniciar el proyecto.

8. Referencias Bibliográficas

[1] Taylor, S. J. (2008). Modelling financial time series. world scientific.
[2] Leigh, W., Modani, N., Purvis, R., & Roberts, T. (2002). Stock market trading rule discovery using technical charting heuristics. Expert Systems with Applications, 23(2), 155-159.
[3] Edwards, R. D., Magee, J., & Bassetti, W. C. (2018). Technical analysis of stock trends. CRC press.
[4] Graham, B. (2019). El inversor inteligente: un libro de asesoramiento práctico. HarperCollins Espanol.
[5] Taylor, M. P., Peel, D. A., & Sarno, L. (2001). Nonlinear mean-reversion in real exchange rates: toward a solution to the purchasing power parity puzzles. International economic review, 42(4), 1015-1042.
[6] Kim, K. J. (2003). Financial time series forecasting using support vector machines. Neurocomputing, 55(1-2), 307-319.
[7] Kazem, A., Sharifi, E., Hussain, F. K., Saberi, M., & Hussain, O. K. (2013). Support vector regression with chaos-based firefly algorithm for stock market price forecasting. Applied soft computing, 13(2), 947-958.
[8] LeBaron, B. (2002). Short-memory traders and their impact on group learning in financial markets. Proceedings of the National Academy of Sciences, 99(suppl 3), 7201-7206.
[9] Kenton, W. (2020) Financial Markets. Recuperado de: https://www.investopedia.com/terms/f/financial-market.asp
[10] Trading (s.f.) Cambridge dictionary. Recuperado de: https://dictionary.cambridge.org/dictionary/english/trading
[11] Kenton, W. (2020) Rate of Return. Recuperado de: https://www.investopedia.com/terms/r/rateofreturn.asp

[12] Kuepper, J. (2020) Volatility Definition. Recuperado de: https://www.investopedia.com/terms/v/volatility.asp
[13] Sharpe, W. F. (1994). The sharpe ratio. Journal of portfolio management, 21(1), 49-58.
[14] Magdon-Ismail, M., Atiya, A. F., Pratap, A., & Abu-Mostafa, Y. S. (2004). On the maximum drawdown of a Brownian motion. Journal of applied probability, 41(1), 147-161.
[15] https://www.mql5.com/
[16] Begel, A. (1998). NinjaScript: A Dataflow Language for Composing Network Services in Ninja. University of California, Berkeley.
[17] https://www.python.org/
[18] https://www.tensorflow.org/