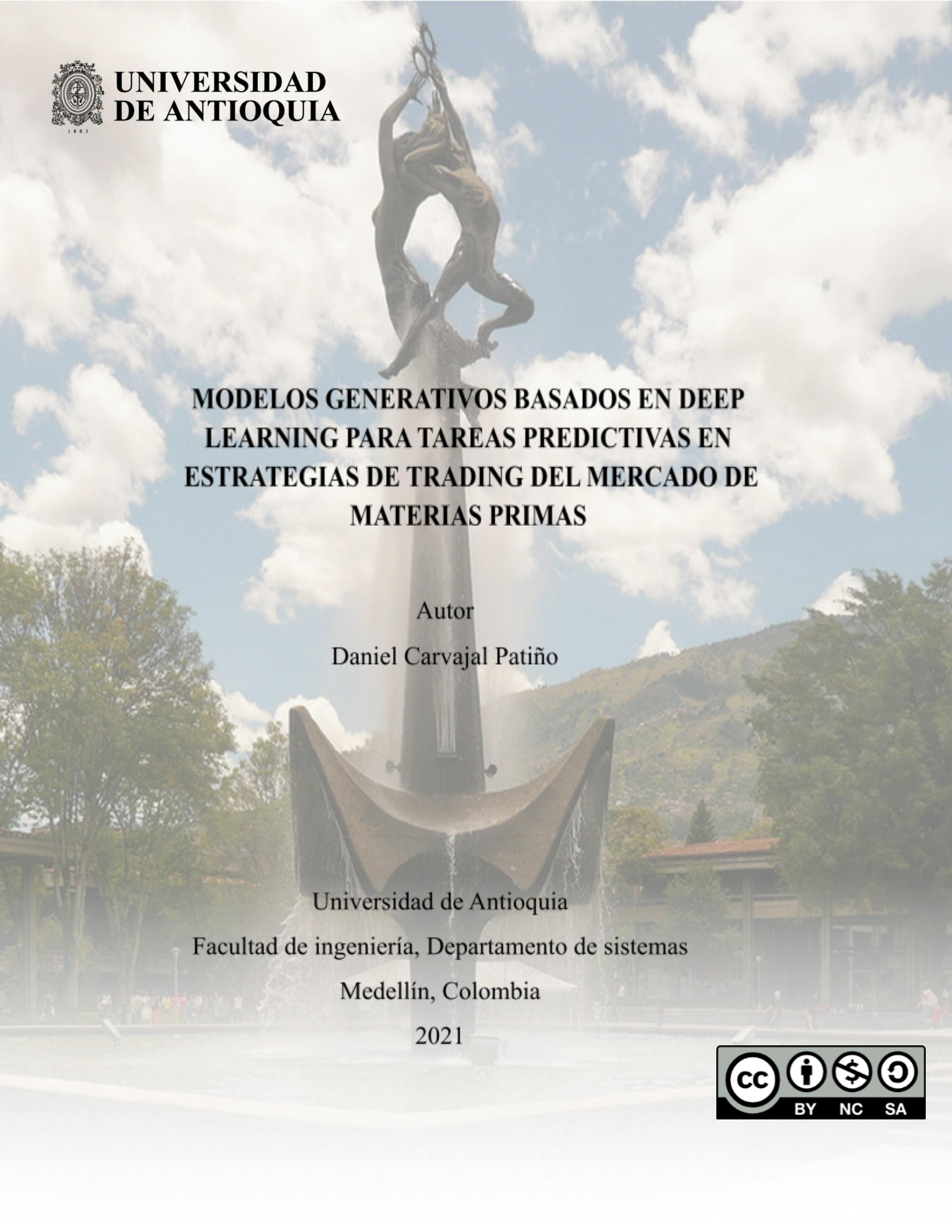


UNIVERSIDAD DE ANTIOQUIA

MODELOS GENERATIVOS BASADOS EN DEEP LEARNING PARA TAREAS PREDICTIVAS EN ESTRATEGIAS DE TRADING DEL MERCADO DE MATERIAS PRIMAS

Autor

Daniel Carvajal Patiño

Universidad de Antioquia

Facultad de ingeniería, Departamento de sistemas

Medellín, Colombia

2021

Modelos generativos basados en deep learning para tareas predictivas en estrategias de trading del mercado de materias primas.

Daniel Carvajal Patiño

Trabajo de investigación presentado como requisito para optar por el título de:
Magister en ingeniería

Asesores
Ph.D Raúl Ramos Pollán

Línea de investigación:
Deep Learning

Grupo de investigación:
In2lab

Universidad de Antioquia
Facultad de ingeniería, Departamento de sistemas
Medellín Colombia
2021

Resumen

Los mercados de materias primas, son mercados a nivel mundial donde se negocian precios de compra y venta de productos básicos como granos, metales, energía, carnes, etc. Muchas compañías realizan compras y ventas en estos mercados con el fin de generar ganancias en la ejecución de estas transacciones (*trading*). La gran mayoría de estas transacciones se basan en especulaciones hechas sobre el precio del mercado. Por ejemplo, si se está seguro de que el precio de un producto va a subir para el día siguiente, conviene comprar en el día actual para luego vender, generando así una ganancia debido a la diferencia de precios entre un día y otro.

De esta manera, tener información con cierto nivel de seguridad sobre el comportamiento del precio para momentos posteriores, resulta ventajoso. Una forma de tener información sobre el comportamiento futuro de un mercado, es realizando predicciones del mismo. Áreas como la econometría, la estadística y el aprendizaje automático (Machine Learning) proponen una gran cantidad de métodos, técnicas y modelos con este propósito. El presente proyecto está ubicado en el área de la inteligencia artificial generando predicciones del mercado del oro para su aprovechamiento en estrategias de *trading*.

Debido a que se trabajó con series temporales y simulaciones, los modelos fueron utilizados bajo un mecanismo de validación cruzada para series de tiempo, lo que limita el uso de datos en los distintos momentos de entrenamiento, sumandole a eso la complejidad de los datos del mercado, lo que podría traducirse en un bajo rendimiento en las predicciones. Por ende, se propone el uso de datos sintéticos producidos por modelos generativos basados en deep learning, para el entrenamiento de los modelos predictivos. Estos datos sintéticos deberán replicar distintos escenarios del mercado de la mejor manera posible para solventar dicha limitante. Cabe mencionar que también se quiso determinar la utilidad de datos sintéticos en contextos financieros.

El desempeño de los modelos predictivos y generativos se midió a través de métricas de desempeño y simulaciones de estrategias de *trading*, al final se desarrolló una metodología experimental, que a través de simulaciones y estrategias de *trading* muestra el impacto de usar datos sintéticos producidos por modelos generativos basados en deep learning en la predicción del mercado de materias primas.

Tabla de contenido

Lista de figuras

Lista de tablas

1. Trabajo de Investigación

1.1. Introducción

Una de las dificultades en la elaboración de proyectos que impliquen el uso de técnicas de Machine Learning es la falta de datos. En la gran mayoría de casos, la obtención de los mismos conlleva un costo elevado, pueden ser datos que reflejan eventos muy escasos o se cuenta con un método de validación que limita el uso de los mismos. Para enfrentar esta problemática existen distintas metodologías, por ejemplo, generar datos sintéticos o hacer sobremuestreo. Hay que tener en cuenta que dependiendo del tipo de datos que se estén manejando alguna técnica podría ser más beneficiosa que otra. En el caso de series de tiempo en ámbitos financieros se presenta esta problemática por varias razones [1].

A pesar de que los mercados de materias primas funcionan a alta frecuencia, es decir, en un corto periodo de tiempo se tiene nueva información del mercado, contar un histórico de este tipo de información es difícil, unicamente grandes compañías que llevan años operando en el mercado podrían contar con esta información, sumado a esto, debido al carácter simulativo de algunos proyectos y para evitar el uso de información del futuro o el ruido generado por datos muy antiguos, los modelos predictivos que se utilizan no cuentan con la información suficiente de la cual sacar provecho y obtener un buen rendimiento. En el caso del presente proyecto se contó con datos frecuentes de 14 meses en el mercado del oro y se desarrolló bajo un carácter simulativo. Estos datos fueron entregados por una compañía que opera en el mercado de materias primas, con el fin de elaborar un proyecto que mejore su rendimiento en el mercado. La ejecución de dicho proyecto permitió la financiación de la maestría, por ende se realiza una alineación en los proyectos.

Teniendo en cuenta que la limitación de datos es un problema, se propuso el uso de modelos generativos basados en deep learning como forma de obtención de datos (data augmentation) [2]. De esta manera se tendrá una mayor cantidad de escenarios de mercado, permitiendo a los modelos predictivos ser más robustos a la hora de enfrentarse a nuevos datos. Sin embargo, el uso de modelos generativos representó un gran reto, ya que obtener un buen funcionamiento de los mismos fue una tarea complicada, [3] para lograrlo, fue necesario hacer varias iteraciones modificando la arquitectura de los modelos aplicando distintos trucos que han funcionado en la literatura, [4] hasta encontrar un modelo que se adaptó a la tarea propuesta mostrando un proceso de entrenamiento deseado.

En la actualidad existen varios modelos generativos basados en deep learning, entre ellos existen las GAN (Generative Adversarial Network), VAE (Variational AutoEncoder), RBM (Restricted Boltzmann Machine) o GTN (Generative Teaching Networks). El objetivo de las GAN, VAE y RBM es generar datos similares a los datos originales, mientras que el objetivo de las GTN se centra en generar datos que ayuden con la tarea de predicción que se tenga, sean similares o no a los datos originales. De esta manera, al entrenar GANs, VAEs o RBMs se debe comprobar que generan datos similares a los del mercado, es decir, que las señales sintéticas muestran un comportamiento o propiedades muy similares o iguales a los del precio del mercado. En cambio, al entrenar GTNs se debe comprobar que los datos generados están ayudando en la tarea predictiva. En el presente proyecto se planteó el uso de algunos de estos modelos para la generación de datos sintéticos con el objetivo incrementar el número de escenarios con los que se entrena cada modelo predictivo. El

desempeño de los modelos generativos fue determinado a través de mediciones de la similitud entre los datos reales y sintéticos como con mediciones del desempeño de los modelos predictivos al usar datos sintéticos y su impacto en estrategias de *trading*.

Como se mencionó las predicciones del mercado sirven como insumo a estrategias de *trading*. En base a como se define la tarea predictiva puede ser difícil conseguir un alto nivel de acierto debido a la naturaleza de los datos. [5] Las series financieras son una secuencia de datos (por ejemplo el precio) que presentan una alta volatilidad y que pueden ser fácilmente influenciadas por la misma señal (technical analysis), como por sucesos o eventos externos (fundamental analysis). En el presente proyecto se definió la estimación de la velocidad del precio como tarea predictiva. Fue necesario un conjunto de iteraciones en busca de un modelo predictivo que se adaptara a la tarea predictiva, esto implicó un poder computacional alto y un extenso periodo de experimentación, hay que añadir que también se comprobó el comportamiento de los modelos al ser entrenados con datos sintéticos producidos por distintos modelos generativos.

Finalmente, se realizaron simulaciones de distintas estrategias de *trading* midiendo su capacidad para generar ganancias realizando operaciones de compra y venta en distintos momentos del mercado. Una estrategia de *trading* consiste en la manera en que se toma una serie de decisiones de compra o venta en distintos momentos del mercado con el objetivo de generar una ganancia, la idea en general consiste en realizar compras a cierto precio y venderlo a uno mayor. Las operaciones pueden ser realizadas en distintos momentos al precio indicado por el mercado, cuando se realiza una de estas operaciones la estrategia gana una posición frente al mercado, por ejemplo en el caso del mercado del oro, si se compran 100oz la estrategia tiene una posición de +100, es decir tiene material y debe venderlo, si en cambio se venden 100oz, la estrategia tiene una posición de -100, es decir tiene una deuda de material y debe comprar, mientras que la estrategia no realice ninguna operación su posición es de 0, es decir no tiene ni debe material. Cabe mencionar que el trámite entre comprar y vender material es denominado *trade*. Por ejemplo, una estrategia tiene una posición de 0, en base a alguna especulación decide vender 250 oz de material, en ese momento se abre un *trade* y se obtiene una posición de -250, luego de un tiempo, cuando la especulación se haya efectuado, se decide comprar 250 oz, en ese momento se cierra la posición, es decir la posición pasa a 0 y se cierra el *trade*.

De esta manera, cada proceso de *trading* (*trade*) tiene dos instantes importantes y están marcados por una operación de venta o de compra. El primer instante consiste en el momento en que se obtiene una posición, el cual es denominado el momento de entrada, el segundo instante es el momento en que se deshace o cierra la posición, denominado momento de salida. Dependiendo de la tendencia del precio, el momento de entrada estará marcado por una operación de compra o de venta, por ejemplo, si el precio está subiendo lo ideal es realizar una compra y obtener una posición, para luego venderla a un precio mayor. Si el precio se encuentra bajando lo ideal es primero realizar una operación de venta y luego comprar por un precio menor. En resumen, comprar barato y vender caro.

1.2. Estado del arte

Los modelos generativos profundos (deep generative models) son modelos que a través del uso de distintas arquitecturas de redes neuronales tratan de aprender la distribución de los datos con los que se les entrena, para luego poder generar muestras de la distribución aprendida. Si el modelo logra determinar la distribución original de los datos, las muestras generadas pueden ser fácilmente tomadas como muestras reales.

Las GANs (Generative Adversarial Networks) son un tipo de estos modelos, con una arquitectura de dos redes neuronales que compiten entre sí, conocidas como discriminador y generador, han tomado gran popularidad en el último par de años. Fueron presentadas por Ian Goodfellow en 2014. [6]

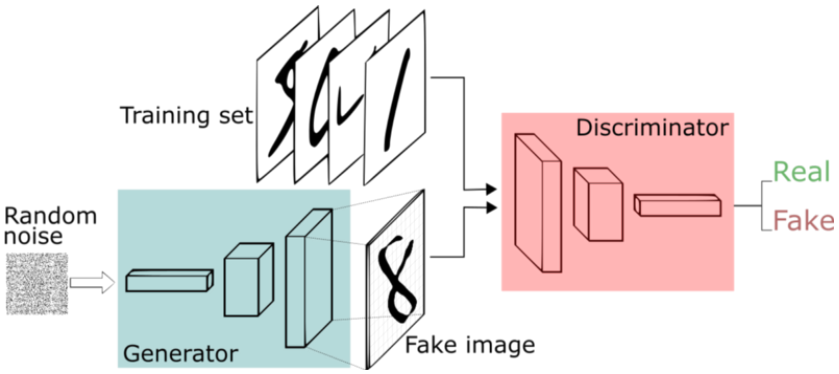


Fig. 1. Arquitectura de una GAN.
Fuente: Adaptado de [7]

Durante el proceso de entrenamiento, la red conocida como generador produce datos los cuales deben ser revisados junto con los datos reales por el discriminador, este, entrenado para clasificar dichos datos como reales o falsos, produce una retroalimentación para el generador, el cual actualiza sus pesos para producir datos que logren engañar la clasificación del discriminador.

Desde su creación, las GANs han sido usadas frecuentemente en la generación de imágenes sintéticas, por ejemplo, existe una página web donde se pueden ver rostros generados por una GAN. En lo que se refiere a la generación de series financieras también se han realizado algunos trabajos.

Por ejemplo, Fernando de Meer Prado [1] toma datos del indicador *S&P500* y genera nuevas señales a través del uso de WGANs, Wassersteins GAN, un tipo especial de estas redes. Luego, comprueba la calidad de las señales generadas verificando Kurtosis, funciones de autocorrelación y otras propiedades de las señales. Finalmente, con los datos reales y sintéticos entrena un modelo ResNet para predecir la tendencia de dicho indicador, obteniendo un nivel de acierto de alrededor del 70%.

Por su parte Antonio Tizzano [8] usa GANs recurrentes para generar nuevos escenarios de mercado, con el objetivo entrenar un modelo de reinforcement learning, una rama del machine learning que se encarga del entrenamiento de agentes con la capacidad de realizar acciones en un entorno con tal de maximizar una recompensa. En este caso la recompensa son los ingresos que se pueden obtener en el mercado (entorno), con una serie de acciones de compra o venta. Tizzano

logra mostrar cómo el uso de datos sintéticos ayuda al agente a obtener mejores resultados, ya que dicho agente es más robusto al contar con más escenarios de los cuales aprender.

Los VAEs (Variational Autoencoders) son modelos generativos cuya arquitectura está basada en los Autoencoders. Mientras que un Autoencoder busca la manera de representar los datos de entrenamiento en un espacio de menor dimensión, con la ayuda de dos redes neuronales, encoder y decoder, un modelo VAE trata de determinar la distribución de la que provienen los datos a través de la misma arquitectura de dos redes, encoder y decoder. Este modelo fue propuesto por Diederik P Kingma y Max Welling en 2014. [9]

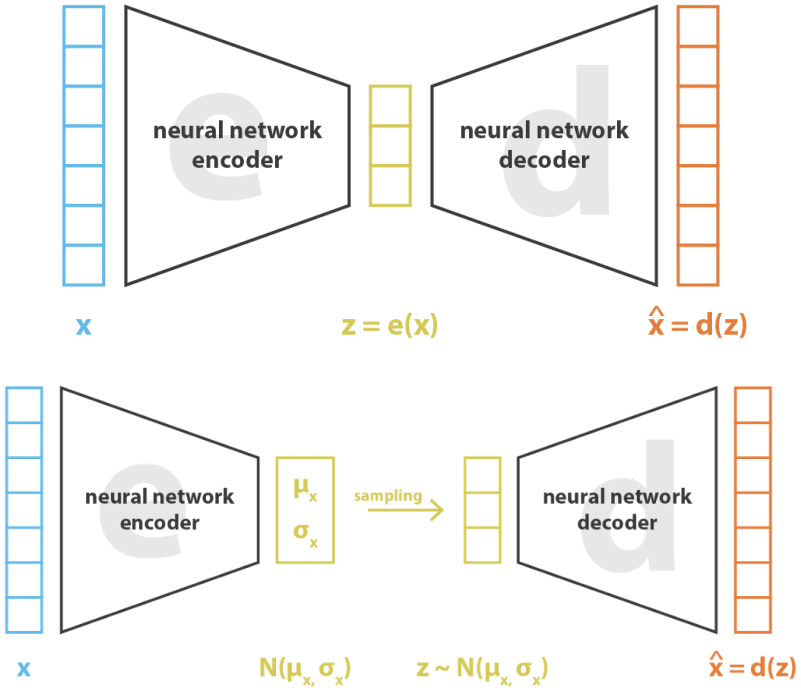


Fig. 2. Arquitectura de una VAE. (a) arquitectura Auto Encoder. (b) arquitectura Variational Auto Encoder.
Fuente: Adaptado de [10]

En un estudio, Chen-Sheng Gu [11] usa modelos VAE para generar características en base a señales del mercado y a indicadores macroeconómicos. Estas nuevas características (salida del modelo VAE) son usadas en el entrenamiento de una LSTM, un tipo especial de red neuronal recurrente, para generar predicciones del mercado. Finalmente, estas predicciones son usadas como insumo a estrategias de *trading*. Los modelos implementados por Chen-Sheng Gu y su equipo, muestran una precisión cercana al 83%. En este estudio se hacen predicciones de la señal del *S&P500*.

Las RBM (Restricted Boltzmann Machines) son modelos que a través de una arquitectura de dos capas es capaz de aprender la distribución de los datos con la que se le entrena. Estos modelos están basados en las máquinas de Boltzmann, las cuales son computacionalmente complejas de entrenar, sin embargo, restringiendo el número de conexiones de estos modelos se logra disminuir dicha complejidad. Estos modelos fueron propuestos por Ruslan Salakhutdinov, Andriy Mnih y Geoffrey Hinton en 2007. [12]

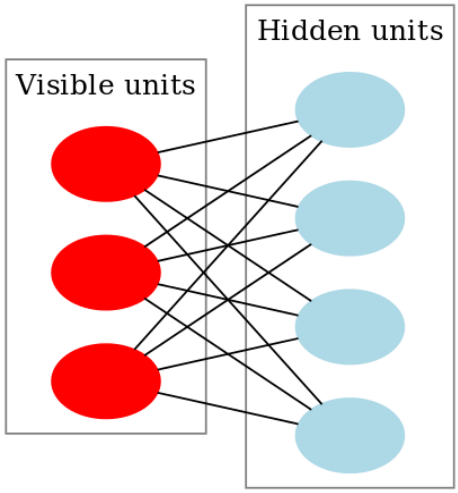


Fig. 3. Arquitectura de una RBM.
Fuente: Adaptado de [13]

Qiubin Liang y su equipo [14] usan modelos RBM para generar nuevas características en base a una señal binarizada del mercado, dichas características son usadas para la predicción del mercado a través de distintos clasificadores, como Random forest, SVM y logistic regression. Realizando comparaciones entre los modelos entrenados con datos originales y generados por las RBM, muestran que el desempeño de los modelos mejora con el uso de las nuevas características. En este estudio se alcanzan niveles de acierto del 61%.

Las GTN [15] (Generative Teaching Networks) son modelos generativos que con una arquitectura de dos redes (basados en las GANs) son capaces de generar datos sintéticos que ayudan a mejorar el entrenamiento de modelos predictivos. Su arquitectura cuenta con un generador que se encarga de producir datos sintéticos con los cuales una red denominada aprendiz es entrenada. Durante el proceso de entrenamiento el aprendiz tratará de predecir datos reales entrenando con los datos sintéticos producidos por el generador, los fallos del aprendiz harán que el generador produzca mejores datos, a través de la actualización de los pesos de ambas redes. Debido a este proceso los datos generados por las GTN no tiene por que ser similares a los datos originales con los que se le entrena.

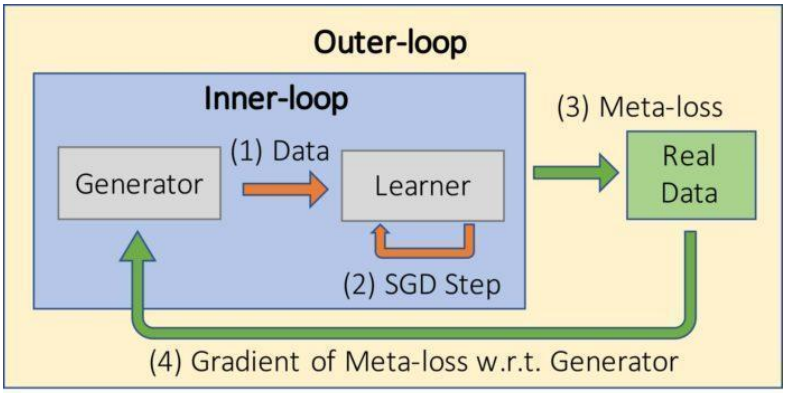


Fig. 4. Arquitectura de un GTN.
Fuente: Adaptado de [15]

Sin embargo, usar datos sintéticos no es el unico metodo para ayudar en el desempeño de los modelos predictivos, por ejemplo, un estudio reciente llevado a cabo por Xing Wang [16] y su equipo, demuestra que los modelos pueden incrementar su desempeño hasta un 20%, usando datos del mercado de acciones transformados a una nueva representación de menor dimensión, a través del uso de una capa embebida, la cual captura la correlación y otras relaciones implícitas entre las distintas acciones del mercado. En otro estudio llevado a cabo en 2018, Zhaojie Luo [17] y su equipo usan la transformada ondícula continua, o continuous wavelet transform en inglés, para descomponer la señal del precio del petróleo y con estos datos entrenar un modelo predictivo profundo, los experimentos realizados en este estudio muestran cómo el uso de la señal descompuesta ayuda en el desempeño de los modelos predictivos.

Como se mencionó, en este proyecto se propone el uso de algunos de los modelos generativos aquí presentados, para obtener nuevos escenarios del mercado con los cuales entrenar modelos predictivos cuyas predicciones servirán como insumo a estrategias de *trading* en el mercado de materias primas.

1.3. Objetivos

General

- Evaluar el impacto que tiene el uso de datos sintéticos generados por modelos generativos basados en deep learning en la predicción del mercado de materias primas durante estrategias de *trading*.

Específicos

- Elaborar un motor de simulación de estrategias de *trading* con datos históricos del mercado de materias primas.
- Implementar modelos predictivos que mediante el uso de datos reales del mercado de materias primas sirvan como insumo a estrategias de *trading*.
- Diseñar modelos generativos basados en deep learning capaces de crear datos sintéticos del mercado de materias primas.
- Implementar modelos predictivos que mediante el uso de datos reales y sintéticos del mercado de materias primas sirvan como insumo a estrategias de *trading*.
- Relacionar el desempeño de las simulaciones, los modelos predictivos y los modelos generativos mediante indicadores financieros y métricas de rendimiento.

2. Generalidades

2.1. Mercado de materias primas

La compañía mencionada anteriormente permitió el acceso a datos del mercado para un periodo de 14 meses que abarcan desde el 2019-01-01 al 2020-02-22. Los datos del mercado proporcionados por la compañía son denominados *fullbooks*. Cada *fullbook* contiene las ofertas de compra y venta realizadas por cada uno de los proveedores del mercado para un instante de tiempo. En promedio se cuenta con un *fullbook* cada 3ms, para todo el periodo de datos, exceptuando fines de semana, ya que el mercado no se encuentra abierto esos días.

Las ofertas de cada *fullbook* son representadas en un par de matrices en donde se muestra la cantidad de material ofrecido, el precio por onza de dicha cantidad y el proveedor que realiza la oferta. De esta manera, cada *fullbook* consiste en dos matrices asociadas a un instante de tiempo, lo que se puede interpretar como el estado del mercado para dicho momento.

```
[[0.0, 1282.2, 200.0],
 [0.0, 1282.4, 300.0],
 [0.0, 1282.5, 100.0],
 [0.0, 1282.6, 3400.0],
 [0.0, 1282.7, 300.0],
 [3.0, 1282.53, 1000.0],
 [3.0, 1282.77, 500.0],
 [3.0, 1282.96, 500.0],
 [3.0, 1283.13, 500.0],
 [3.0, 1283.31, 1000.0],
 [3.0, 1283.5, 1000.0],
 [3.0, 1283.85, 2000.0],
 [3.0, 1284.03, 1000.0],
 [3.0, 1284.75, 2000.0],
 [3.0, 1286.02, 5000.0]]
```

Fig. 5. Ejemplo de una matriz de ofertas perteneciente a un *fullbook*. La primera columna indica el proveedor, la segunda columna el precio y la tercera columna indica el volumen de la oferta. En cada *fullbook* se encuentran dos matrices, una para operaciones de venta y otra para operaciones de compra. Cada vez que una nueva oferta es puesta en el mercado, se obtiene un *fullbook* diferente.

Con estos datos se tiene una descripción detallada del mercado durante los 14 meses con los que se cuenta. Recorriendo cronológicamente cada uno de los *fullbooks* e interpretándose como eventos que van surgiendo en cada momento, es posible ejecutar simulaciones de distintas estrategias de *trading* realizando operaciones de compra y venta en varios momentos de dicho recorrido, si se lleva un registro de cada una de las operaciones realizadas por cada estrategia simulada, se puede determinar el desempeño de cada una de ellas, ya que se puede calcular la ganancia total generada por cada estrategia al final de la simulación, teniendo en cuenta el dinero gastado y obtenido después de cada operación de compra o venta.

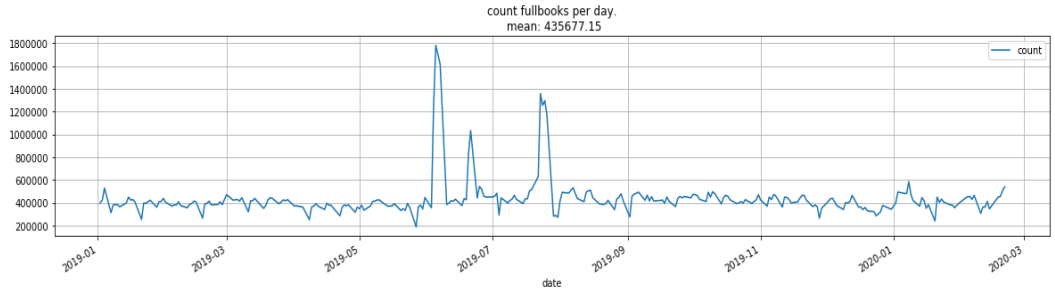


Fig. 6. Cantidad de *fullbooks* por día durante el periodo de datos con los que se cuenta.

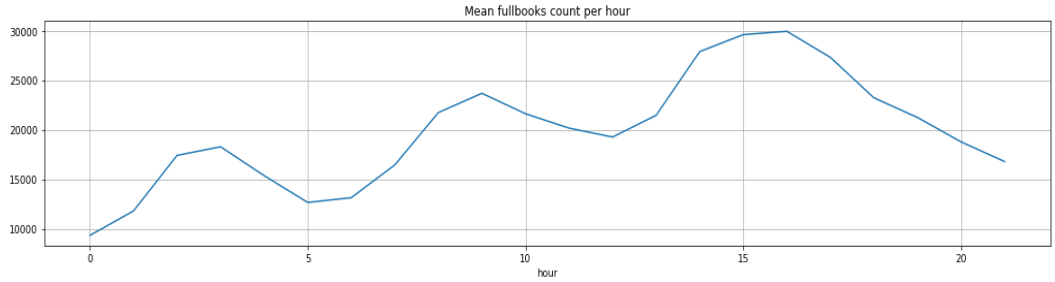


Fig. 7. Cantidad promedio de *fullbooks* por cada hora durante el periodo de datos con los que se cuenta.

En conversaciones con la compañía se determinó que las estrategias de *trading* simuladas realizarán operaciones de compra y venta con un volumen fijo de 100 oz. De esta manera se disminuye la complejidad de las estrategias de *trading* y el riesgo que representa hacer operaciones con un volumen mayor. Teniendo esto en cuenta, es posible construir una señal del precio del mercado calculando la media de comprar y vender 100 oz al mejor postor en cada instante del mercado (en cada *fullbook*). Realizando esto para los 14 meses de datos, se obtiene la siguiente señal.

Fig. 8. Precio del mercado de 100oz. Promedio entre comprar y vender 100oz de material en cada momento del mercado (fullbook).

Cuando se menciona el precio del mercado o la señal del mercado se hace referencia a la señal mostrada en la gráfica anterior. Cabe mencionar que durante la simulación las estrategias de *trading* estarán recorriendo cronológicamente esta señal y cada operación de venta o compra estará marcada por el precio del mercado correspondiente al instante en que se realiza dicha operación.

2.2. Estrategias de trading

Se realizó un conjunto de reuniones con los *traders* de la compañía (personal encargado de las operaciones de *trading*) para conocer y comprender el comportamiento que tienen en el mercado, es decir, conocer cuál es su estrategia, para luego implementarla y simularla.

La compañía realiza sus operaciones de *trading* basándose en la cantidad de movimiento que lleva el precio (*momentum*) en cada instante, si ven que el precio tiene un *momentum* del cual pueden sacar ventaja, realizan operaciones de compra o venta, abriendo un *trade*, cuando consideran que el *momentum* está cambiando o se está desvaneciendo, cierran la posición obtenida. Mientras que el precio no presente un nivel considerable de *momentum* no realizan operaciones de *trading*, ya que no sacarían ganancias con dicho comportamiento del mercado. Por ejemplo, si en algún instante determinan que el precio se encuentra subiendo con un *momentum* considerable, realizan una operación de compra, cuando determinan que dicho comportamiento está terminando, es decir el precio dejará de subir o empezará a bajar, realizan una operación de venta, generando de esta manera una ganancia, ya que vendieron material a un precio mayor al que fue comprado.

De esta manera, hay que tener dos situaciones en cuenta, el momento en que se abre un *trade* (el precio tienen *momentum*) y el momento en que se cierra (el precio a perdido el *momentum*), ya que si se logra tener una diferencia alta, entre el precio del momento de entrada y de salida, asegurando el mayor precio en la operación de venta, se obtendrán ganancias. Hay que aclarar que un momento de entrada puede estar marcado tanto por una operación de compra como de venta, dependiendo de si el precio se encuentra subiendo o bajando. Algunos ejemplos para ilustrar a continuación.

Fig. 9. Ejemplo de un *trade* con una tendencia de bajada en el precio. La línea verde indica el momento de entrada mientras que la línea roja indica el momento de salida del trade. Para generar ganancias lo ideal consiste en realizar una operación de venta en el momento de entrada.

La figura 9 muestra un ejemplo de un *trade* que es abierto una vez se detecta una cantidad de movimiento considerable en el precio con tendencia de bajada (línea verde), y cerrado cuando se determina que dicho comportamiento se pierde (línea roja). Ya que se decide abrir un *trade*, se asume que el precio continuará con dicha tendencia (bajada), de esta manera, la operación que marca el momento de entrada es una venta, obteniendo así una posición negativa. En el momento en que se detecta que dicho comportamiento en el precio ha cambiado, la posición es cerrada con una operación de compra. De esta manera, se obtiene una ganancia, ya que el precio de venta fue mayor que el de compra. Visto de otra manera, lo ideal es vender material en el momento indicado por la línea verde (momento de entrada) y comprar material en el momento indicado por la línea roja (momento de salida), de esta manera, se vende a un precio mayor que el de compra. En este caso el *trade* ocurre entre la línea verde y la roja.

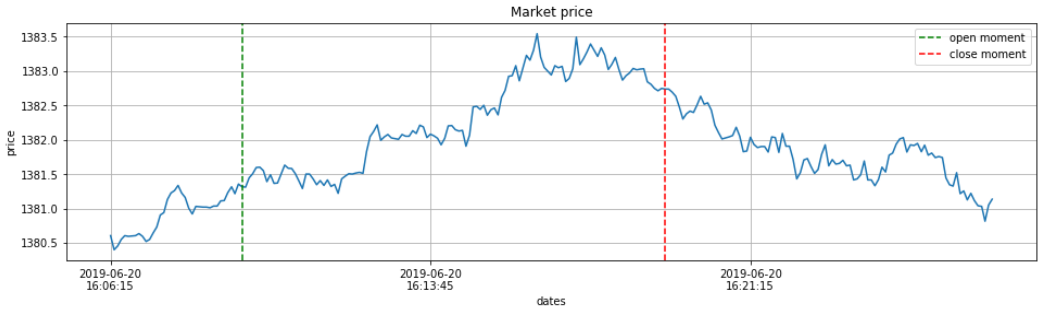


Fig. 10. Ejemplo de un *trade* con una tendencia de subida en el precio. La línea verde indica el momento de entrada mientras que la línea roja indica el momento de salida del trade. Para generar ganancias lo ideal consiste en realizar una operación de compra en el momento de entrada.

De igual manera que el ejemplo anterior, la figura 10 muestra dos momentos idóneos para realizar operaciones de *trading*, pero esta vez, con una tendencia de subida en el precio. Para este ejemplo, luego de detectar una cantidad de movimiento en el precio, se realiza una operación de compra, luego al determinar que dicho comportamiento ha cesado, se vende el material obtenido con anterioridad, generando de igual manera una ganancia. De manera ideal y para los ejemplos anteriores, los momentos de entrada y salida podrían estar ubicados algunos minutos antes para generar ganancias aún mayores, sin embargo hay que tener en cuenta que estos procesos de *trading* son realizados en tiempo real y en cada instante no se sabe que puede pasar con el precio.

Hay que aclarar que el determinar un buen momento de entrada no asegura un buen momento de salida, ya que el precio podría tener comportamientos bruscos o inesperados, aun así, determinar un buen momento de entrada es algo importante, ya que determina una oportunidad de generar ganancias. Como se mencionó anteriormente, durante las simulaciones, las operaciones de *trading* serán realizadas con un volumen fijo de 100 oz, cabe mencionar que con una mayor cantidad de material negociada en cada operación, se podría generar una mayor ganancia, pero a su vez esto genera un mayor riesgo, ya que con cada movimiento del precio se movería una mayor cantidad de dinero.

Para implementar esta estrategia surgen un par de inconvenientes, el primero consiste en definir o cuantificar dicho *momentum*, ya que en la compañía no poseen alguna fórmula, sino que es un criterio que los *traders* se han formado con la experiencia de observar el comportamiento del precio durante años. Dicha cuantificación del *momentum* debe ser robusta ante la volatilidad del precio y estar acorde a las tendencias del mismo, por otra parte, es necesario un conjunto de mecanismos, de monitoreo al comportamiento del precio, que decidan cuándo deshacerse de una posición. La compañía maneja un par de mecanismos denominados *trailingstop* y stoploss. Sin embargo, estos dos mecanismos podrían no ser suficientes ante algunos comportamientos del precio.

Trailingstop: Es el cambio necesario en el precio respecto al último pico o valle observado para considerar que el precio ha cambiado su tendencia, cuando este cambio es observado se cierra la posición abierta que se tenga. Por ejemplo, se decide obtener una posición comprando material ya que el precio cuenta con *momentum* y una tendencia de subida, en cuanto el precio baje, se mide la diferencia respecto al último máximo, cuando esta diferencia sea mayor al *trailingstop*, la posición es cerrada vendiendo el material comprado. En una tendencia de bajada se abre con una operación de venta y el *trailingstop* se mide en base al último mínimo observado.

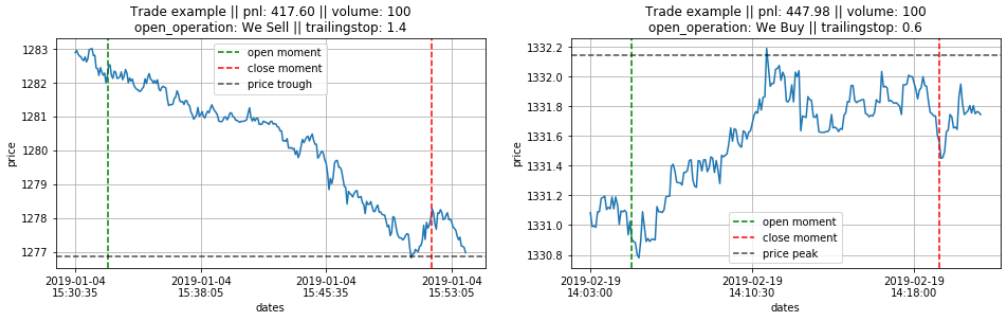


Fig. 11. Ejemplos del mecanismo de *trailingstop* en operaciones de *trading*. Luego de haber abierto un *trade* y una vez se haya sobrepasado la diferencia de precio entre el último pico o valle y el precio actual se cierra el *trade*.

Lo ideal es que el precio haya subido o bajado lo suficiente para no ser afectado por la diferencia entre el precio de cierre y el último pico o valle observado. Establecer de manera correcta el *trailingstop* es importante, ya que un *trailingstop* muy grande puede conllevar que el *trade* dure demasiado tiempo sin cerrarse, mientras que un *trailingstop* muy pequeño puede conllevar a que el *trade* sea cerrado por pequeñas volatilidades del precio.

Stoploss: Límite máximo de pérdida permitido para una posición abierta. Cuando se obtiene una posición y se mantiene abierta, en cada momento tiene una valoración en el mercado, si esta valoración es a pérdida y es mayor al límite definido, la posición es cerrada. Este caso puede observarse al haber un cambio muy brusco en el mercado el cual no permite el cierre de la posición por el *trailingstop*, o se define un *trailingstop* no acorde a la volatilidad del mercado.

2.3. Redes generativas antagónicas

Las redes generativas antagónicas o GANs, por sus siglas en inglés, son un tipo de modelo generativo profundo, fueron presentadas en 2014 por Ian Goodfellow y han obtenido gran popularidad en los últimos años, esto debido a los buenos resultados que se han obtenido con su uso en la producción de datos sintéticos. Este modelo generativo fue implementado para la generación de datos sintéticos del mercado de materias primas planteado en el presente proyecto. Hay que decir que de manera general las GANs presentan un nivel de complejidad alto en su proceso de entrenamiento, es necesario hacer varios ajustes en la arquitectura del modelo para conseguir que este modelo converja. Es debido a dicha complejidad que surgen distintos estudios donde muestran algunos trucos, técnicas o consideraciones a la hora de entrenar e implementar estos modelos.

Con el objetivo principal de obtener experiencia en la implementación, entrenamiento y uso de modelos generativos profundos, se llevó a cabo una primera actividad que consistió en generar imágenes sintéticas basadas en un dataset de radiografias del torax, determinando el impacto que tiene utilizar algunas técnicas para mejorar el desempeño de los modelos generativos. Cabe mencionar que la actividad tuvo éxito en su objetivo y una vez obtenida la experiencia con estos modelos, se procedió a trabajar con modelos que se alimentarán de señales y usaran métricas de similitud, para comparar los datos reales y sintéticos, como se muestra en la sección 3.4 del presente texto. En esta primera actividad se empleó un dataset de imágenes para identificar de manera visual, rápida y sencilla, el impacto de algunas técnicas, trucos o consideraciones como lo son:

- Escalar los datos de entrada a valores entre [-1, 1] y utilizar *tanh* como función de salida del generador.
- Implementar una arquitectura DCGAN de ser posible.
- Usar Dropout en el generador.
- Si se cuenta con las etiquetas utilizarlas en el entrenamiento del modelo.
- Usar LeakyReLU como activación en las capas del generador y el discriminador.
- Entrenar el modelo con mini-batches que contengan solamente datos reales o solamente datos sintéticos.
- Usar un valor de 0.9 como etiqueta para los datos reales en el discriminador en vez de 1.
- Utilizar como función de costo la distancia wasserstein.

El dataset que se utilizó consiste en una serie de imágenes de rayos X de la caja torácica como la que se muestra a continuación.

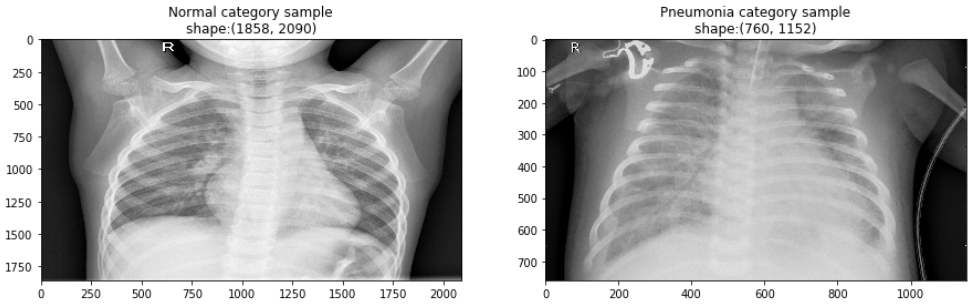


Fig. 12. Muestra del dataset usado para la primera actividad de generación de datos propuesta.

Para esta tarea se implementaron distintas arquitecturas de GANs con las cuales se pudiese observar el impacto de usar los trucos mencionados anteriormente. Dicho impacto se determinó basándose en la similitud visual entre los datos reales y sintéticos, y el desempeño del proceso de entrenamiento de cada modelo. Cabe mencionar que se propuso esta actividad ya que es más sencillo, a la hora de ganar experiencia con estos modelos, determinar la similitud entre datos reales y sintéticos usando imágenes y no series de tiempo.

Se implementa una primera arquitectura que sirva como línea de base para determinar el impacto de los trucos sugeridos. Cabe mencionar que en todas las arquitecturas diseñadas se hace uso de datos cuyos valores están entre 1 y -1, también se usan distintos mini-batches que contienen solamente imágenes reales o solamente imágenes sintéticas. Los resultados para dicho modelo base son los siguientes:

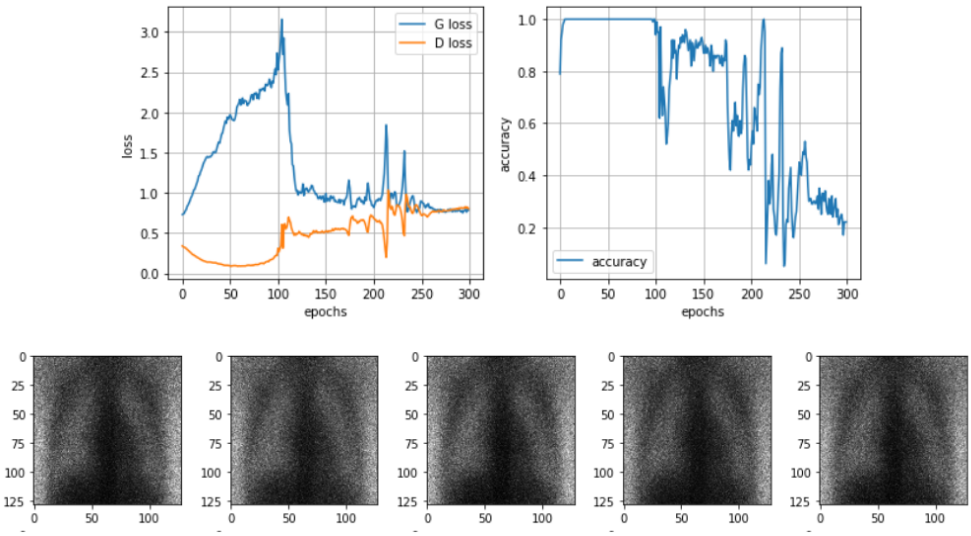


Fig. 13. Resultados modelo base tarea generativa de imágenes del tórax. Arriba: métricas de desempeño en el entrenamiento del modelo base. Abajo: Muestras de datos generados por el modelo base.

Con la línea de base definida, se procede con la experimentación. Hay que mencionar que nos gustaría obtener una GAN que:

- Genere imágenes sintéticas iguales que las reales
- Presenten un proceso de entrenamiento donde el discriminador y el generador convergen
- El discriminador al final del proceso de entrenamiento muestra un nivel de acierto del 50%.

El ideal sería un comportamiento parecido al siguiente:

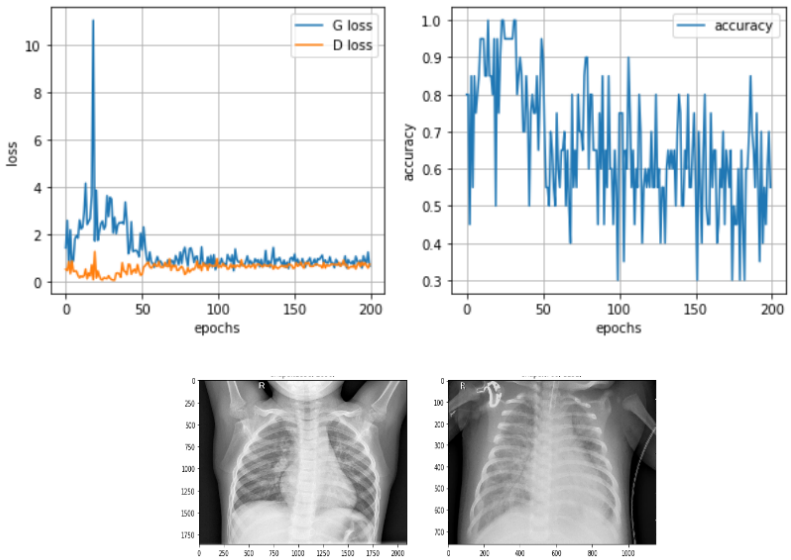


Fig. 14. Métricas de desempeño en entrenamiento y muestras de datos generados por una GAN ideal.

Truco LeakyReLU: Usando leakyReLU como función de activación tanto para el generador como para el discriminador, se puede observar como el discriminador mejora su desempeño, ya no se observa una pérdida creciente y se ve una mejora en el acierto, sin embargo se ve como el

generador pierde un poco de rendimiento. A pesar de todo esto visualmente las imágenes generadas se ven igual al modelo base.

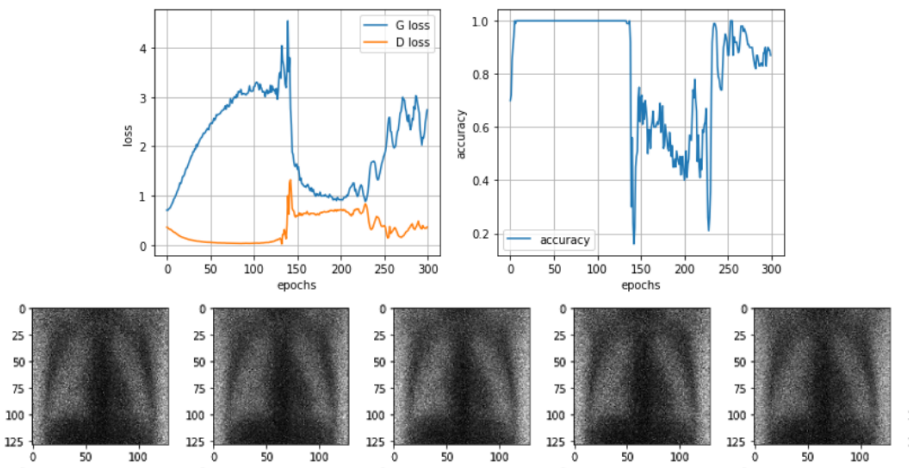


Fig. 15. Métricas de desempeño en entrenamiento y muestras de datos generados por una GAN usando el truco de LeakyReLU.

Truco scaling-target: Usar un valor decimal como 0.9 para la etiqueta de los datos reales en el discriminador no ayuda al mismo, el generador vuelve a presentar una convergencia, sin embargo las imágenes generadas siguen siendo iguales al modelo base.

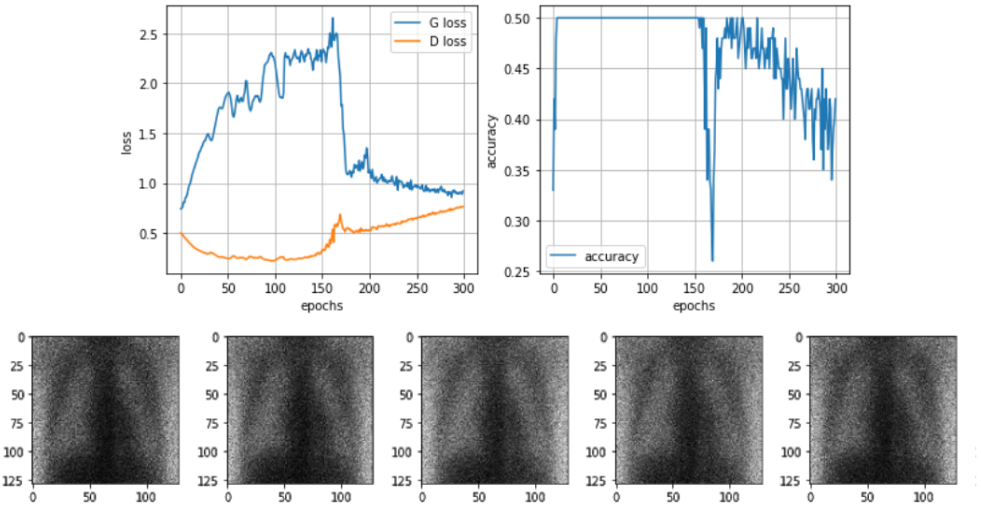


Fig. 16. Métricas de desempeño en entrenamiento y muestras de datos generados por una GAN usando el truco de scaling-target.

Truco DCGAN: En el caso de utilizar capas convolucionales en vez de densas, se puede observar como la calidad de las imágenes generadas mejora, aunque para la arquitectura diseñada el discriminador no parece tener buen desempeño ya que al final del proceso de entrenamiento presenta un muy bajo nivel de acierto.

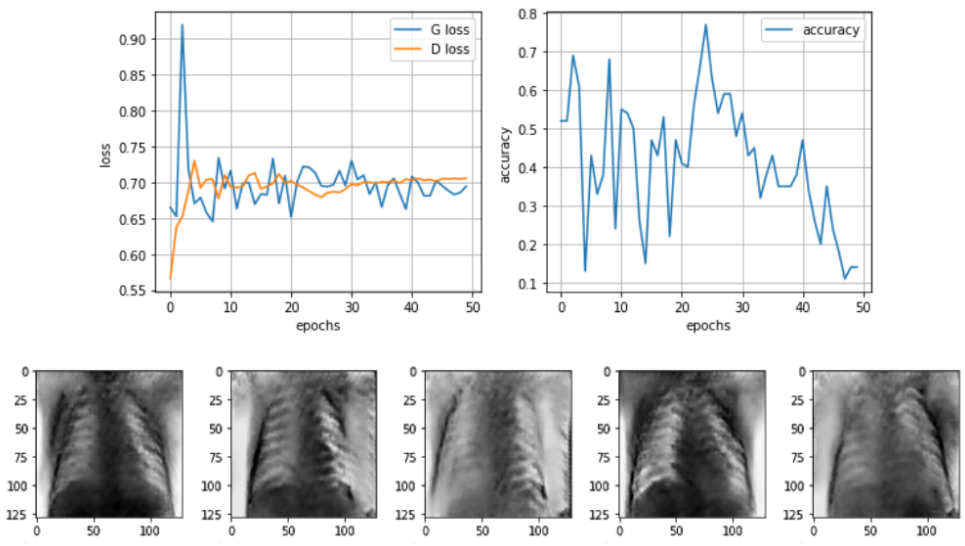


Fig. 17. Métricas de desempeño en entrenamiento y muestras de datos generados por una GAN usando el truco de DCGAN.

Truco Dropout: Finalmente, el último truco que se probó fue usar dropout en el generador, este truco ayuda incluso en el desempeño del discriminador aunque la calidad de las imágenes parece ser igual al de la arquitectura anterior.

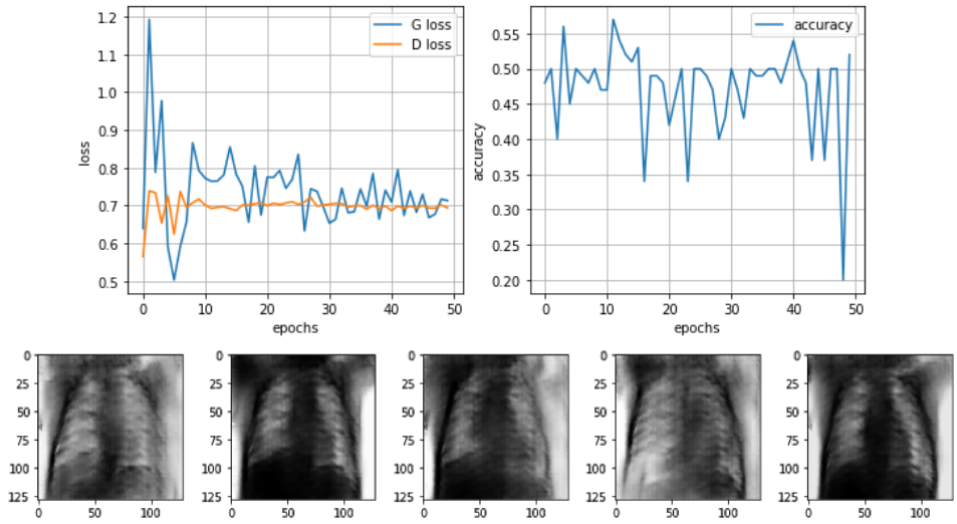


Fig. 18. Métricas de desempeño en entrenamiento y muestras de datos generados por una GAN usando el truco de Dropout.

Con varias arquitecturas desarrolladas y varios de los trucos explorados se puede proseguir con la implementación de modelos generativos para la producción de datos sintéticos del mercado de materias primas.

3. Metodología propuesta

Para llevar a cabo todos los experimentos que se mencionan en esta y las siguientes secciones, fueron desarrollados un conjunto amplio de funciones en el lenguaje de programación Python 3 haciendo uso de librerías como Pandas, NumPy, Scikit-learn, TensorFlow entre otras.

3.1. Mecanismo de entrada

Con el objetivo de implementar el enfoque usado por la compañía en sus estrategias de *trading*, se propuso el desarrollo de un conjunto de funciones, que durante el recorrido de los datos del mercado, detectara aquellos momentos adecuados para abrir y cerrar un *trade*, mientras se llevaban cuentas de las métricas de desempeño para poder comparar el rendimiento de las estrategias simuladas. Cabe aclarar que las simulaciones serán llevadas a cabo abriendo un *trade* a la vez, es decir, un trade no es abierto hasta que no se haya cerrado el anterior.

Para la detección de momentos de entrada, se diseñó e implementó una tarea predictiva que determina el *momentum* del precio en cada instante a través de estimaciones de la velocidad del mismo, en otras palabras, la noción de *momentum* que tiene la compañía se representó como la velocidad del precio. Por ejemplo, si en un instante se estima una velocidad positiva, se interpreta que el precio se encuentra subiendo, por ende, es recomendable abrir un *trade* con una operación de compra, en cambio, si se estima una velocidad negativa, se interpreta que el precio se encuentra bajando, por ende, es recomendable abrir un *trade* con una operación de venta. Esta tarea predictiva fue implementada usando algoritmos de aprendizaje supervisado y un esquema de validación cruzada para series de tiempo, donde un modelo predictivo va generando estimaciones de la velocidad, mientras que es re-entrenado con cierta frecuencia para reconocer los nuevos comportamientos del mercado y evitar el ruido generado por comportamientos muy antiguos.

Al ser una tarea predictiva de aprendizaje supervisado es necesario contar con un objetivo a predecir, el cual recree la velocidad del precio en cada momento. Esta velocidad se produjo utilizando el filtro de Savitzky–Golay [18] con un tamaño de ventana de 841 valores para toda la señal del precio de los 14 meses con los que se cuenta. El filtro de Savitzky–Golay permite el suavizado y cálculo de las derivadas de una señal realizando un cómputo con datos anteriores y posteriores a cada valor de dicha señal. Al calcular la primera derivada de la señal del precio usando dicho filtro se obtiene una señal de velocidad, esto debido a que la señal del precio se puede interpretar como una posición a través del tiempo. Esta señal será denominada *velocidad offline* ya que se calcula directamente sobre toda la señal del precio en donde cada valor fue estimado usando datos anteriores y posteriores al mismo a través del filtro de Savitzky–Golay, lo cual puede ser interpretado como usar datos del futuro y pasado para estimar un valor actual. Esta *velocidad offline* será únicamente usada como target de los modelos predictivos.

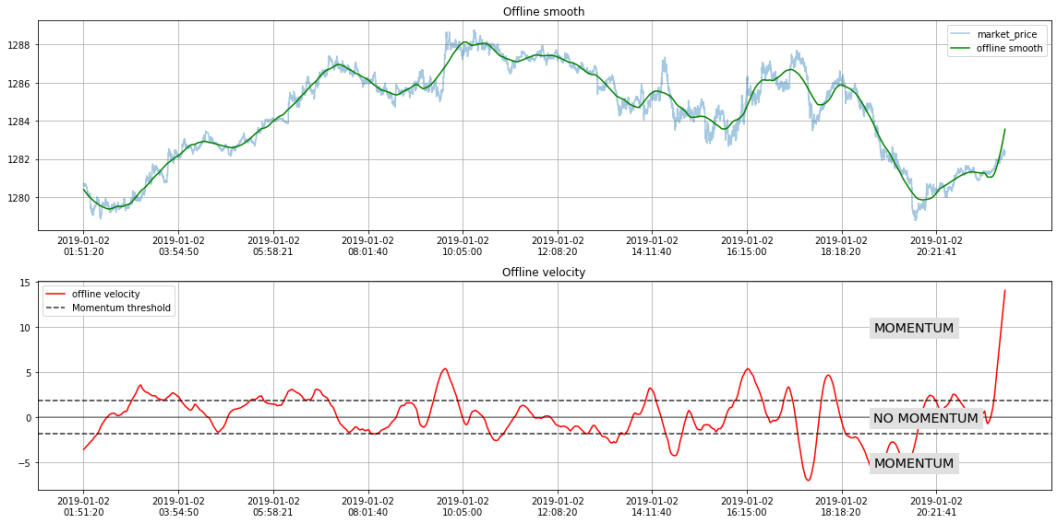


Fig. 19. Ejemplo de la señal de velocidad y señal suavizada obtenidas usando el filtro de Savitzky–Golay en la señal del precio. Valores altos o bajos en la señal de la velocidad pueden indicar la presencia de *momentum* en la señal del precio.

Ya que la compañía realiza sus operaciones basándose en la existencia o ausencia de *momentum* en el precio, la señal de *velocidad offline* es transformada en una señal digital utilizando intervalos, los cuales marcan la presencia de *momentum* en una tendencia de subida como en una tendencia de bajada. Los intervalos seleccionados fueron (-inf, -1.85), [-1.85, 1.85], (1.85, inf). De esta manera, los valores de la *velocidad offline f*ueron divididos en 3 clases, la primera clase (-1) indica que el precio se encuentra bajando, es decir, el valor de la velocidad es menor a -1.85. La segunda clase (0) indica que el precio se encuentra estable, es decir el valor de la velocidad está entre -1.85 y 1.85. La última y tercera clase (1) indica que el precio se encuentra subiendo, osea, el valor de la velocidad es mayor a 1.85. Para definir los límites de cada intervalo a los cuales se les asignaría una de las clases mencionadas, se calculó aquel valor negativo y positivo para el cual la clase 0 contará con un porcentaje de población de velocidad del 62%.

Class distribution		
-1	0	1
18%	62%	20%

Tab. 1. Distribución de clases para la estimación de la velocidad aplicando los intervalos (-inf, -1.85), [-1.85, 1.85], (1.85, inf) sobre la señal de velocidad *offline* del precio.

Cabe mencionar que esta tarea de predicción y los parámetros mencionados anteriormente fueron seleccionados luego de una serie de simulaciones de estrategias de trading que hicieron uso de distintos *targets* diseñados, con el objetivo de determinar una buena manera de generar ganancias. El *target* que generó mejores resultados fue el utilizado para los modelos predictivos.

3.2. Metodología

En el esquema de validación cruzada para series de tiempo, las predicciones se van realizando periódicamente simulando una ejecución en tiempo real. Durante esta simulación el modelo debe ser entrenado cada cierto tiempo para hacer uso de la información más “reciente” y evitar el ruido

de la información más “antigua”. Teniendo esto en cuenta, cada simulación requiere de un conjunto de parámetros a definir:

- **model:** Modelo predictivo a utilizar durante toda la simulación.
- **sample_freq:** Frecuencia de muestreo de las señales utilizadas. Cada señal puede contar con datos con una frecuencia irregular, en el caso del precio del mercado se cuenta un dato cada 3ms en promedio.
- **train_period:** Periodo con el cual se entrenará el modelo predictivo o el tamaño del histórico de las señales necesarias para entrenar un modelo.
- **test_period:** El periodo en que un modelo estará realizando predicciones o la frecuencia con que se entrena el modelo seleccionado.

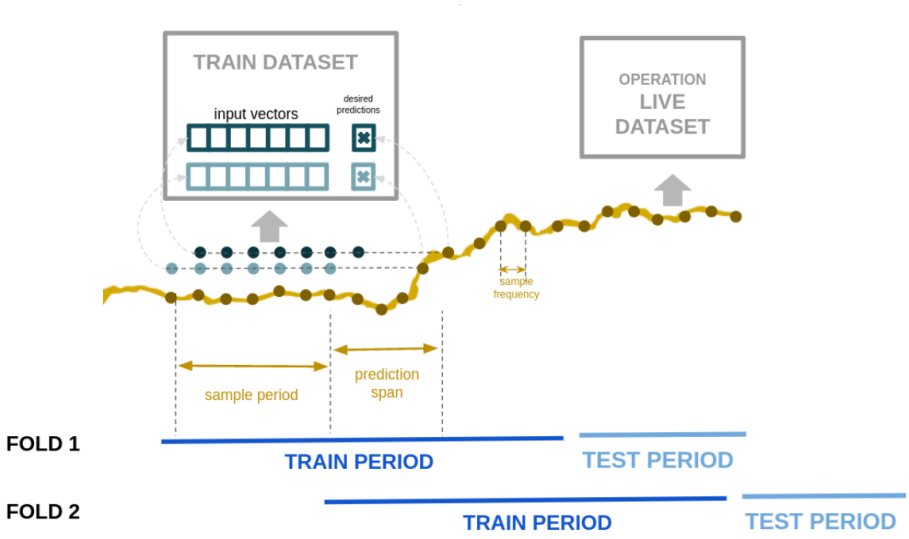


Fig. 20. Esquema de validación cruzada para series de tiempo en el uso de modelos predictivos y generativos.

Para el entrenamiento de cada modelo, en la estimación de la velocidad, se decide usar un conjunto de características de varias señales, el precio del mercado, dos señales suavizadas del precio y dos señales de velocidad del precio. Las señales suavizadas y de velocidad son generadas usando el filtro de Savitzky–Golay simulando el tiempo real a través de ventanas deslizantes, evitando así el uso de datos del futuro. Cabe aclarar que las señales calculadas en tiempo real difieren del comportamiento real que tuvo el precio. Para que estos datos puedan ser usados para entrenar el modelo predictivo fue necesaria la implementación de un conjunto de funciones que transformarán dichos datos en un dataset. Entre el *sample_freq*, *train_period* y el conjunto de características a usar se determina el tamaño del dataset que se usará para entrenar cada modelo predictivo.

Las velocidades calculadas en tiempo real que se mencionaron anteriormente y de las cuales se derivan las características con las que se entrenan los modelos, son señales calculadas a través de ventanas deslizantes sobre la señal del precio con la que se cuenta, es decir, el precio para el periodo de 2019-01-01 a 2020-02-22, este método de ventana deslizante se usa con el objetivo de simular el ‘tiempo real’. En cada ventana se usa un filtro de Savitzky–Golay tomando el último valor de la señal generada, en donde la primera derivada entregada por el filtro se interpreta como la velocidad y la derivada 0 como la señal suavizada. Cada una de las velocidades usadas en la experimentación fueron calculadas usando los siguientes valores en los parámetros del filtro de Savitzky–Golay.

Savitzky-Golay filter	
window length	polyorder
721	3
1441	--

Tab. 2. Configuraciones del filtro Savitzky–Golay usadas durante la experimentación para el cálculo de velocidades en tiempo real usadas para construir los conjuntos de datos con los que fueron entrenados los modelos predictivos.

En el conjunto de características con el que se entrena cada modelo se usa el rango y la desviación estándar para periodos de 30 segundos, 1 minuto y 1 hora de las 5 señales. También se toman las diferencias de cada señal suavizada con el primer y segundo momento anterior para periodos de 1 minuto, es decir la señal diferenciada. Por último y como características extra, cada valor de un periodo de 1 minuto de una de las velocidades es transformado en un valor de -1, 0 o 1, basados en los intervalos de la *velocidad offline*. Para verificar el comportamiento de las funciones implementadas se llevó a cabo una primera experimentación con los siguientes parámetros.

model	sample freq	train period	test period
RandomForestClassifier	5s	10h	10min

Tab. 3. Conjunto de valores usados en la primera experimentación que sirvió para verificar la implementación realizada para realizar las simulaciones de trading y como línea de base para la experimentación con datos sintéticos.

El modelo RandomForest usado contaba con una máxima profundidad de 4, 30 estimadores y la opción *balanced* en el peso de las clases. Estos parámetros fueron seleccionados luego de una experimentación que tuvo el objetivo de determinar el modelo, *sample_freq* y *test_period* a utilizar, y como se mencionó, para verificar la implementación realizada para las predicciones y simulaciones de estrategias de trading. En el presente trabajo se presenta con el objetivo de explicar la metodología diseñada. Cabe mencionar que en la sección 4 del presente texto, se muestra una experimentación más completa, variando otros parámetros como el *train_period*. Por último y para seleccionar los parámetros referentes a las velocidades en tiempo real y las características del conjunto de datos con los que se entrenó cada modelo predictivo, hay que mencionar que también se realizaron experimentos que buscaban tener un conjunto adecuado para la tarea predictiva diseñada.

Realizando predicciones para todo el periodo de datos con el que se cuenta y usando la parametrización anterior se obtuvieron los siguientes resultados.

accuracy		Accuracy per class		
train	test	-1	0	1
0,8314	0,590	0,468	0,663	0,478

Tab. 4. Desempeño de las predicciones resultantes de la experimentación realizada con los parámetros mostrados en la tabla 3.

Como se puede observar el modelo utilizado presenta un nivel de acierto mejor que el aleatorio, ya que consiste en una tarea de 3 clases, pero también presenta cierto grado de overfitting. Sin embargo, ¿Con este nivel de acierto es posible generar ganancias en las estrategias de *trading*?, ¿Otras configuraciones en los parámetros de predicción podrían tener un impacto distinto en las

predicciones?, ¿El uso de datos sintéticos en los modelos predictivos mejoraría su desempeño? son preguntas que se contestarán más adelante del presente estudio. Cabe mencionar que las predicciones realizadas por los modelos predictivos y que luego serán usadas en estrategias de *trading*, son realizadas usando únicamente los datos reales del precio.

3.3. Mecanismos de salida

Durante la simulación, cada vez que se decide abrir un trade, un nuevo valor de trailingstop es asignado en base a la volatilidad del mercado. Un valor alto de *trailigstop* cuando existe una alta volatilidad en el precio, permitirá que el *trade* aproveche mejor los cambios del precio y no se verá afectado por aquellos cambios pertenecientes a la propia volatilidad. En este caso, y luego de una serie de conversaciones con la compañía y de experimentación, se define la volatilidad como el histórico del *spread* del precio, es decir, la diferencia entre el precio de compra y venta de 100oz de materia en cada instante de tiempo. Cada vez que se decide abrir un *trade*, se compara el valor actual del *spread* con su histórico y haciendo uso de la función f(x) = x/10 se asigna el *trailingstop*. donde x es el percentil al cual pertenece el *spread* actual.

También se diseñaron e implementaron mecanismos de monitoreo al precio adicionales a los dos usados por la compañía, ya que dichos mecanismos resultan insuficientes. Cerrar un *trade* en base al *trailingstop* no es una metodología perfecta, una vez asignado a un *trade*, el precio puede tener comportamientos que lo burlen, por ejemplo, se tiene una tendencia de subida en el instante de abrir un *trade*, el precio, en vez de seguir subiendo o empezar a bajar se podría mantener en un tipo de meseta con una baja volatilidad, este caso causaría que un *trade* durará sin cerrarse varias horas o días ya que el *trailingstop* no sería superado en ningún momento posterior. Cabe mencionar que estos mecanismos son usados por las estrategias mientras exista un *trade* abierto. Dichos mecanismos implementados son:

No momentum mitigation: En cualquier momento posterior al que se tomó la decisión de abrir un trade, se puede mirar atrás y verificar si en el momento de dicha decisión el precio tenía alguna tendencia, o presentaba una volatilidad muy baja, si dicha volatilidad es menor a un umbral definido, el *trade* será cerrado, ya que al no existir alguna tendencia incluso tiempo después de tomada la decisión, el *trailingstop* no será útil. Para fines de la simulación esta mitigación es puesta en uso 20 minutos después de abrir el *trade* y serán cerrados aquellos cuyos la desviación estándar de los 10 minutos anteriores al momento de entrada es inferior a 0.25.

Trend capture late mitigation: En cualquier momento posterior al que se tomó la decisión de abrir un *trade*, se puede verificar si la decisión fue tomada cerca al final de la tendencia predicha. De ser así, lo mejor es cerrar el *trade*, ya que se ha entrado a una tendencia diferente a la que se esperaba hubiera mientras el *trade* estuviera abierto. Esta mitigación es puesta en uso luego de 30 minutos de haber abierto un *trade* y serán cerrados aquellos donde la tendencia de 15 minutos antes de entrar al *trade*, es diferente a la tendencia de los 15 minutos posteriores al momento de entrada. La tendencia de dichos periodos es calculada con la diferencia entre el primer y último precio de cada periodo.

Trend subsumed mitigation: Durante la simulación y de manera global se puede presentar una tendencia de subida, y existir pequeños picos de bajada. Si se toma la decisión de abrir un *trade* basado en estas tendencias de muy corto plazo, se generarán pérdidas. La idea es cerrar un *trade* si se detecta dicho comportamiento. Esta mitigación es puesta en uso luego de 10 minutos de

haber abierto un *trade* y serán cerrados aquellos donde la tendencia de los 15 minutos anteriores a abrir el *trade* difiere de la tendencia predicha.

Wrong prediction mitigation: Una vez haya transcurrido el tiempo suficiente para calcular la velocidad real que tuvo el precio en el momento de abrir un *trade*, se verifica si dicha decisión fue correcta o no, de no ser correcta, se cierra el *trade*. Acá es necesario mencionar que dicha velocidad real que tuvo el precio hace referencia a la *velocidad offline* de ese momento, como se mencionó cada valor de esta velocidad es calculado con valores posteriores y anteriores al mismo, por ende se hace necesario esperar por los valores posteriores durante la simulación para poder hacer el cálculo del valor de *velocidad offline.* Cabe mencionar que dicha velocidad fue calculada de la señal del precio muestreada cada 5s, lo que significa que al usar un valor de 841 en el filtro cada valor de la *velocidad offline* fue estimado con 35 minutos de datos anteriores y posteriores al mismo. Es decir, durante la simulación cada vez que se abre un *trade*, hay que esperar 35 minutos para poder determinar el valor de *velocidad offline* del momento en que se abrió el *trade*.

Hold trading mitigation: Luego de haber cerrado un *trade*, se vuelve a verificar la predicción para tomar la decisión de abrir uno nuevo, al usar las distintas mitigaciones anteriormente mencionadas surge el inconveniente de que serán abiertos más *trades* en comparación de no usar las mitigaciones, ya que los *trades* durarán menos tiempo. Para poder controlar la cantidad de *trades* que se abren durante la simulación, se decide no abrir nuevos *trades* hasta que el *trailingstop* del tade anterior sea alcanzado, así el *trade* anterior haya sido cerrado por una mitigación.

End day: Ya que la compañía cierra todas sus posiciones cerca de las 10 de la noche, durante las simulaciones de *trading*, no son abiertos nuevos *trades* después de esta hora, y el *trade* que esté abierto cuando llegue dicho momento, será cerrado.

Liquidity start: La idea de este mecanismo es evitar tener *trades* abiertos cuando no hay liquidez en el mercado, es decir, cuando hay pocos proveedores participando en el mismo, esto suele suceder a tempranas horas del día. Este mecanismo no permite abrir *trades* si no hasta pasadas las 7 u 8 de la mañana.

Los parámetros de cada mecanismo presentado, fueron seleccionados en base a comportamientos de la compañía o para darle espacio al *trailingstop* de ser validado, esto último en el caso de las mitigaciones.

3.4. Modelos generativos en el mercado

Ya que los modelos predictivos muestran cierto nivel de overfitting y con el objetivo de determinar el impacto de usar datos sintéticos en ámbitos financieros, se propuso el uso de modelos generativos profundos para producir nuevos datos con los cuales entrenar modelos predictivos, esperando un mejor desempeño en los mismos y en las estrategias de *trading*.

Sin embargo, no basta con simplemente implementar un modelo generativo y hacer uso de sus datos, hay que asegurarse de que el modelo presenta un buen proceso de entrenamiento, ya que el uso de datos producidos por modelos mal entrenados, solo agregaría ruido a las predicciones y a las estrategias de *trading*, de la misma manera se espera que los datos generados presenten similitud con los datos reales.

Existen bastantes modelos generativos profundos, como por ejemplo las GAN, RBM, VAE o GTN, Estos modelos a través de arquitecturas de redes neuronales tienen como objetivo generar datos sintéticos que, por lo general, son utilizados en el entrenamiento de modelos predictivos en tareas en las cuales no se cuente con una cantidad suficiente de datos, en el caso del presente proyecto el uso de una metodología de validación para series de tiempo, limita el uso de los datos con los que se cuenta.

Se realizó una serie de experimentos donde se verificó la estabilidad del proceso de entrenamiento y el nivel de similitud producido entre los datos sintéticos y los reales para distintos modelos generativos implementados. En las simulaciones de *trading* se usaron los datos producidos por el modelo que mejor presentó resultados en la experimentación que se menciona. La experimentación consistió en producir datos sintéticos del mercado de materias primas utilizando los primeros 15 días del mes de agosto del 2019, cada modelo fue entrenado para generar una señal del mercado similar a la de la media de comprar y vender 100oz. El dataset con el que se entrenó cada modelo generativo fue construido usando dicha señal real, con los siguientes parámetros:

Dataset parametrization	
sample freq	5s
sample period	1h
overlap spam	1min
rescale rows	minmax

Tab. 5. Parámetros con los que se construyó el dataset para entrenar los modelos generativos.

El parámetro *overlap_spam* hace referencia al solapamiento de datos entre cada fila del dataset, el parámetro *rescale_rows* hace referencia al escalado que tienen los datos en el dataset, en este caso, cada fila es escalada entre valores de 0 y 1 usando el maximo y minimo de cada fila, mientras que el parametrio *sample_period* hace referencia al tamaño de muestra de la señal usados en cada fila del dataset.

Cada modelo generativo produjo la cantidad de datos suficiente para ser comparados con los 15 días mencionados de datos reales, en otras palabras, cada modelo generó 15 días de datos sintéticos. Hay que mencionar que la salida del modelo generativo tiene el mismo formato que la entrada con la que se le entrena, es decir una matriz de datos con los parámetros de la tabla anterior. Por ello, una vez generados los datos, se hace un proceso para obtener una única señal y compararla con la señal real del precio del mercado. Este proceso consiste en concatenar cada una de las filas de la matriz usando las diferencias entre cada uno de los valores con el valor siguiente, teniendo en cuenta que hay solapamiento entre las filas.

Una vez se obtiene la señal producida por cada modelo, se procede a determinar la similitud con los datos reales, dicha similitud es medida usando la distancia Wasserstein (Earth mover's distance) [19]. La distancia Wasserstein, también conocida como distancia de movimiento de tierra, es una función de distancia entre distribuciones de probabilidad que considera el costo mínimo de convertir una distribución en otra como si se tratara de dos pilas de tierra. Hay que aclarar que lo que se comparó fue la distribución de diferencias de cada momento de la señal con un periodo anterior, en otras palabras, las distribuciones de las señales diferenciadas una vez.

Se implementaron alrededor de 130 modelos generativos para la experimentación descrita anteriormente, dichos modelos estuvieron basados en arquitecturas de GAN y VAE, variando su complejidad usando distintos tipos de capas, como las densas, las convolucionales, Dropout, entre otras, y modificando la cantidad de capas y nodos por capa. Hay que mencionar que algunos de los trucos de entrenamiento de GAN mencionados en secciones anteriores, fueron aplicados en los modelos con arquitectura GAN implementados.

Los modelos que mejor desempeño mostraron en la experimentación mencionada anteriormente son los siguientes

Base architecture	Layers	Wasserstein	**epochs:** 200 || **batch_size:** 10			
			Optimizer	Network	Network architecture	
VAE	Dense	1,801	Adam(1e-3) latent dim: 3	encoder	Dense(512) Batchnormalization() ReLU()	Dense(512) Batchnormalization() ReLU()
				decoder	Dense(512) ReLU()	Dense(512) ReLU()
GAN	Conv1D	2,15566	Adam(1e-3)	discriminator	Conv1D(256, kernel_size=5) ReLU()	Conv1D(256, kernel_size=5) ReLU()
				generator	Conv1DTranspose(256) LeakyReLU()	Conv1DTranspose(256) LeakyReLU()
GAN	Dense	1,35476	Adam(1e-3)	discriminator	Dense(256) LeackyReLU() Dropout(0.5)	Dense(256) LeackyReLU() Dropout(0.5)
				generator	Dense(128) Dropout(0.5)(x) ReLU()	Dense(512) ReLU()
GAN	LSTM	2,58	Adam(1e-4)	discriminator	LSTM(256, return_sequences=True) ReLU()	LSTM(256, return_sequences=False) ReLU()
				generator	Dense(128) LeakyReLU()	Dense(128) LeakyReLU()

Tab. 6. Arquitectura de los modelos generativos que presentaron un mayor desempeño en la experimentación realizada. El desempeño fue medido a través de la distancia Wasserstein. Un buen desempeño en los modelos generativos significa que produjeron los datos más parecidos a los datos reales.

Dado que el proyecto tiene un enfoque de simulación y validación cruzada para series de tiempo, la producción de datos sintéticos con el modelo generativo seleccionado debe ser realizada de esta misma manera. De este modo, el modelo generativo es entrenado cada cierto tiempo con nueva información, para evitar el ruido que pueda generar la información más antigua y de esta manera conseguir que el modelo generativo produzca datos sintéticos acordes a las nuevas tendencias o comportamientos del mercado.

Hay que tener en cuenta que en la estimación de la velocidad se requiere el cálculo de velocidades en tiempo real de la señal del mercado, los datos sintéticos y reales deben estar en el mismo formato una vez vayan a ser usados en los modelos predictivos y que cada modelo predictivo a entrenar debe tener la posibilidad de acceder a una cantidad amplia y suficiente de datos sintéticos. Por ende, se desarrollaron una serie de funciones que permitieran el uso de modelos generativos y datos sintéticos de manera periódica. Cada vez que se entrena un modelo generativo, se produce una cantidad considerable de datos y se realiza el procesado necesario para la estimación de la velocidad. En el momento que se usa un modelo predictivo, este simplemente accede a los datos sintéticos que se hayan generado más recientemente.

Para la producción de datos sintéticos se definió un parámetro denominado *generative_period*, el cual indica el periodo de entrenamiento de cada modelo generativo y a su vez, el periodo válido para usar un conjunto de datos sintéticos. En concreto se produjeron datos sintéticos con un *generative_period* de 15 días, esto quiere decir que el modelo generativo era entrenado cada 15 días de simulación con 15 dias de datos del precio del mercado, lo que significa que durante 15 días de simulación el modelo predictivo usará los datos sintéticos de un mismo modelo generativo. El modelo generativo seleccionado para esta tarea fue el GAN-Dense que se muestra en la tabla 6.

El uso de datos sintéticos en modelos predictivos consiste en que cada vez que un modelo predictivo vaya a ser entrenado, se entrene con los datos reales y una cantidad extra de datos sintéticos, lo que producirá predicciones distintas a las que ya se tenían, por esto, es necesario volver a ejecutar las simulaciones de modelos predictivos que se habían realizado sin usar datos sintéticos, pero esta vez, haciendo uso de ellos.

4. Experimentación

4.1. Predicciones del mercado

Con todo lo mencionado hasta el momento existen varios factores que podrían afectar el comportamiento de las estrategias de *trading* y las predicciones, uno de ellos, y la motivación de este proyecto, es la cantidad de datos con los que se cuenta en cada entrenamiento, ya sea de datos reales o sintéticos. De manera general en un proyecto, no contar con una amplia cantidad de datos reales y poder contar con datos sintéticos podría ser una gran ventaja. Entonces, ¿De qué manera se ven afectadas las estrategias de *trading* y predicciones con la variabilidad de datos sintéticos y reales usados en el entrenamiento de modelos predictivos?

Para contestar esta pregunta se realizó un conjunto de simulaciones de estrategias de *trading* que hicieran uso de modelos predictivos entrenados con distintas cantidades tanto de datos sintéticos como reales. Dicha experimentación fue acompañada con un conjunto de gráficas que pudieran dar una visión global del impacto de los datos sintéticos y reales en las estrategias de *trading*. En el momento en que se entrena un modelo predictivo se accede a una cantidad de datos reales o sintéticos para este proceso, la cantidad de datos reales usado para entrenar cada modelo, fue denominada *train_period*, mientras que la cantidad de datos sintéticos usado para entrenar cada modelo, fue denominada *synthetic_size*. El *train_period* es dado en unidades de tiempo, ya que los datos reales están asociados a un instante de tiempo, mientras que *synthetic_size* representa el número de nuevas filas añadidas al dataset provenientes de la señal producida por el modelo generativo.

Anteriormente se mostraron los resultados para una experimentación con valores de 10h y 0 para los parámetros de *train_period* y *synthetic_size* respectivamente, es decir, cada modelo fue entrenado con 10 horas de datos reales anteriores al momento de entrenamiento y ninguno de datos sintéticos. Sin embargo, en ese mismo momento, el modelo pudo haber sido entrenado con una mayor o menor cantidad de datos reales, o pudo haber hecho uso de una cantidad pequeña o grande de datos sintéticos. De esta manera y con el objetivo de determinar el impacto de la variación de datos reales y sintéticos en la tarea predictiva, se realizaron predicciones usando los siguientes valores de *train_period* y *synthetic_size*.

train period	synthetic size
1h	0
3h	14000
10h	21000

Tab. 7. Parámetros usados para determinar el impacto de datos sintéticos en estimaciones de velocidad y estrategias de *trading.*

Con dicha experimentación se obtuvieron los siguientes resultados. Hay que mencionar que para el modelo y demás parámetros de experimentación requeridos se usaron los ya mencionados en la prueba anterior.

train period	real size	synthetic size	pct synthetic in exp dataset	accuracy (on real data)		accuracy per class in test (on real data)		
				train	test	-1	0	1
1h	720	0	0,0%	0,9999	0,495	0,243	0,648	0,260
1h	720	14000	95,1%	0,9287	0,367	0,208	0,453	0,252
1h	720	21000	96,6%	0,8768	0,348	0,207	0,425	0,244
3h	2160	0	0,0%	0,9733	0,557	0,303	0,714	0,313
3h	2160	14000	86,6%	0,8700	0,527	0,369	0,613	0,409
3h	2160	21000	90,6%	0,8409	0,518	0,373	0,594	0,419
10h	7200	0	0,0%	0,8314	0,590	0,468	0,663	0,478
10h	7200	14000	66,0%	0,7542	0,575	0,519	0,597	0,561
10h	7200	21000	74,4%	0,7414	0,577	0,523	0,597	0,565

Tab. 8. Resultados para la estimación de la velocidad variando los parámetros *train_period* y *synthetic_size* en los datos reales. En la tabla se puede observar como el con el uso de una mayor cantidad de datos reales y sintéticos el desempeño por clase mejora.

Como se puede observar en la tabla, con el incremento de datos reales, los modelos predictivos muestran mejores resultados, ya que el desempeño (*accuracy test*) que muestran los experimentos con un *train_period* de 10h, es mayor que el desempeño mostrado por los experimentos con un *train_period* de 1h, lo cual es algo totalmente esperado. Por otro lado, en el uso de datos sintéticos, se puede observar como el desempeño empeora con el aumento del *synthetic_size* para cada uno de los *train_period* utilizados, lo cual es algo que no favorece al presente estudio. Sin embargo, al observar el desempeño por clase al aumentar el *synthetic_size* para cada uno de los *train_period* utilizados, se observa como aumenta para las clases 1 y -1, esto genera una nueva motivación ya que es en las predicciónes de las clases 1 y -1 donde existe la oportunidad de generar ganancias en las estrategias de trading, puesto que es en estas predicciones donde se detecta una tendencia fuerte en el precio. Si se predice 0, así sea de manera equivocada, la estrategia simplemente no abrirá un nuevo *trade* y espera a la siguiente predicción realizada por el modelo.

De esta manera surge la necesidad por ejecutar estrategias de trading que hagan uso de estas predicciones y comprobar el impacto en las ganancias producidas por cada estrategia, debido al comportamiento observado en el desempeño de las predicciones. Cabe mencionar que también se observa como los niveles de overfitting disminuyen un poco con el incremento de datos reales y sintéticos.

Antes de continuar con la simulación de estrategias de *trading* y entrando un poco más a fondo a la exploración de los experimentos de predicción, se encontró que las probabilidades entregadas por los modelos para cada predicción en cada experimento también muestran comportamientos interesantes. Hay que aclarar que cada modelo entrenado entrega una lista de probabilidades en cada momento en que realiza una predicción, cada valor en esta lista corresponde a una probabilidad por cada clase, la clase con mayor probabilidad es la asignada como predicción para ese instante. Tomando la diferencia que hay entre la mayor probabilidad y la suma de las dos menores se obtiene un valor que se denominó *proba_diff*, este indicador fue desarrollado con el objetivo de determinar la “seguridad” que tiene el modelo cuando entrega cada predicción. Valores bajos de *proba_diff* indican que el modelo no se encuentra muy seguro de la predicción que realiza, ya que los valores de probabilidad para cada una de las clases, son cercanos entre sí, en cambio,

valores altos de *proba_diff* indican que el modelo se encuentra seguro de la predicción realizada, ya que los valores de probabilidad para cada una de las clases se distancian entre sí. Verificando la distribución de *proba_diff* para cada uno de los experimentos se observa lo siguiente.

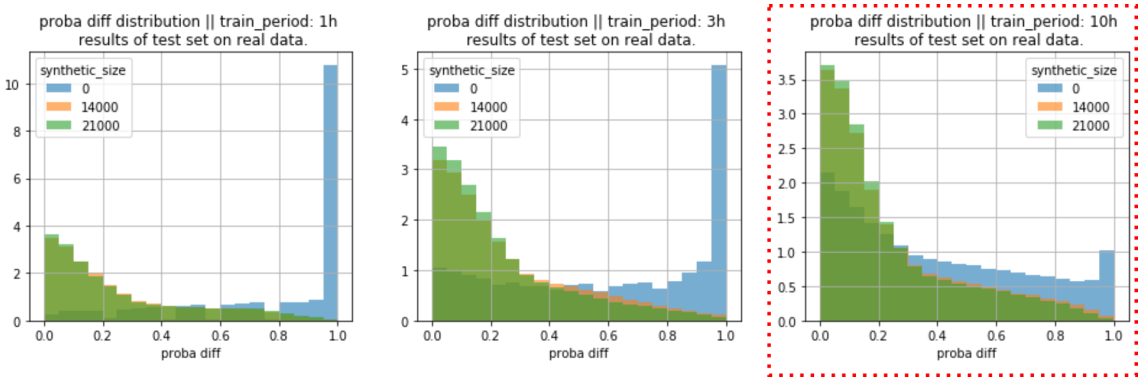


Fig. 21. Distribución de *proba_diff* para los distintos experimentos de predicción en el conjunto de test de datos reales. En la gráfica se puede observar como la distribución de predicciones que no hacen uso de datos sintéticos tienen mayor población alrededor de 1. Esto significa mayor certeza en estos modelos. Una discriminacion por predicciones y target de los experimentos con *train_period* de 10h puede ser observado en la figura 22.

Hay una ventaja en los modelos que no hacen uso de datos sintéticos, ya que estos muestran una mayor 'seguridad' en sus predicciones, dado que las distribuciones de *proba_diff* para estos modelos se encuentran más cercanas a 1, que las distribuciones de modelos que usaron datos sintéticos. Sin embargo, y basándose en el comportamiento observado en el desempeño por clase de las predicciones, ¿cómo es el comportamiento de estas distribuciones al hacer una discriminacion por clase? Verificando el conjunto de experimentos con un *train_period* de 10h se observa lo siguiente.

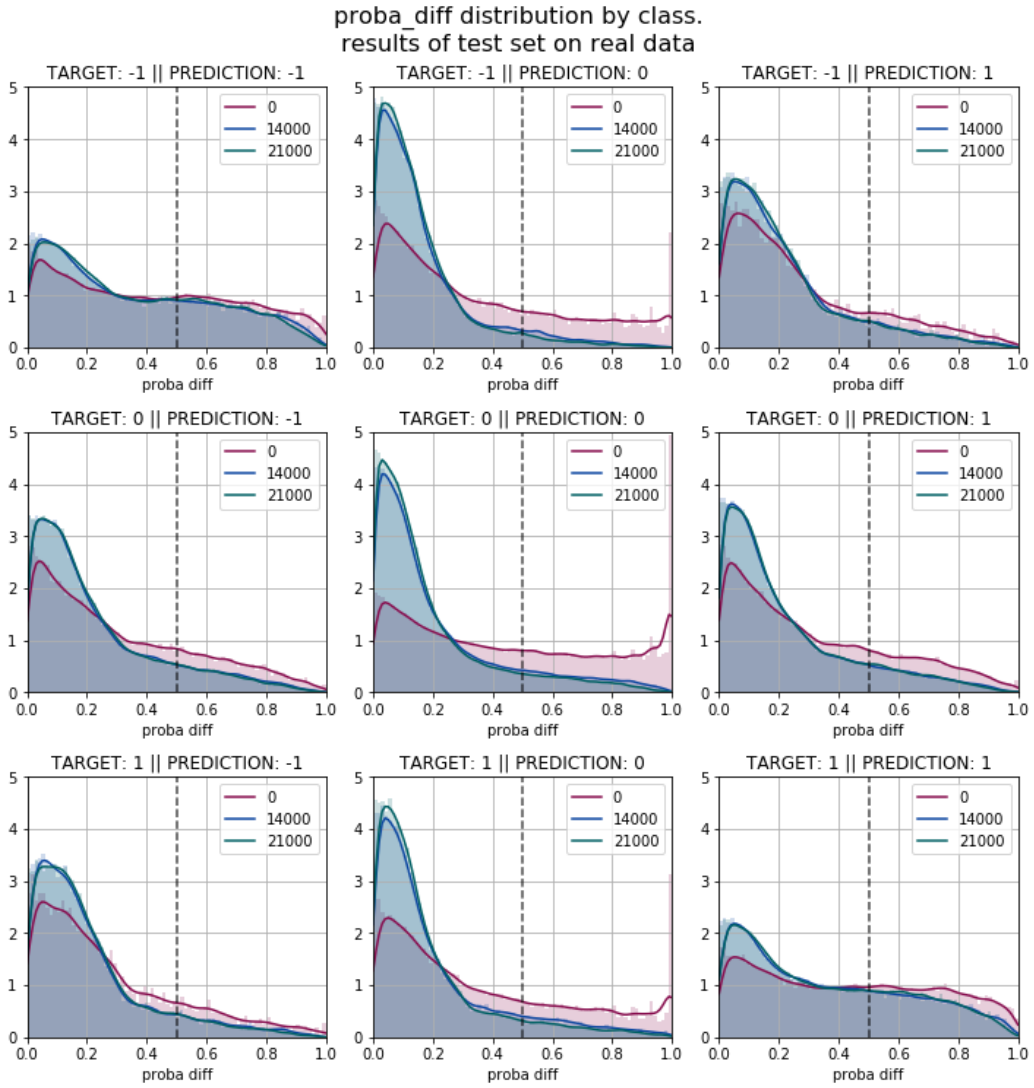


Fig. 22. Distribución de proba_diff discriminado por las combinaciones de predicción y target de cada experimento con un *train_period* de 10h en el conjunto de test de datos reales. Las predicciones con el mismo valor muestran una menor población de estimaciones erróneas mientras el valor de *proba_diff* aumenta.

A pesar de haber una menor “seguridad” en las predicciones de los modelos que usaron datos sintéticos, de manera general, hay cierto comportamiento en el uso de datos sintéticos que podría favorecer a las estrategias de *trading*. Hay que aclarar que realizar predicciones erróneamente de las clases 1 o -1 en las estrategias de *trading* genera un mayor riesgo que predecir erróneamente la clase 0 ya que se podrían generar pérdidas al abrir un *trade* en un momento equivocado. Como se ve en la figura 22 si se hace una discriminacion en las predicciones basados en el *proba_diff* se pueden evitar en mayor medida las predicciones erróneas, ya que la población de predicciones correctas es mayor que la población de predicciones incorrectas cuando se utiliza un umbral de *proba_diff*, esto se puede observar claramente en la figura 22 cuando se comparan las 3 gráficas de cada columna entre sí. Debido a este comportamiento, se decide entregar a las estrategias de *trading* la capacidad de determinar la validez de cada predicción en base a la probabilidad entregada por el modelo. En otras palabras, cada vez que una estrategia hace uso de una predicción para decidir si abrir o no un *trade*, esta, está en la capacidad de descartar dicha predicción si considera que la probabilidad entregada por el modelo es muy pequeña. De ser así, la

predicción es descartada y el *trade* no es abierto. Varias estrategias fueron ejecutadas con niveles de validación diferentes para las predicciones o en otras palabras, umbrales de probabilidad distintos.

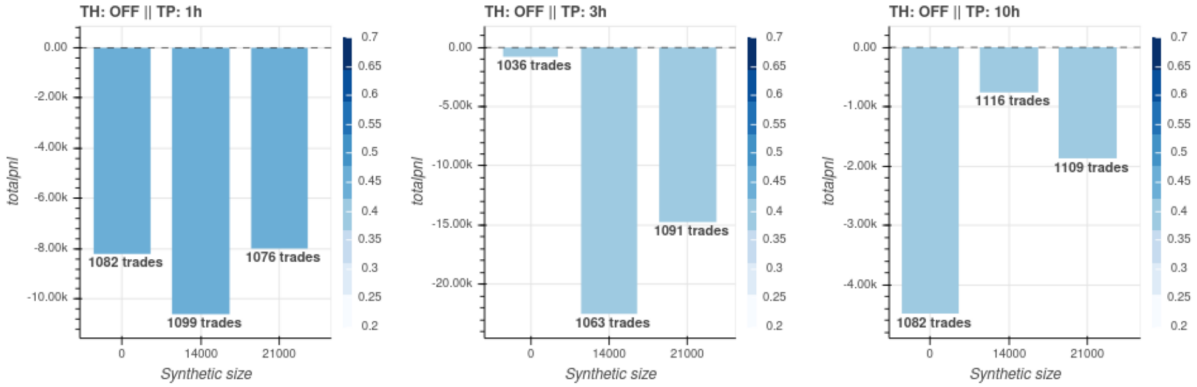
4.2. Simulación de estrategias

Se realizaron simulaciones de *trading* para cada uno de los experimentos de predicción mostrados anteriormente variando el umbral de probabilidad con los siguientes valores.

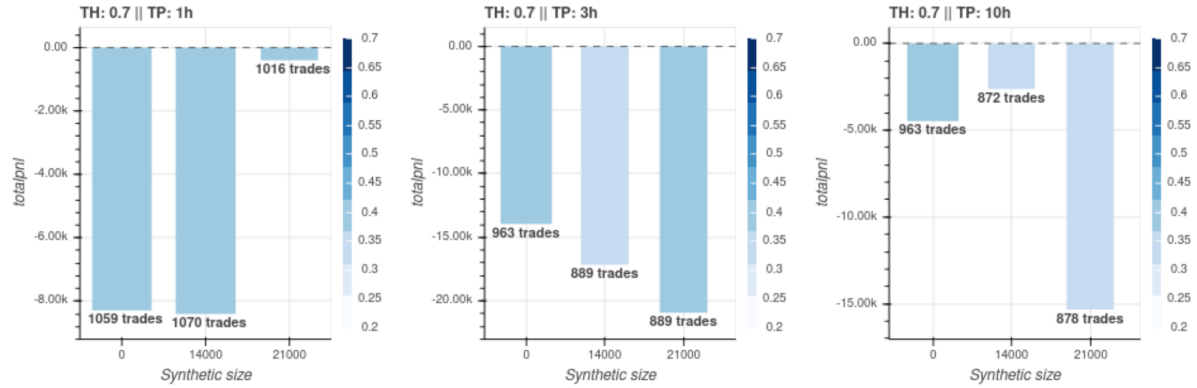
proba diff threshold
0
0,7
0,9

Tab. 9. Umbrales de *proba_diff* usados en las estrategias de *trading.*

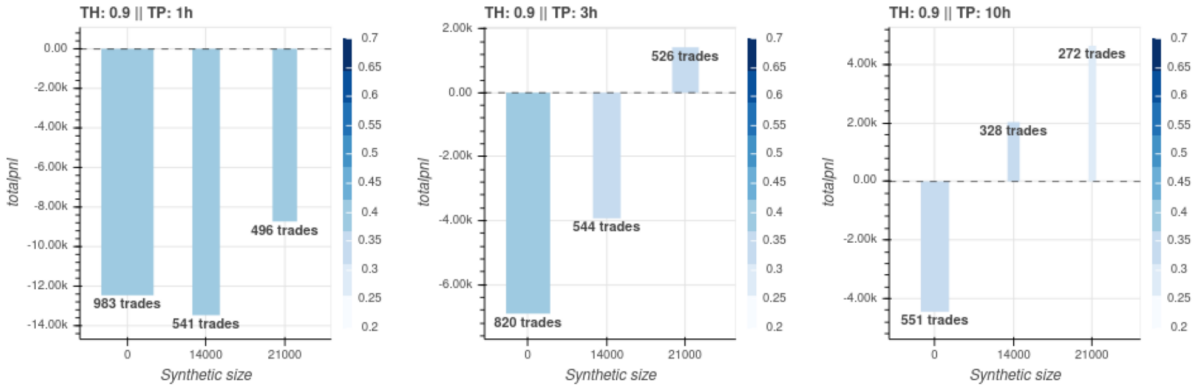
Donde 0 significa que se consideran válidas todas las predicciones que se realicen mientras que 0.9 significa que se consideran válidas aquellas predicciones con un *proba_diff* mayor o igual a 0.9. Ejecutando dichas estrategias para las predicciones mencionadas, para todo el periodo de datos con el se cuenta y haciendo comparaciones con la ganancia total generada (*totalpnl)* por las estrategias, se obtuvieron los siguientes resultados.



(a) Estrategias que usaron un umbral de 0.



(b) Estrategias que usaron un umbral de 0.7.



(c) Estrategias que usaron un umbral de 0.9.

Fig. 23. Resultados de las simulaciones de estrategias de *trading*. Las estrategias hacen uso o no uso de datos sintéticos producidos por modelos generativos profundos. Resultados de simulación para datos entre 2019-01-01 y 2020-02-22. El color de las barras representa el porcentaje de *trades* positivos, el ancho representa la cantidad de *trades*. TP significa *train_period* y TH *threshold*. La gráfica muestra cómo las ganancias generadas por cada estrategia aumentan cuando se hace uso de un mayor número de datos sintéticos mientras un umbral de probabilidad es aplicado a las predicciones.

Los distintos comportamientos de los modelos predictivos causados por el uso de datos sintéticos tienen un impacto en las estrategias de *trading*, como se ve en la figura 23, ya que aquellas estrategias que usaron de mayor manera los datos sintéticos presentaron mejores resultados, la métrica *totalpnl*, hace referencia al dinero total obtenido por cada estrategia durante la simulación.

Esta mejora en el rendimiento de las estrategias de *trading* se observa en especial cuando se hace uso de un umbral de 0.9 en el *proba_diff* de las predicciones, (última fila de la figura 23), allí se puede observar como con el incremento de datos sintéticos en los modelos predictivos, las estrategias pasan de perder a ganar dinero durante la simulación, demostrando de esta manera el impacto de usar datos sintéticos producidos por modelos generativos profundos en tareas predictivas en estrategias de *trading* del mercado de materias primas, sin dejar de lado el hecho de que usar una mayor cantidad de datos reales también tiene un impacto positivo en las estrategias de trading, ya que las estrategias que hicieron uso de modelos predictivos entrenados con un *train_period* de 1h producen menos pérdidas que las estrategias que hicieron uso de modelos predictivos entrenados con un *train_period* de 10h.

5. Conclusiones

La obtención de ganancias en estrategias de *trading* es un reto al cual se enfrenta cada participante de los distintos mercados, muchos de ellos optan por el monitoreo manual del precio y su experiencia en el mercado para realizar las operaciones de *trading*. En el presente trabajo se muestra una metodología de *trading* que puede ser implementada para que funcione en tiempo real y de manera automática, ya que los experimentos mostrados fueron simulados bajo un esquema de validación cruzada para series de tiempo, la cual simula la noción de tiempo real.

Esta metodología demuestra producir ganancias para el periodo comprendido entre 2019-01-01 y 2020-02-22 para el mercado de materias primas del oro haciendo uso de algoritmos de inteligencia artificial y mecanismos de monitoreo del precio. También se logra demostrar que el uso de datos sintéticos en contextos financieros puede tener un impacto positivo, campo en el cual aún hay mucho por estudiar y desarrollar.

Hay diferentes puntos desde los cuales se podría continuar con el desarrollo del proyecto:

- La metodología propuesta, solo se probó en un mercado. Tener una noción del funcionamiento de esta metodología en varios mercados como el de las divisas podría generar un gran impacto.
- Los modelos predictivos y generativos usados en la presente metodología son modelos relativamente sencillos, se podría generar una experimentación más amplia para determinar un modelo más robusto que mejore el desempeño de los modelos presentados.
- Los modelos predictivos fueron entrenados con características de 5 señales derivadas de la señal del precio, podrían crearse conjuntos de datos con más características o que hagan uso de más señales derivadas del precio, incluso, con señales externas.

Referencias

[1] F. De Meer Pardo. *Enriching Financial Datasets with Generative Adversarial Networks.* Agosto 22, 2019. Accedido en: Octubre 16, 2019. [En línea]. Disponible en: https://repository.tudelft.nl/islandora/object/uuid%3A51d69925-fb7b-4e82-9ba6-f8295f96705c

[2] D. A van Dyk, X. Meng. *The Art of Data Augmentation.* Enero 01, 2012. Accedido en: Abril 11, 2021. [En línea]. Disponible en: https://www.tandfonline.com/doi/abs/10.1198/10618600152418584?casa_token=7_X89ALa4YUAAAAA:1w_8TSjeEj0aygpe49Yv_KbTA_Gt_X4NagVrzncLl9w4l8CvNusV7UW35P3er74s6DKjhoT1W12RV2Vv

[3] J. Hui. *GAN, Why it is so hard to train Generative Adversarial Networks!.* Junio 21, 2018. Accedido en: Enero 31, 2020. [En línea]. Disponible en: https://medium.com/@jonathan_hui/gan-why-it-is-so-hard-to-train-generative-advisory-networks-819a86b3750b

[4] Z. Pan, W. Yu, X. Yi, A. Khan, F. Yuan, Y. Zheng. *Recent Progress on Generative Adversarial Networks (GANs): A Survey.* Marzo 14, 2019. Accedido en: Mayo 26, 2020. [En línea]. Disponible en: https://ieeexplore.ieee.org/abstract/document/8667290

[5] X. Zhou, Z. Pan, G. Hu, S. Tang, and C. Zhao. *Stock Market Prediction on High-Frequency Data Using Generative Adversarial Nets.* Abril 15, 2018. Accedido en: Octubre 25, 2019. [En línea]. Disponible en: https://www.hindawi.com/journals/mpe/2018/4907423/

[6] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio. *Generative Adversarial Networks.* Junio 10, 2014. Accedido en: Abril 11, 2021. [En línea]. Disponible en: https://arxiv.org/abs/1406.2661

[7] T. Silva. *An intuitive introduction to Generative Adversarial Networks (GANs).* Enero 7, 2018. Accedido en: Abril 11, 2021. [En línea]. Disponible en: https://www.freecodecamp.org/news/an-intuitive-introduction-to-generative-adversarial-networks-gans-7a2264a81394/

[8] A. Tizzano. *Direct Reinforcement Learning for the DVA Hedging through Recurrent Generative Adversarial Networks for dataset augmentation.* Octubre 3, 2018. Accedido en: Octubre 25, 2019. [En línea]. Disponible en: https://www.politesi.polimi.it/handle/10589/142609

[9] D. P. Kingma, M. Welling. *Auto-Encoding Variational Bayes.* Mayo 1, 2014. Accedido en: Marzo 30, 2020. [En línea]. Disponible en: https://arxiv.org/abs/1312.6114

[10] J. Rocca. *Understanding Variational Autoencoders.* Septiembre 23, 2019. Accedido en: Abril 11, 2021. [En línea]. Disponible en: https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73

[11] C. Gu, H. Hsieh, C. Wu, R. Chang, J. Ho. *A fund Selection Robo-Advisor with Deep learning driven market prediction.* Octubre 9, 2019. Accedido en: Mar 30, 2020. [En línea]. Disponible en: https://ieeexplore.ieee.org/abstract/document/8914183

[12] R. Salakhutdinov, A. Mnih y G. Hinton. *Restricted Boltzmann machines for collaborative filtering.* Junio 2007. Accedido en: Marzo 30, 2020. [En línea]. Disponible en: https://dl.acm.org/doi/abs/10.1145/1273496.1273596

[13] A. Sharma. *Restricted Boltzmann Machines — Simplified*. Octubre 1, 2018. Accedido en: Abril 11, 2021. [En línea]. Disponible en: https://towardsdatascience.com/restricted-boltzmann-machines-simplified-eab1e5878976

[14] Q. Liang , W. Rong, J. Zhang, J. Liu , Z. Xiong. *Restricted Boltzmann Machine Based Stock Market Trend Prediction.* Mayo 19, 2017. Accedido en: Marzo 30, 2020. [En línea]. Disponible en: https://ieeexplore.ieee.org/abstract/document/7966014

[15] P. Such, A. Rawal, J. Lehman, K. Stanley, J. Clune. *Generative Teaching Networks: Accelerating Neural Architecture Search by Learning to Generate Synthetic Training Data.* Diciembre 17, 2019. Accedido en: Mayo 26, 2020. [En línea]. Disponible en: https://arxiv.org/abs/1912.07768

[16] X. Wanga, Y. Wangb, B. Wengc, A. Vinela. *Stock2Vec: A Hybrid Deep Learning Framework for Stock Market Prediction with Representation Learning and Temporal Convolutional Network.* Septiembre 29, 2018. Accedido en: Octubre 27, 2020. [En línea]. Disponible en: https://arxiv.org/abs/2010.01197

[17] Z. Luo, J. Chen, X. Cai, K. Tanaka, T. Takiguchi, T. Kinkyo, S. Hamori. *Oil Price Forecasting Using Supervised GANs with Continuous Wavelet Transform Features*. Agosto 24, 2018. Accedido en: Octubre 27, 2020. [En línea]. Disponible en: https://ieeexplore.ieee.org/document/8546240

[18] A. Savitzky, M. Golay. *Smoothing and Differentiation of Data by Simplified Least Squares Procedures*. Julio 1, 1964. Accedido en: Abril 11, 2021. [En línea]. Disponible en: https://pubs.acs.org/doi/abs/10.1021/ac60214a047

[19] E. Levina and P. Bickel. *The Earth Mover's distance is the Mallows distance: some insights from statistics.* Julio 14, 2001. Accedido en: Apr 11, 2021. [En línea]. Disponible en: https://ieeexplore.ieee.org/abstract/document/937632