# Análisis del Índice S&P 500: Descubrimiento de Conocimiento a través de Datos Financieros

Nicolas Vega Muñoz
Jorge Saenz de Miera

## Descripción del Dominio

El índice Standard & Poor's 500, comúnmente conocido como S&P 500, es uno de los indicadores financieros más influyentes y ampliamente seguidos en el mundo de las inversiones y las finanzas. Este índice representa la capitalización bursátil de las 500 principales empresas cotizadas en los mercados de Estados Unidos.

Este proyecto se enfoca en el análisis del índice S&P 500, con el objetivo de aplicar técnicas de minería de datos y aprendizaje automático para descubrir conocimiento a partir de los datos financieros asociados con las empresas que lo componen. A través de la exploración y el análisis de series temporales de precios de acciones, volúmenes de negociación y otros datos fundamentales, este proyecto tiene como propósito identificar patrones, tendencias y relaciones que puedan proporcionar información valiosa para inversores y analistas financieros.

En las siguientes secciones, exploraremos en detalle el dominio del S&P 500, los tipos de datos disponibles, los objetivos de descubrimiento de conocimiento y las técnicas de preprocesamiento de datos que se aplicarán en este proyecto.

## Características y tipos de datos

Se trata de un conjunto de varias series temporales cuyos valores son el precio de cierre de la acción en el intervalo deseado. Se podrían tener datos de distinta granularidad, desde cierre de precios por intervalos de minutos a días o semanas. Nosotros trabajaremos con un muestreo diario, pero todo lo que se estudiará será aplicable al resto de intervalos.

Es importante tener en cuenta la dificultad de analizar éstas series temporales debido a la presencia de la aleatoriedad por la interacción humana. Aún así se caracterizan por tener generalmente una tendencia (alcista/bajista) y ciertos patrones que se suelen seguir, pero nada garantiza que estos se repitan. En ocasiones puede haber cierta periodicidad, pero ésta no se rige por ninguna norma y no es trivial ni seguramente frecuente.

Lo que sí caracteriza a este tipo de series temporales es la presencia de máximos y mínimos que representan los precios alcanzados a los que por algún motivo la gente comienza a vender/comprar ya que se considera que el precio alcanzado es superior/inferior al valor de la acción, y esto viene acompañado de un cambio de tendencia que puede ser a micro o macro escala, pues en un día puede haber una tendencia bajista a la vez que la tendencia semanal es alcista por ejemplo.

# Objetivos de Descubrimiento de Conocimiento

El principal objeto de estudio en bolsa será la predicción de los próximos valores de cierre de acciones de una empresa concreta. De aquí se pueden sacar 2 planteamientos:

- Clasificación: Saber si el precio de cierre será superior o inferior al último dado.
- Regresión: Estimar el precio concreto de la próxima acción (o próximas acciones)

# Dataset y preprocesamiento

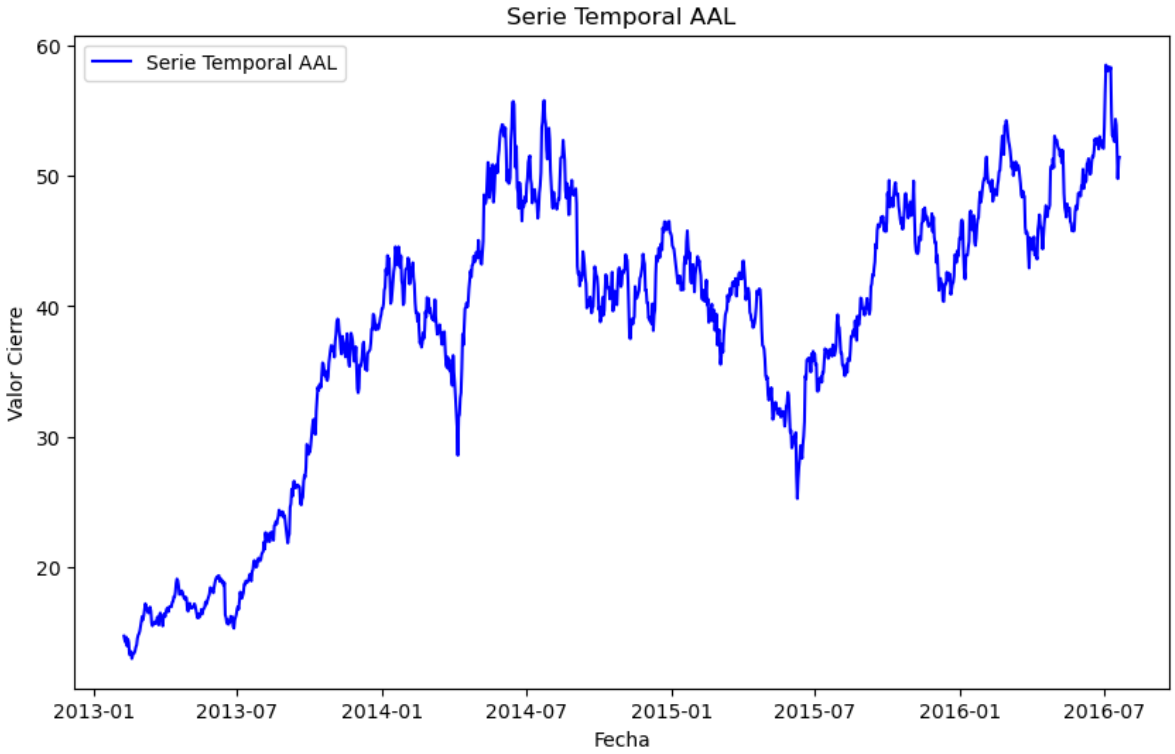
https://www.kaggle.com/datasets/camnugent/sandp500/

Contamos con un dataset que contiene 500 series temporales, 1 por cada empresa perteneciente a S&P. Disponemos de datos desde 2013-02-08 a 2018-02-07, 5 años.

Cada serie temporal cuenta con los siguientes datos:

- Fecha - en formato: yy-mm-dd
- Apertura - precio de la acción en la apertura del mercado (estos son datos de la NYSE, por lo que todos están en USD)
- Alto - precio más alto alcanzado durante el día
- Bajo - precio más bajo alcanzado durante el día
- Cierre - valor de la acción en el momento de cierre de ese dia
- Volumen - número de acciones negociadas
- Nombre - el nombre de la empresa

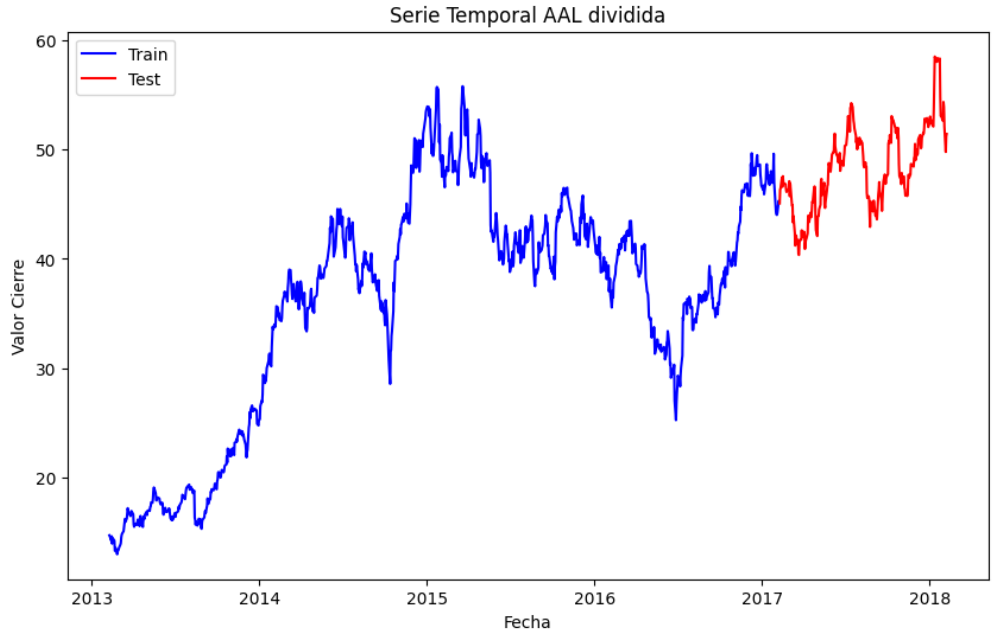
Ejemplo American Airlines:



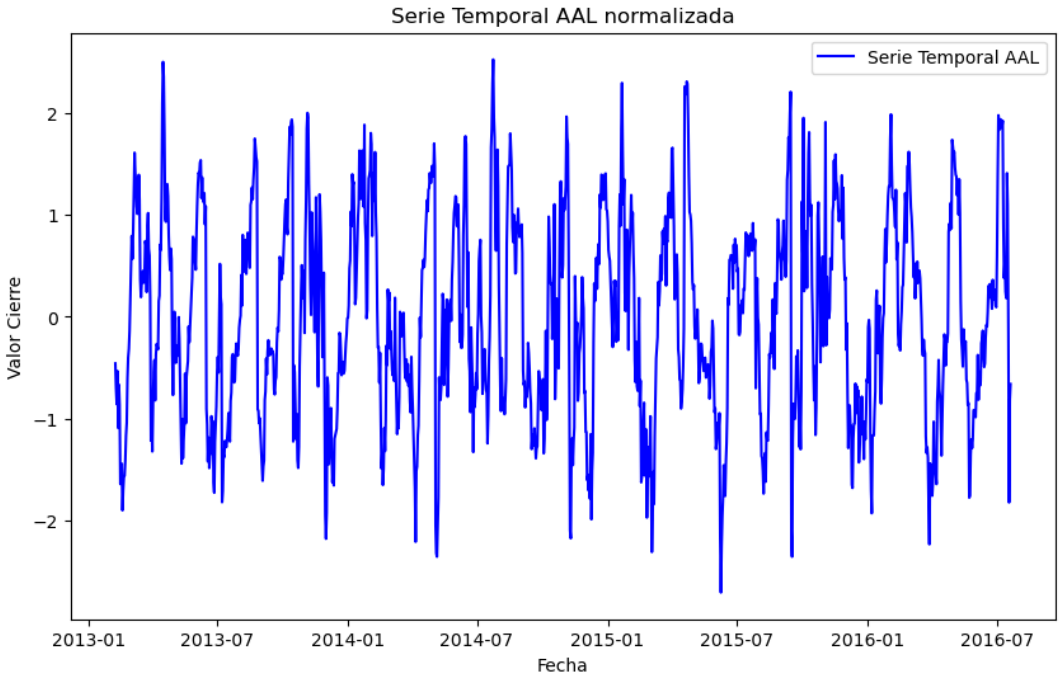

En cuanto a preprocesamiento el primer paso es obtener una serie temporal con el precio de cierre para cada una de las empresas. Lo que nos lleva a considerar, en primer lugar, la eliminación de las demás variables.

Una vez hecho esto, eliminamos aquellas empresas en las que hay valores faltantes en sus valores de cierre. De este modo eliminamos los datos de 35 empresas, quedándonos con un total de 1259 valores de cierre para cada una de las 470 empresas restantes.

Antes de aplicar técnicas de preprocesamiento, dividimos los datos en Train y Test. Para ello hemos decidido dejar el último año de los 5 que tenemos para Test, lo que equivaldría a una separación 80% - 20%

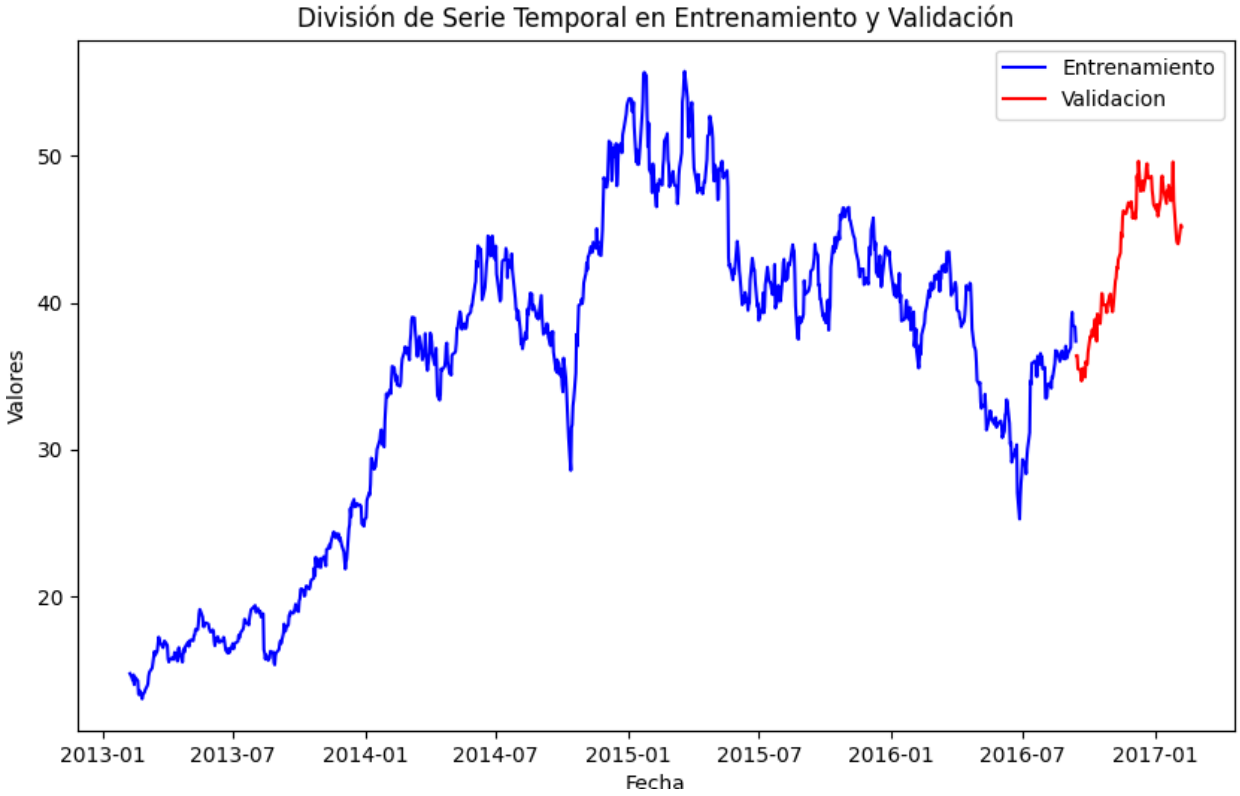



Una vez tenemos series temporales de entrenamiento completas de longitud 1007 realizamos una normalización dinámica pues sabemos que la bolsa es muy volátil a lo largo del tiempo y consideramos que establecer una media común a todas las empresas, o incluso una media para cada empresa sería un error pues se puede dar el caso de una empresa que en un inició los valores de cierre estuviesen en el rango 10-20€ y en el presente estén en el rango 1200-1300. En este caso no creemos que una simple normalización sea la opción correcta, y por ello realizamos una normalización por ventanas (normalización dinámica). En nuestro caso hemos optado por realizar una estandarización.



Una vez tenemos la serie temporal estandarizada dinámicamente, la transformamos en secuencias de datos, que son conjuntos de puntos consecutivos en la serie, que se utilizarán para entrenar el modelo. Este enfoque ayuda al modelo a aprender patrones y relaciones temporales en los datos.

Por último, antes de pasar a entrenar nuestro modelo, dividimos los datos en entrenamiento y validación. Vamos a utilizar el primer 90% de los datos para entrenar el modelo, y el 10% restante para evaluar los resultados del modelo.




# REGRESIÓN

Para hacer una predicción en nuestra serie temporal, vamos a probar con 2 tipos de modelos basados en redes neuronales, una red Long-Short Term Memory (LSTM) y una red convolucional (CNN). Para cada uno de los modelos implementaremos un modelo entrenado con datos sin normalizar y otro con datos normalizados.

## LSTM

```
lstm = Sequential()
lstm.add(LSTM(50, input_shape=(X_train.shape[1], 1)))
lstm.add(Dense(1))
lstm.compile(optimizer='adam', loss='mse')
```

Para el modelo LSTM, vamos a utilizar la red más básica posible, con 2 capas, la primera LSTM, y la segunda, una totalmente conectada, con una neurona de salida, para obtener una única salida, que corresponda a la predicción del siguiente valor de la serie temporal.

Las redes LSTM son muy utilizadas en la predicción de series temporales debido a su capacidad para capturar dependencias temporales a largo plazo en los datos. Las series temporales a menudo tienen patrones complejos y dependencias a largo plazo entre los puntos de datos, lo que hace que los modelos tradicionales de series temporales, como los métodos lineales o ARIMA, puedan resultar limitados, y utilizar este tipo de modelos, puede resultarnos muy útil.

## CNN

```
cnn = Sequential()
cnn.add(Conv1D(filters=64, kernel_size=3, activation='relu', input_shape=(X_train.shape[1], 1)))
cnn.add(MaxPooling1D(pool_size=2))
cnn.add(Flatten())
cnn.add(Dense(50, activation='relu'))
cnn.add(Dense(1))
cnn.compile(optimizer='adam', loss='mse')
```

Para el modelo de CNN, también definimos una arquitectura muy básica, la primera capa es una convolucional y tiene el objetivo de detectar patrones locales en la serie temporal, luego añadimos una capa pooling y otra Flatten, para reducir la dimensionalidad de las características, resaltando así las características más importantes, y convertir las características bidimensionales en un vector unidimensional, y por último dos capas completamente conectadas Dense, de las que, la primera utiliza la función de activación ReLU para introducir no linealidades adicionales y aprender representaciones más complejas, y la segunda tiene una sola neurona de salida, al igual que en LSTM, para determinar la predicción del siguiente valor de la serie

Las redes neuronales convolucionales (CNN) son también utilizadas en la predicción de series temporales debido a su habilidad para aprender automáticamente patrones espaciales y temporales en los datos. Aunque originalmente diseñadas para el procesamiento de imágenes, las CNN también pueden ser aplicadas a datos unidimensionales, como va a ser nuestro caso.

# Resultados

Vemos las predicciones de cada uno de los modelos:

**LSTM**:

- Sin normalización:

```
Errores LSTM:
Mean Squared Error (MSE): 8225.327508136937
Root Mean Squared Error (RMSE): 90.69359132891881
Mean Absolute Error (MAE): 44.70350518466598
R-squared (R^2): 6.388196922213485e-05
```

- Normalizado:

```
Errores LSTM normalizado:
Mean Squared Error (MSE): 1.9890293810518618
Root Mean Squared Error (RMSE): 1.4103295292419646
Mean Absolute Error (MAE): 1.1559882517338047
R-squared (R^2): 0.00030020314506706836
```

**CNN**:

- Sin normalización:

```
Errores CNN:
Mean Squared Error (MSE): 8256.562607997897
Root Mean Squared Error (RMSE): 90.86562940957322
Mean Absolute Error (MAE): 47.57789044116947
R-squared (R^2): -0.0037333047653334006
```

- Normalizado:

```
Errores CNN normalizado:
Mean Squared Error (MSE): 3.204628761330171
Root Mean Squared Error (RMSE): 1.7901476926025324
Mean Absolute Error (MAE): 1.4167694709673142
R-squared (R^2): -0.6106683753475006
```

Como se puede observar obtenemos unos resultados francamente malos, obteniendo incluso valores de R^2 negativos, lo cual no tiene sentido pues este toma valores en el rango (0,1). Observamos cómo realizando una regresión, tanto normalizando como sin normalizar, obtenemos resultados bastante malos. Este resultado sugiere la complejidad y la limitación de predecir eventos futuros en dominios como el mercado de valores, donde factores humanos y aleatorios juegan un papel crucial.

Una vez concluido que este enfoque no es el más adecuado procedemos a tratar de predecir si el próximo valor será mayor o menor que el actual, en vez de tratar de predecir el precio concreto. De ésta forma, convertimos el problema de regresión a uno de clasificación.

# CLASIFICACIÓN

## Implementación a nuestro dataset

Debemos adaptar nuestro dataset consistente de series temporales en la bolsa a un problema de clasificación. Nuestro objetivo será predecir, dados n valores temporales, si el siguiente valor será mayor o menor que el último, es decir, si las acciones de una empresa subirán o bajarán en el próximo día. De este modo nos enfrentaremos a un problema de clasificación binaria:

- 1: El precio de la acción sube.
- 0: El precio de la acción baja

Para ello debemos hacer un sencillo preprocesado de datos en el que dada la serie temporal etiquete como '1' aquellas en las que el último valor sea mayor que el penúltimo, y '0' las restantes. Finalmente eliminaremos el último valor de la serie temporal pues es este el que deseamos predecir, no numéricamente, sino si será mayor o menor que el último disponible.

Como hemos comentado previamente nuestro dataset completo consta de los datos de 500 empresas distintas, conformando así una serie temporal cada una de ellas.

## Conversion de TS a IMG

Se nos plantea convertir un conjunto de series temporales a imágenes para su futura clasificación mediante una CNN 2D.

En primer lugar deberemos establecer un procedimiento para convertir cada serie temporal a una imagen. Definimos las siguientes transformaciones:

1. Gramian Angular Field
2. Markov Transition Field

Mediante éstas técnicas lograremos transformar una serie temporal unidimensional en una matriz bidimensional, que tomará la forma de imagen.

### Gramian Angular Field

La construcción del Gramian Angular Field implica la transformación de la serie temporal en una matriz similar a la de Gram, que es una matriz simétrica que contiene información sobre los productos escalares entre pares de puntos en la serie. Ésta implementación sin embargo no usa la matriz de Gran directamente, pues el producto escalar que realiza lo hace sobre coordenadas polares.

$$G(x_1, \ldots, x_n) = \begin{vmatrix} \langle x_1, x_1 \rangle & \langle x_1, x_2 \rangle & \ldots & \langle x_1, x_n \rangle \\ \langle x_2, x_1 \rangle & \langle x_2, x_2 \rangle & \ldots & \langle x_2, x_n \rangle \\ \vdots & \vdots & & \vdots \\ \langle x_n, x_1 \rangle & \langle x_n, x_2 \rangle & \ldots & \langle x_n, x_n \rangle \end{vmatrix}$$

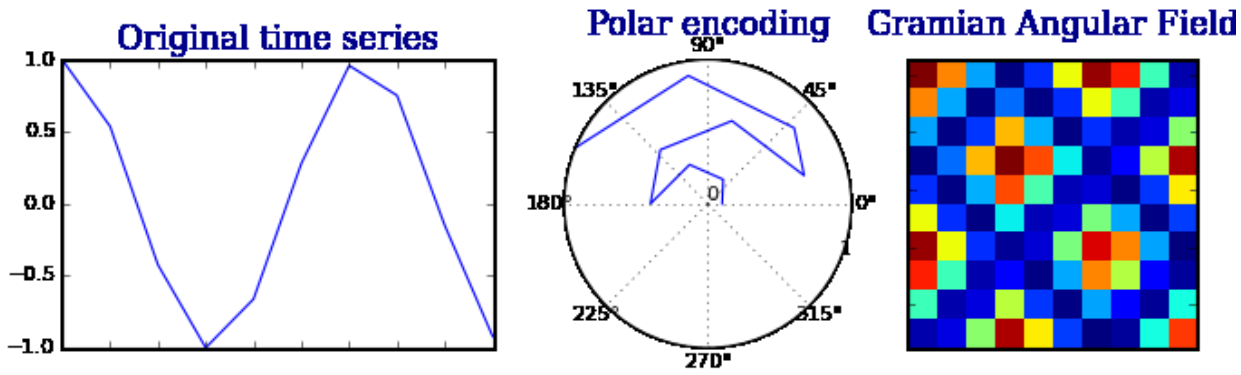
Matriz de Gram

El primer paso para la obtención de la matriz GAF es la normalización de los valores al intervalo [-1, 1]. Posteriormente procedemos a realizar un cambio de coordenadas a coordenadas polares, y finalmente definimos el producto escalar como:

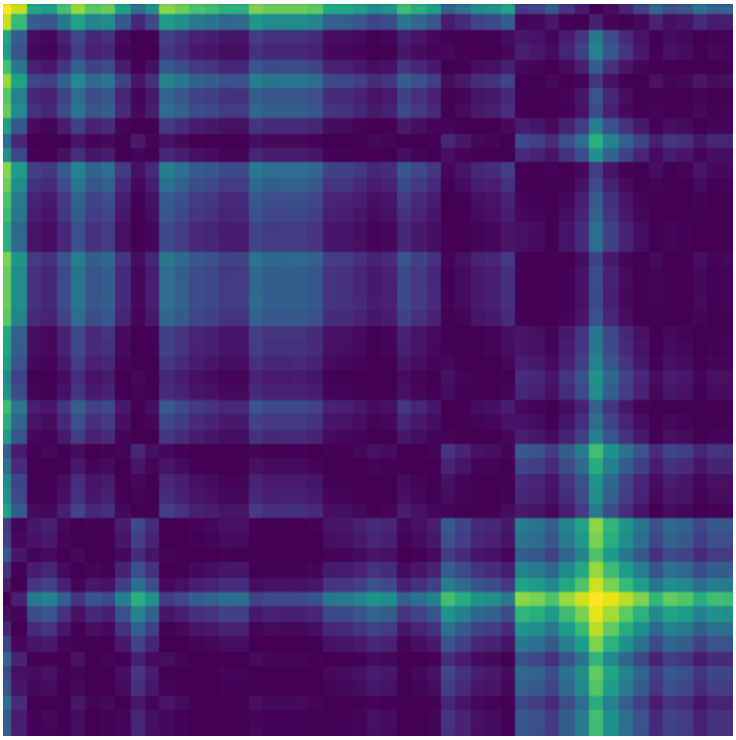
$$\langle x1,x2\rangle = \cos(\phi 1 + \phi 2); \text{ con } \phi i = \arccos(\text{norm}(x\_i))$$

La matriz se construye exactamente igual que la matriz de Gram ilustrada arriba, pero con el producto escalar definido.

La transformación angular tiene la propiedad de preservar las relaciones de orden entre los valores de la serie temporal, lo que es útil para capturar la información relevante sin perder la estructura temporal.




Ejemplo de serie temporal transformada a imagen mediante GAF:



**Markov Transition Field**

Supongamos que tenemos una serie temporal de longitud N. Comenzamos colocando cada valor en la serie temporal en cuantiles (es decir, "agrupamos" cada valor). Por ejemplo, si usamos cuartiles (4 grupos), el 25% más pequeño de los valores definiría los límites del primer cuartil, el segundo 25% más pequeño de los valores definiría los límites del segundo cuartil, etc. Lo que se pretende  hacer es formar la matriz de transición de estados.

La matriz será compuesta por valores:

$$Aij=P(st=j|st-1=i)$$

Siendo Aij la probabilidad de pasar del estado i al j.

estimamos este valor mediante máxima verosimilitud. Aij  es el recuento de transiciones de  i a j, dividido por el número total de veces que estuvimos en el estado i. Ten en cuenta que si tenemos Q cuantiles (es decir, tenemos Q "estados"), entonces A es una matriz de Q×Q.

Llamaremos a la Matriz M:

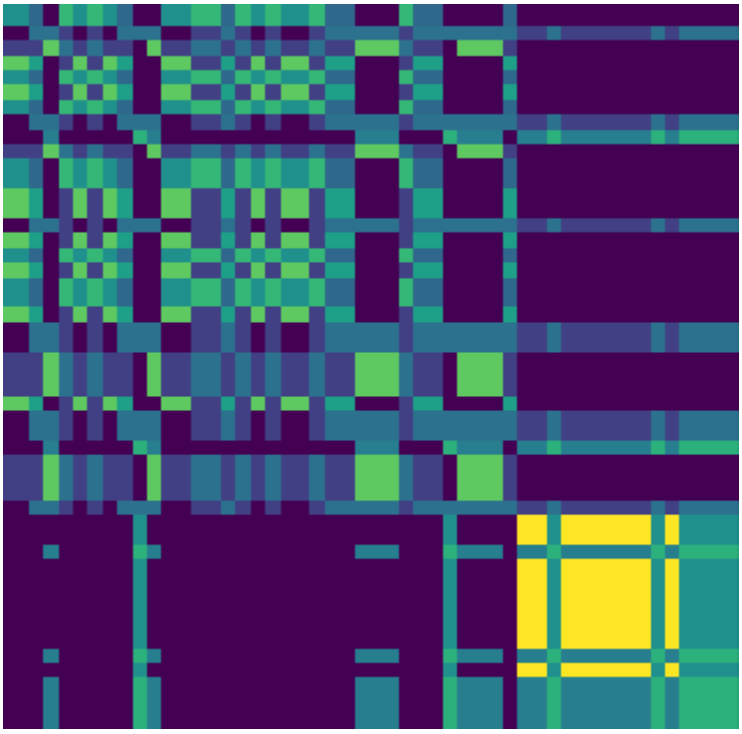
$$M_{kl} = A_{q_k q_l}$$

Mkl será la probabilidad de una transición el contenedor xk  al contenedor xl.
qk  es el cuantil ("contenedor") para xk  , y ql  es el cuantil para xl.

Es decir, se observa xk  y xl, que son 2 puntos en la serie temporal en pasos de tiempo arbitrarios k y  l. qk  y ql  son los cuantiles correspondientes. Mkl  es simplemente la probabilidad de que hayamos visto una transición directa de un paso (es decir, Markoviana) de qk  a ql  en la serie temporal.

Mkl proporciona una medida de cuán relacionados están dos puntos arbitrarios en la serie temporal en comparación con la frecuencia con la que aparecen uno al lado del otro en la serie temporal. En otras palabras, refleja la probabilidad de una transición directa de un paso desde el cuantil qk  al cuantil ql  , lo que puede interpretarse como una medida de la relación o dependencia entre esos dos puntos específicos en la serie temporal.

Ejemplo serie temporal a imagen mediante MTF:



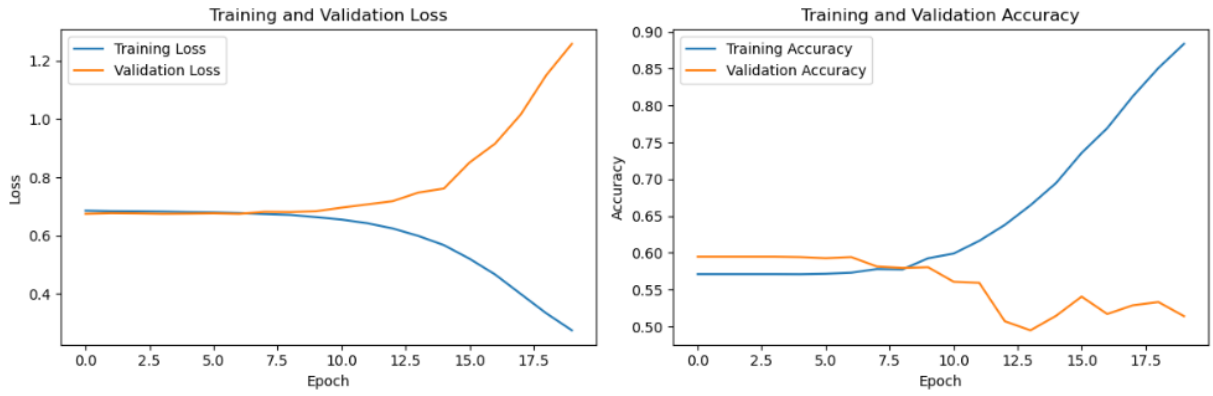
## Modelos CNN

Procedemos a entrenar una red convolucional (CNN) para la clasificación de series temporales transformadas a imágenes mediante GAF y MTF. La estructura de la CNN será la misma para ambos casos:

```
model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu',
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))

model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(1, activation='sigmoid'))
```

**MODELO GAF:**




Como se puede observar a partir de las 10 épocas se produce un overfitting, parece que con aproximadamente 5  épocas obtenemos los mejores resultados.

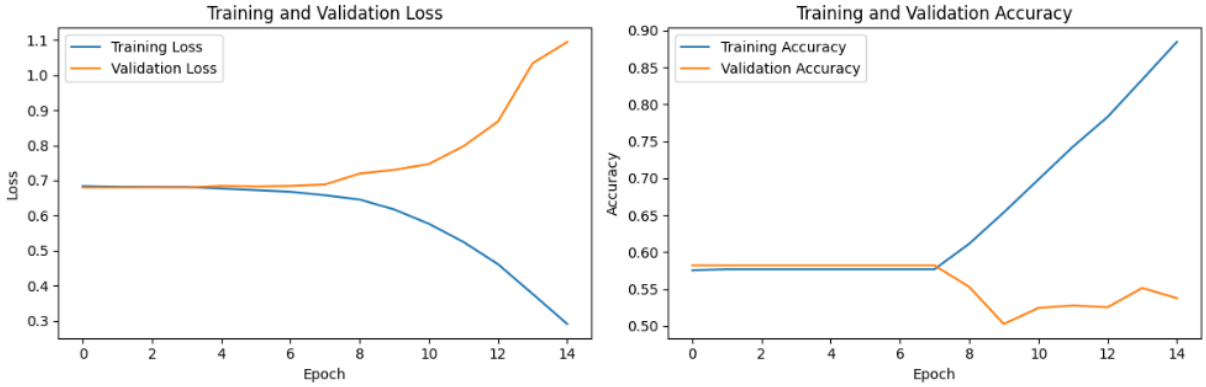
----------Confusion Matrix----------

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| False | 0.58 | 0.66 | 0.62 | 633 |
| True | 0.47 | 0.39 | 0.42 | 495 |
| accuracy | | | 0.54 | 1128 |
| macro avg | 0.52 | 0.52 | 0.52 | 1128 |
| weighted avg | 0.53 | 0.54 | 0.53 | 1128 |

Al evaluar el modelo con los datos de test observamos una accuracy de un 54%. Este podría parecer un resultado relativamente positivo al tener en cuenta el dominio en el que estamos, pero realmente este no es un buen resultado debido a que su accuracy es menor que la simple elección de la clase mayoritaria (0.54 < 633/(633+495) = 0.56).

**Modelo MTF:**

Evaluaremos ahora los resultados obtenidos mediante otra transformación, MTF.




Al igual que en el anterior caso observamos un rápido overfitting desde las primeras épocas y muy poca variación en la accuracy del modelo. También parece que los mejores resultados se obtienen con pocas épocas.

----------Confusion Matrix----------

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| False | 0.57 | 0.60 | 0.59 | 633 |
| True | 0.46 | 0.43 | 0.44 | 495 |
| accuracy | | | 0.53 | 1128 |
| macro avg | 0.52 | 0.52 | 0.52 | 1128 |
| weighted avg | 0.52 | 0.53 | 0.52 | 1128 |

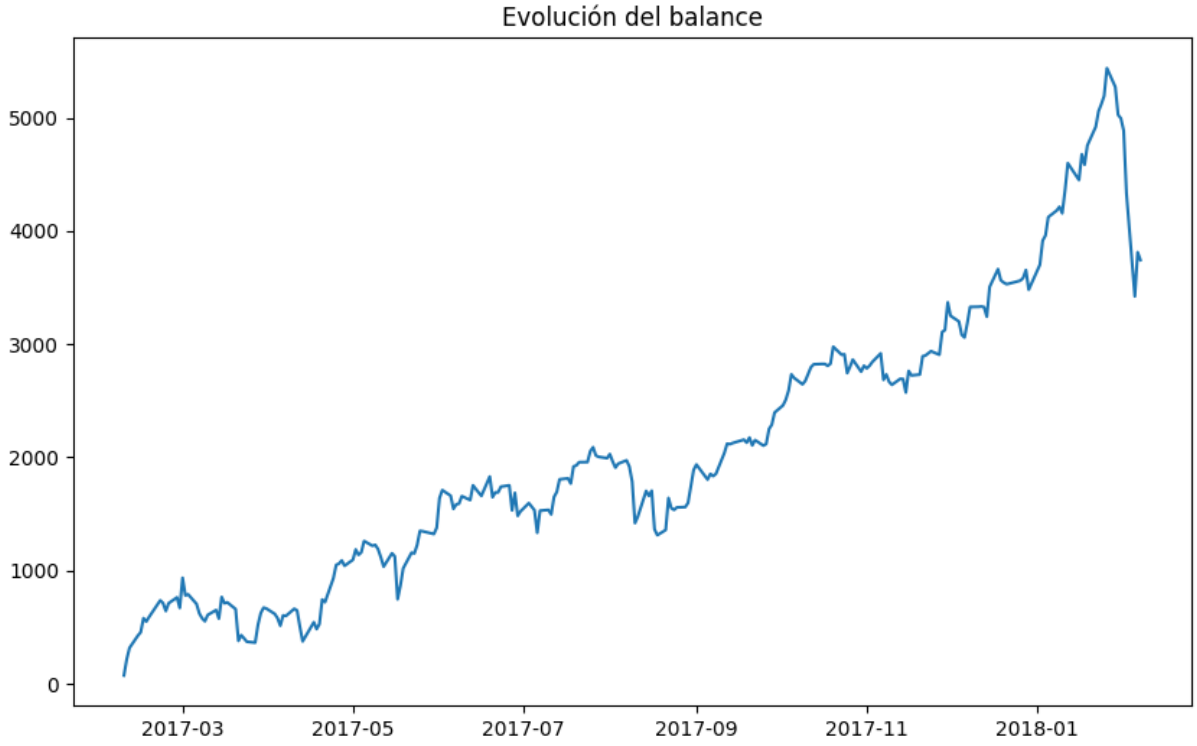
Obtenemos unos resultados peores que en el anterior modelo. Tenemos una accuracy de 53%, muy cercano al modelo aleatorio.

# PUESTA EN PRÁCTICA CON DATOS DE TEST

Para tener una visión más clara de cómo funcionan cada uno de nuestros modelos, en lugar de evaluarlos con las métricas típicas de regresión y clasificación, vamos a hacer una simulación de cómo sería aplicar estos modelos durante el año que habíamos reservado para Test. Para

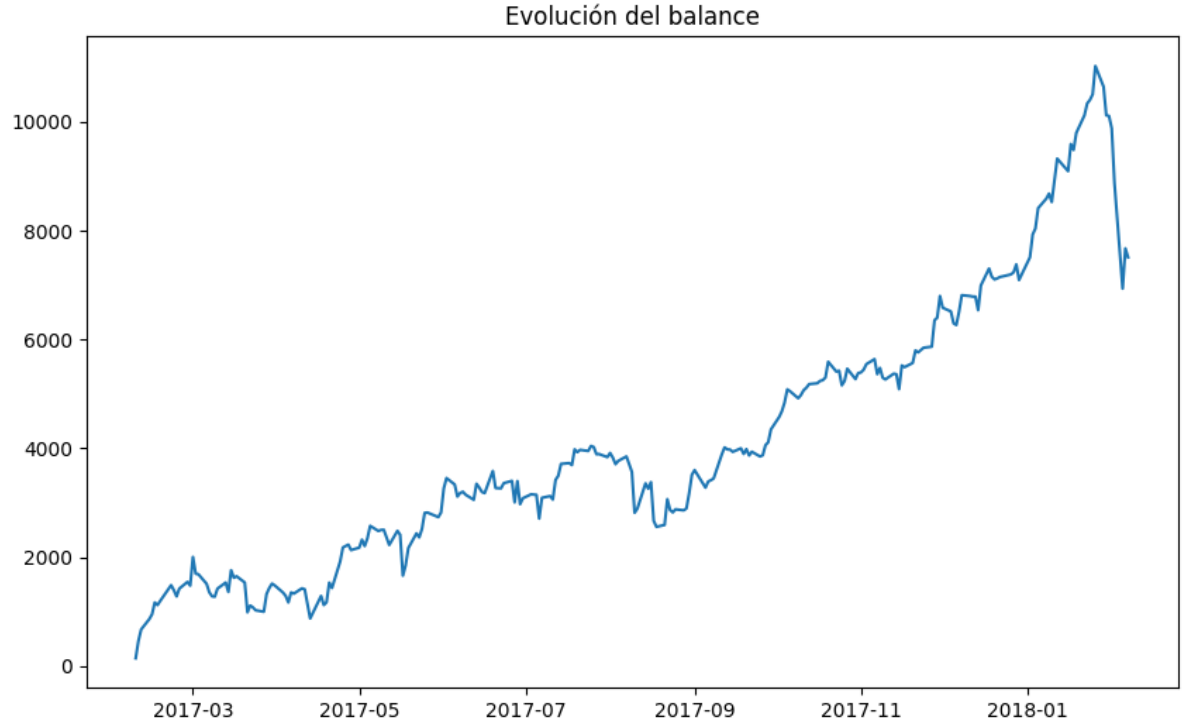
ello, vamos a simular que cada día del año vamos a predecir qué acciones van a subir el día siguiente, y vamos a simular que compramos una acción de cada una de las empresas predichas. Luego, fijándonos en los datos de test reales, vamos a simular que vendemos todas esas acciones al día siguiente con el precio real. La idea es repetir esto todos los días, y ver si podríamos llegar a obtener beneficios con alguno de nuestros modelos.
La única parte de la simulación en la que entran nuestros modelos es en predecir, a partir de la última ventana de cada serie temporal, si cada empresa va a subir o no. Para tener una base, vamos a crear una función que clasifica en comprar - no comprar de manera aleatoria

```
El balance final es de 3743.25 $
Con una inversión total de 6055074.67 $
El día con la mayor inversión fueron 29928.24 $
El día con mayor beneficio fueron 390.21 $,
El día con mayores pérdidas fueron -908.69 $,
Se han comprado un total de 59314 acciones, con una media de 235.37 por día
```




Vemos que con una clasificación aleatoria obtendríamos un beneficio a lo largo del año de unos 3600 $. Vamos a utilizar esta predicción como base para evaluar los modelos que hemos visto previamente. Obtener beneficios de esta manera, puede ser porque el año en el que estamos haciendo estas comprobaciones, ha sido en general bueno para todas las empresas del índice, y por eso, comprando cada día las acciones que fuesen, vamos a obtener beneficios al final del año. También debemos tener en cuenta que estamos considerando una compra sin comisiones. Lo más normal en el mundo real es que haya un porcentaje de comisión que reduce considerablemente estos beneficios.
Para comprobar que los resultados obtenidos con el clasificador aleatorio son coherentes, vamos a probar también qué sucedería si nuestro clasificador clasifica todas las empresas siempre como que van a subir.

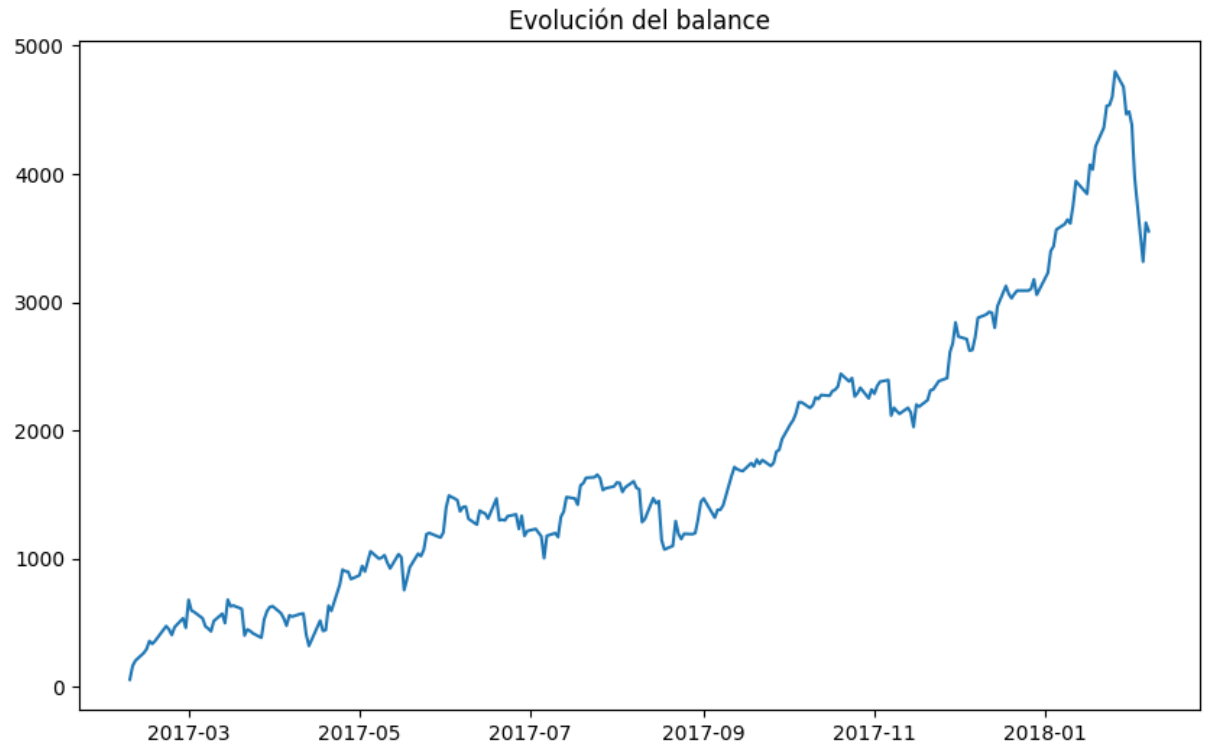
```
El balance final es de 7506.39 $
Con una inversión total de 12104427.01 $
El día con la mayor inversión fueron 54997.28 $
El día con mayor beneficio fueron 739.55 $,
El día con mayores pérdidas fueron -1938.37 $,
Se han comprado un total de 118440 acciones, con una media de 470.0 por día
```




Efectivamente vemos que al comprar todas las acciones todos los días se consigue un resultado muy parecido, incluso con un beneficio mayor. Esto puede resultar engañoso, ya que, lo único que estamos haciendo con respecto al caso anterior es aumentar todos los valores de los resultados obtenidos. Con una buena predicción no solo queremos maximizar el balance final que tenemos, sino que queremos hacerlo invirtiendo la menor cantidad posible. En concreto, el valor de día con la mayor inversión, quiere decir que necesitamos tener disponible en un día esa cantidad de dinero, y en este último caso son 55.000 $, un gasto para un único día irreal. Para futuras evaluaciones, vamos a fijarnos en el valor (balance final)/(máxima inversión en un día), que en los dos casos probados es similar, 0.13.

Probamos ahora con nuestro modelo de clasificación basado en una CNN que clasifica imágenes obtenidas con MTF

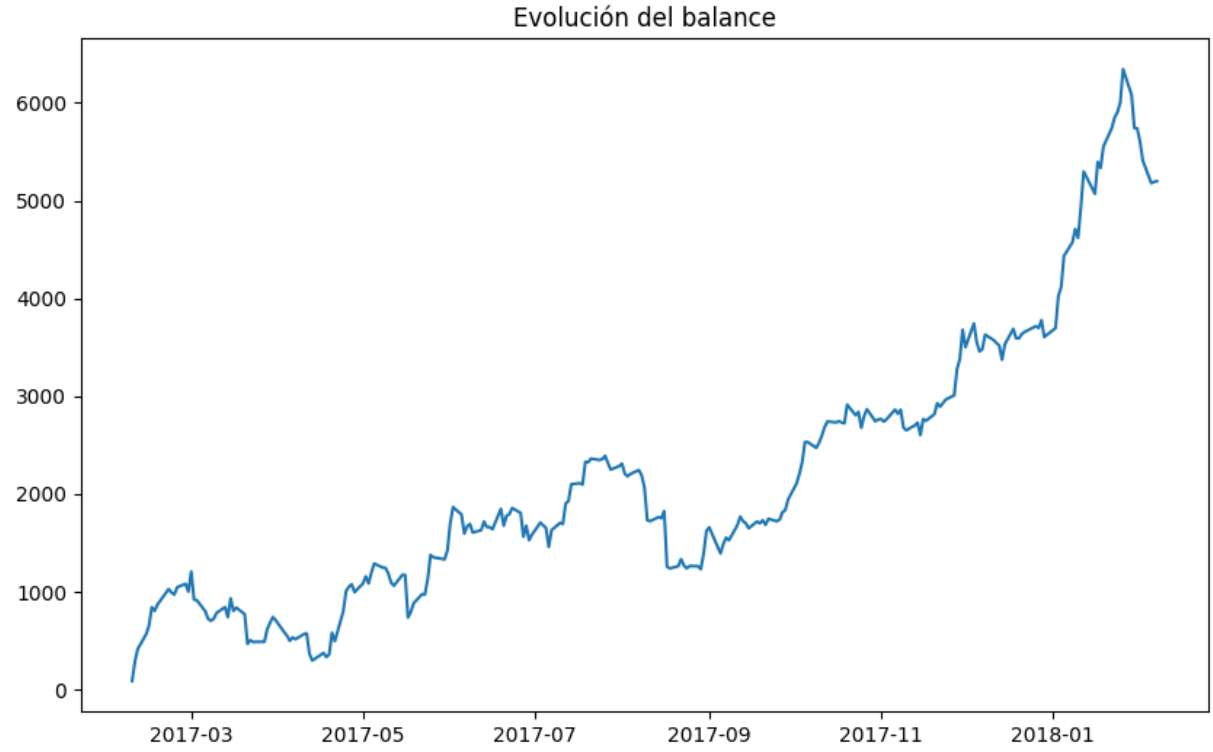
```
El balance final es de 3553.23 $
Con una inversión total de 4974887.41 $
El día con la mayor inversión fueron 28574.44 $
El día con mayor beneficio fueron 303.33 $,
El día con mayores pérdidas fueron -654.45 $,
Se han comprado un total de 48561 acciones, con una media de 192.7 por día
```




Vemos que los resultados son muy similares a los obtenidos con una clasificación aleatoria. La primera diferencia que apreciamos, es que el día con más inversión son 28.500$, más de 1000$ menos que en una clasificación aleatoria. Además el día con mayores pérdidas, se pierden 300$ menos que en la aleatoria, lo que significa que se están detectando bien algunas de las empresas en las que no deberíamos invertir.

El resto de modelos tienen todos un resultado muy similar, que no mejoran significativamente el azar, pero los resultados son ligeramente mejores. Para probar si podemos mejorar el azar, fuera de los modelos propuestos, vamos a probar a usar otro tipo de reglas sencillas para predecir si una serie va a subir o no.
La primera que hemos implementado, se basa en la tendencia, si la ventana a analizar, presenta una tendencia alcista, se predice que esta acción puede seguir subiendo, y se simula su compra

```
El balance final es de 5198.2 $
Con una inversión total de 7020170.01 $
El día con la mayor inversión fueron 44915.74 $
El día con mayor beneficio fueron 363.67 $,
El día con mayores pérdidas fueron -568.02 $,
Se han comprado un total de 66206 acciones, con una media de 262.72 por día
```




Vemos que sigue la línea de la predicción aleatoria, obtenemos un mayor beneficio, pero con una mayor inversión

# CONCLUSIÓN FINAL

En este proyecto de Descubrimiento de Conocimiento aplicado al análisis de series temporales, hemos abordado de manera integral diversas etapas, desde la exploración y preprocesamiento de datos hasta la implementación y evaluación de modelos de aprendizaje automático.

En el proyecto, abordamos el análisis del índice S&P 500, un indicador financiero complejo, con el objetivo de aplicar técnicas de minería de datos y aprendizaje automático en series temporales de precios de acciones. Trabajamos con un conjunto de datos que incluía 500 series temporales diarias desde 2013 hasta 2018, centrándonos en el precio de cierre.

Realizamos un preprocesamiento que implicó la eliminación de variables innecesarias, una normalización dinámica por ventanas y la transformación de las series en secuencias para el entrenamiento de modelos de regresión (LSTM y CNN). Ante los resultados insatisfactorios de los modelos de regresión, convertimos el problema a clasificación binaria, implementando modelos de clasificación con imágenes generadas a partir de Gramian Angular Field (GAF) y Markov Transition Field (MTF). Aunque los resultados fueron modestos, se superó ligeramente el rendimiento aleatorio. Evaluamos estrategias prácticas de compra y venta, reconociendo la complejidad del mercado financiero y destacando el potencial de los modelos como herramientas de apoyo en la toma de decisiones financieras.

En resumen, este proyecto brindó una comprensión profunda de los desafíos asociados con la predicción en el mercado financiero y resaltó la importancia de contextualizar los resultados en función de la complejidad inherente a este dominio. Las estrategias propuestas podrían requerir mejoras y ajustes adicionales para lograr resultados más sólidos y aplicables en un entorno práctico.

# Referencias


- https://www.tensorflow.org/?hl=es-419
- https://pandas.pydata.org/docs/
- https://www.aprendemachinelearning.com/pronostico-de-series-temporales-con-redes-neuronales-en-python/
- https://www.linkedin.com/pulse/tiempedia-modelos-cnn-aplicados-series-de-tiempo-mora-caballero-lwmne/?trk=article-ssr-frontend-pulse_more-articles_related-content-card&originalSubdomain=es
- https://lazyprogrammer.me/convert-a-time-series-into-an-image/
- https://pyts.readthedocs.io/en/stable/modules/image.html
- Z. Wang and T. Oates, “Encoding Time Series as Images for Visual Inspection and Classification Using Tiled Convolutional Neural Networks.” AAAI Workshop (2015).