TRABAJO DE FIN DE MÁSTER

Máster Interuniversitario en Estadística e Investigación Operativa UPC-UB

Título:Redes neuronales convolucionales, reconocimiento de imágenes y pronósticos financieros.

Autor: Eduardo Sepulveda Valdivia

Director: Argimiro Arratia Quesada

Ponente: Pedro Delicado Useros

Departamento: Estadística e investigación operativa.

Universidad: Universidad Politécnica de Cataluña

Convocatoria: **Octubre 2019**

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat de Matemàtiques i Estadística

UNIVERSITAT DE BARCELONA

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat de Matemàtiques i Estadística

UNIVERSITAT DE BARCELONA

Facultat de Matemàtiques i Estadística
Universitat Politècnica de Catalunya

Tesis de máster

Master en Estadística e Investigación Operativa

Redes Neuronales Convolucionales, reconocimiento de imágenes y pronósticos financieros

Autor: Eduardo Sepulveda Valdivia

Director: Argimiro Arratia Quesada

Departamento de Computación

Barcelona, octubre de 2019

Agradecimientos

A Argimiro Arratia, por su inestimable guía, paciencia e inigualable forma de transmitir sus conocimientos.

A mis amigos, los cuales conoci durante estos años en Barcelona y que sin duda comparti grandes alegrías y experiencias durante este tiempo.

Y a Franziska, por su paciencia y amor incondicional.

Resumen

Antecedentes: El descubrimiento, reconocimiento y análisis de patrones en el mundo que nos rodea es un factor fundamental para los progresos científicos y tecnológicos. En la actualidad se observa que se generan grandes volúmenes de datos mediante internet y en un estudio realizado por International Business Machines (IBM) durante el 2017 [1], se esbozó que el 90% de todos los datos de internet en ese entonces se habían creado en los últimos dos años. Lo cual se atribuia principalmente al increíble crecimiento de Internet, donde en el 2012, se tenía 2.500 millones, en el 2014 se tenía 3.000 millones, y en 2019 se tienen 4.100 millones de personas en línea los cuales generan alrededor de 2.5 quintillones de bytes de datos cada día.

Según la revista Forbes, los ingresos mundiales para Big Data y análisis de negocios crecerán a más de $ 203 mil millones en el año 2020, a una tasa de crecimiento anual compuesta del 11.7% y que la industria con la mayor inversión en big data y soluciones de análisis de negocios se verá en la banca [2].

Por estas razones, nacen nuevos conceptos en el mundo empresarial como por ejemplo la cultura “Data Driven”, la cual busca mejorar el proceso de toma de decisiones considerando que el futuro del trabajo gira en torno a los datos y su análisis, mediante la aplicación de nuevas tecnologías y técnicas de aprendizaje automático e inteligencia artificial. Estas tecnologías buscan una mejor predicción a pronósticos financieros, gestión del riesgo e identificar nuevos clientes potenciales, etc. Las implicaciones de esta tecnología son enormes, pero la mayoría de los bancos todavía están en las primeras etapas de la adopción.

Objetivo: Las redes neuronales convolucionales (CNN) son mejor conocidas como buenos clasificadores de imágenes. Estos modelos se han utilizado recientemente para pronósticos financieros. El propósito de este trabajo es mostrar que al convertir la información financiera en imágenes y alimentar con estas representaciones de imágenes financieras a la CNN, se obtiene una mejora en la clasificación respecto a utilizar datos numéricos sin conversión a imagen.

Métodos: Se construyeron dos tipos de modelos de red neuronal convolucional. En ambos casos, se utilizó un modelado secuencial, el cual, es una forma fácil de construir un modelo capa por capa en Keras. Dado que el objetivo es probar la posible mejora en la clasificación de una CNN con gráficos de recurrencia (RP), no existe la preocupación de ajustar de manera sofisticada el funcionamiento interno de la CNN si no, más bien, trabajar con una estructura básica a la cual se alimenta la entrada procesada como imagen a través del método de gráfico de recurrencia, o no.

Resultados y conclusiones: La comparación de los resultados obtenidos mediante la utilización de Redes Neuronales Convolucionales (CNN) vs Redes Neuronales Convolucionales (CNN) y los Gráficos de Recurrencia (RP), demostró que, al realizar un procesamiento previo a los datos numéricos de entrada convirtiéndolos en imágenes mediante la transformación del gráfico de recurrencia (RP), se obtienen mejores resultados de clasificación en ambos experimentos. Igualmente se destaca que la varianza entre las generaciones (100) disminuye mediante la utilización de RP, siendo asi un resultado mas confiable. Los resultados se midieron en base a cuatro métricas diferentes de rendimiento de clasificación (acc, loss, K-fold, AUC). Ademas, se utilizó una norma estándar (norma euclidiana) para la definición de Gráficos de Recurrencia. Es posible utilizar otras normas, en particular las más adecuadas para capturar similitudes entre series de tiempo financieras, como, por ejemplo, una métrica basada en correlación y sería interesante ver la diferencia en el rendimiento del modelo Conv2D + RP bajo diferentes normas subyacentes a la definición del método RP.

Difusión de los resultados: Los resultados de esta tesis han sido presentados en la conferencia:
A. Arratia & Eduardo Sepúlveda (2019). Convolutional Neural Networks, image recognition and financial time series forecasting . In: ECML-PKDD. Proc. 4th Workshop MIDAS (MIning DAta for financial applicationS). September 16, 2019 - Wurzburg, Germany.

Palabras clave: Redes Neuronales Convolucionales, Gráficos de Recurrencia, Series de Tiempo, Predicción.

Abstract

Background: The discovery, recognition and analysis of patterns in the world around us is a fundamental factor for scientific and technological progress. At present we observe that we generate large volumes of data through the internet and in a study conducted by International Business Machines (IBM) during 2017 [1] it was outlined that 90% of all data from internet at that time had been created in the last two years. Which was mainly attributed to the incredible growth of the Internet, where in 2012, we had 2,500 million, in 2014 we had 3,000 million, and in 2019 we have 4,100 million people online which generate about 2.5 quintillion bytes of data every day.

According to Forbes magazine, it is said that global revenue for Big Data and business analysis will grow to more than $ 203 billion in 2020, at a compound annual growth rate of 11.7% and that the industry with the largest investment in Big Data and business analysis solutions will be seen in banking [2].

That is why new concepts are born in the business world, such as the “data driven” culture, which seeks to improve the process of making decisions considering that the future of work revolves around data and its analysis, through the application of new technologies and machine learning techniques and artificial intelligence. These technologies seek a better prediction of financial forecasts, risk management and identify new potential clients, etc. The implications of this technology are enormous, but most banks are still in the early stages of adoption.

Objective: Convolutional Neural Networks (CNN) are best known as good image classifiers. This model is recently been used for financial fore- casting. The purpose of this work is to show that by converting financial information into

images and feeding these financial-image representations to the CNN, it results in an improvement in classification.

Methods: There were built two types of Convolutional Neural Network models using the Keras library in Python. In both cases it was used sequential modeling because is the easiest way to build a model layer by layer in Keras. Since the goal is to test the possible enhancement in classification of a CNN with a RP it will not care about tuning in any sophisticated way the inner workings of the CNN and rather work with a basic structure upon which we feed the input processed as image through the recurrence plot method, or not. In the following list it is detailed the structure of the Convolutional Neural Network in the models and difference between them. The main difference is the pre-process of data inputs.

Results and conclusions: The comparison of the results obtained through the use of CNN vs CNN + RP, showed that, by pre-processing the input by transforming the recurrence graph (RP) of the numerical data into images, it is obtained a better classification results in Both experiments. It can also be highlighted that the variance between generations (100) decreases through the use of RP, thus being a more reliable result. The results were measured based on four different metrics of classification performance (acc, loss, K-fold, AUC).
In addition, a standard standard (Euclidean standard) was used for the definition of recurrence graphs. It is possible to use other standards, particularly the most appropriate ones to capture similarities between financial time series, such as a correlation-based metric and it would be interesting to see the difference in performance of the Conv2D + RP model under different standards underlying the RP method definition.

Communication of results: The results of this thesis have been presented at the conference:
A. Arratia & Eduardo Sepúlveda (2019). Convolutional Neural Networks, image recognition and financial time series forecasting . In: ECML-PKDD. Proc. 4th Workshop MIDAS (MIning DAta for financial applicationS). September 16, 2019 - Wurzburg, Germany.

Keywords: Convolutional neural networks, recurrence plots, time series, forecasting

Índice General

1 Introducción 1
2 Conceptos metodologicos 5
2.1 Graficos de recurrencia (RP) 5
2.2 Redes Neuronales Convolucionales (CNN). 7
2.3 Series de tiempo (TS). 10
2.4 Metricas para evaluación del modelo 11
2.4.1 Precisión de clasificación 11
2.4.2 Pérdida logarítmica o pérdida 11
2.4.3 Validacion cruzada (K-Fold) 12
2.4.4 Area bajo la curva (AUC) 12
3 Procesamiento de los datos 14
3.1 Indice S&P500 14
3.2 Data de bancos 15
4 Experimentos y resultados 17
4.1 Especificaciones de la CNN 18
4.2 Experimentos realizados 20
4.2.1 Experimiento 1: Prediccion de la dirección del SP&500 20
4.2.2 Experimento 2: Prediccion de quiebra de bancos 23
5 Conclusiones 26
Bibliografía 28
A Codigo python 29

Índice de figuras

Figura 1: Ejemplo de grafico de recurrencia 6
Figura 2: Estructura general de las redes neuronales Convolucionales 8
Figura 3: Graficos de Recurrencia distintos periodos S&P500 21
Figura 4: Resultados graficos S&P500 22
Figura 5: Graficos de recurrencia de distintos bancos de EEUU 24
Figura 6: Resultados graficos quiebra de bancos 25

Índice de tablas

Tabla 1: Resultados experimento S&P500 21
Tabla 2: Resultados experimento quiebra de bancos 24

Capítulo I

Introducción

En la actualidad se observa que se generan grandes volúmenes de datos, especialmente mediante internet, en un estudio realizado por International Business Machines (IBM) durante el 2017 [1], se esbozó que el 90% de todos los datos de internet en ese entonces se habían creado en los últimos dos años. Se atribuia principalmente al increíble crecimiento de Internet, donde en el 2012, se tenían 2.500 millones, en el 2014 se tenían 3.000 millones, y en 2019 se tienen 4.100 millones de personas en línea los cuales generan alrededor de 2.5 quintillones de bytes de datos cada día. Según, la revista Forbes los ingresos mundiales para Big Data y análisis de negocios crecerán a más de $ 203 mil millones en el año 2020, a una tasa de crecimiento anual compuesta del 11.7% y que la industria con la mayor inversión en Big Data y soluciones de análisis de negocios se vera en la banca [1].

La gran cantidad de datos históricos y la alta precisión que poseen las instituciones financieras son activos valiosos, pero no se comprenden ni explotan completamente en los procesos de toma de decisiones. Es por esto, que nacen nuevos conceptos en el mundo empresarial como por ejemplo la cultura “data driven”, la cual busca mejorar el proceso de toma de decisiones considerando que el futuro del trabajo gira en torno a los datos y su análisis, mediante la aplicación de nuevas tecnologías y técnicas de aprendizaje automático e inteligencia artificial. Estas tecnologías buscan una mejor predicción a pronósticos financieros, gestión del riesgo e identificar nuevos clientes potenciales, etc. Las implicaciones de esta tecnología son enormes, pero la mayoría de los bancos todavía están en las primeras etapas de la adopción.

De esta manera, la adopción de técnicas matemáticas mas elaboradas ha tomado fuerza en los últimos años, los modelos de predicción o clasificación basados en el aprendizaje profundo comenzaron a emerger empíricamente como los mejores que lograron el rendimiento en diversas aplicaciones, superando a los métodos clásicos de inteligencia computacional como Support Vector Machine. Sin embargo, en el procesamiento de imágenes estos modelos de aprendizaje profundo superan a las otras técnicas [3]. En la literatura, los métodos de aprendizaje profundo han comenzado a aparecer en los estudios financieros y existen algunas implementaciones de técnicas de aprendizaje profundo como, las Redes Neuronales Convolucionales o Convolutional Neural Networks (CNN), cuya primera arquitectura se remonta al modelo propuesto por [3], las cuales han demostrado empíricamente a lo largo del tiempo que estas son mucho mejores para clasificar imágenes y estructuras generales de alta dimensión que las redes neuronales clásicas o tradicionales. Una característica distintiva de CNN es que explota la coherencia espacial local de las imágenes. Esto se logra al aplicar convoluciones a pequeños cuadrados de la entrada, a través de una ventana deslizante que recorre toda la imagen, una CNN aprende las características locales de la imagen y, en consecuencia, conserva la relación espacial entre sus píxeles. Cada convolución está determinada por un filtro (o núcleo) que define el tamaño de la ventana deslizante para capturar la información local (extracción de características) de la imagen. Aplicando diferentes filtros (y convoluciones), la CNN aprende diferentes mapas de características, mejorando así su capacidad de reconocer patrones específicos en los datos de entrada. Para mayores detalles matemáticos del modelo CNN revisar [4].

En el ámbito de las series de tiempo financieras, la motivación para usar las CNN es claramente su capacidad para aprovechar la información estructural (local) dentro de los datos, idealmente mediante el aprendizaje de mapas de características (o sus filtros originales). Estos filtros representan patrones específicos en la serie temporal los cuales son significativos para la predicción de valores futuros, como los extraídos del análisis técnico; o patrones en variables exógenas dependientes del tiempo consideradas como predictores

potenciales, tales como los obtenidos del análisis fundamental [5]. Sin embargo, la imagen limitada que ofrece la naturaleza 1D de los datos de series temporales podría restringir la capacidad de reconocimiento de una CNN. Como se ha estudiado bien en la teoría de señales, algunas características de las series temporales se identifican mejor en el dominio de la frecuencia que en el dominio del tiempo. Del mismo modo, hacer una representación de mayor dimensión de una serie temporal puede dar una imágen más rica de sus diferentes características que una CNN puede reconocer y aprovechar mejor. Es aquí donde la metodología de la gráfica de recurrencia para transformar matrices numéricas en una matriz de bits (una imagen), tiene un papel fundamental para comparar si aplicando una transformación a los datos de entrada de la CNN a 2D se pueden obtener mejores resultados que utlizando datos de entrada de 1D.

El contenido del trabajo se estructura de la siguiente manera:

Capítulo 2: En este capítulo explicaremos los conceptos más importantes utilizados, tanto de los Gráficos de Recurrencia, Redes Neuronales Convolucionales, Series de Tiempo, Loss, K-fold y AUC. Daremos una visión general de cada tema y además una explicación matemática lo cual tendra gran importancia a lo largo del trabajo.

Capítulo 3: Se informa sobre el procesamiento de los datos, de donde se obtuvieron, cuales son los periodos de tiempo que componen y la motivación para realizar los experimentos con estos. Indice S&P 500, y los otros sobre la predicción de dificultades financieras de una colección de bancos estadounidenses.

Capítulo 4: Se informa sobre los experimentos, que básicamente consisten en comparar el rendimiento al hacer predicciones por un CNN estándar versus un CNN dotado de una unidad de preprocesamiento RP de la entrada, y en dos escenarios diferentes: uno en la predicción de la dirección del precio del índice

S&P 500, y el otro sobre la predicción de dificultades financieras de una colección de bancos estadounidenses.

Capítulo 5: Se realiza un resumen sobre los resultados y conclusiones obtenidas mediante los experimientos realizados en el Capítulo anterior. Ademas, donde se busca explicar la mejora de CNN en la predicción de series de tiempo financieras al preprocesar los datos de entrada como imágenes antes de alimentarlos a nuestro modelo de CNN. La transformación de datos numéricos a imagen se realiza a través de Gráficos de Recurrencia (RP).

Apéndice: Al final del trabajo se incluye un apéndice con código de R utilizado para el análisis de los datos.

Capítulo II

Conceptos metodológicos

En este capítulo de carácter metodológico introduciremos los conceptos y métodos más relevantes empleados para el reconocimiento de imágenes y pronósticos financieros mediante Redes Neuronales Convolucionales.

Se dara una explicacion metodológica sobre las técnicas aplicadas para la transformación a imágenes de series de tiempo mediante graficos de recurrencia y su posterior análisis y evaluación con Redes Neuronales Convolucionales.
Este capítulo explicará, los Gráficos de Recurrencia, Redes Neuronales Convolucionales, Series de Tiempo y métricas empleadas para la evaluacion de resultados, tales como precisión, perdidas, AUC y validación cruzada. La validación cruzada en series de tiempo se justifica en [6].

Para cada tópico primero se explicará una pequeña introducción, aplicación de este y luego se describirá la formulación matemática que estos tienen.

2.1 Graficos de recurrencia (RP)

La recurrencia es una propiedad fundamental de los sistemas dinámicos, que puede utilizarse para caracterizar el comportamiento del sistema en el espacio fase [7]. Desde 1980 se creó una herramienta de visualización de esta recurrencia la cual se denomino graficos de recurrencia. Estos inicialmente se vieron ligados al mundo de la salud, en concreto uno de los primeros usos tenía que ver con el análisis de los intervalos del latido del corazón [7]. Los Recurrence Plots se han ocupado del estudio del estado de la fase espacio en muchos otros campos, los más conocidos son la economía, astrofísica,

geología etc. En la figura 1 podemos observar un grafico de recurrencia obtenido en este experimento.

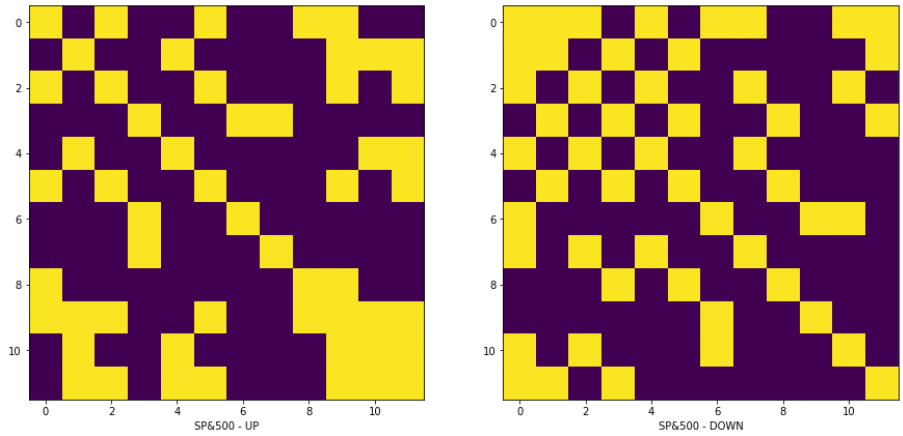


Figura 1: Ejemplo de grafico de recurrencia

Recientemente los graficos de recurrencia se han utilizado para la clasificación de imágenes con Series de Tiempo (TS). En base a esto, son dos los tipos de trabajo que se pueden realizar aplicando CNN a TS, la diferencia entre ellos es la siguiente:

1) Los inputs del primer tipo de aplicación modifican la arquitectura de los CNN y utiliza 1D señal de Serie de Tiempo (TS) como input o dato de entrada.
2) La otra aplicación consiste en transformar la señal 1D Serie de Tiempo con una matriz en una señal 2D y luego aplicar las CNN.

El segundo tipo de aplicación se puede ver reflejado en la investigación de [8]. En la cual utilizan las Series de Tiempo, a las cuales aplican una matriz basada en los graficos de recurrencia, para transformar las series de tiempo en una trayectoria de la fase espacio 2D a partir de una incrustación de retraso de tiempo (time delay emmbeding).

Los gráficos de recurrencia, introducidas por [9], proporcionan una visualización de la naturaleza periódica de una trayectoria a través de un espacio de fase. Los gráficos de recurrencia se han utilizado en la clasificación de series de tiempo [10]. Se formaliza como una matriz donde cada entrada está dada por la ecuación

$$R(x,j) = \Theta\left(\varepsilon - \left\|x_{(i)} - x_{(j)}\right\|\right), x_{(\cdot)} \in \mathbb{R}^m, i,j = 1 \ldots N$$

donde N es el número de estados, $x_{(i)}$ es la subsecuencia observada en el tiempo $i, \|\cdot\|$ es una norma, ε es un umbral para la cercanía y Θ es la función Heaviside $(\Theta(z) = 0, si\ z < 0, o\ 1\ en\ caso\ contrario)$. Por lo tanto, si la trayectoria m-dimensional de la serie temporal en el tiempo i está cercana (con respecto a alguna métrica) a la subsecuencia observada en el tiempo j, habrá un 1 (o un cuadrado amarillo, como en nuestra representación de imagen) en entrada (i, j) de la matriz de recurrencia; de lo contrario, el valor es 0 (un cuadrado morado oscuro).

En los experimentos llevados a cabo en este estudio se utiliza la distancia euclidiana por pares para la norma RP $\|\cdot\|$ Ademas, se observa que se puede usar RP más allá de las series de tiempo como una representación gráfica general de una métrica de similitud entre las características multidimensionales de algún otro tipo de datos numéricos además de las series de tiempo. Es asi como se considera para el problema de clasificación de bancarrota.

2.2 Redes Neuronales Convolucionales (CNN).

Las redes neuronales convolucionales (CNN) están formadas por nodos ocultos (o neuronas), distribuidos a través de varias capas, con pesos y sesgos aprendibles. Cada neurona recibe varias entradas y calcula una suma ponderada sobre ellas, y luego pasa el resultado a través de una función de activación que proporciona una salida.

La importancia de la convolucion radica en que cada neurona convolucional procesa datos solo para su campo receptivo. Si bien las redes neuronales clásicas (sin convolucion) se pueden usar para aprender características y clasificar datos, no es práctico aplicar esta arquitectura a las imágenes. Dado que sería necesaria una gran cantidad de neuronas, incluso en una arquitectura poco profunda, debido a los tamaños de entrada muy grandes asociados con las imágenes, donde cada píxel es una variable relevante. La operación de convolución brinda una solución a este problema, ya que reduce

el número de parámetros libres, permitiendo que la red sea más profunda con menos parámetros.

En la figura 2 se puede observar un modelo generalizado de las redes neuronales convolucionales las cuales son una red multicapa que consta de capas convolucionales y de reducción alternadas, y que finalmente tiene capas de conexión total.

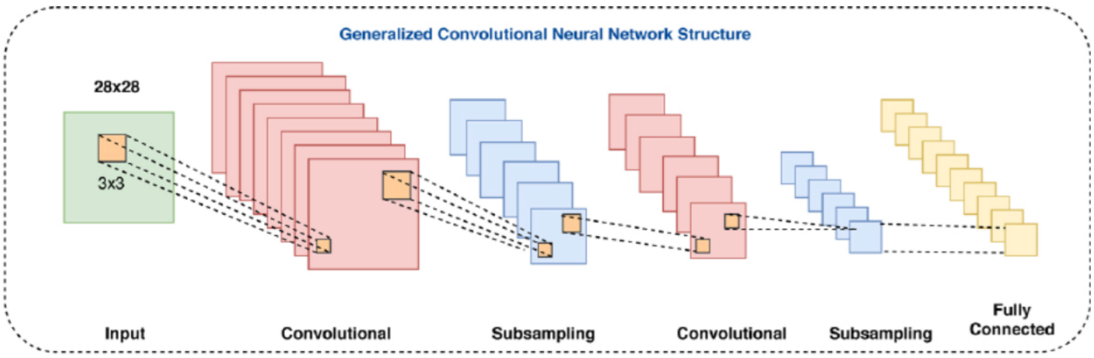


Figura 2: Estructura general de las redes neuronales Convolucionales [11]

A continuación, explicamos las funciones de activación mas popularmente utilizadas las cuales son:

- sigmoide logístico, se define formalmente como la ecuación

$$\left(\sigma\,(\mathrm{x}) = \frac{1}{(1\ + \exp(-\mathrm{x}))}\right)$$

- Tangente hiperbólica, se define formalmente como la ecuación

$$\left(\tanh\,(\mathrm{x}) = \frac{2}{(1\ + \exp(-2\mathrm{x}))} - 1\right)$$

- Unidades lineales rectificadas, se define formalmente como la ecuación

$$(\mathrm{ReLu}, \mathrm{R}(\mathrm{x}) = \max(0, x))$$

Estos parámetros de la red se ajustan minimizando alguna función de pérdida.

Una de las principales diferencias de las redes neuronales normales, es que los nodos en una CNN no están completamente conectados, sino que solo están

conectados a una región local en las entradas. Por lo cual, esta conectividad local se logra mediante el uso de convoluciones en lugar de sumas ponderadas.

En cada capa de la CNN, las entradas están enrevesadas con una matriz de peso denominado filtro o núcleo para crear un mapa de características; es decir, la matriz de peso (núcleo) que tiene una dimensión más pequeña que la entrada, se desliza sobre la entrada y calcula su producto de punto sobre cada región local.
La salida de este mapa de características se pasa a través de una función de activación la cual es no lineal (algunas de las tres funciones de activación mencionadas anteriormente), y la dimensión de este mapa de características transformado se reduce posteriormente mediante agrupación espacial (o submuestreo). Por ejemplo, considerando max, promedio o suma de valores. En la práctica, se ha demostrado que el conocido Max Pooling funciona mejor.
El objetivo de la agrupación es reducir la muestra de una representación de entrada, reducir su dimensión y permitir suposiciones sobre las características contenidas en las subregiones agrupadas.

El procedimiento se repite capa por capa (el número de capa determina la profundidad de la red); es decir, en cada capa subsiguiente $i = 2, \dots, D,$ el mapa de características de entrada obtenido de la capa anterior $l - 1,$ por convoluciones y transformación a través de la función de activación y agrupado, se convoluciona con un conjunto de núcleos (*kernels*) para crear un nuevo mapa de características correspondiente a la capa actual.

La dimensión del operador de convolución se adapta a la dimensión de los datos. Para la entrada unidimensional, como es el caso de las series de tiempo, $\mathrm{x} = \{\mathrm{x}\,(\mathrm{t}) \colon \mathrm{t} = 1, \dots, \mathrm{N}\}$, convoluciones unidimensionales (1D) con núcleos wh, para $h = 1, \dots, M. \mathrm{con}\ \mathrm{M} < \mathrm{N}$, se definene formalmente como

$$S(t,h) = (w_h * x)(t) = \sum_{n=-\infty}^{\infty} x(n) w_n(t-n)$$

Para la entrada bidimensional, como es el caso de las imágenes, $(l(i,j))_{l<i,j<N}$, las convoluciones apropiadas deben ser bidimensionales (2D) en las matrices del núcleo $(Kh(i,j))_{1<i,j<M}$:

$$S(i,j,h) = (K_h * I)(i,j) = \sum_m \sum_n I(i-m, j-n) K_h(m,n)$$

Se utilizo, la biblioteca de Python Deep Learning, Keras, para los cálculos de CNN. Adoptamos la nomenclatura de Keras, donde una CNN 1D se denomina Conv1D, y una CNN 2D es Conv2D.

2.3 Series de tiempo (TS).

Por serie de tiempo se refiere a datos estadísticos que se recopilan, observan o registran en intervalos de tiempo regulares (diario, semanal, semestral, anual, entre otros) y se utilizan para explicar la relación causal entre diversas variables que cambian en el tiempo e influyen entre ellas. Desde el punto de vista estadistico una serie temporal es una sucesión de variables aleatorias indexadas según parámetro creciente con el tiempo.

En una serie existen cinco tipos básicos de variación. Estos componentes son:

1. **Tendencia secular**: indica la marcha general y persistente del fenómeno observado, es una componente de la serie que refleja la evolución a largo plazo.
2. **Variación estacional**: Es el movimiento periódico de corto plazo. Se trata de una componente causal debida a la influencia de ciertos fenómenos que se repiten de manera periódica en un año (las estaciones), una semana (los fines de semana) o un día (las horas puntas) o cualquier otro periodo. Recoge las oscilaciones que se producen en esos períodos de repetición.
3. **Variación cíclica**: Es el componente de la serie que recoge las oscilaciones periódicas de amplitud superior a un año, movimientos normalmente irregulares alrededor de la tendencia, en las que a diferencia de las variaciones estacionales, tiene un período y amplitud variables, pudiendo clasificarse como cíclicos, cuasicíclicos o recurrentes.

4. **Variación aleatoria**: De carácter errático, también denominada **residuo**, no muestran ninguna regularidad (salvo las regularidades estadísticas), debidos a fenómenos de carácter ocasional como pueden ser tormentas, terremotos, inundaciones, huelgas, guerras, avances tecnológicos, etc.
5. **Variación transiente**: De carácter errático debidos a fenómenos aislados que son capaces de modificar el comportamiento de la serie (tendencia, estacionalidad variaciones cíclicas y aleatoria).

2.4 Métricas para evaluación del modelo

Evaluar los algoritmos utilizados es una parte esencial por lo cual utilizaremos los siguientes criterios para comparar nuestros modelos:

2.4.1 Precisión de clasificación

Es lo que generalmente se refiere cuando se utiliza el término precisión. Es la relación entre el número de predicciones correctas y el número total de muestras de entrada.

$$Precisión = \frac{Numero\ de\ precisiones\ correctas}{Numero\ total\ de\ predicciones\ realizadas}$$

En general la precisión de clasificación trabaja bien cuando existe una cantidad similar de muestras para cada categoría.

2.4.2 Pérdida logarítmica o pérdida

Esta funciona penalizando las clasificaciones falsas, un número que indica qué tan incorrecta fue la predicción del modelo en un solo ejemplo. Si la predicción del modelo es perfecta, la pérdida es cero; de lo contrario, la pérdida es mayor.

La función de perdida utilizada fue *Pérdida de error cuadrática media* (MSE) la cual se calcula como el promedio de las diferencias cuadráticas entre los valores predichos y reales.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \gamma_i)^2$$

El resultado siempre es positivo, independientemente del signo de los valores predichos y reales, y un valor perfecto es 0.0.

2.4.3 Validación cruzada (K-Fold)

Es un procedimiento de remuestreo utilizado para evaluar modelos de aprendizaje automático en una muestra de datos limitada y se utiliza para evitar un sesgo causado por la muestra particular elegida.

El procedimiento tiene un único parámetro llamado k que se refiere al número de grupos en los que se dividirá una muestra de datos determinada. Como tal, el procedimiento a menudo se llama validación cruzada k-fold. En nuestro caso particular utilizamos un K = 10 para ambos experimentos lo cual significa que fueron con una validación cruzada de 10 veces.

La validación cruzada se utiliza principalmente en el aprendizaje automático aplicado para estimar la habilidad de un modelo de aprendizaje automático en datos no vistos. Es decir, usar una muestra limitada para estimar cómo se espera que el modelo funcione en general cuando se usa para hacer predicciones sobre datos que no se usaron durante el entrenamiento del modelo.

Es un método popular porque es fácil de entender y porque generalmente da como resultado una estimación menos sesgada u optimista de la habilidad del modelo que otros métodos.

2.4.4 Área bajo la curva (AUC)

Es una de las métricas más utilizadas especificamente en problemas de clasificación binaria. El AUC de un clasificador es igual a la probabilidad de que el clasificador clasifique un ejemplo positivo elegido al azar más alto que un ejemplo negativo elegido al azar. Se deben considerar dos situaciones:

1. **Tasa positiva verdadera (sensibilidad):** Es la proporción de puntos de datos positivos que se consideran correctamente como positivos, con respecto a todos los puntos de datos positivos.

$$Tasa\ positiva\ verdadera = \frac{Verdaderos\ Positivos}{Falsos\ Negativos + Verdaderos\ Positivos}$$

2. **Tasa de falso positivo (especificidad):** Corresponde a la proporción de puntos de datos negativos que se consideran erróneamente como positivos, con respecto a todos los puntos de datos negativos.

$$Tasa\ falsos\ positivos = \frac{Falsos\ Positivos}{Falsos\ Positivos + Verdaderos\ Negativos}$$

La tasa de falsos positivos y la tasa de verdaderos positivos tienen valores en el rango [0, 1]. Cuanto mayor sea el valor, mejor es el rendimiento de nuestro modelo.

Capítulo III

Procesamiento de los datos

En el trabajo realizado se llevaron a cabo dos experimentos, uno el índice S&P500 y el otro la posibilidad de quiebra de bancos de EEUU. Se realizará una pequeña explicación de cada data set, la importancia que tiene en el ámbito económico – empresarial mediante esto justificar su selección para el análisis.

Posteriormente, se explicará tanto las variables dependientes como la variable independiente categórica mencionando como fue construida y por último las fechas seleccionadas para nuestra data.

3.1 Indice S&P500

El indicador financiero S&P 500 Index, creado por Standard & Poor's en 1957, es uno de los índices bursátiles más seguidos en el mundo, que consta de 500 de las compañías más grandes que cotizan en la Bolsa de Nueva York. El precio del índice se calcula utilizando un rendimiento real, que es a la vez los cambios en el precio de las acciones y los pagos de dividendos. Como se explicaba este índice ocupa la ponderación, es decir, Standard & Poor's calcula la capitalización de mercado de cada empresa, considerada en el índice utilizando solo la cantidad de acciones disponibles para ser compradas o vendidas por el público, conocido como capital flotante. Para que una empresa entre en este Índice, debe valorar y analizar 8 parámetros que explican el estado financiero de la compañía: capitalización bursátil, liquidez, domicilio, capital flotante, clasificación del sector, viabilidad financiera, periodo de tiempo durante el cual ha cotizado en bolsa y ser negociada en la bolsa de valores.

Para el experimento de predicción del mercado de valores, la variable dependiente o el objetivo para la predicción es el signo de la prima de equidad Standard & Poors 500 (GSPCep). Esta es la diferencia de los logaritmos de los rendimientos de los precios, incluidos los dividendos, y la tasa de interés libre de riesgo:

$$\mathrm{GSPCep(t)} = \log\left(\frac{\mathrm{P(t)} + \mathrm{D12(t)}}{P(t-1)}\right) - \log\left(r(t)+1\right)$$

Donde P(t) y D12(t) son, respectivamente, el precio y las sumas móviles de 12 meses de dividendos pagados en el índice S&P 500 en el momento t, y r(t) es la tasa de interés del Tesoro de los Estados Unidos a tres meses cuenta. El objetivo de predicción es esencialmente la dirección del precio con referencia a la tasa de interés libre de riesgo. Se consideran como variables exploratorias o características el historial pasado de la prima de equidad, hasta tres rezagos, y el historial pasado de su varianza estimado como el cuadrado de GSPCep, hasta dos rezagos.

Estos datos financieros se han recuperado de la página web de Amit Goyal, que se basa en datos recopilados por Robert Schiller para su análisis estadístico de S&P 500. Los datos se muestrean mensualmente y varían de 1990 a 2015.

3.2 Data de bancos

Existe una amplia literatura sobre los modelos de predicción de quiebra o banca rota de los bancos, la cual se remonta con Altman (1968) donde el objetivo era evaluar los determinantes de la falla financiera y diseñar reglas de predicción. En el análisis de [12] sobre los papers relacionados con la predicción numérica de la quiebra de bancos de los últimos 40 años, se recogen los cuatro tópicos más tratados, siendo estos los siguientes:

1) Técnicas de modelación para la prediccion de posibilidad quiebra mediante las técnicas de modelado como análisis discriminante, regresión logística, redes neuronales, máquinas de soporte vectorial.
2) Selección de variables utilizadas para la predicción de la quiebra, cuyo objetivo era entregar la información adecuada para alimentar los modelos y asi obtener una mayor precisión.
3) Analisis de los tipos de falla que un modelo puede pronosticar; como por ejemplo la mayoría de los modelos solo pueden hablar del estado de quiebra como binario (quiebra frente a no quiebra).
4) Desarrollo de modelos multiestatales (es decir, múltiples estados de salud financiera), como los modelos capaces de predecir una resolución final de bancarrota (liquidación, reorganización, adquisición por otra empresa, etc).

Por estas razones, fue una motivación especial realizar el experimento sobre quiebra de bancos. La información financiera ha sido recuperada de la Federal Deposit Insurance Corporation (FDIC), que mantiene una lista de los bancos asegurados de EE. UU. que se declararon en quiebra durante el período 1992-2017. El número total de bancos estadounidenses que se declararon en quiebra entre 1992 y 2017 llegó a 845.

Los datos consisten en matrices con 106 variables exploratorias pertenecientes a indicadores financieros extraídos de informes financieros trimestrales de bancos de EE. UU., más una variable de respuesta que es categórica y toma el valor 1 para indicar una entidad en quiebra, o 0 en caso contrario. Hay 5152 de estas matrices, cada una de las cuales representa la situación financiera de un banco. Cada banco tiene al menos 8 trimestres de informes de datos financieros.

Capítulo IV

Experimentos y resultados

El siguiente Capítulo consta de dos partes, primero se darán las expecificaciones técnicas para el modelo CNN propuesto. Se recuerda que la única diferencia en los experimentos es el procesamiento diferente de los datos de entrada. Son dos modelos los cuales se utilizaron en experimentos.
El primer modelo consiste en alimentar con datos numéricos unidimensionales directamente como entrada a las CNN. El segundo modelo consta de un preprocesamiento a los datos de entrada numéricos convirtiéndolos en imágenes utilizando la técnica de graficos de recurrencia. Cada imagen representa una serie temporal que se clasifica según su variable de respuesta. Estas imágenes proporcionan la entrada para nuestra CNN.

Se realizan los experimentos en dos escenarios financieros diferentes:
En un comiezo, se darán las especificaciones técnicas del modelo CNN, mencionando el tipo de convolucion, el optimizador, función de perdida y las métricas para evaluar el modelo.

Se continua con una explicación en detalle de cada experimento recordemos que los experimentos fueron realizados en dos escenarios financieros diferentes:

1) Predecir la dirección del precio del índice S&P 500.
2) Predecir la posibilidad de quiebra en un conjunto de bancos estadounidenses.

En cada uno de estos escenarios se describe la preparación de los datos con la cual se alimentarán nuestros modelos, mencionando las variables dependientes y la variable independiente categorica. Ademas, se menciona

cuantos periodos tiene nuestra serie de tiempo en cada caso. Luego se observa una selección aleatoria de graficos de recurrencia para ilustrar el pre-proceso de las imágenes (graficos de recurrencia categorizados según su variable de respuesta), y por ultimo se mencionan los resultados obtenidos a partir de las 100 simulaciones. Los resultados incluyen las cuatro métricas explicadas anteriormente en el Capítulo II.

4.1 Especificaciones de la CNN

Se construyeron dos tipos de modelos de Red Neuronal Convolucional utilizando la biblioteca Keras en Python. En ambos casos, se utilizó modelado secuencial porque es la forma más fácil de construir un modelo capa por capa en Keras. Dado que el objetivo es probar la posible mejora en la clasificación de una CNN con un RP, no será una preocupación ajustar de manera sofisticada el funcionamiento interno de la CNN si no, más bien trabajar con una estructura básica sobre la cual se alimenta la entrada procesada como imagen a través de el método de grafico de recurrencia, o no. En la siguiente lista se detalla la estructura de la red neuronal convolucional en nuestros modelos y la diferencia entre ellos. La principal diferencia es el preprocesamiento de las entradas de datos.

Conv1D: Consiste en una capa convolucional hecha de convoluciones unidimensionales con núcleos de tamaño 2 y 64 nodos. Las entradas son matrices numéricas. La función de activación que utilizamos es ReLU, que funciona bastante bien en la práctica y es menos costosa desde el punto de vista computacional que tanh y sigmoide porque implica operaciones matemáticas más simples. La capa de agrupación se producirá con la agrupación máxima.

Conv2D + RP: Para este modelo, las entradas (matrices de números) se procesan primero con el método de diagrama de recurrencia (RP) para producir las imágenes correspondientes (matrices de 0 y 1). Luego sigue una capa convolucional hecha de convoluciones bidimensionales con núcleos de tamaño

2 × 2 y 64 nodos, una función de activación ReLu y una capa de agrupación Max.

Compilando el modelo: La compilación del modelo requiere tres parámetros: optimizador, pérdida y métrica.

1. El optimizador controla la tasa de aprendizaje. Se usará "Adam" como optimizador porque generalmente es un buen optimizador para usar en muchos casos. El optimizador Adam ajusta la tasa de aprendizaje durante el entrenamiento. La tasa de aprendizaje determina qué tan rápido se calculan los pesos óptimos para el modelo. Una tasa de aprendizaje menor puede conducir a pesos más precisos (hasta cierto punto), pero el tiempo que lleva calcular los pesos será más largo.
2. Se utilizará el error cuadrático medio (MSE) para la función de pérdida. Esta es la opción más común para problemas de clasificación con dos etiquetas.
3. Métrica: se utilizará la métrica de precisión para ver el puntaje de precisión en el conjunto de validación cuando entrenamos el modelo.

Entrenar el modelo: Se utilizará la función "fit ()" con los siguientes parámetros: datos de entrenamiento (Entrenamiento X), datos de destino (Entrenamiento Y), datos de validación, número de épocas y número de lote.

El número de épocas es cuántas veces se pasa todo el conjunto de datos hacia adelante y hacia atrás a través de la red neuronal. Cuantas más épocas se corran, más mejorará el modelo, hasta cierto punto. Después de ese punto, el modelo dejará de mejorar durante cada época. El tamaño del lote es el número total de ejemplos de entrenamiento presentes en un solo lote. Por ejemplo, se puede dividir el conjunto de datos de 60 ejemplos en lotes de 12, luego se necesitarán 5 iteraciones para completar 1 época. En los experimentos se establece epoch en 10 y batch en 1.

Datos de validación: Se utilizará el conjunto de pruebas que se proporcionó en el conjunto de datos de los modelos, que se han dividido en validacion X e validacion Y.

4.2 Experimentos realizados

4.2.1 Experimiento 1: Predicción de la dirección del SP&500

Preparación de datos: Como se detalla en la sección 3.1, los datos consisten en conjuntos de 5 variables exploratorias (retrazados 1, 2 y 3 de la serie GSPCep y retrazados 1 y 2 del cuadrado de la serie, muestreados mensualmente), más 1 variable de respuesta que es categórica tomando el valor 1 para indicar un aumento del precio (mensual) de S & P500, o 0 en caso contrario.

Hay 300 de estas matrices, que representan la serie multivariante que va de 1990 a 2015. Aplicamos una ventana variable de tamaño 12 (12 meses de datos o matrices) para predecir la dirección del precio del próximo mes y adelantar pasos de 1 mes. Por lo tanto, cada RP utiliza 12 períodos de datos financieros.

En el primer modelo (Conv1D) usamos como "entrenamiento X" una matriz de entrada con los valores numéricos de las variables de los 12 períodos de tiempo y como "entrenamiento Y" usaremos una matriz de entrada categórica de 0 y 1 para cada serie.

Por otro lado, para el modelo Conv2D + RP convertimos esta matriz de entrada numérica de entrenamiento (X, Y) a imagen usando gráficos de recurrencia. Los graficos se pueden observar en la Figura 1. De esta forma se obtienen imágenes (una entrada de matriz) que luego se alimentan a la Red Neuronal Convolucional 2D.

Se observa en la Figura 3, que las imágenes creadas por la gráfica de recurrencia presentan diferencias al compararlas entre sus categorías (Abajo = 0 y Arriba = 1).

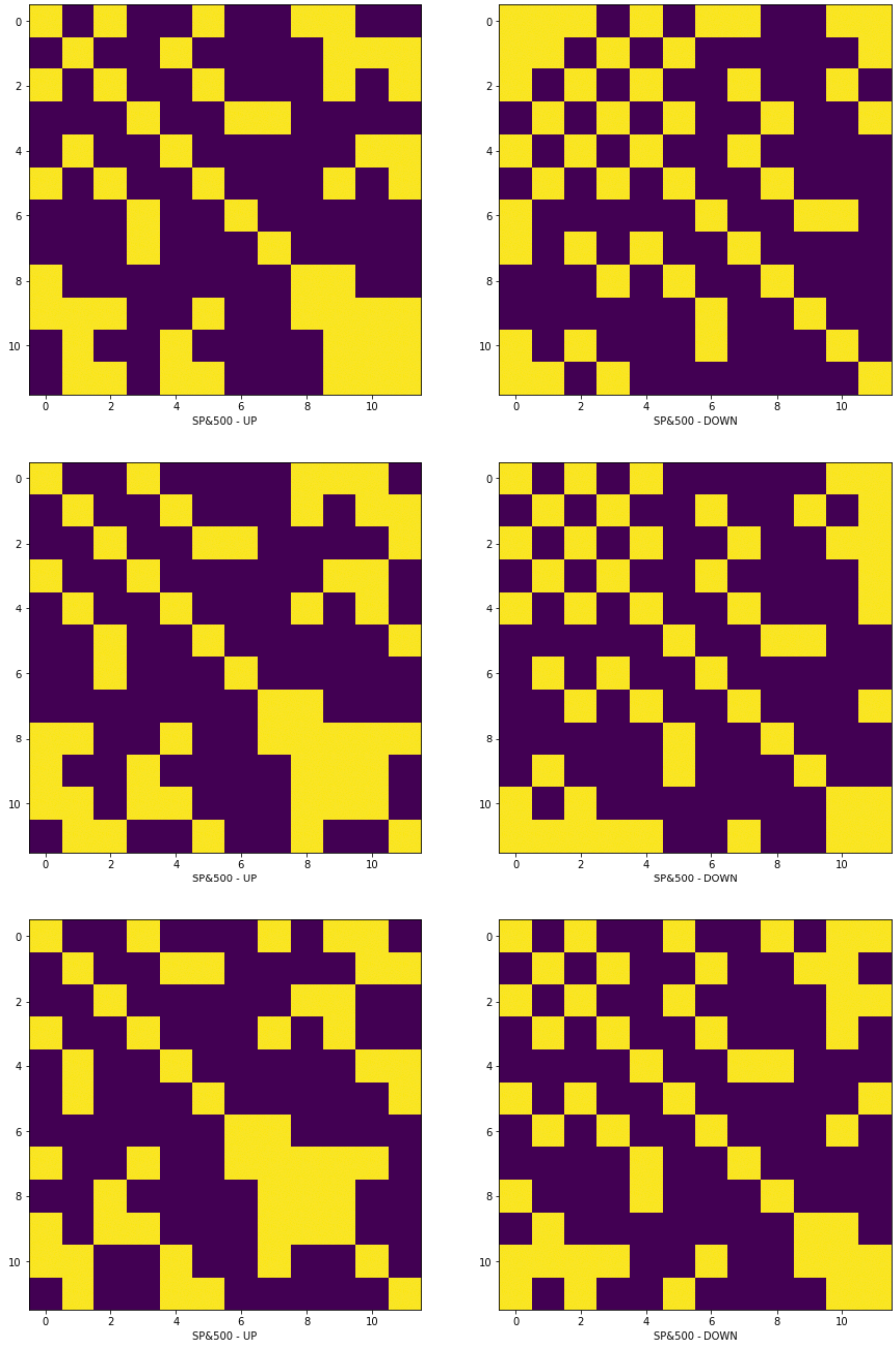


Figura 3: Graficos de Recurrencia distintos periodos S&P500

Resultados: Se repite cada experimento 100 veces y se obtiene los siguientes resultados promedio para cada modelo CNN reportado en la Tabla 1.

CNN	Acc Train	Acc Test	Loss Train	Loss Test	K-Fold Cross V(10)	AUC
Conv1D	62.25% (+/- 2.15%)	56.30% (+/- 1.87%)	2.52 (+/- 0.39)	2.35 (+/- 0.42)	59.83% (+/- 1.97%)	0.65 (+/- 0.057)
Conv2D + RP	69.16% (+/- 0.97%)	65.67% (+/- 0.86%)	3.80 (+/- 0.20)	3.83 (+/- 0.24)	66.74% (+/- 1.06%)	0.68 (+/- 0.025)

Tabla 1: Resultados experimento S&P500

Igualmente se grafican los resultados de estos 100 experimentos comparando la precisión de entrenamiento versus la de validación tanto para Conv1D y Conv2D.

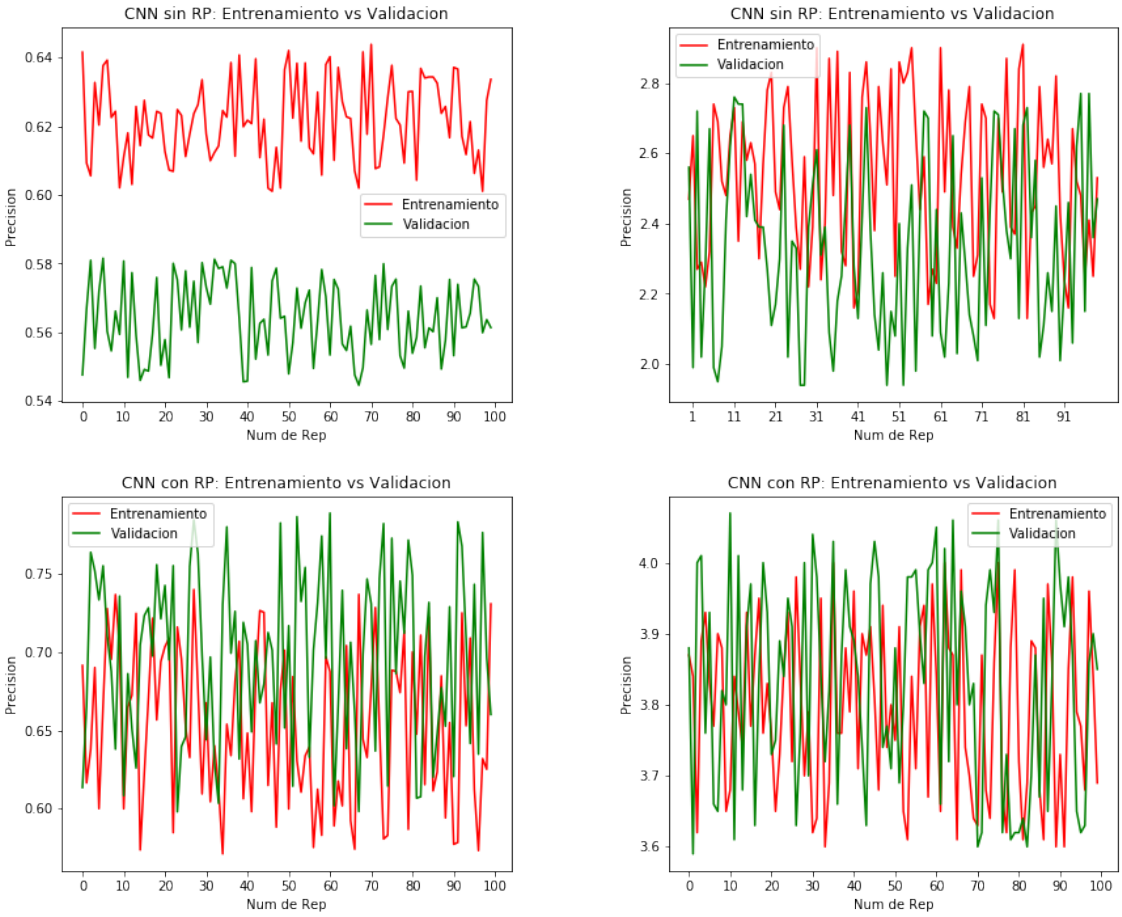


Figura 4: Resultados graficos S&P500

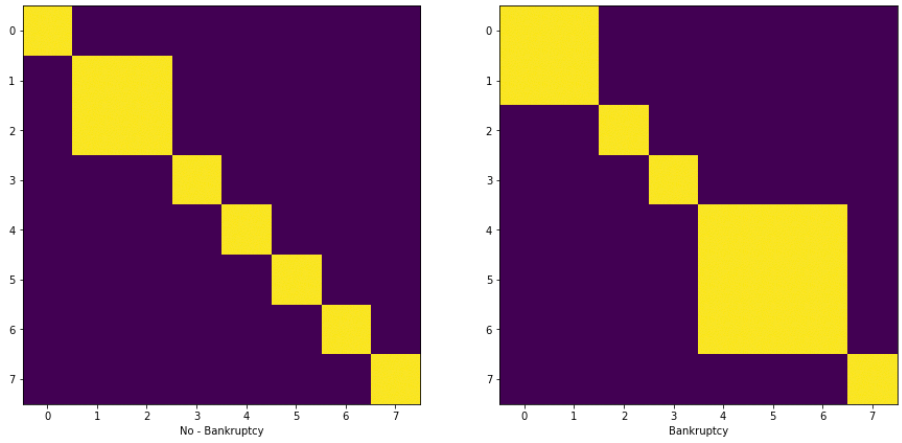
Se observa que al realizar un procesamiento previo de los datos mediante un gráfico de recurrencia el cual, transforma nuestros datos numéricos en imágenes, obtenemos mejores resultados que mejoran constantemente la precisión, la correlación AUC, al tiempo que reducen la pérdida, y el CV de 10 veces muestra que el modelo combinado Conv2D + RP generaliza bien. De igual manera, se observa que la variación de los resultados entre generaciones con los gráficos de recurrencia disminuye considerablemente (aproximadamente un 50% en cada métrica), dando un resultado con mayor confiabilidad.

4.2.2 Experimento 2: Predicción de quiebra de bancos

Preparación de datos: Como se detalla en la sección 3.2, los datos consisten en conjuntos de 106 variables exploratorias (exploratorias pertenecientes a indicadores financieros extraídos de informes financieros trimestrales de bancos de EE. UU., más 1 variable de respuesta que es categórica). Se seleccionan los datos de ocho períodos de tiempo (trimestres) para cada banco y se escalan los datos columna por columna por el método min-max. Se agrega la variable de respuesta que clasifica si el banco fallará o no en el próximo período de tiempo y la cual toma el valor 1 para indicar una entidad en quiebra, o 0 en caso contrario.

Para el primer modelo (Conv1D) se utilizan los datos de entrada como se indicó anteriormente para nuestra CNN. Pero para el segundo modelo (Conv2D + RP) se convierten los datos numéricos de series temporales en imágenes (gráfico de recurrencia) y se da la imagen como entrada a nuestra CNN.

En la Figura 2 se observa la gráfica de recurrencia de tres pares de bancos clasificados como no en quiebra y otro como en quiebra. Se puede ver una clara diferencia entre los gráficos que indica una clasificación adecuada ya dada por el método RP solo.



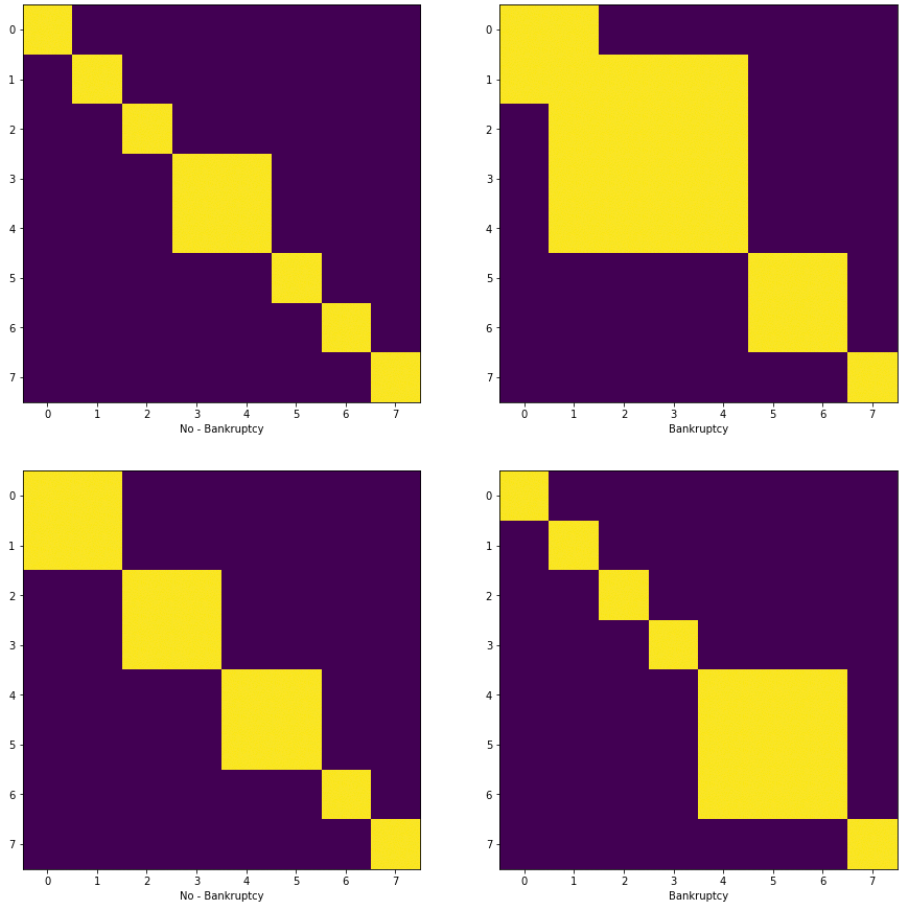


Figura 5: Graficos de recurrencia de distintos bancos de EEUU

Resultados: Usando CNN con una configuración similar en ambos casos, y reproduciendo cada experimento 100 veces, se obtuvieron los resultados (promedio) reportados en la Tabla 2.

CNN	Acc	Acc Test	Loss Train	Loss Test	K-Fold Cross V (10)	AUC
Conv1D	76.82% (+/- 7.22%)	76.14% (+/- 7.10%)	3.03 (+/- 1.28)	3.94 (+/- 1.64)	72.82% (+/- 6.95%)	78.91% (+/- 17.98%)
Conv2D + RP	91.87% (+/- 4.25%)	88.07% (+/- 4.96%)	1.97 (+/- 1.07)	1.98 (+/- 1.09)	89.07% (+/- 5.92%)	92.75% (+/- 0.42%)

Tabla 2: Resultados experimento quiebra de bancos

Igualmente se graficaron los resultados de estos 100 experimentos comparando la precisión de entrenamiento versus la de validación tanto para Conv1D y Conv2D.

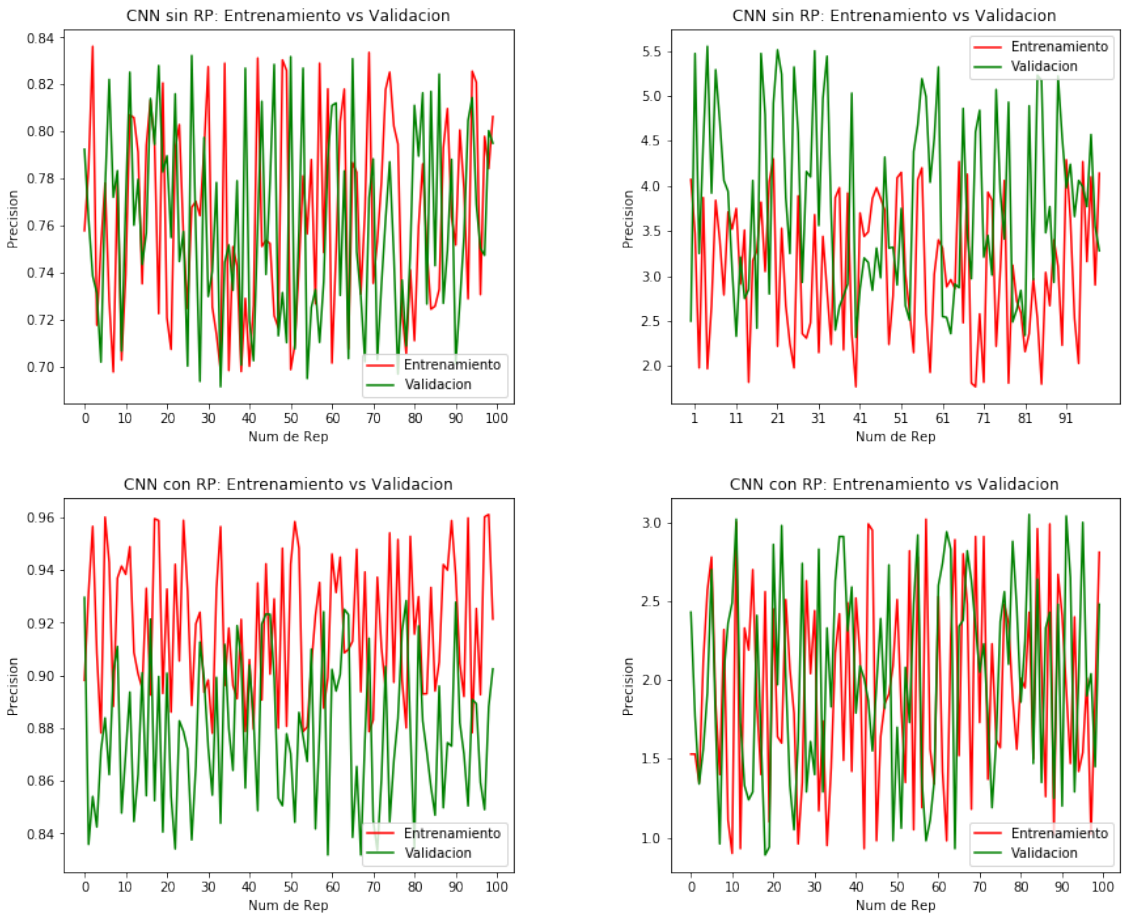


Figura 6: Resultados graficos quiebra de bancos

Se observa igual que el experimento anterior al realizar un procesamiento previo de los datos mediante un gráfico de recurrencia que transforma los datos numéricos en imágenes, mediante el cual se obtienen mejores resultados que mejoran constantemente la precisión, la correlación AUC, al tiempo que reducen la pérdida, y el CV de 10 veces muestra que el modelo combinado Conv2D + RP generaliza bien. A diferencia del ejercicio anterior, la disminucion de variación entre generaciones no fue tan significativa pero aun asi utilizar gráficos de recurrencia aumenta la confiabilidad de los resultados.

Capítulo V

Conclusiones

Mediante los experimentos realizados y los resultados obtenidos, se ha demostrado que, al realizar un procesamiento previo a nuestros datos mediante la transformación de estos datos numéricos a imágenes con la técnica graficos de recurrencia, se obtienen mejores resultados de clasificación. Estos resultados fueron medidos por cuatro métricas diferentes de rendimiento de clasificación y donde se observo lo siguiente:

Para ambos experimentos al realizar el procesamiento previo a los datos mediante la tranformacion a imágenes con los graficos de recurrencia la variabilidad entre las simulaciones disminuye de una manera considerable en ambos experimentos, tanto en la precisión, perdidas, validación cruzada y AUC, igualmente la presicion aumentò en ambos experimentos al utilizar la transformación mediante graficas de recurrencia. Por esto, podemos concluir que la tranformacion a imágenes nos entrega un resultado mas confiable que trabajar con datos numéricos sin procesamiento.

Mediante el experimento quiebra de bancos, inferimos empíricamente que al utilizar una mayor cantidad de variables y aplicar graficos de recurrencia se puede observar una mejora en cuanto a la disminución de la perdida del ejercicio y un aumento considerable en la precisión, igualmente la validación cruzada reafirma nuestros resultados (aumento de precisión y disminución de la variabilidad entre simulaciones). Deducimos que, al existir mayor cantidad de variables en la serie de tiempo, el procesamiento grafico se aprovecha de mejor manera.

Sobre el experimento del S&P500, no se obtuvieron resultados significativos en cuanto a la precisión (aumento, pero no considerablmente como en el otro experimento), lo cual atribuimos principalmente que el modelo CNN es bastante sencillo. Aun asi, el procesamiento previo mediante transformación a imágenes muestra una mejora en la precisión y la validación cruzada.

4.1 Posibles extensiones

En ambos experimentos se ha utilizado una norma estándar (norma euclidiana) para la definición de gráficos de recurrencia. Es posible considerar otras normas, que capturen de manera mas adecuada las similitudes entre series de tiempo financieras, como, por ejemplo, una métrica basada en correlación. Sería interesante ver la diferencia en el rendimiento de Modelo Conv2D + RP bajo diferentes normas subyacentes a la definición del método graficos de recurrencia.

De igual manera, seria interesante comparar nuestro modelo CNN de arquitectura simple versus un modelo CNN de arquitectura compleja (con más capas), ambos modelos ser alimentados mediante graficos de recurrencia y comparar los resultados obtenidos incluyendo el tiempo de procesamiento. De esta manera observar en base al tiempo de procesamiento si realmente es conveniente añadir capas a la CNN o trabajar con una sencilla y un buen procesamiento a imagen de los datos.

Bibliografía

[1] IBM, «IBM Marketing Trends 2017», *IBM*, 2017. [En línea]. Disponible en: https://public.dhe.ibm.com/common/ssi/ecm/wr/en/wrl12345usen/watson-customer-engagement-watson-marketing-wr-other-papers-and-reports-wrl12345usen-20170719.pdf. [Accedido: 29-sep-2019].

[2] L. Columbus, «IBM Predicts Demand For Data Scientists Will Soar 28% By 2020», *Forbes*, 2017. [En línea]. Disponible en: https://www.forbes.com/sites/louiscolumbus/2017/05/13/ibm-predicts-demand-for-data-scientists-will-soar-28-by-2020/. [Accedido: 29-sep-2019].

[3] Y. Lecun, L. Bottou, Y. Bengio, y P. Haffner, «Gradient-based learning applied to document recognition», *Proc. IEEE*, vol. 86, n.º 11, pp. 2278-2324, nov. 1998.

[4] I. Goodfellow, Y. Bengio, y A. Courville, *Deep Learning*. MIT Press, 2016.

[5] A. Arratia, *Computational Finance: An Introductory Course with R*. Atlantis Press, 2014.

[6] C. Bergmeir, R. Hyndman, y B. Koo, «A note on the validity of cross-validation for evaluating autoregressive time series prediction», *Comput. Stat. Data Anal.*, vol. 120, n.º C, pp. 70-83, 2018.

[7] N. Marwan y J. Kurths, «Recurrence Plots for the Analysis of Complex Systems in Earth Sciences», 2009.

[8] N. Hatami, Y. Gavet, y J. Debayle, «Classification of Time-Series Images Using Deep Convolutional Neural Networks», oct. 2017.

[9] J.-P. Eckmann, S. Kamphorst, y D. Ruelle, «Recurrence Plots of Dynamical Systems», *Europhys. Lett. Epl*, vol. 4, pp. 973-977, nov. 1987.

[10] D. F. Silva, V. M. A. de Souza, y G. E. A. P. A. Batista, «Time Series Classification Using Compression Distance of Recurrence Plots», *2013 IEEE 13th Int. Conf. Data Min.*, pp. 687-696, 2013.

[11] O. Sezer y M. Ozbayoglu, «Algorithmic Financial Trading with Deep Convolutional Neural Networks: Time Series to Image Conversion Approach», *Appl. Soft Comput.*, vol. 70, abr. 2018.

[12] P. du Jardin, «Bankruptcy prediction models: How to choose the most relevant variables?», *Bank. Mark. Invest.*, pp. 39–46, ene. 2009.

A Codigo python

A continuación, se presenta la programación de las funciones principales y secundarias elaboradas para la obtencion de los resultados.

CODIGO GENERAL UTILIZADO

```
#-------------------------------------------------------------
# INSTALACION DE LIBRERÍAS NECESARIAS
#-------------------------------------------------------------

import numpy as np
from numpy import array
from numpy import hstack
from numpy import mean
from numpy import std
from numpy import dstack

import os

from keras.models import Model
from keras.layers import Input
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Flatten
from keras.callbacks import ModelCheckpoint
from keras.layers.convolutional import Conv1D
from keras.layers.convolutional import MaxPooling1D
from keras.layers.merge import concatenate
from keras.layers import Convolution2D, MaxPooling2D, LeakyReLU

from sklearn import metrics
import sklearn as sk
import sklearn.metrics.pairwise
from sklearn.model_selection import train_test_split
from sklearn.model_selection import StratifiedKFold
from sklearn.metrics import matthews_corrcoef
```

```
from sklearn.metrics import roc_auc_score
from sklearn.metrics import confusion_matrix, classification_report

from skimage.transform import resize

from matplotlib.ticker import NullFormatter  # useful for `logit` scale
import matplotlib.pyplot as plt
from matplotlib.pyplot import imshow
%matplotlib inline
```

CODIGO ESPECIFICO EN S&P500

```
#------------------------------------------------------------
# Funciones creadas a utilizar
#------------------------------------------------------------
 # split a multivariate sequence into samples
def split_sequences(sequences, n_steps):
    X, y = list(), list()
    for i in range(len(sequences)):
        # find the end of this pattern
        end_ix = i + n_steps
        # check if we are beyond the dataset
        if end_ix > len(sequences):
            break
        # gather input and output parts of the pattern
        seq_x, seq_y = sequences[i:end_ix, :-1], sequences[end_ix-1, -1]
        X.append(seq_x)
        y.append(seq_y)
    return array(X), array(y)

#Recurrence plot
def recurrence_plot(s, eps=0.1):
    d = sk.metrics.pairwise.pairwise_distances(s)
    d = np.heaviside(eps-d,1)
    return d

def Model_CNN_without_RP():
    model = Sequential()
    model.add(Conv1D(filters=64,       kernel_size=2,       activation='relu',
input_shape=(n_steps,5)))
    #quizas donde puse 5 es n_features
    model.add(MaxPooling1D(pool_size=2))
    model.add(Flatten())
    model.add(Dense(32, activation='relu'))
    model.add(Dense(1, activation='relu'))
```

```
    return model

def Model_CNN_with_RP():
    model = Sequential()
    model.add(Convolution2D(32,        (3,        3),        activation='relu',
input_shape=(1,32,32), data_format='channels_first'))
    model.add(MaxPooling2D(pool_size=(2,2)))
    model.add(Flatten())
    model.add(Dense(32, activation='relu'))
    model.add(Dense(1, activation='relu'))
    return model


#------------------------------------------------------------
# Lectura de mis datos y procesamiento de estos
#------------------------------------------------------------
# define input sequence
in_seq1   = var1[Inicio_p:(n_periodos+Inicio_p)] # retrasada 1
in_seq2   = var2[Inicio_p:(n_periodos+Inicio_p)] # retrasada 2
in_seq3   = var3[Inicio_p:(n_periodos+Inicio_p)] # retrasada 3
in_seq4   = var4[Inicio_p:(n_periodos+Inicio_p)] # cuadrado 1
in_seq5   = var5[Inicio_p:(n_periodos+Inicio_p)] # cuadrado 2
out_seq_c = out2[Inicio_p:(n_periodos+Inicio_p)] # Index

# convert to [rows, columns] structure
in_seq1   = in_seq1.reshape((len(in_seq1), 1))
in_seq2   = in_seq2.reshape((len(in_seq2), 1))
in_seq3   = in_seq3.reshape((len(in_seq3), 1))
in_seq4   = in_seq4.reshape((len(in_seq4), 1))
in_seq5   = in_seq5.reshape((len(in_seq4), 1))
out_seq_c = out_seq_c.reshape((len(out_seq_c), 1))

# horizontally stack columns
dataset = hstack((in_seq1, in_seq2, in_seq3, in_seq4, in_seq5, out_seq_c))

# Convert into input/output
x_train, y_train = split_sequences(dataset, n_steps)

# The dataset knows the number of features, e.g. 2
# without RP
n_features    = x_train.shape[2]
# with RP
n_features_rp = x_train.shape[1]

x_train_rp = np.zeros((x_train.shape[0],32,32))
for c in range(0,x_train.shape[0]):
```

```
    dat = recurrence_plot(x_train[c,:,:])
    dat = resize(dat, (32,32),mode='constant')
    x_train_rp[c,:,:] = dat
    c = c + 1

#reshape to include depth
X_train              =    x_train.reshape(x_train.shape[0],    x_train.shape[1],
x_train.shape[2])
X_train_rp         =         x_train_rp.reshape(x_train_rp.shape[0],         1,
x_train_rp.shape[1],x_train_rp.shape[1])

#convert to float32 and normalize to [0,1]
X_train = X_train.astype('float32')
X_train /= np.amax(X_train)

X_train_rp = X_train_rp.astype('float32')
X_train_rp /= np.amax(X_train_rp)

Y_train = y_train
Y_train_rp = y_train

# split into input (X) and output (Y) variables para cross validation
# without RP
X2_without_rp = X_train
Y2_without_rp = Y_train
# with RP
X2_with_rp = X_train_rp
Y2_with_rp = Y_train_rp

# split in train and test set
X_train_1, x_test_1, Y_train_1, y_test_1 = train_test_split(X_train, Y_train,
test_size=0.1)
X_train_2,   x_test_2,   Y_train_2,   y_test_2   =   train_test_split(X_train_rp,
Y_train_rp, test_size=0.1)

#-------------------------------------------------------------
# Code: CNN without RP vs with RP (class)
# CNN (Clasificador) Clasificador 1 = baja 0 = sube
#-------------------------------------------------------------
acc_without_rp = []
val_acc_without_rp = []
loss_without_rp = []
val_loss_without_rp = []
acc_mean_without_rp = list()
acc_std_without_rp = list()
```

```
val_acc_mean_without_rp = list()
val_acc_std_without_rp = list()
loss_mean_without_rp = list()
loss_std_without_rp = list()
val_loss_mean_without_rp = list()
val_loss_std_without_rp = list()
auc_mean_without_rp = list()
mtc_mean_without_rp = list()

acc_with_rp = []
val_acc_with_rp = []
loss_with_rp = []
val_loss_with_rp = []
acc_mean_with_rp = list()
acc_std_with_rp = list()
val_acc_mean_with_rp = list()
val_acc_std_with_rp = list()
loss_mean_with_rp = list()
loss_std_with_rp = list()
val_loss_mean_with_rp = list()
val_loss_std_with_rp = list()
auc_mean_with_rp = list()
mtc_mean_with_rp = list()

for r in range(repe):
    # Compilo Modelo
    CNN_without_RP_Model = Model_CNN_without_RP()
    CNN_without_RP_Model.model.compile(loss='binary_crossentropy',
optimizer='adam', metrics=['acc', 'mse'])
    # Fit and test the model
    CNN_Model_without_rp  = CNN_without_RP_Model.fit(x=X_train_1, y=Y_train_1,
                                                     batch_size=batch_size_1,
epochs=epochs_1, verbose=0,

validation_data=(x_test_1, y_test_1), shuffle=True)
    # Prediction
    CNN_Model_without_rp_auc = CNN_without_RP_Model.predict_classes(x_test_1)

    # Evaluate the model
    acc_without_rp.append(CNN_Model_without_rp.history['acc'])
    val_acc_without_rp.append(CNN_Model_without_rp.history['val_acc'])
    loss_without_rp.append(CNN_Model_without_rp.history['loss'])
    val_loss_without_rp.append(CNN_Model_without_rp.history['val_loss'])
    acc_mean_without_rp.append(np.mean(acc_without_rp))
    acc_std_without_rp.append(np.std(acc_without_rp))
```

```
    val_acc_mean_without_rp.append(np.mean(val_acc_without_rp))
    val_acc_std_without_rp.append(np.std(val_acc_without_rp))
    loss_mean_without_rp.append(np.mean(loss_without_rp))
    loss_std_without_rp.append(np.std(loss_without_rp))
    val_loss_mean_without_rp.append(np.mean(val_loss_without_rp))
    val_loss_std_without_rp.append(np.std(val_loss_without_rp))
    auc_mean_without_rp.append(np.mean(roc_auc_score(y_test_1,
CNN_Model_without_rp_auc)))
    mtc_mean_without_rp.append(np.mean(roc_auc_score(y_test_1,
CNN_Model_without_rp_auc)))

for r in range(repe):
    # Compilo Modelo
    CNN_with_RP_Model = Model_CNN_with_RP()
    CNN_with_RP_Model.model.compile(loss='binary_crossentropy',
optimizer='adam', metrics=['acc', 'mse'])
    # Fit and test the model
    CNN_Model_with_rp  = CNN_with_RP_Model.fit(x=X_train_2, y=Y_train_2,
                                                batch_size=batch_size_1,
epochs=epochs_1, verbose=0,

validation_data=(x_test_2, y_test_2), shuffle=True)
    # Prediction
    CNN_Model_with_rp_auc = CNN_with_RP_Model.predict_classes(x_test_2)

    # Evaluate the model
    acc_with_rp.append(CNN_Model_with_rp.history['acc'])
    val_acc_with_rp.append(CNN_Model_with_rp.history['val_acc'])
    loss_with_rp.append(CNN_Model_with_rp.history['loss'])
    val_loss_with_rp.append(CNN_Model_with_rp.history['val_loss'])
    acc_mean_with_rp.append(np.mean(acc_with_rp))
    acc_std_with_rp.append(np.std(acc_with_rp))
    val_acc_mean_with_rp.append(np.mean(val_acc_with_rp))
    val_acc_std_with_rp.append(np.std(val_acc_with_rp))
    loss_mean_with_rp.append(np.mean(loss_with_rp))
    loss_std_with_rp.append(np.std(loss_with_rp))
    val_loss_mean_with_rp.append(np.mean(val_loss_with_rp))
    val_loss_std_with_rp.append(np.std(val_loss_with_rp))
    auc_mean_with_rp.append(np.mean(roc_auc_score(y_test_2,
CNN_Model_with_rp_auc)))
    mtc_mean_with_rp.append(np.mean(roc_auc_score(y_test_2,
CNN_Model_with_rp_auc)))

print("acc      without      RP:      %.2f%%      (+/-      %.2f%%)      "      %
(np.mean(acc_mean_without_rp)*100, np.std(acc_std_without_rp)*100))
```

```
print("acc with RP: %.2f%% (+/- %.2f%%) " % (np.mean(acc_mean_with_rp)*100,
np.std(acc_std_with_rp)*100))
print("val_acc without RP: %.2f%% (+/- %.2f%%) " %
(np.mean(val_acc_mean_without_rp)*100, np.std(val_acc_std_without_rp)*100))
print("val_acc with RP: %.2f%% (+/- %.2f%%) " %
(np.mean(val_acc_mean_with_rp)*100, np.std(val_acc_std_with_rp)*100))
print("loss without RP: %.2f%% (+/- %.2f%%) " %
(np.mean(loss_mean_without_rp)*100, np.std(loss_std_without_rp)*100))
print("loss with RP: %.2f%% (+/- %.2f%%) " % (np.mean(loss_mean_with_rp)*100,
np.std(loss_std_with_rp)*100))
print("val_loss without RP: %.2f%% (+/- %.2f%%) " %
(np.mean(val_loss_mean_without_rp)*100, np.std(val_loss_std_without_rp)*100))
print("val_loss with RP: %.2f%% (+/- %.2f%%) " %
(np.mean(val_loss_mean_with_rp)*100, np.std(val_loss_std_with_rp)*100))
print("AUC without RP: %.2f%% (+/- %.2f%%) " %
(np.mean(auc_mean_without_rp)*100, np.std(auc_mean_without_rp)*100))
print("AUC with RP: %.2f%% (+/- %.2f%%) " % (np.mean(auc_mean_with_rp)*100,
np.std(auc_mean_with_rp)*100))
print("Mathew Corr with RP: %.2f%% (+/- %.2f%%) " %
(np.mean(mtc_mean_without_rp)*100, np.std(mtc_mean_without_rp)*100))
print("Mathew Corr RP: %.2f%% (+/- %.2f%%) " % (np.mean(mtc_mean_with_rp)*100,
np.std(mtc_mean_with_rp)*100))
```

CODIGO ESPECIFICO EN QUIEBRA DE BANCOS

```
#------------------------------------------------------------
# Funciones creadas a utilizar
#------------------------------------------------------------

# split a multivariate sequence into samples
def split_sequences(sequences, n_steps):
    X, y = list(), list()
    for i in range(len(sequences)):
        # find the end of this pattern
        end_ix = i + n_steps
        # check if we are beyond the dataset
        if end_ix > len(sequences):
            break
        # gather input and output parts of the pattern
        seq_x, seq_y = sequences[i:end_ix, :-1], sequences[end_ix-1, -1]
        X.append(seq_x)
        y.append(seq_y)
    return array(X), array(y)

#Recurrence plot
```

```
def recurrence_plot(s, eps=0.1):
    d = sk.metrics.pairwise.pairwise_distances(s)
    d = np.heaviside(eps-d,1)
    return d

#CNN without RP
def Model_CNN_without_RP():
    model = Sequential()
    model.add(Conv1D(filters=64,       kernel_size=2,       activation='relu',
input_shape=(n_steps,n_features)))
    model.add(MaxPooling1D(pool_size=2))
    model.add(Flatten())
    model.add(Dense(32, activation='relu'))
    model.add(Dense(1, activation='relu'))
    return model

#CNN WITH RP
def Model_CNN_with_RP():
    model = Sequential()
    model.add(Convolution2D(32,        (3,        3),        activation='relu',
input_shape=(1,32,32), data_format='channels_first'))
    model.add(MaxPooling2D(pool_size=(2,2)))
    model.add(Flatten())
    model.add(Dense(32, activation='relu'))
    model.add(Dense(1, activation='relu'))
    return model

#-------------------------------------------------------------
# Lectura de mis datos y procesamiento de estos
#-------------------------------------------------------------
cols                                                                                =
["asset","asset2","asset5","AVASSETJ","bkprem","chbal","dep","depdom","depi","
eq","eqsur","eqtot","equptot","ernast","idoa","idoliab","liab","liabeq","lnatr
es","lnlsnet","nclnls","numemp","oaienc","RBCT1J","RBCT2","RWAJT","sc","uc","u
cln","voliab","astempm","depdastr","eeffr","eq5","eqv","ernast5","intexpy","in
tincy","lnatresr","lnlsdepr","lnlsgr5","lnlsntv","nclnlsr","nimy","noijy","non
iiay","nonixay","nperfv","rbc1aaj","rbc1rwaj","rbcrwaj","roa","roaptx","roe","
roeinjr","coredep","ddt","depidom","depins","depnidom","idtrni","irakeogh","nt
r","ntrsmmda","ntrsoth","ntrtime","trn","ts","eintexp","epremagg","esal","ibef
xtr","IDEOTH","idothnii","idpretx","intinc","nim","noij","nonii","nonix","idln
ls","lnci","lncon","lnconoth","lnlsgr","lnre","lnredom","lnrenres","lnreres","
scage","scrdebt","scus","eqcprev","netinc","ilndom","isc","LNRERSFM","eothnint
","chbalni","lnls","ntripc","edepdom","trnipcoc"]
# convert folders to class labels
# downstairs/upstairs = 0,walking/jogging  = 1, standing/sitting = 2
```

```
class_translate = {"Bueno" : 0, \
                   "1_Q" : 1,   }

#pre allocate arrays
#x_all = np.zeros((644,n_steps,103))
x_all = np.zeros((644,n_steps,103))
x_rp = np.zeros((644,103,n_steps))
y_all = np.zeros(644)
c_all = 0

for i in class_translate.keys():
    for j in range(1,323):
        dat_all  =  pd.read_excel("~/Documents/TFM  MESIO/Data/Bankruptcy/"+  i
+"/"+ i +"_" +str(j) + ".xlsx")[cols].values
        dat_all = resize(dat_all, (n_steps,103),mode='constant')
        dat_rp = dat_all.transpose()
        x_all[c_all,:,:] = dat_all
        x_rp[c_all,:,:] = dat_rp
        y_all[c_all] = class_translate[i]
        c_all = c_all + 1


#Cambio nombre las variables para aprovechar mi codigo anterior
x_train = x_all
x_train_2 = x_rp
y_train = y_all
# The dataset knows the number of features, e.g. 2
# without RP
n_features    = x_train.shape[2]
# with RP
n_features_rp = x_train_2.shape[1]

x_train_rp = np.zeros((x_train_2.shape[0],32,32))
for c in range(0,x_train_2.shape[0]):
    dat = recurrence_plot(x_train_2[c,:,:])
    dat = resize(dat, (32,32),mode='constant')
    x_train_rp[c,:,:] = dat
    c = c + 1

#reshape to include depth
X_train             =    x_train.reshape(x_train.shape[0],    x_train.shape[1],
x_train.shape[2])
X_train_rp         =         x_train_rp.reshape(x_train_rp.shape[0],         1,
x_train_rp.shape[1],x_train_rp.shape[1])
```

```
#convert to float32 and normalize to [0,1]
X_train = X_train.astype('float32')
X_train /= np.amax(X_train)

X_train_rp = X_train_rp.astype('float32')
X_train_rp /= np.amax(X_train_rp)

Y_train = y_train
Y_train_rp = y_train

# split into input (X) and output (Y) variables para cross validation
# without RP
X2_without_rp = X_train
Y2_without_rp = Y_train
# with RP
X2_with_rp = X_train_rp
Y2_with_rp = Y_train_rp

# split in train and test set
X_train_1, x_test_1, Y_train_1, y_test_1 = train_test_split(X_train, Y_train,
test_size=0.1)
X_train_2,  x_test_2,  Y_train_2,  y_test_2  =  train_test_split(X_train_rp,
Y_train_rp, test_size=0.1)

#-------------------------------------------------------------
# Code: CNN without RP vs with RP (class)
# CNN (Clasificador) Clasificador 1 = baja 0 = sube
#-------------------------------------------------------------
# acc_without_rp = []
val_acc_without_rp = []
loss_without_rp = []
val_loss_without_rp = []
acc_mean_without_rp = list()
acc_std_without_rp = list()
val_acc_mean_without_rp = list()
val_acc_std_without_rp = list()
loss_mean_without_rp = list()
loss_std_without_rp = list()
val_loss_mean_without_rp = list()
val_loss_std_without_rp = list()
auc_mean_without_rp = list()
mtc_mean_without_rp = list()

acc_with_rp = []
val_acc_with_rp = []
```

```
loss_with_rp = []
val_loss_with_rp = []
acc_mean_with_rp = list()
acc_std_with_rp = list()
val_acc_mean_with_rp = list()
val_acc_std_with_rp = list()
loss_mean_with_rp = list()
loss_std_with_rp = list()
val_loss_mean_with_rp = list()
val_loss_std_with_rp = list()
auc_mean_with_rp = list()
mtc_mean_with_rp = list()

for r in range(repe):
    # Compilo Modelo
    CNN_without_RP_Model = Model_CNN_without_RP()
    CNN_without_RP_Model.model.compile(loss='binary_crossentropy',
optimizer='adam', metrics=['acc', 'mse'])
    # Fit and test the model
    CNN_Model_without_rp  = CNN_without_RP_Model.fit(x=X_train_1, y=Y_train_1,
                                                     batch_size=batch_size_1,
epochs=epochs_1, verbose=0,

validation_data=(x_test_1, y_test_1), shuffle=True)
    # Prediction
    CNN_Model_without_rp_auc = CNN_without_RP_Model.predict_classes(x_test_1)

    # Evaluate the model
    acc_without_rp.append(CNN_Model_without_rp.history['acc'])
    val_acc_without_rp.append(CNN_Model_without_rp.history['val_acc'])
    loss_without_rp.append(CNN_Model_without_rp.history['loss'])
    val_loss_without_rp.append(CNN_Model_without_rp.history['val_loss'])
    acc_mean_without_rp.append(np.mean(acc_without_rp))
    acc_std_without_rp.append(np.std(acc_without_rp))
    val_acc_mean_without_rp.append(np.mean(val_acc_without_rp))
    val_acc_std_without_rp.append(np.std(val_acc_without_rp))
    loss_mean_without_rp.append(np.mean(loss_without_rp))
    loss_std_without_rp.append(np.std(loss_without_rp))
    val_loss_mean_without_rp.append(np.mean(val_loss_without_rp))
    val_loss_std_without_rp.append(np.std(val_loss_without_rp))
    auc_mean_without_rp.append(np.mean(roc_auc_score(y_test_1,
CNN_Model_without_rp_auc)))
    mtc_mean_without_rp.append(np.mean(roc_auc_score(y_test_1,
CNN_Model_without_rp_auc)))
```

```
for r in range(repe):
    # Compilo Modelo
    CNN_with_RP_Model = Model_CNN_with_RP()
    CNN_with_RP_Model.model.compile(loss='binary_crossentropy',
optimizer='adam', metrics=['acc', 'mse'])
    # Fit and test the model
    CNN_Model_with_rp  = CNN_with_RP_Model.fit(x=X_train_2, y=Y_train_2,
                                                batch_size=batch_size_1,
epochs=epochs_1, verbose=0,

validation_data=(x_test_2, y_test_2), shuffle=True)
    # Prediction
    CNN_Model_with_rp_auc = CNN_with_RP_Model.predict_classes(x_test_2)

    # Evaluate the model
    acc_with_rp.append(CNN_Model_with_rp.history['acc'])
    val_acc_with_rp.append(CNN_Model_with_rp.history['val_acc'])
    loss_with_rp.append(CNN_Model_with_rp.history['loss'])
    val_loss_with_rp.append(CNN_Model_with_rp.history['val_loss'])
    acc_mean_with_rp.append(np.mean(acc_with_rp))
    acc_std_with_rp.append(np.std(acc_with_rp))
    val_acc_mean_with_rp.append(np.mean(val_acc_with_rp))
    val_acc_std_with_rp.append(np.std(val_acc_with_rp))
    loss_mean_with_rp.append(np.mean(loss_with_rp))
    loss_std_with_rp.append(np.std(loss_with_rp))
    val_loss_mean_with_rp.append(np.mean(val_loss_with_rp))
    val_loss_std_with_rp.append(np.std(val_loss_with_rp))
    auc_mean_with_rp.append(np.mean(roc_auc_score(y_test_2,
CNN_Model_with_rp_auc)))
    mtc_mean_with_rp.append(np.mean(roc_auc_score(y_test_2,
CNN_Model_with_rp_auc)))

print("acc      without      RP:      %.2f%%      (+/-      %.2f%%)      "      %
(np.mean(acc_mean_without_rp)*100, np.std(acc_std_without_rp)*100))
print("acc with RP: %.2f%% (+/- %.2f%%) " % (np.mean(acc_mean_with_rp)*100,
np.std(acc_std_with_rp)*100))
print("val_acc     without     RP:     %.2f%%     (+/-     %.2f%%)     "     %
(np.mean(val_acc_mean_without_rp)*100, np.std(val_acc_std_without_rp)*100))
print("val_acc     with     RP:     %.2f%%     (+/-     %.2f%%)     "     %
(np.mean(val_acc_mean_with_rp)*100, np.std(val_acc_std_with_rp)*100))
print("loss      without      RP:      %.2f%%      (+/-      %.2f%%)      "      %
(np.mean(loss_mean_without_rp)*100, np.std(loss_std_without_rp)*100))
print("loss with RP: %.2f%% (+/- %.2f%%) " % (np.mean(loss_mean_with_rp)*100,
np.std(loss_std_with_rp)*100))
```

```
print("val_loss without RP: %.2f%% (+/- %.2f%%) " %
(np.mean(val_loss_mean_without_rp)*100, np.std(val_loss_std_without_rp)*100))
print("val_loss with RP: %.2f%% (+/- %.2f%%) " %
(np.mean(val_loss_mean_with_rp)*100, np.std(val_loss_std_with_rp)*100))
print("AUC without RP: %.2f%% (+/- %.2f%%) " %
(np.mean(auc_mean_without_rp)*100, np.std(auc_mean_without_rp)*100))
print("AUC with RP: %.2f%% (+/- %.2f%%) " % (np.mean(auc_mean_with_rp)*100,
np.std(auc_mean_with_rp)*100))
print("Mathew Corr with RP: %.2f%% (+/- %.2f%%) " %
(np.mean(mtc_mean_without_rp)*100, np.std(mtc_mean_without_rp)*100))
print("Mathew Corr RP: %.2f%% (+/- %.2f%%) " % (np.mean(mtc_mean_with_rp)*100,
np.std(mtc_mean_with_rp)*100))


         d <-  as.matrix(conf.table$overall)[-2,]
         d <- as.matrix(d)

         g       <-       round((as.matrix(cohen.kappa(cbind(data,       data2),
alpha=0.05)$confid)[1,]),6)
         names(g) <- kap
         g <- t(t(g))
         d <- rbind(d,g)
         matrix2 <- cbind(c,d)
         c=matrix2
    }
  }
return(matrix2)
}
```