Luis Buceta Ojeda: 100303573 | Lucia Alvargonzalez Azqueta: 100315935 | Lauro Bravar Abril-Martorell: 100303876 | Adrián Rodríguez Grillo: 100316457

IBEXPredictor

PRACTICA 3

Inteligencia Artificial en las Organizaciones

# Introducción y objetivos

# Estado del arte

# Obtención y procesamiento de los datos

## Obtención de los datos

Para la obtención de datos de las diferentes bolsas y sus históricos utilizamos la API en python de Quandl, que es una empresa que se dedica a la recopilación de diferentes datos económicos. Esta API nos permitió acceder a los datos históricos de las plazas mundiales en formato *csv*, pudiendo modificar la forma de obtenerlos. Los parámetros que podíamos usar eran los siguientes:

* **Valores o conjuntos de valores:** Este parámetro nos permitía seleccionar que mercado o conjunto de mercados queríamos datos.
* **Datos deseados:** Este parámetro por defecto devolvía todos los posibles datos disponibles, como el precio de apertura, cierre, máximo, mínimo y volumen. Si se deseaba se podía acotar a los deseados.
* **Fecha de inicio y final:** El rango de fechas que se devolverían en el csv
* **Tipo de datos:** Este parámetro, que fue de los más útiles, nos permitió acceder a diferentes representaciones de datos como fueron los valores numéricos normales, los valores de cambio con respecto al día anterior, los porcentajes de subida y bajada con respecto al día anterior y los datos normalizados.
* **Frecuencia de recogida:** Este parámetro permitía ver cada cuanto se recogían los datos, siendo las opciones diario, mensual, semanal. Visto que queríamos una predicción diaria usamos la opción diaria.

Para la selección de plazas mundiales a elegir, observamos las principales mostradas en diferentes páginas de mercado como Yahoo Finance[[1]](#endnote-1), Expansión[[2]](#endnote-2) o Bolsamanía[[3]](#endnote-3), eligiendo finalmente seis bolsas:

* El IBEX35, para tener los datos del día.
* El Dow Jones, que es la bolsa estadounidense y, por tanto, la que más influencia en el resto.
* El Nikkei 225, que es la bolsa de Japón y la principal de la parte oriental del mundo.
* El Euro Stoxx50, que contiene los 100 principales valores de las bolsas europeas y, por tanto, es un generalizador de Europa.
* El Dax, que es la bolsa alemana y unas de las principales europeas.
* El Cac40, que es la bolsa francesa y otra de las principales europeas.

Una de las bolsas que es de las más importantes en el mundo es la de Reino Unido, pero debido a que utilizamos la versión gratuita de la API no estaba disponible.

## Decisiones

Para predecir la tendencia del día siguiente, consideramos que los datos más útiles serían los de cierre, debido a que los demás, como el máximo o mínimo del día, no serían muy influyentes sin conocer el resultado final y otros, como el precio de apertura, directamente no resultaban útiles.

Por tanto, para el proceso de obtención de datos lo que hicimos fue recoger, para las variables de entrada, los valores de cierre de las seis bolsas desde el 2000 hasta el día anterior a la recogida y, para la salida, los datos de cierre del IBEX35 para las mismas fechas. Una vez obtenidos estos datos, el siguiente paso era procesarlos para obtener los ficheros deseados para ejecutar las pruebas.

Este problema, que es la predicción de la tendencia del día siguiente de un mercado dados los datos históricos del mismo mercado y otros más, lo planteamos como un problema de clasificación donde la salida serían únicamente dos clases, sube y baja, que representaríamos como 1 y 0 respectivamente y las variables de entrada serían los datos de las diferentes plazas.

Para las pruebas planteamos dos formas diferentes de solucionar el problema, mediante la predicción a un día, donde con los datos de cierre de las bolsas de un día se predeciría el día siguiente, y con la predicción a cinco días, donde se utilizarían los datos de cinco días de las bolsas. El motivo de esta elección fue debido a dos razones, por una parte, diversos artículos de especialistas en la materia indicaban que la relación temporal entre los datos no era duradera en el tiempo y, por otra, queríamos evitar los patrones estacionales y temporales que se pueden producir en ciertas épocas del año.

## Procesamiento

El primer paso para la creación de los ficheros de datos fue obtener la salida, para ello, como hemos indicado anteriormente, tomamos los datos de cierre del IBEX35, en concreto los de porcentaje de cambio y tomamos los valores superiores al 0% como días de subida y el resto como día de bajada. Posteriormente, subimos la columna de resultados un día para convertirlo en la salida del día anterior. Es decir, si un día sube un 3% este pasaría a ser día de subida y posteriormente pasaría a ser la salida del día anterior.

El segundo paso fue juntar todas las columnas con los diferentes datos de entrada y la salida mediante la combinación de las tablas tomando como índice las fechas. Uno de los problemas que encontramos fue que no todas las bolsas abren los mismos días debido a los días festivos, por lo que, para solucionar esto, tuvimos que tomar dos medidas.

Primero, eliminar los días en los que el IBEX no operaba, ya que no tiene sentido predecir un día en el que la plaza no abrirá. En estos casos, la salida lo que predecirá será el siguiente día que se abra. Y segundo, en el caso del resto de bolsas, si en un determinado día una plaza no abre lo que se hace es arrastrar el valor del último día abierto hasta encontrarse otro día en el que se encuentren datos.



Tras estos dos pasos, ya tendríamos los datos preparados en un nuevo fichero *csv*, donde cada línea del mismo será un día con su conjunto de variables de entrada, que serán los datos de cierre de los mercados (6 si es a un día, 30 si es a cinco) y la salida, que será la tendencia del IBEX35 el día siguiente.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Fecha | Cierre – IBEX35 | Cierre – Dow Jones | Cierre – Nikkei225 | Cierre – Stoxx50 | Cierre – Dax | Cierre – Cac40 | Sube o baja |

Los ficheros utilizados son:

* ProcessedNormal.csv
* ProcessedChange.csv
* ProcessedRChange.csv
* ProcessedNormalized.csv
* 5DayNormal.csv
* 5DayChange.csv
* 5DayRChange.csv
* 5DayNormalized.csv

# Resolución del problema

## Weka

## Pyhthon

El objetivo principal en esta práctica era usar Python para desarrollar el predictor, usando principalmente las redes de neuronas. Sin embargo, tras observar que la implementación de otros tipos de clasificadores parecía sencilla, acabamos implementando diferentes clasificadores basados en máquinas de soporte vectorial, randomforest y k-nearest neighbours.

Para la implementación utilizamos diferentes librerías y frameworks, todos ellos venían incluidos en Anaconda Python[[4]](#endnote-4), que es una versión de Python que incluye las librerías más importantes para la ciencia de datos. En nuestro caso, utilizamos las siguientes librerías:

* **Quandl:** Obtención de datos.
* **Numpy y Pandas:** Procesamiento de datos y lectura de *csv*.
* **Tensorflow y keras:** Implementación de las redes de neuronas.
* **Scikit-learn:** Implementación de las máquinas de soporte vectorial, randomforest, kNN.

### Procesamiento de datos

Aparte de lo explicado en el apartado de obtención de datos, para el funcionamiento de las técnicas de aprendizaje automático hay que adaptar los datos a estructuras válidas de Python. Además, para comprobar la efectividad de los clasificadores hay que separar los datos en dos conjuntos, el de entrenamiento con el que se enseña a los modelos y el de test con el que se comprueba su efectividad.

Por tanto, el primer paso necesario para la lectura de los datos era convertirlos a una estructura de Python compatible con las librerías utilizadas, esto se consigue leyendo el *csv* con Pandas y convirtiéndolo en una dataTable. Esta tabla de datos, posteriormente se dividirá en dos conjuntos, uno que contendrá las variables de entrada y, otro, con las salidas.

El siguiente paso fue la división de estos datos en los conjuntos de entrenamiento y de test, este proceso se realiza en un método que permite seleccionar el porcentaje que queremos para el conjunto de entrenamiento.

### Redes de neuronas

Como hemos indicado anteriormente, para la implementación de las redes de neuronas utilizamos TensorFlow[[5]](#endnote-5), que es una librería open-source de google, y keras[[6]](#endnote-6), que es una abstracción de más alto nivel para usar la anterior librería.

El primer requisito para hacer funcionar una red de neuronas es establecer su diseño, por ello creamos un método donde poder establecer el diseño deseado de forma separada al resto del código, para facilitar su configuración. Por otro lado, para su entrenamiento y posterior prueba, utilizamos dos métodos de entrenamiento diferentes, uno donde los conjuntos de entrenamiento y test se introducen de forma manual y, otro, donde utilizamos validación cruzada, con número de conjuntos modificable por parámetro. Ambos métodos permiten modificar la configuración de los siguientes parámetros de la red de neuronas:

* **Epochs:** Que es el número de ciclos en los que se recorrerá el conjunto de entrenamiento entero, es decir, son las iteraciones del algoritmo. Por defecto son 500 iteraciones.
* **Batch:** Que indica el número de ejemplos del conjunto se propagarán por la red de forma simultáneamente. Es decir, si el batch es de 20, el entrenamiento se hará tomando de 20 en 20 los ejemplos hasta recorrer el conjunto. Esto permite un entrenamiento más rápido y menos memoria.

### Configuraciones utilizadas

Para las redes de neuronas utilizamos diferentes modelos con los que probamos, con diferentes números de neuronas, que explicaremos más adelante. Sin embargo, tras ciertas pruebas establecimos varios parámetros que dejamos fijos debido a su mejor desempeño.

Uno de los parámetros fue el optimizador de la red de neuronas, utilizando el optimizador Adam[[7]](#endnote-7), debido a que su ratio de aprendizaje adaptativo, su buen desempeño con gran cantidad de datos y su baja necesidad de memoria realizaban una convergencia más rápida. Otro parámetro fijo fue el uso de la entropía cruzada para comprobar el proceso de entrenamiento de la red debido a su mejor desempeño en problemas de clasificación con respecto al error medio. Con respecto al número de iteraciones establecimos que 500 fue un número más que suficiente para entrenar las redes, debido a que el aumento a partir de ese momento no producía grandes medidas para el costo en tiempo necesario. Con respecto al tamaño del ‘batch’ lo establecimos en número entre 20 y 50 ejemplos.

Finalmente, la activación de las neuronas también se mantuvo constante para toda la experimentación tras comprobar que sus resultados mejoraban con la siguiente configuración: En la capa de entrada y ocultas la activación es ‘ReLu’ y para la capa de salida es ‘sigmoidea’. La razón de ser de esto es que la activación ‘ReLu’ no tiende a hacer cero el gradiente, evitando bloqueos en el aprendizaje, además de ser más rápida en aprendizaje. Sin embargo, en la capa de salida, que solo hay una neurona, no podíamos usar este método debido a que hay veces que algunas neuronas se bloquean momentáneamente y perderíamos información.

Con respecto al número de neuronas, estableceremos los diferentes modelos utilizados en el apartado siguiente junto con sus resultados.

### Modelos y resultados

En este subapartado explicaremos los resultados obtenidos con las diferentes configuraciones utilizadas, tanto para la predicción a un día como a cinco días. Con respecto a los ficheros de datos utilizados, en las primeras pruebas utilizamos todos los ficheros (normal, cambio, porcentaje de cambio y normalizado). Sin embargo, tras estas pruebas vimos que el potencial se encontraba principalmente en el conjunto de datos de porcentaje de cambio, ya que este no tenía en cuenta el tamaño de los mercados, evitando que una pequeña caída en mercados grandes como Dow Jones afectase más de lo necesario, y, debido a que eran porcentajes, que se establecen en un rango, es más fácil generalizar.

En todos los modelos, lo que se mantiene contaste es el número de neuronas de la capa inicial, que tendrá tantas como variables de entrada tenga el fichero, es decir, 6 en el caso de la predicción a un día y 30 en el caso de cinco.

#### Predicción a un día

Con respecto a la predicción a un día, los mejores resultados los obtuvimos con una sola capa intermedia. Esto es así debido a las pocas neuronas de entrada hacía que más de una capa oculta no tuviesen mucho sentido. Los resultados con 6 neuronas en la capa de entrada y 3 en la capa oculta.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 1DayNormal | 1DayChange | 1DayRChange | 1DayNormalized |
| Cross Validation | 51,54% | 56,75% | 55,47% | 52,54% |
| Manual | 51,63% | 56,28% | 56,98% | 51,63% |

Como vemos en estos resultados, el potencial se observa en los ficheros de cambio, que son más fáciles de generalizar ya que pueden darse en varios ejemplos de forma similar.

Como hemos indicado, meter más capas ocultas no tiene sentido, los resultados con una capa oculta de 3 neuronas y otra de 2 son:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 1DayNormal | 1DayChange | 1DayRChange | 1DayNormalized |
| Cross Validation | 49.51% | 51.8% | 51.98% | 50.24% |
| Manual | 50.25% | 52.14% | 52.20% | 51,63% |

Por lo que descartamos su uso para este problema.

#### Predicción a cinco días

Para la predicción a cinco días sí que utilizamos varias configuraciones multicapa, junto con configuraciones de una sola capa oculta, aunque no acabaron aportando mejores resultados.

Los resultados obtenidos con una capa oculta de 15 neuronas fueron los siguientes:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 5DayNormal | 5DayChange | 5DayRChange | 5DayNormalized |
| Cross Validation | 51,54% | 52,79% | 56,15% | 52,54% |
| Manual | 48,37% | 52,49% | 56,51% | 51,63% |

Donde volvemos a ver que el archivo de porcentaje de cambio vuelve a obtener mejores predicciones. Con respecto al uso de dos capas ocultas, obtuvimos los mejores resultados utilizando 20 neuronas en la primera capa y 10 en la segunda. Aunque también utilizamos otra configuración con 15 neuronas en la primera y 7 en la segunda.

|  |  |
| --- | --- |
| Neuronas en capa | 5DayRChange |
| 30 – 20 – 10 – 1 | 57,44% |
| 30 – 15 – 7 – 1 | 56,05% |
| 30 – 20 – 1 | 54,42% |

### Conclusiones

Como observamos en los resultados, las redes de neuronas consiguieron un 57% de acierto en el mejor de los casos, tratándose este ejemplo de la previsión a cinco días. Aunque hicimos más pruebas con más configuraciones, todos rondaban entre el 54% y 58%, si se trataba del fichero con los porcentajes de cambio.

Con respecto al resto de ficheros, sus estimaciones no superaban el 53% y algunas se situaban por debajo del 50%, por lo que los descartamos para su uso. La explicación de esto ya fue expuesta anteriormente y aquí confirmamos la teoría.

# Conclusiones

# Trabajos futuros

# Referencias

1. Yahoo Finanzas: <https://es.finance.yahoo.com/> [↑](#endnote-ref-1)
2. Expansión: <http://www.expansion.com/> [↑](#endnote-ref-2)
3. Bolsamanía: <http://www.bolsamania.com/> [↑](#endnote-ref-3)
4. Anaconda Python: <https://www.continuum.io/downloads> [↑](#endnote-ref-4)
5. TensorFlow: <https://www.tensorflow.org/> [↑](#endnote-ref-5)
6. Keras: <https://keras.io/> [↑](#endnote-ref-6)
7. Adam optimizer: <https://arxiv.org/abs/1412.6980> [↑](#endnote-ref-7)