Trabajo Final Base de Datos Masivas

Creación de dataset y aplicación de random forest

Rodriguez Agustin 96898
agustinrodriguez206@gmail.com

Introducción

En este trabajo se realiza un proceso de *Knowledge discovery in databases* (**KDD**), basado en mercado de valores (**MV**) internacionales. Para explorar la posibilidad de utilizar el algoritmo RandomForest **[1]** en un conjunto de datos que cuentan con un orden en el tiempo y poder predecir su comportamiento en el futuro.
Se realizó un análisis entre distintos sitios web especializados en MV y se seleccionó entre ellos los que cuentan con mayor cantidad de datos y facilidad para obtener los históricos de diferentes acciones en un periodo de tiempo determinado.
Con el conjunto de datos obtenidos se formó el dataset ordenado por sus respectivas marcas de tiempo(**ts**), y se agrupó las acciones por sus *ts,* generando en cada momento un tupla con los datos de todas las acciones.
Se continua con el procesamiento de los datos, para lograr una mejora en la calidad de los mismos. Para esto se utilizan distintas técnicas según las características propias de los datos como saltos abruptos o datos faltantes. También se realiza una etapa que cuenta con varias transformaciones para dar temporalidad, eliminar tendencias y por último agregar atributos relacionados al análisis técnico de MV y generación de variables objetivos para poder entrenar los modelos y su posterior testeo.
Luego de varias iteraciones para lograr un conjunto de datos confiables, se realizaron nuevas transformación de los mismos, como la normalización de variables para unificar el dominio de los atributos entre otras que serán explicadas en el documento.
En la etapa final del trabajo se dividen los datos en conjuntos de distintos tamaños separando en grupos de entrenamiento y testing que se utilizan con distintas políticas de re entrenamiento para la clasificación con RandomForest y luego se evaluaran los modelos con los grupos de datos separados. Para lograr la mejor parametrización de los modelos se realizaron varios experimentos con distintos parámetros y se utilizaron los que que tuvieron mejor performance y requieren menor tiempo para realizar el entrenamiento y predicción

Esquema de trabajo

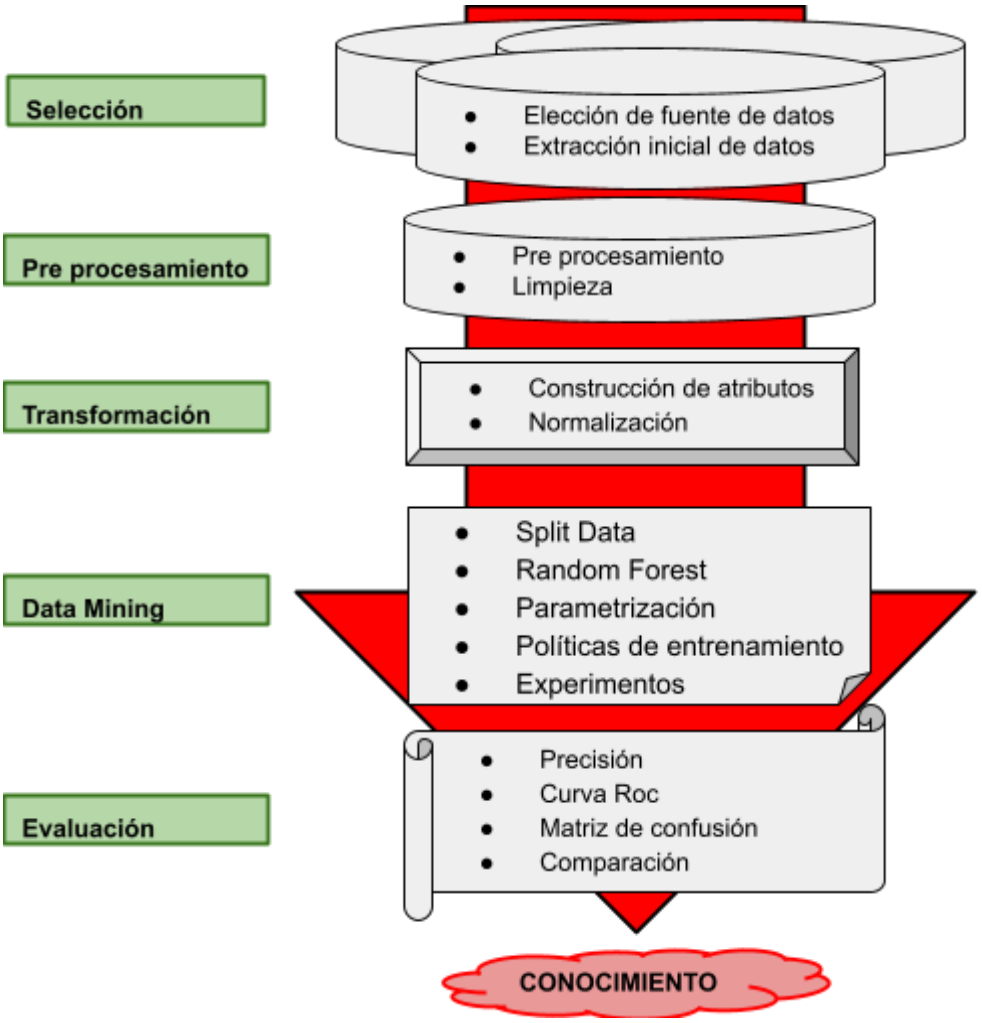


Figura 1: Esquema de trabajo donde se resumen 5 grandes etapas del proceso de KDD aplicado en este trabajo.

En la *Figura 1* se pueden observar el orden cronológico de los pasos llevados a cabo para realizar el trabajo . Pasos que serán detallados en el resto del documento.

Objetivo

El Objetivo de este trabajo es generar un dataset a partir de varias series de tiempo de acciones del mercado financiero y poder utilizarlo con un método de clasificación como es Random Forest y examinar la posibilidad de utilizar este tipo de técnica en datos que tienen un orden en el tiempo.

Objetivos particulares:

- Extracción de datos de acciones del mercado de valores de U.S.A.
- Exploración de los datos para la creación de un dataset con las acciones seleccionadas.
- Utilizar los datos anteriores para darle temporalidad el dataset.
- Generar atributos calculados que permitan explotar mejor la temporalidad de los datos.
- Explorar y seleccionar los mejores parámetros del algoritmo RandomForest.
- Utilizar y comparar distintas políticas de re entrenamiento de los modelos.
- Observar y analizar los resultados para determinar si es posible utilizar Random Forest para este tipo de datos.

Estructura del dataset

El formato habitualmente utilizado en análisis técnico de mercado de valores, cuenta con datos de apertura, cierre, mayor precio, menor precio y volumen (**O,C,L,H,V,T**) de un activo financiero, en un intervalo de tiempo definido que puede ser un periodo de minutos hasta meses. Para este trabajo se combinarán 5 activos distintos, unidos por marcas de tiempo, de esta forma se contará con **O,C,L,H,V,T** para cada activo en cada intervalo de tiempo, en un intervalo diario, es decir que los valores corresponden a la apertura, cierre, etc. en un día de actividad.

Temporalización del dataset

Este trabajo se basa en el algoritmo de Random Forest.(**RF**) Algoritmo que no considera una cronología de los datos o un orden temporal. Para poder llevar a cabo este trabajo se realiza una transformación del dataset para dar temporalidad y poder utilizar ocurrencias anteriores con RF. Estas transformaciones utilizan ocurrencias previas para darle un temporalidad a los datos.

Para lograr esto se agregan a cada tupla, los valores de todas las instancias hasta X instantes de tiempo anteriores, donde cada instante de tiempo representa un dia, generando una segmento temporal de tamaño X. De esta forma en cada ocurrencia se cuenta con una ventana con los atributos de las X marca de tiempos anteriores. Siendo $X \in \{5, 10, 30\}$, para generar distintos rangos de tiempo y poder comparar sus resultados.

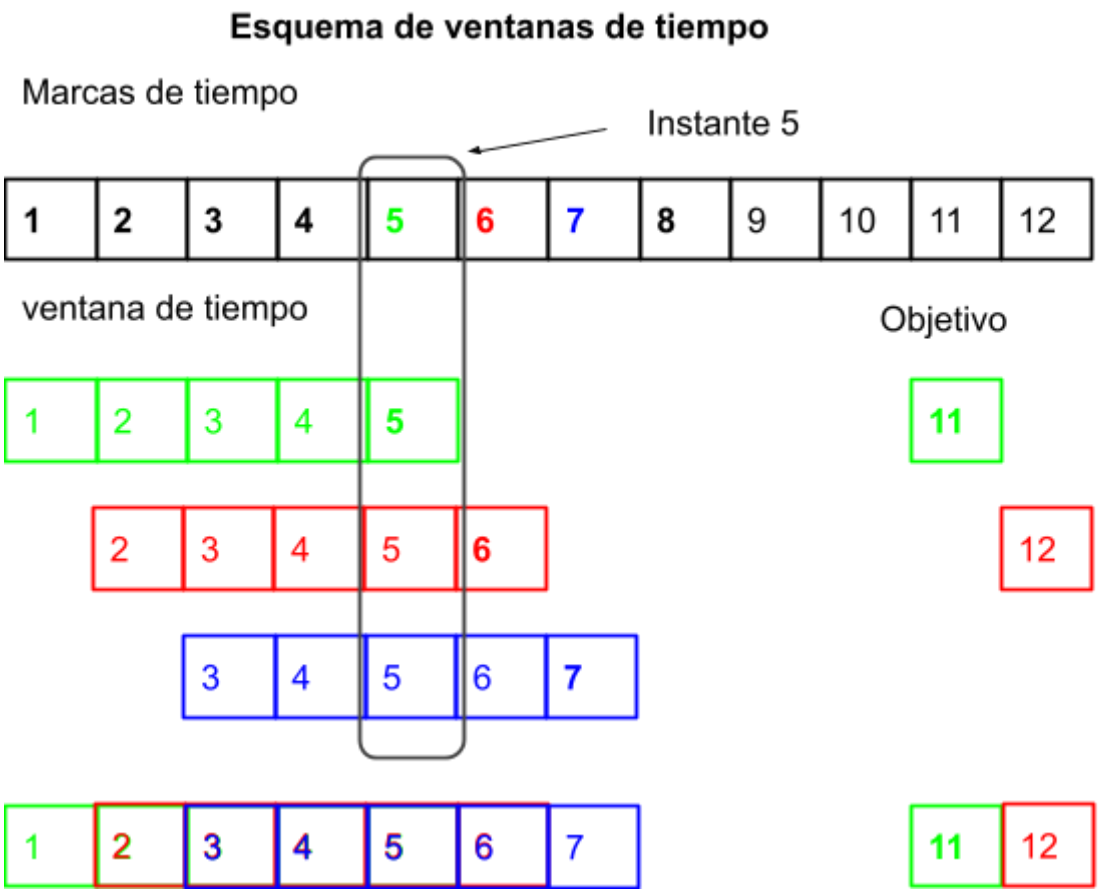


Figura 2

En la *Figura 2* se muestran 12 t*imestamps*, y tres ventanas de tiempo correspondientes a las marcas **5, 6 y 7** de color **verde, rojo y azul** respectivamentes. Al instante de tiempo 5, se ve que la tupla está formada por las marcas desde el 5 hasta el 1.

Selección de datos

En esta sección del trabajo se van a inspeccionar distintos sitios web, y ver las facilidad para extraer los valores de las acciones, teniendo en cuenta disponibilidad de datos y la calidad de los mismos.

Elección de fuente de datos

Para la formación del dataset se analizaron varios aspectos, entre estos están las fuentes donde extraer datos históricos del MV, la definición de los intervalos de tiempo y los activos que se van a agrupar.

Fuentes de datos: Para la extracción de datos se analizaron 4 sitios web conocidos en el ámbito del mercado de valores.

Los sitios analizados fueron:
https://finviz.com/
https://www.investing.com/
https://stockcharts.com/
https://finance.yahoo.com/

En un primer análisis se descartaron los sitios https://finviz.com/ y https://finance.yahoo.com/ por no ser completamente gratuitos.
Para los sitios restantes existe la posibilidad de realizar web scraping. En ambos casos se puede replicar un pedido a una api, y de esta forma obtener los datos requeridos en un periodo de tiempo pre definido.
Mientras que el sitio https://stockcharts.com/ respondia con un string de datos que requería un parseo especial, el sitio https://www.investing.com/ retornaba un contenido en formato JSON, que facilita el manejo del contenido.
Otra característica que inclino la elección del sitio https://www.investing.com/, es que cuenta con mayor cantidad de información, como noticias relacionadas a un símbolo en un periodo de tiempo que podría ser utilizado a futuro en una extensión de este trabajo.

Tabla a: Tabla que compara las características del los sitios seleccionados

Sitio	Gratis	Cantidad de datos	Formato Respuesta	Información Extra
https://finviz.com/	Parcialmente	Media	Json	NO
https://www.investing.com/	Si	Alta	Binario	Si
https://stockcharts.com/	SI	Alta	Strings	NO
https://finance.yahoo.com/	Parcialmente	Alta	Json	NO

Request

La vista general de los datos en los sitios web se muestra en gráficos (Gráfico 3 y 3 bis), pero para la formación del dataset, se utilizó un conjunto de archivos json utilizados por el sitio para dibujar los gráficos.
Luego de seleccionar el sitio para extraer los datos, con la herramienta de inspección del browser extraemos la información necesaria para poder replicar el pedido a la api del sitio y utilizar los parámetros dentro de un script que guarda los resultados en un archivo Json para su posterior tratamiento.

Gráfico de velas chinas

Gráfico 1: Gráfico de velas chinas del sitio stockcharts.com

Gráfico lineal

Gráfico 1 bis: Gráfico Lineal del sitio investing.com

Gráfico 1 y 1 bis son las vistas estándar de los sitios de algunos de los sitios web dedicado al mercado financiero

Exploración del sitio para implementar el scraping

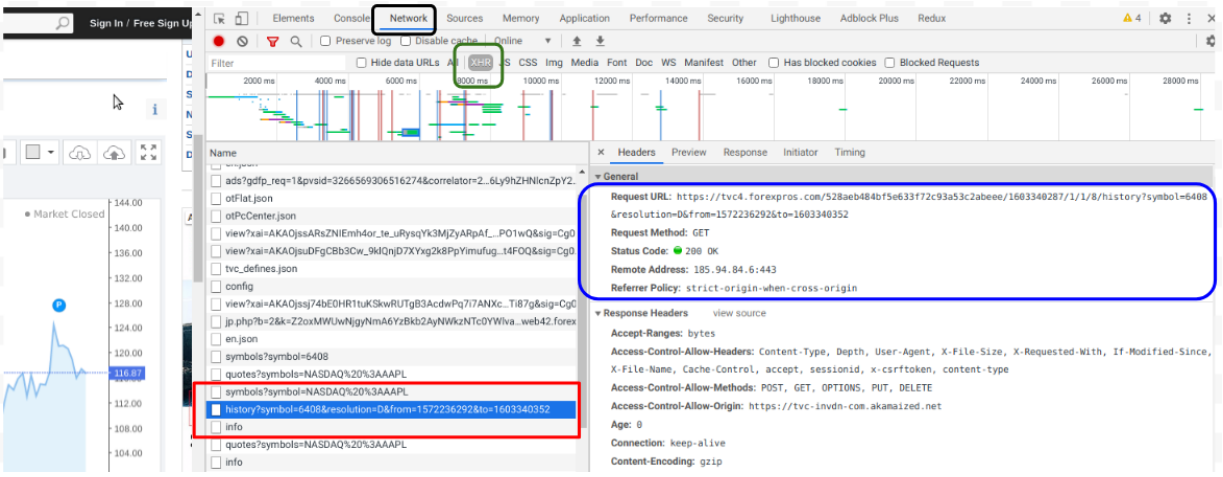


Figura 3

En La *Figura 6* podemos ver como en la pestaña Network del inspector , filtrando el tráfico por "*xhr and fetch"* y buscando cual es pedido que tiene la información que deseamos guardar. Con el pedido localizado solo queda ver que la url y los parámetros que utiliza

URL
https://tvc4.forexpros.com/3f07fad0d63e4e81d7ee5059f4cf7c43/1582662962/1/1/8/history?symbol=**6408**&resolution=**D**&from=**1582586960**&to=**1582873620**

Symbol: *6408*
resolution: D
from: *1582586960*
to: *1582873900*

Parámetros del request

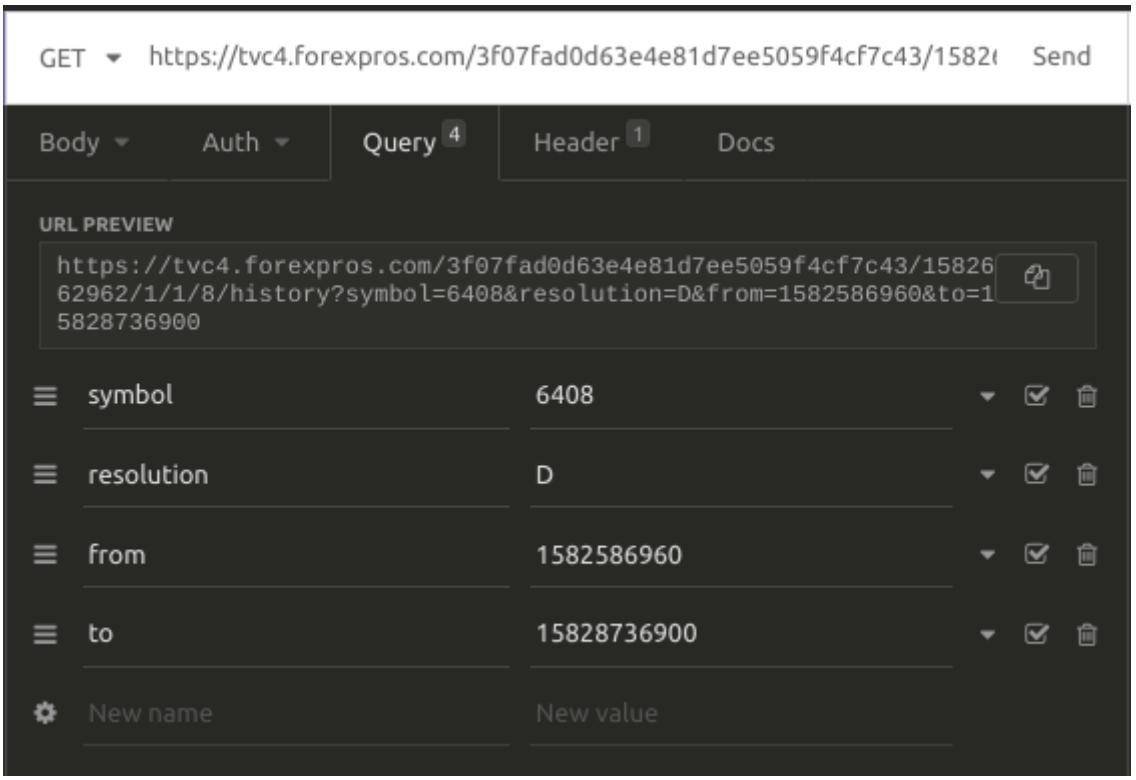


Figura 4: Parámetros utilizados en el request para obtener los datos

Respuesta:

{
"t": [1582588800],
"c": [288.07998657227],
"o": [300.95001220703],
"h": [302.5299987793],
"l": [286.13000488281],
"v": [56214132],
"vo": [0],
"s": "ok"
}

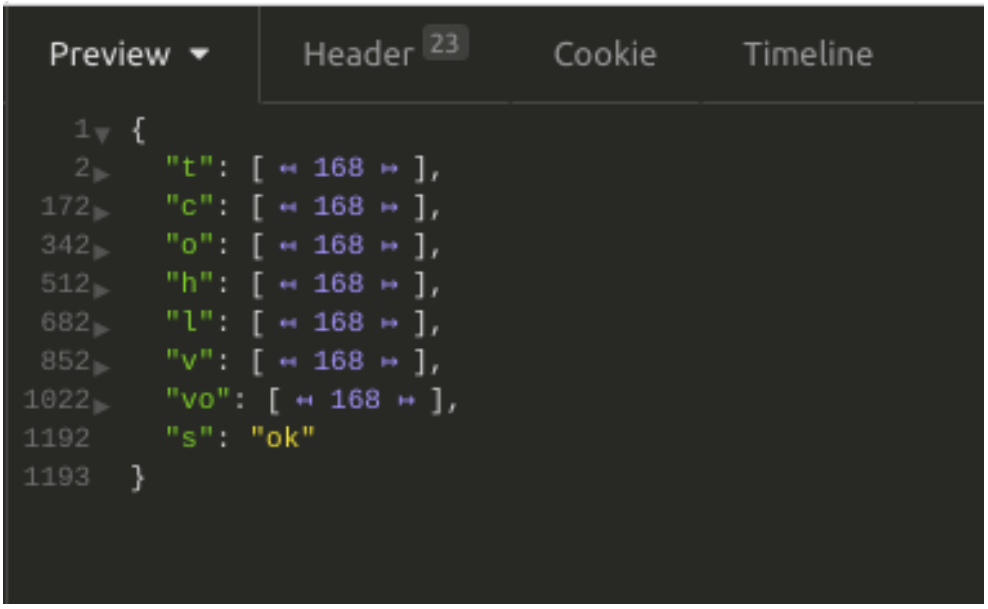


Figura 5: Response ilustrativa a la request de datos

Selección de símbolos y periodo de tiempo

En este caso se tuvo en cuenta la disponibilidad de distintos símbolos en el tiempo, y con la condición de que coexistieron durante un periodo de tiempo extenso.
Luego de observar varias opciones, se escogieron los símbolos del ORO (8830), WTI (8849), INDICIE USD (8827), APPLE (6408),EXXON (7888), Y se optó por un periodo que arranca en 17-09-2001(1000684800) hasta 31-12-2018 (1546300799). Este periodo son más de 4300 ocurrencias para cada acción.

Extracción inicial de datos:

Para la extracción de datos, se generó un script que itera entre un conjunto de símbolos seleccionados y los almacena todos juntos en formato csv para su posterior manipulación.

El formato del archivo generado, es un json que cuenta con el siguiente formato
[{
name:'número del símbolo'
info: {"t": [...],"c": [...],"o": [...],"h": [...],"l": [...],"v": [...] }
}]
donde cada atributo dentro de info, contiene la lista de valores para el periodo de tiempo definido.

Este archivo parcial se puede ver en **##Scraping##**
Para este proceso se tuvo en cuenta no generar una sobrecarga sobre el sitio. y para ello se tuvo la precaución de generar una espera entre pedido y pedido.

Los datos generados también se pueden ver en **##Resultado Scraping##**

Preprocesamiento

Como se ve en el paso anterior los datos de cada activo vienen dados por una lista de valores que corresponden a cada atributo del objeto seleccionado.
Esto significa que se requieren varios pasos para poder crear un conjunto de datos que pueda ser adaptado para ser utilizado con el método RF.

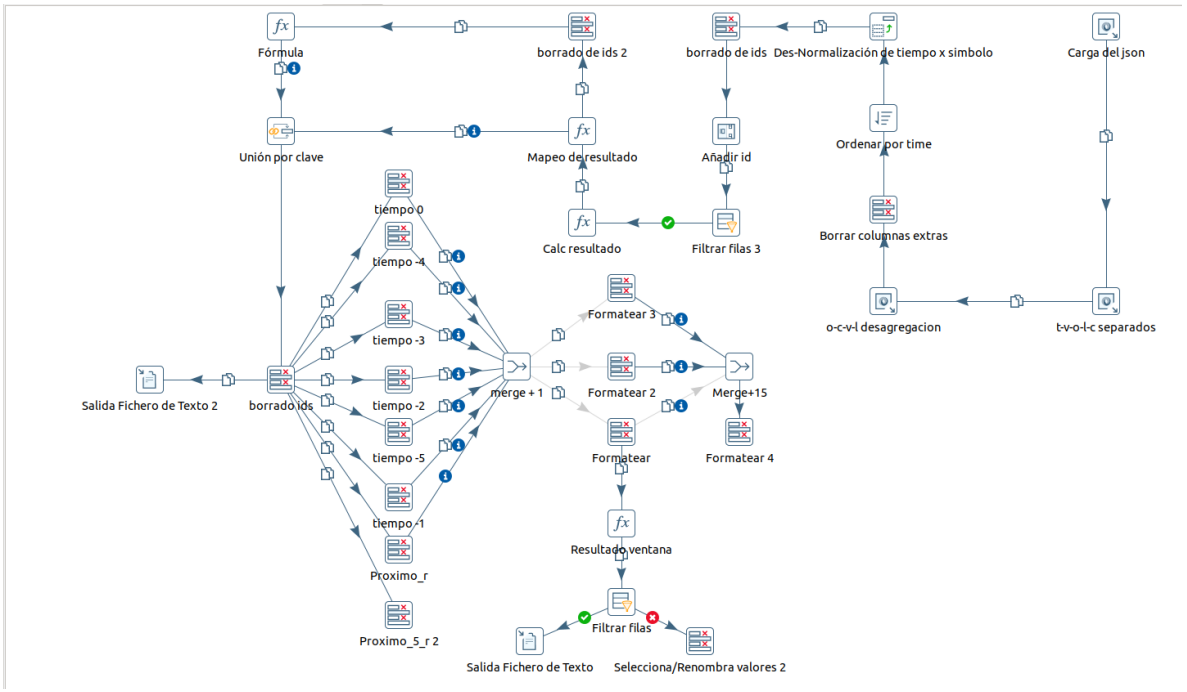


Figura 6: Esquema de trabajo con spoon

Transposición de valores

La primera parte consiste en generar una instancia por TS para cada valor dentro de la lista de cada acción, para luego unir todas las acciones dentro de una marca de tiempo común. Para esto se realiza una transposición de cada lista manteniendo el orden de aparición para poder unir los distintos atributos sin cambiar su orden y mantener de esta forma la integridad de los datos. Con esto quedan los datos agrupados por acción y sus respectivos valores incluyendo las marcas de tiempo y los valores que tomaron en esas respectivas marcas para luego ordenarlos por el atributo "**T**" de esta forma se respetó el orden de aparición.

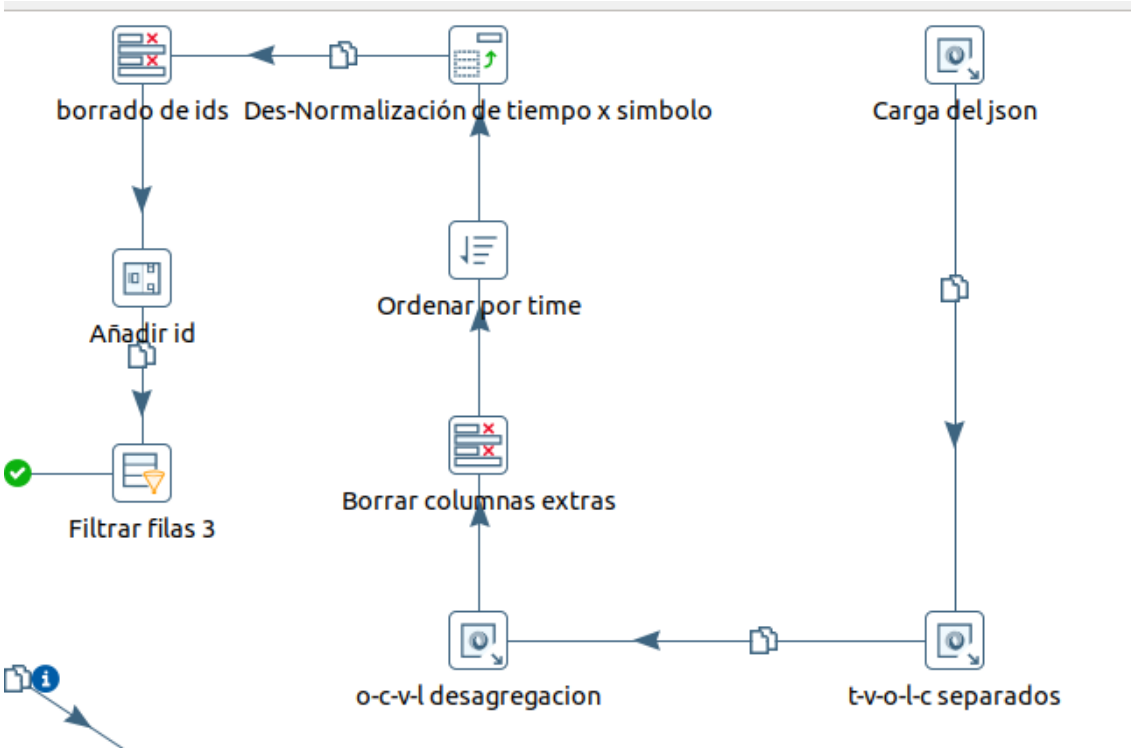


Figura 6 bis: Segmento del esquema de trabajo con spoon

pasando de tener el conjunto :

[{
name:'número del símbolo'
info: {"t": [...],"c": [...],"o": [...],"h": [...],"l": [...],"v": [...] }
}]

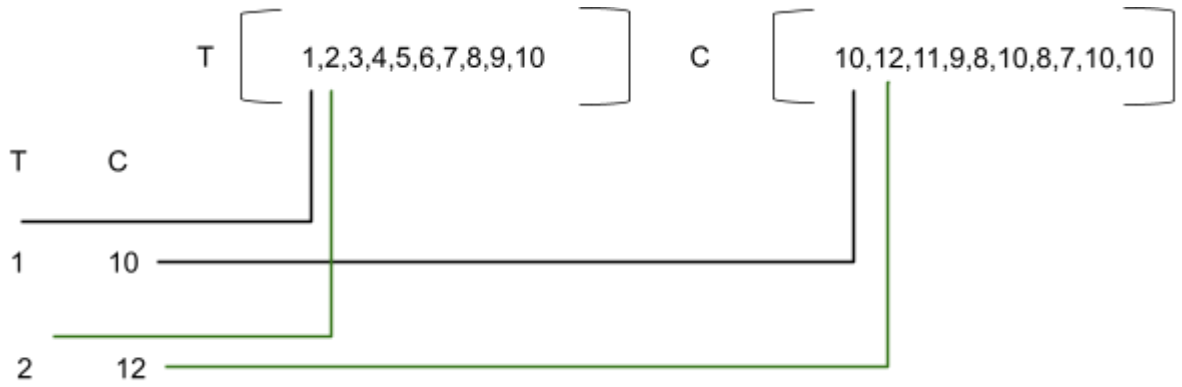


Figura 7: Esquema de reorganización de los datos

Luego de esto se forma una tabla por acción como el que se muestra en el *Tabla 1*

Tabla 1: Resultado de un acción agrupada por marca de tiempo

T	**C**	**O**	**H**	**L**	**V**
5456481	10	9	11	9	1200
5456482	11	10	12	10	1500
...	...	...	..	..	...
5456490	20	18	20	15	1250

Agrupación por marca de tiempo

Ahora contamos con todos las marcas de tiempo, agrupadas por acción, el siguiente paso es juntar todas las acciones dentro de una misma marca de tiempo en un solo conjunto de datos. Para este paso se generó un proceso que une todas las acciones por TS, generando nuevos atributos para cada acción, de esta forma multiplicamos por 5 las cantidades de atributos, agregando un sufijo con el nombre de la acción para cada atributo existente.
Este paso se realizó para poder tener en una misma marca de tiempo todos los datos con los que se cuentan para ese momento que permite nutrir el proceso de entrenamiento del modelo y facilita las próximas etapas del procesamiento.

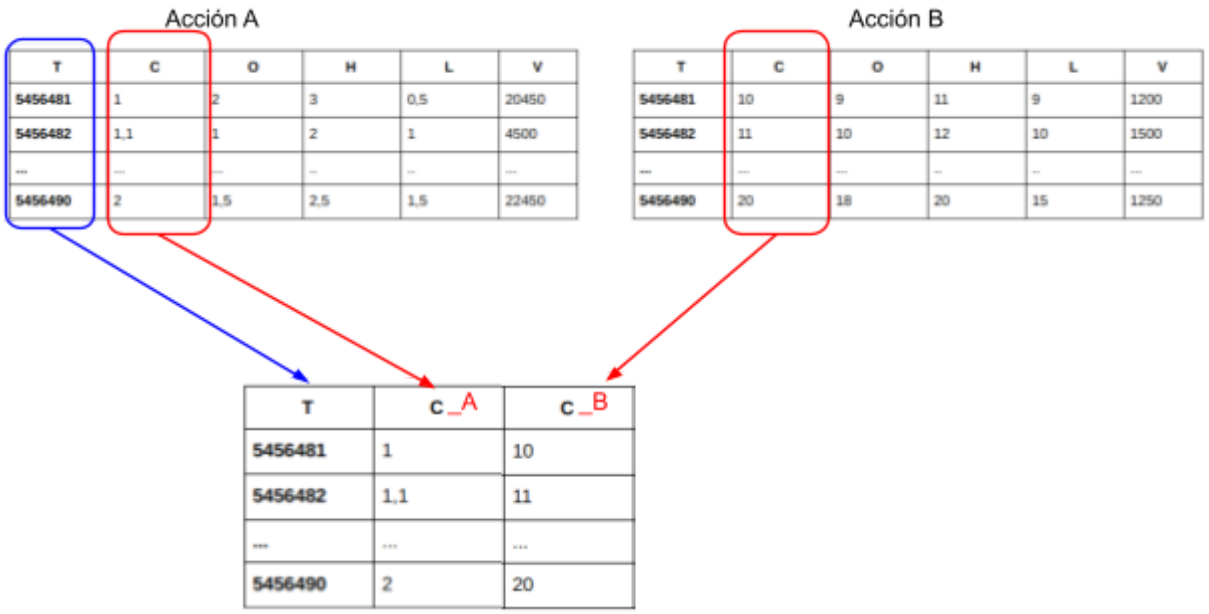


Figura 8: En la figura se muestra como se le asigna un sufijo a las columnas por acción agrupada

Este paso nos deja como resultado un conjunto de datos representado con la estructura que se muestra en la Tabla 2. Donde quedan apareadas cada una de las acciones con sus respectivas variaciones diarias.

Tabla 2

T	C_A	C_B	O_A	O_B	H_A	H_B	L_A	L_B	V_A	V_B
5456481	10	10	9	6	11	7	9	11	0	1200
5456482	11	11	10	10	12	6	10	2	0	1500
...	**...**	**...**	**...**	**...**	**...**	**...**	**...**	**...**	**...**	**...**
5456490	20	20	18	10	20	9	15	3	0	1250

Con esta estructura puede observar que algunas de las acciones no registraban el volumen, de esta manera no se contaba con ningún valor de volumen para varias de las acciones seleccionadas, por esto se decidió eliminar todos los atributos de volumen para todas las acciones, y contar con la misma cantidad de información para todos los activos.

Eliminación de faltantes en las marcas de tiempo

La integración de los datos por marca de tiempo habilita la aparición de valores faltantes. Siendo el valor dentro de las marcas de tiempo un valor irrecuperable, ya que no se puede calcular y un error en este cálculo podría incorporar un dato futuro, se tomó la decisión de eliminar todas las instancias que tenían el valor de **TS** como nulo. Esto se realizó para garantizar que no se está colocando un dato en un orden temporal erróneo.

Join temporal

Con el objetivo de agregar los atributos correspondientes a las ventanas de tiempos establecida en 5 días, es decir los valores ocurridos para las distintas acciones en los anteriores 5 observaciones y las observaciones a 1, 5 y 30 días posteriores. Se generaron un conjunto de ids auxiliares que indican cuales son los los valores de los ids que se van a utilizar para cada uno de estos atributos nuevos.
El segundo paso de esta etapa es agregar a cada marca temporal las observaciones anteriores trayendo los valores que indican cada id auxiliar, solo dejando con los valores de las variables = O,C,L,H.

Generación de variables Objetivo

El próximo paso consiste en generar una variable categórica, que será el resultado de cada acción en ese día, esta variable tomará los valores de “Subió” y “Bajo”. Para considerar que una acción subió su valor de cierre para ese dia tiene que ser mayor a su valor de apertura de no ser así se considerara que la acción bajo. Este procedimiento es igual para los valores a 5 y 30 días, teniendo en cuenta que se considera subió si el valor de cierre dentro de 30 días es mayor al valor de cierre de ese dia.

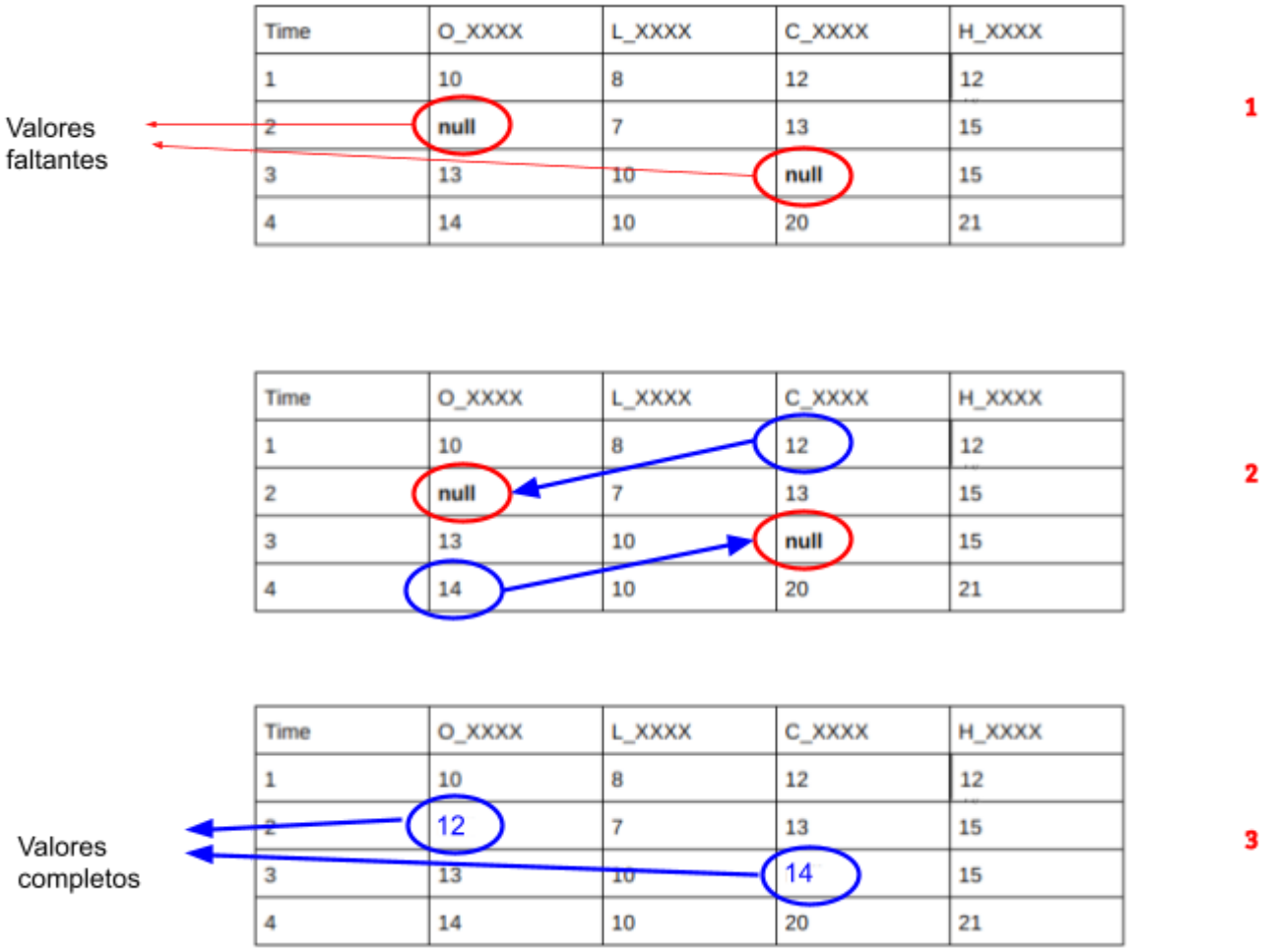
Eliminación de los atributos innecesarios

Este paso se encarga de limpiar todos los atributos generados para tareas propias del proceso solo dejando información extraída y los generados para los resultados.
Esto concluye con un archivo **##Transformacion##** que luego se continuará procesando.

Todas estas tareas se pueden encontrar en **##Proceso ktr##**

Imputación de nulos

En un paso anterior se eliminaron ocurrencias por no contar con marca de tiempo, ya que este era el único valor que podía garantizar orden en los datos. Pero el resto de los datos pueden ser procesados para rellenar los datos faltantes. Para este paso se tomaron dos decisiones distintas para dos casos diferentes.
El primer caso viene dado por el faltante en un dato de apertura o cierre.Estos casos tienen la posibilidad de ser completados con un valor de un caso anterior o posterior. Se llenó el campo faltante de apertura (O_XXXX) con el valor del cierre anterior (C_xxxx -1). Algo análogo se hizo para los valores faltantes en el campo de cierre (C_XXXX) donde se completó con el valor de apertura del instante posterior (O_XXXX + 1).
Algo a tener en cuenta es que si el valor que busco también es faltante se buscará la ocurrencia previa que tenga un valor no nulo.



Time	O_XXXX	L_XXXX	C_XXXX	H_XXXX
1	10	8	12	12
2	null	7	13	15
3	13	10	null	15
4	14	10	20	21

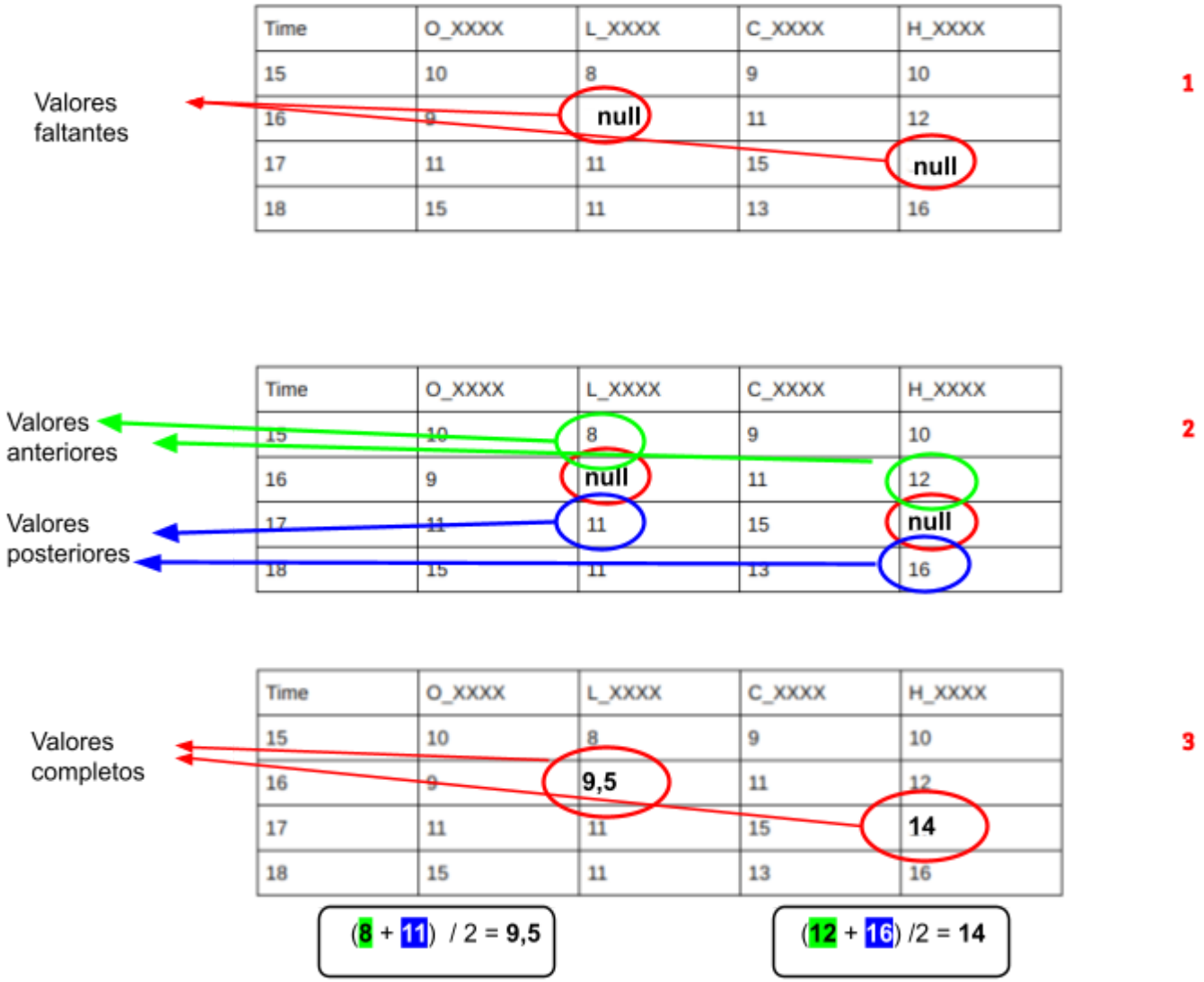
Time	O_XXXX	L_XXXX	C_XXXX	H_XXXX
1	10	8	12	12
2	null	7	13	15
3	13	10	null	15
4	14	10	20	21

Time	O_XXXX	L_XXXX	C_XXXX	H_XXXX
1	10	8	12	12
2	12	7	13	15
3	13	10	14	15
4	14	10	20	21

Figura 9: Estrategia de imputación de datos faltantes de apertura o cierre.

En la *figura 9* se pueden observar 3 pasos para la imputación de los valores faltantes:
Paso 1: Se detectan los campos donde los valores de apertura y cierre tienen valor nulo.
Paso 2: Se localiza el valor de cierre anterior o apertura posterior según corresponda.
Paso 3: Se reemplaza el valor faltante con el valor escogido para ese campo.

El segundo caso está dado por los valores faltantes en los campos High y Low (Figura 10). En estos casos se decidió utilizar el valor medio entre las ocurrencias anteriores y posteriores. También teniendo en cuenta que si algunos de estos valores también es nulo, se busca la ocurrencia próxima con un valor no nulo.



Time	O_XXXX	L_XXXX	C_XXXX	H_XXXX
15	10	8	9	10
16	9	null	11	12
17	11	11	15	null
18	15	11	13	16

Time	O_XXXX	L_XXXX	C_XXXX	H_XXXX
15	10	8	9	10
16	9	null	11	12
17	11	11	15	null
18	15	11	13	16

Time	O_XXXX	L_XXXX	C_XXXX	H_XXXX
15	10	8	9	10
16	9	9,5	11	12
17	11	11	15	14
18	15	11	13	16

Figura 10: Estrategia de imputación de Low o High

Los pasos para el segundo caso de imputación son:
Paso 1: Se detectan los casos nulos para las columnas de High y Low
Paso 2: Se detecta el valor posterior y anterior para cada valor nulo
Paso 3: Se reemplaza el valor nulo con el promedio del valor anterior y posterior

Todas estas modificaciones se pueden ver en la notebook **##[Notebook](#)##**

Transformación

Esta etapa incluye todas las tareas que modifican de alguna forma los datos. La transformación de los datos es una tarea bastante laboriosa, ya que en un primer momento es difícil saber exactamente cuales son todas las transformaciones necesarias.
El enfoque tomado para realizar estos pasos fueron dos, el primero incorporar algunas features que puedan agregar valores y métricas que ayuden a los modelos de random forest a mejorar su desempeño.

Creación de Features

Para agregar features se utilizaron algunas métricas conocidas en el mercado de valores que ayudan a la temporalidad y dan alguna información extra para conocer un poco el contexto de los valores. También se utilizan sumarización de algunas ocurrencias que pueden también agregar información al conjunto de datos. Por último, se crean variables objetivo que serán modeladas con Random Forest para clasificar suba o baja de precios de las acciones.

A continuación se listan las variables calculadas compuestas por índices y descriptores propios del dominio bursátil. Estos son:

- **Bollinger Bands:** Son dos bandas que están dados por la media de los valores en un periodo de tiempo, +/- el desvío estándar en el periodo de tiempo que se está tomando , para más detalle ver la *formula*.
- **RSI**: es un indicador que muestra la magnitud y velocidad del cambio en los precios, para más detalle ver en *formula*.

- **EMA** (media móvil ponderada): es la media de los valores en un rango de tiempo, y pondera a los valores más recientes dentro del rango que se calcula la media (ver *formula*).

- **SMA** (Media móvil simple): es la media de los valores en un rango de tiempo
- **Pendiente:** Es básicamente la pendiente de la recta que une los puntos de cierre de la ocurrencia anterior menos la apertura en una ocurrencia X días anteriores dividido la cantidad de días.
- **Cantidad de bajas:** son la cantidad de ocurrencias consecutivas de bajas en los precios.
- **Cantidad de altas:** son la cantidad de ocurrencias consecutivas de subas en los precios.
- **Ocurrencias anteriores:** Son las k instancias y valores generados anteriores al momento actual
- **Variables objetivos:** en este trabajo se va a utilizar un mismo conjunto de datos para predecir si varias acciones van a subir o bajar en distintas ventanas de tiempo,

para esto necesitamos calcular estos resultados, que luego el modelo va a usar como objetivo. Se crearon 3 para cada acción, uno a un día en el futuro, a 5 días en el futuro y 30 días a futuro.

Se pueden ver las funciones para la creación de los indicadores en la **##Celda##** de la notebook utilizada para transformar estos datos.
Para la creación de indicadores cantidad de altas y bajas, y ocurrencias anteriores se puede observar la siguiente **##Celda##**

Eliminación de tendencias

La eliminación de la tendencia surgió en el trabajo luego de notar que los modelos que se utilizaban siempre realizaban la misma predicción, según la tendencia de los gráficos en el tiempo, es decir, como a lo largo de los años la tendencia de un gráfico era a la suba, el modelo para esa acción siempre predice suba, si bien se lograba un nivel aceptable de predicción, esto no está bien. Por lo tanto se decidió que los valores para los atributos tenían que ser transformados de manera que la tendencia de los activos no afecte a las predicciones.
Para realizar esta tarea se generó un algoritmo que calcula la diferencia entre una ocurrencia anterior y la actual, y se reemplaza el valor actual con esta diferencia, así los modelos no son afectados por las tendencias de las series.

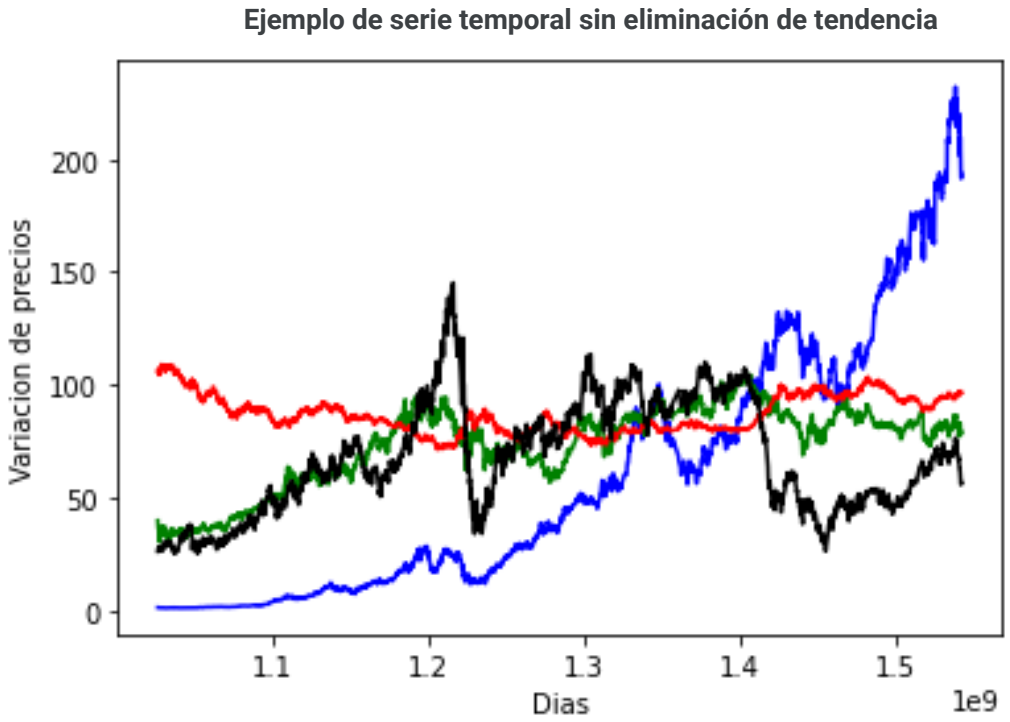


Figura 10: variación de los precios de los activos

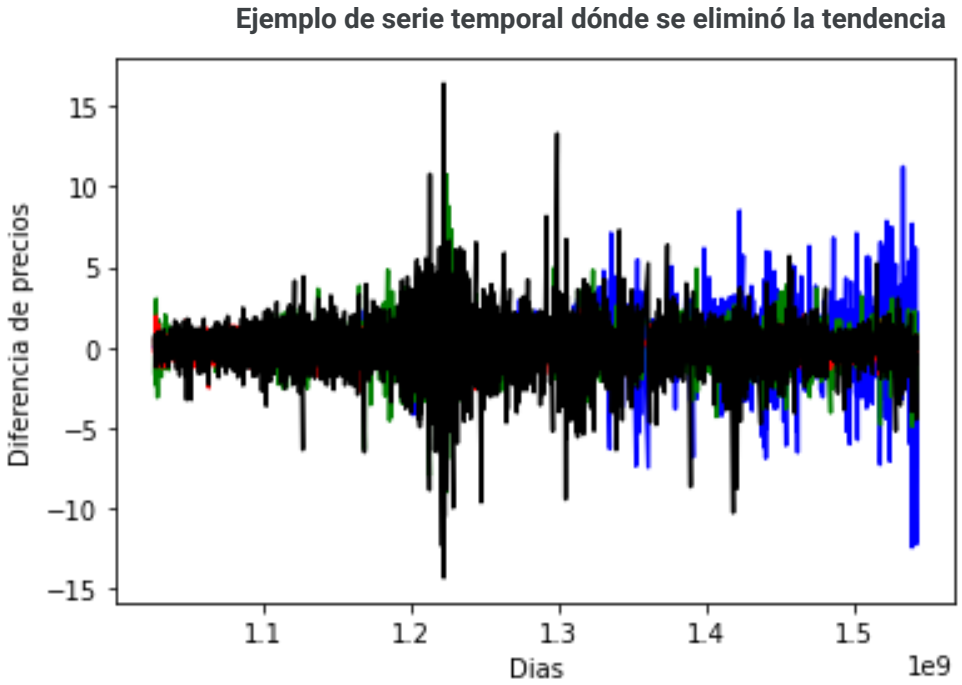


Figura 11: Variación de los la diferencia de precios de las acciones

Como se puede ver en la *Figura 11* ahora los datos no están influidos por las tendencias en el tiempo, logrando de esta manera que los modelos estimen mucho mejor con respecto a que las predicciones ahora no son siempre iguales.
Para una próxima iteración es tentador explorar la posibilidad de transformar los valores al porcentaje de diferencia entre los valores anteriores y los actuales .

Tabla 3: Matriz de confusión en clasificaciones de casos donde no se eliminó la tendencia.

	Predicted label		
True label		Subió	Bajo
	Subió	381	287
	Bajo	0	0

En la tabla 3 se puede ver que el modelo solo predice subas, y su resultado sólo se explica por coincidencia, ya que siempre predice que va a subir una acción por su tendencia general

Normalización

La normalización de los datos se realizó para ajustar los valores a una nueva escala en todos los campos menos los creados en la sección de *Creación de Features*. *(* Pendiente, Cantidad de bajas,Cantidad de altas,Ocurrencias anteriores,Variables objetivos).

La normalización fue implementada con Z-score mediante StandarScaler un método que se apoya en la media (m) y desvío ($\boldsymbol{\sigma}$) de las muestras para generar la transformación de los datos. Para respetar la temporalidad de los datos y procurar no usar información del futuro. Situación que no se cumpliría si usamos el dataset completo para los cálculos de la media y el desvío. Para cumplir esta restricción y poder utilizar Z-score se separa la normalización en dos procesos distintos, el primero de entrenamiento y transformación y el segundo solo de transformación.
El primer paso realiza un modelo que guarda las variables m **y** $\boldsymbol{\sigma}$, calculada con ese conjunto de datos, y estos valores son utilizados para realizar la transformación.
El segundo paso simplemente realiza la transformación apoyándose en el modelo generado en la etapa anterior, de esta forma los últimos datos son normalizados sin utilizar información del futuro, ya que utilizaron la m **y** $\boldsymbol{\sigma}$ de los datos pasados.
Estos pasos están reflejados en la ##celda## .

El Resultado de todo este proceso es el archivo ##ResultadoTransformacion##

Data mining

En la sección data mining se realizan varias tareas para desarrollar un modelo capaz de entrenar y clasificar datos con temporalidad. El proceso incluye la separación del dataset en sub sets más pequeños para probar los algoritmos. También se elaboran 5 políticas distintas para actualizar los modelos utilizados.
Todo los resultados se almacenarán de forma que permitan una comparación completa.

Split Data

Las primeras separación de los datos son obligadas, para conseguir un conjunto de entrenamiento y otro de testing y estos dos conjuntos a su vez se separan en atributos y etiqueta. En resumen necesitamos separar los atributos para entrenar y testear el modelo. Esta separación también se da para las etiquetas, vamos a tener que tener un grupo de etiquetas para entrenar el modelo y otro para el testing. Las etiquetas o variables objetivo son los valores discretos que tiene que clasificar el modelo.

En este trabajo además se utilizó otra separación de datos, que consiste en generar fracciones iguales del dataset, para poder probar los modelos en datasets con distintas cantidad de días, esto permite probar el comportamiento de los modelos en el transcurso del tiempo.
Todas las separaciones tienen en cuenta el orden de los datos, de esta forma nunca encontraremos en los conjuntos de entrenamientos datos que son más nuevos que los datos en los conjuntos de testing.

Figura 12: Separación del dataset en atributos y etiquetas, y separación en entrenamiento y test

En la figura 12 se puede observar la separación del dataset tradicional donde quedan bien definidos 4 grupos distintos.

1. Los atributos de entrenamiento dentro de los cuadros azul y verde,
2. Las etiquetas de entrenamiento dentro de los cuadros roja y verde
3. Los atributos de *testing* dentro de los cuadros azul y naranja

4. Las etiquetas de testing son las incluidos en los cuadros rojo y naranja

Para poder probar los modelos en periodos de tiempo de distintas duración. la separación de datos se modificó para generar varios dataset más pequeños, los subconjuntos generados tienen un tamaño X ∈ {200,350,500,700} .

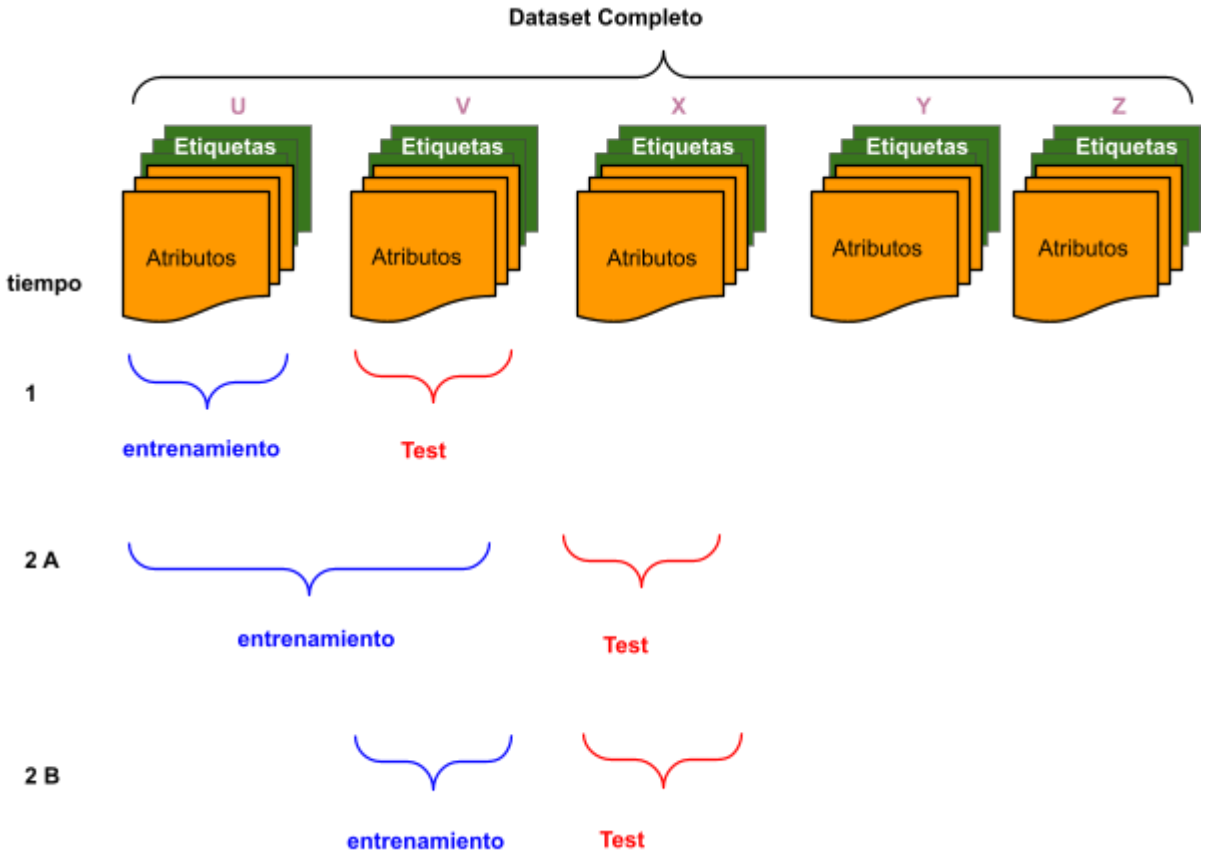


Figura 13: Diferencia de agrupamiento de los datos al simular el avance del tiempo

La *figura 13* muestra cómo se realiza la división del dataset para probar distintos enfoques. Podemos ver que el conjunto completo de datos se divide en nuevos dataset más pequeños (U,V,X,Y,Z), teniendo todos estos la misma cantidad de TS, menos el último que puede tener menos.
En la *Figura 13* se pueden ver dos tipos distintos de agrupación de los datos en el punto 2A y 2B, que forman subconjuntos de datos distintos con el mismo propósito, probar la eficiencia de los modelos en el tiempo y poder testear distintas políticas de entrenamiento de los modelos.
En **1** es común para todos los experimentos, se toma **U** que es el grupo de datos más viejos, es decir que ocurrieron primero y se lo utiliza como entrenamiento para probar el grupo **V.**
En 2**A** también se forman dos conjuntos uno para entrenamiento y otro para testing, tomando los datos de **U** y **V** como grupo de entrenamiento y utilizando **X** como testing.
De esta forma se van armando distintos conjuntos, simulando un avance temporal hasta terminar con los datos de U,V,X,Y para entrenamiento y Z como testing

La diferencia en 2**B** es que a partir del momento dos esta separación solo se queda con los datos en **V** para entrenamiento(en 2 A tenia U y V) y **X** como conjunto de testing.
Estos conjuntos nos permiten comparar cómo se comportan los modelos con el correr del tiempo.

Random Forest

El objetivo de este trabajo es utilizar el algoritmo de clasificación Random Forest, en un conjuntos de datos con un orden temporal, para esto se generó un datasets con las características explicadas en pasos anteriores.

Algoritmo

El algoritmo de random forest (**RF**) es un método de aprendizaje supervisado basado en árboles de decisión. En RF cada árbol de decisión es construido con un subconjunto aleatorio de las muestras de entrenamiento, y luego utiliza promedios de las predicciones de los árboles para seleccionar las mejores divisiones del árbol. Al tomar promedio de las predicciones algunos errores pueden anularse , y así RF logra una varianza reducida al combinar varios árboles **[2].**
Vale aclarar que el algoritmo original de RF utiliza el método de votación de cada estimador por una clase, y el de scikit-learn utiliza el promedio de las predicciones.

Parámetros

Para utilizar el algoritmo de RF hay que definir un grupo de parámetros que son los que pueden marcar la diferencia a la hora de evaluar los resultados.
Para seleccionar los parámetros que se van a utilizar en los experimentos , se utilizaron dos herramientas de selección de parámetros que se compararon con los que vienen de base en el algoritmo.
Parámetros base:

- criterion: “gini”
- n_estimators: 100
- max_depth: “None”
- min_samples_split: 2
- min_samples_leaf: 1
- max_features: “auto”
- bootstrap: True

La primera herramienta de comparación fue *RandomizedSearchCV que utiliza un conjunto de valores para cada parámetro y los combina de forma aleatoria, generando un conjunto de evaluaciones para los datos y los parámetros aleatorios que seleccionó.*

Para esta evaluación se generó una prueba de los parámetros de forma aleatoria para cada acción con el total de los datos, separados en *training y test*.
Los resultados de estas pruebas no superaron en ninguno de los casos a los resultados del mismo conjunto de datos para los parámetros bases, por lo que se descartaron estos resultados y se dejaron los parámetros bases como punto de partida para mejorar el modelo y se pasó a la segunda herramienta para mejorar la performance de los parámetros bases.

La segunda herramienta utilizada es *GridSearchCV*. Qué prueba todo los conjuntos de parámetros especificados y los valida mediante *cross-validation*. De esta forma se genera una inspección más exhaustiva que con el método *RandomizedSearchCV*.
Este proceso resultó requerir una gran cantidad de procesamiento y tiempo de ejecución, razón por la cual colab y una máquina personal se quedaban cortas. Dada esta situación los algoritmos se corrieron en una máquina virtual en el cidetic aprovechando los recursos de un servidor que cuenta con 164 gb de ram y 24 procesadores. La tarea en este servidor llevo casi 3 días.
Este experimento devolvió un conjunto de parámetros que consiguió superar a los bases.

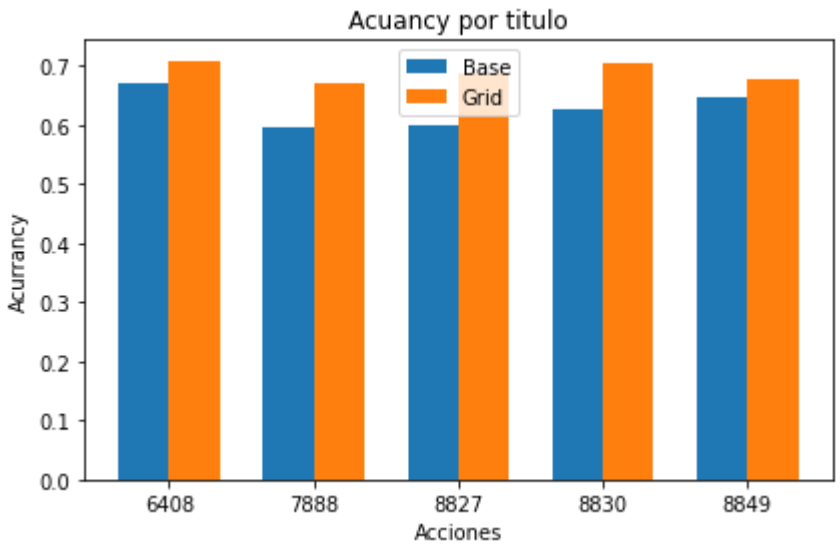


FIGURA 14: Comparación de los resultados de *Accuracy* con los parámetros base y los encontrados con gridSearch

Como se puede observar en la *figura 14* para cada acción se logró mejorar la precisión con respecto a los parámetros de base.

Algo que también se tuvo en cuenta es el tiempo que requería el algoritmo para correr con los distintos parámetros, ya que si la mejora no es significativa en algunos casos podría ser conveniente utilizar los parámetros que no requieran un gran esfuerzo de tiempo para ser ejecutados.

Estos experimentos se encuentran en este [script](), y los resultados se graficaron en esta [notebook]()

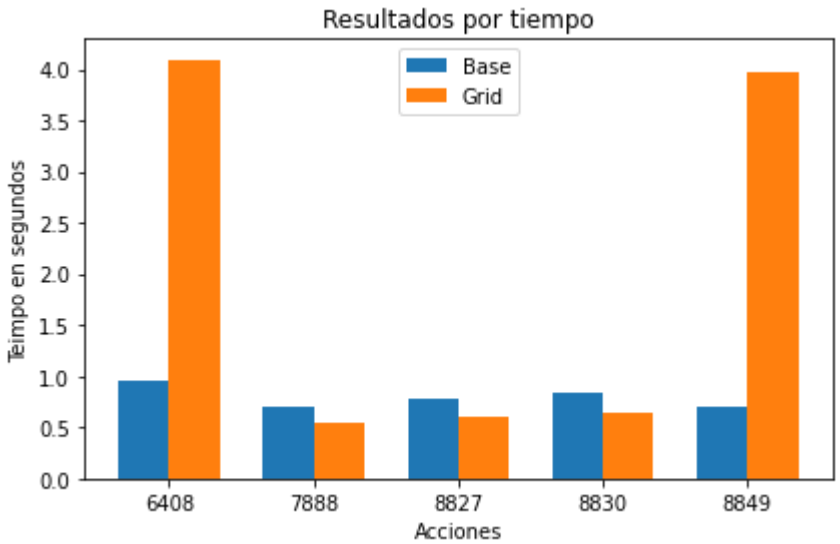


Figura 15: Comparación de tiempo para los parámetros base y los encontrados con gridSearch

En la *figura 15* se observa que para la mayoría de las acciones el gasto de tiempo es inferior para los parámetros obtenidos mediante grid, que los bases, pero también se observan dos casos donde el tiempo es 4 veces mayor para grtd que para base, un que esto no es un problema ya que son magnitudes pequeñas y solo son 3 segundos de diferencia.

Conjuntos de entrenamiento

Para probar RF en distintos escenarios, se decidió tener varios conjuntos de entrenamiento para cada acción. La idea de estas agrupaciones fue ver el rendimiento RF en el tiempo y analizar la performance en los distintos grupos de entrenamiento que se describieron en la sección “split data”.
Como se puede ver en esa sección, los datos de entrenamiento tienen dos comportamientos distintos a medida que se analiza un grupo posterior. El primer caso (2A) todos los datos anteriores al momento que se está analizando son utilizados como conjunto de entrenamiento y son evaluados con los datos del instante actual, de esta forma estaremos analizando la performance al usar toda la información anterior.
En el segundo caso (2B) solo usaremos como grupo de entrenamiento el subconjunto inmediatamente anterior al actual (A - 1) y descartando los datos que sucedieron antes que A -1, de esta forma se puede ver si el modelo es más eficiente con datos que son más parecidos o cercanos a los que se evaluaron.
En resumen estas dos formas de separar los datos marcarán si hay diferencia en los modelos al tomar todos los datos anteriores o solo un subgrupo inmediatamente anterior.

Tamaños de los conjuntos de entrenamiento

Para poder evaluar si la cantidad de datos que se usan para entrenar el modelo genera un cambio significativo en los resultados, se tomaron 5 tamaños distintos de los datos de entrenamiento, esto quiere decir que vamos a tener varios modelos que son entrenados con una cantidad diferente de días. Para esto se decidió utilizar sub datasets de 200, 350, 400, 500 y 700 días. Esta separación se utilizará para formar segmentos de tiempo que respetaron los dos modelos explicados en la sección anterior (Conjuntos de entrenamiento)

Políticas de re entrenamiento

Para completar los dos pasos anteriores se decidió tener distintas políticas de re entrenamiento del modelo para analizar en qué casos es conveniente utilizar recursos para volver a entrenar el modelo y si esto genera algún cambio significativo en las predicciones de RF.
Se optaron por 3 políticas de re entrenamiento.
La primera es **nunca** re entrenar el modelo, y utilizar siempre el generado con los primeros X datos y con estos predecir todos los conjuntos posteriores.
La segunda política, es re entrenar el modelo cuando el **AUC es inferior a 0,5**, así solo se regenera el modelo cuando la última predicción es inferior al rango de aleatoriedad.
Por último se toma como decisión **siempre** re entrenar el modelo, esta es una política costosa ya que siempre se está generando un nuevo modelo , pero puede resultar beneficiosa ya que siempre se cuenta con un modelo actualizado con los últimos datos.

Tabla 4: Tabla de modelos de entrenamiento

Model	Cuando vuelve a entrenar	Datos que utiliza
1	NUNCA	Solo el primer grupo
2	Cuando AUC < 0,5	TODOS
3	Cuando AUC < 0,5	ANTERIOR
4	Siempre	Anterior
5	Siempre	TODOS

Experimentos

Teniendo definido las distintas formas de generar los grupos de datos para entrenar los modelos, tanto en tamaño como datos históricos incluidos, teniendo definida las distintas políticas de re entrenamiento del modelo (tabla 4), y con intención de diferenciar las distintas acciones que se utilizan y las distintas ventanas que se determinaron para predecir, se procede a realizar todos los experimentos correspondientes.
Todas estas variables en juego generan un gran número de experimentos a realizar, aproximadamente
variables en juego:

A. **5** políticas de entrenamiento
B. **5** tamaños de dataset
C. **2** agrupamientos de datos anteriores
D. **5** Acciones distintas
E. **3** ventanas de tiempo

esto genera un total de 5 *5 * 2 * 5 * 3= 750 experimentos distintos

Todos estos experimentos se pueden ver con sus resultados en esta ##notebook##

Resultados

Para describir los diversos resultados que se encuentran con tantas variables a analizar, se decidió separar los resultados en 4 grupos distintos.
El primer grupo será un análisis general de todas las variables juntas , esto quiere decir que los resultados serán comparados todos con todos sin tener en cuenta sus características individuales.
El segundo análisis será comparar las distintas ventanas de tiempo, para saber si el modelo ajusta mejor para corto, medio o largo plazo de predicción.
El siguiente análisis será comparando las distintas políticas de re entrenamiento de los modelos. Para ver si es conveniente re entrenar o no, y con qué datos.
Por último, se comparan los resultados de los distintos tamaños de grupos tomados para los entrenamientos.

Resultados generales

A continuación se presentan las mediciones de AUC y Accuracy para los principales resultados.

Tabla 5: Mediciones de AUC para los top 20 modelos

Media	- Desvio	+ Desvio	Symbolo	Tamaño	Modelo	Ventana
0.6377	0.5155	0.7600	8849	200	4	30
0.6198	0.5109	0.7288	8849	200	1	30
0.6174	0.5187	0.7162	8827	400	2	30
0.5996	0.5059	0.6933	8827	400	1	30
0.5984	0.5027	0.6941	7888	200	4	5
0.5937	0.5368	0.6506	8827	500	1	30
0.5937	0.5368	0.6506	8827	500	2	30
0.5937	0.5368	0.6506	8827	500	3	30
0.5937	0.5368	0.6506	8827	500	5	30
0.5903	0.4777	0.7029	8849	200	5	30
0.5901	0.4578	0.7223	8849	200	2	30
0.5898	0.5272	0.6523	8849	400	2	30
0.5864	0.5444	0.6283	8849	350	4	30
0.5853	0.5087	0.6620	8849	400	1	30
0.5849	0.5177	0.6520	8830	400	4	30
0.5841	0.4746	0.6936	8849	400	4	30
0.5832	0.4742	0.6921	8849	400	3	30
0.5830	0.4941	0.6719	8827	400	3	30
0.5823	0.4953	0.6694	8849	200	3	5
0.5821	0.5188	0.6453	8830	350	4	30

Tabla 6: Mediciones de Acurancy para los top 20 modelos

Media	- Desvio	+ Desvio	Symbolo	Tamaño	Modelo	Ventana
0.6050	0.4998	0.7103	8849	200	4	30
0.6006	0.5195	0.6817	6408	350	4	30
0.5867	0.4381	0.7353	6408	200	4	30
0.5675	0.4674	0.6675	7888	500	3	30
0.5656	0.5220	0.6091	7888	700	4	30
0.5623	0.4932	0.6314	8827	400	1	30
0.5597	0.4796	0.6398	6408	500	4	30
0.5587	0.4601	0.6572	8849	200	3	5
0.5583	0.4926	0.6241	8830	500	1	5
0.5583	0.4926	0.6241	8830	500	2	5
0.5583	0.4926	0.6241	8830	500	3	5
0.5583	0.4926	0.6241	8830	500	5	5
0.5581	0.4520	0.6642	8849	400	3	30
0.5580	0.4139	0.7022	6408	200	3	30
0.5576	0.4119	0.7033	8849	200	2	30
0.5547	0.4363	0.6730	8849	400	4	30
0.5544	0.4843	0.6244	6408	400	3	30
0.5510	0.5081	0.5939	6408	700	3	30
0.5493	0.4993	0.5993	7888	500	4	5
0.5484	0.4455	0.6513	7888	500	1	30

Tomando los primeros 20 resultados, en orden de mayor a menor, donde la media es el valor medio de todos los resultados para cada experimento. Max y min son los mejores y

peores valores para ese modelo , y los últimos valores hacen referencia a las variables que se utilizaron para los experimentos.
Donde symbol es la acción predicta, tamaño es la cantidad de días con la que se armaron los segmentos del dataset, modelo hacer referencia cuando se re entrena el modelo y con qué datos, y por último ventana hace referencia a la cantidad de días a futuro que está prediciendo el modelo.
De estos resultados se puede observar que a primera vista los resultado no son muy alentadores, y solo los primeros tienen algún resultado por encima del 0,5 y que la mayoría son a largo plazo(30 días) y muy pocos a medio plazo (5 días). Análisis que se profundizará más adelante.
También se puede ver que el modelo 4 parece ser el que cuenta con mejores resultados, esto significa (los modelos están detallados en la tabla 3) que es conveniente siempre re entrenar el modelo con los el grupo de datos inmediatamente anteriores, análisis que también se detalla más adelante.

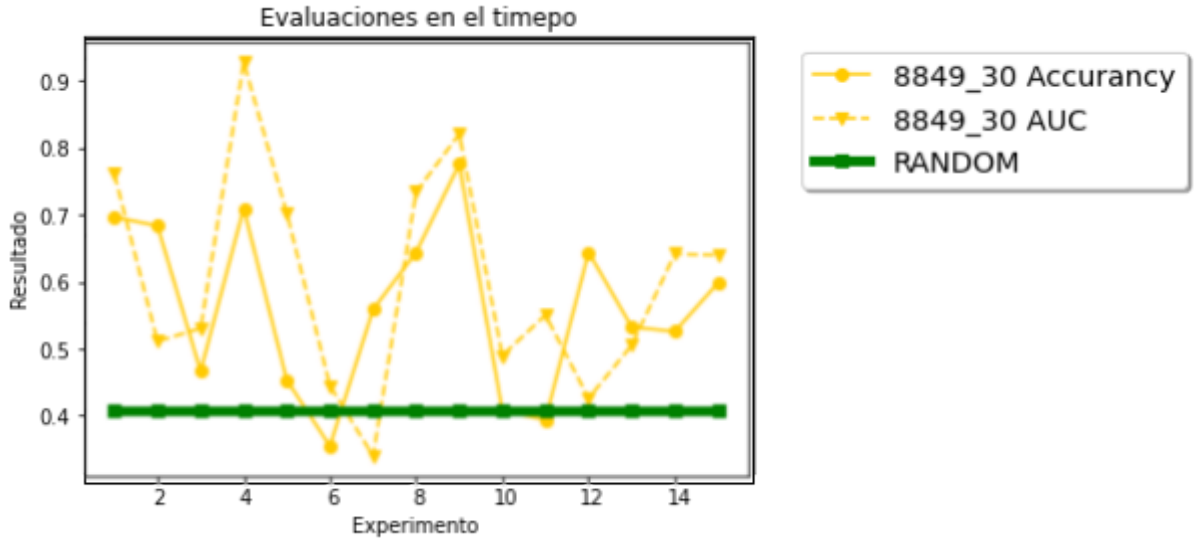


Figura 16: Gráfico de los resultados la acción 8849 correspondiente a la top 1

En el gráfico se puede ver que en este caso el modelo se comporta bien, y es el primero en la tabla general, y casi siempre se mantiene por encima de 0,5. alcanzando en el mejor de los casos un 89%

Para ver todos los resultados individualmente se creó el sitio https://masivasf.herokuapp.com/

Resultado ventanas de tiempo

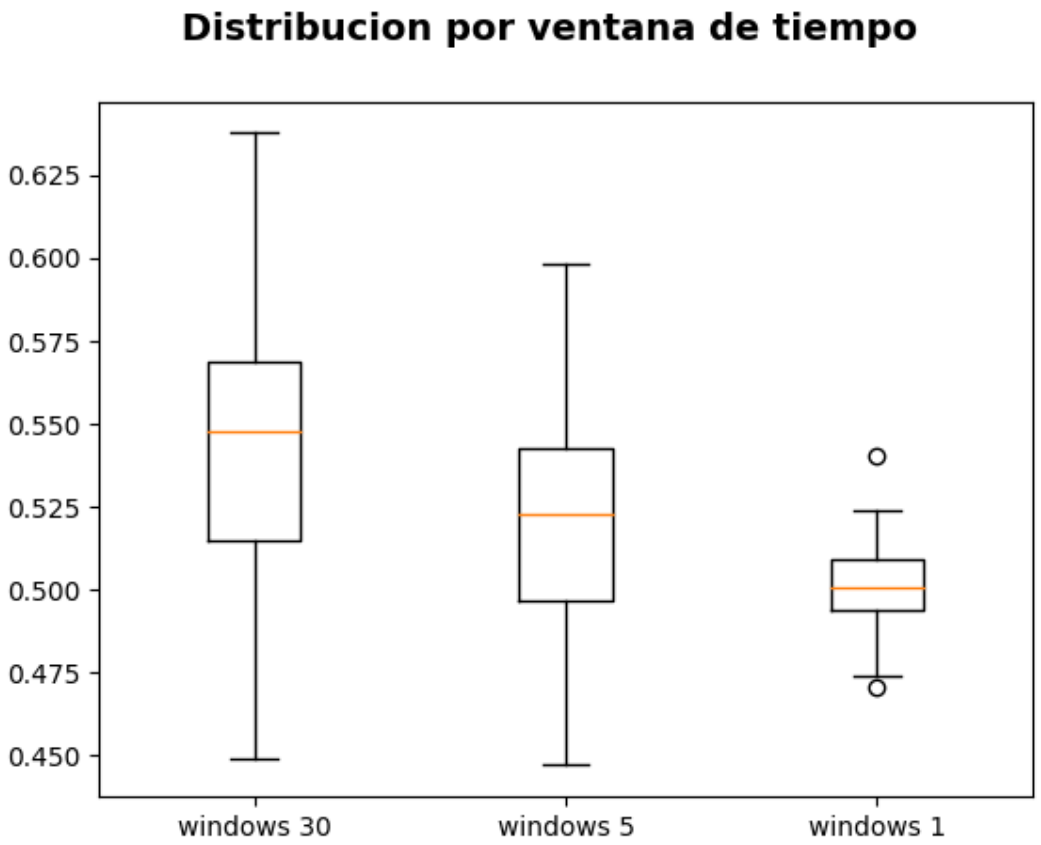


Figura 17: Boxplots de AUC agrupados por ventanas de tiempo

En figura 17 se observa la distribución de las medias de AUC para cada experimento separados por ventanas de tiempo.

En el gráfico se se nota claramente como los modelos funcionan bastante mejor para predicciones a largo plazo , y teniendo un AUC bastante más bajo en predicciones a corto plazo , aunque en términos generales los modelos no son muy buenos, si podemos afirmar que a medida que la ventan disminuye la calidad del modelo empora.

Resultado por Modelos

Como se explicó en la sección de experimento se utilizaron 5 modelos de entrenamiento distintos, para comparar los resultados se tomaron en cuentas los experimentos que tuvieron un AUC mayor a 0.5 y se representan en un gráfico de tortas para ver cual es el que tiene mayores resultados con un AUC > 0.6

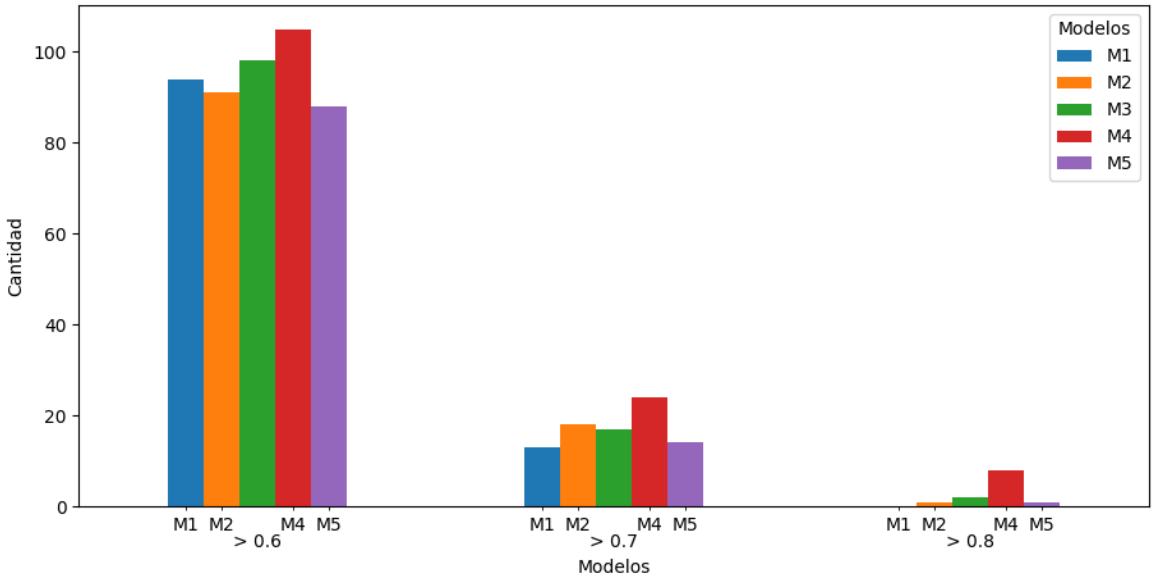


Gráfico 4: Acumulados de los modelos para AUC 0,6 0,7 y 0,8

Como se puede ver en el gráfico 4 para el acumulado de experimentos con AUC > 0,6 todos los modelos son similares, pero ya para un AUC > 0,7 el modelo 4 se destaca para los resultados con AUC > 0,7. Y por último con un AUC > 0,8 el modelo 4 es el que más acumula, aunque la cantidad es 8 podemos afirmar que el modelo 4 en comparación con los otros es el que mejor se comporta.

Resultados por tamaños de los subset de datos

En esta última sección se comparan los resultados teniendo en cuenta los tamaños de los datasets que se usaron para entrenar los modelos es decir la cantidad de días incluidos en cada segmentos para ir simulando avance en el tiempo y el comportamiento de los modelos a medida que esto sucede. Hay que tener en cuenta que aunque se escogieron distintos tamaños de segmentos según el modelo elegido este puede incluir varios segmentos anteriores.

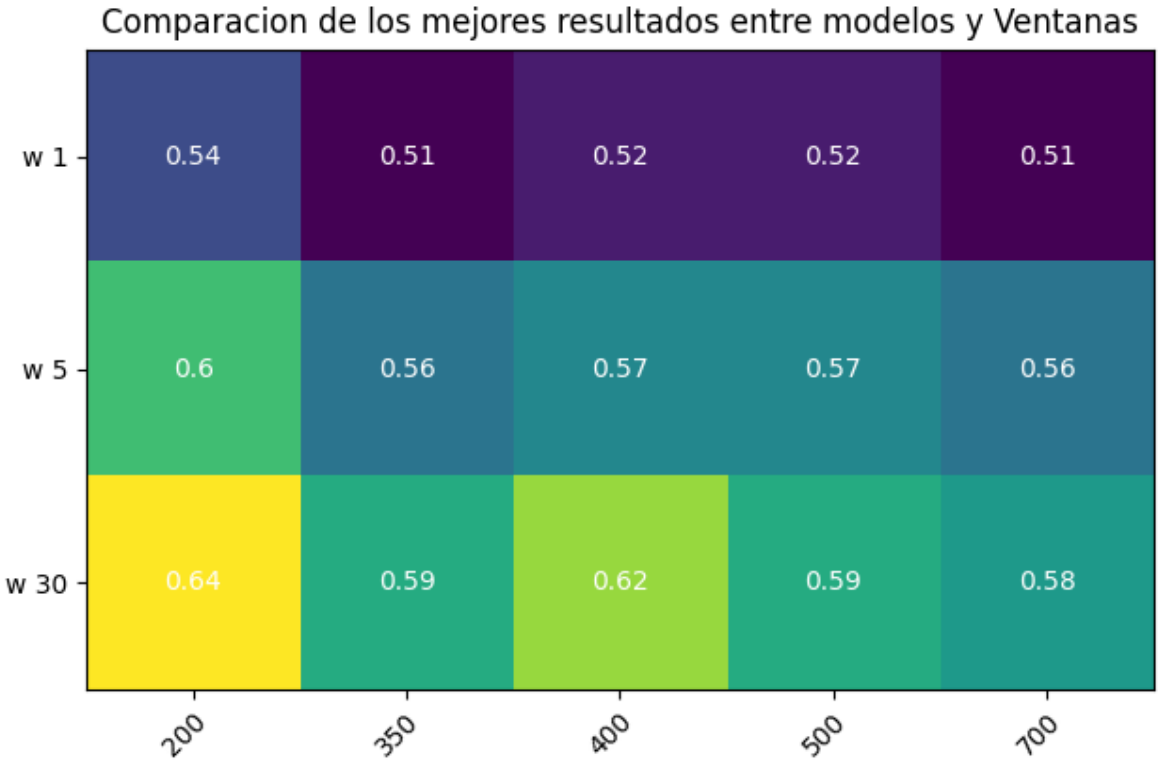


Gráfico 7: Heat map comparando ventanas y cantidad de datos

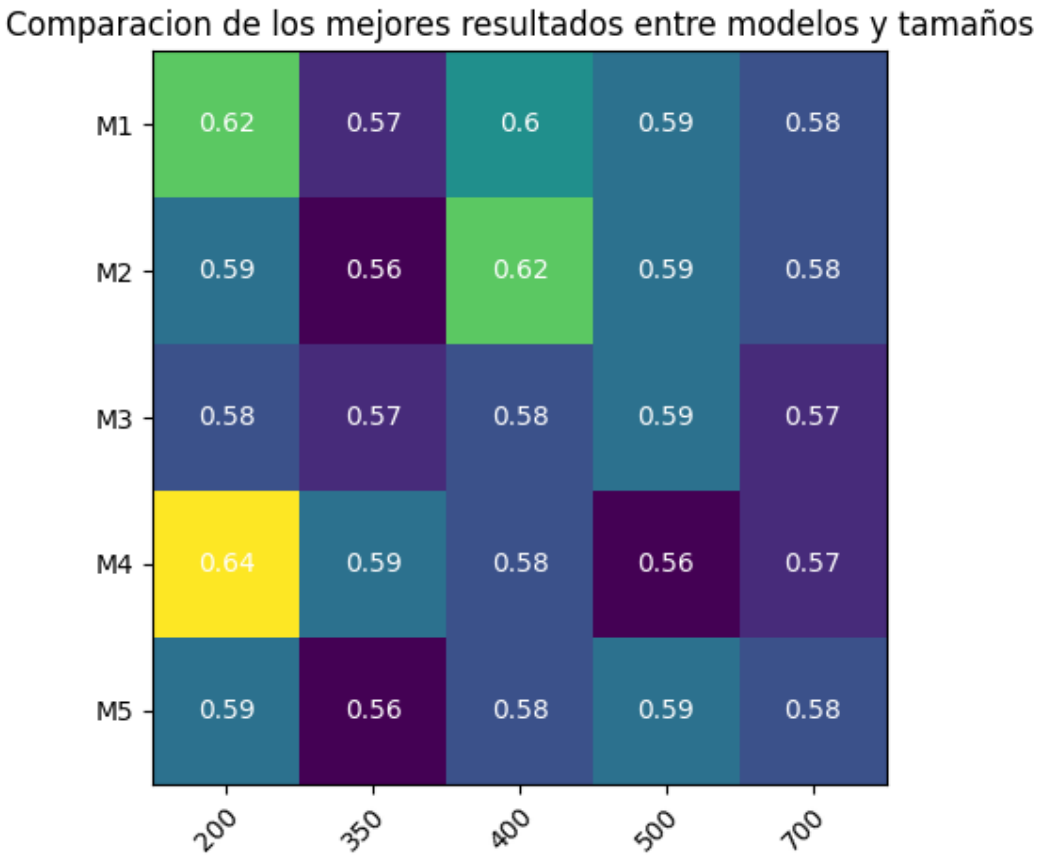


Gráfico 8: Heat map comparando modelos y cantidad de datos

En el gráfico 7 donde se relacionan la cantidad de días en los datasets con el tamaño de la ventana a futuro para predecir las tendencias medido en AUC, y se puede observar que los mejores resultados se dan para el tamaños de 200 días y como habíamos observado antes la ventana de 30 días a futuro tiene los mejores resultados.

En el gráfico 8 que muestra las relaciones entre modelos usados y tamaño se sigue observando que los mejores resultados están dados para un tamaño de 200 días, tanto para el modelo 1 como el modelo 4, respetando observaciones anteriores donde se mostraba al modelo 4 como uno de los mejores.

Conclusiones

Conclusiones personales

En cuanto a las tareas desarrolladas para realizar el trabajo puedo considerarme satisfecho, ya que realice un gran número de tareas distintas permitiendo aprender un gran abanico de herramientas y reforzar conceptos . También puedo considerar que realice un pequeño primer paso en las ciencia de datos, que me dejó ver que requiere un gran esfuerzo desde

la obtención de datos y su procesamiento y caer en la idea que obtener algún conocimiento desde los datos requiere muchas iteraciones con prueba y error.

Conclusiones Sobre el trabajo

Observando los resultados de muchos experimentos con tantos parámetros pude concluir que utilizar un algoritmo de RF para predecir series de tiempo no es muy eficiente, ya que los resultados obtenidos son muy fluctuantes y no están muy distantes al nivel de azar.
No obstante esto, en caso de usar este algoritmo para intentar predecir el comportamiento de una acción en el tiempo, puedo decir que los mejores resultados se dan cuando utilizo ventanas de tiempo a futuro grandes, entrenando el modelo con grupos chicos de datos, y que es mejor siempre volver a entrenar el modelo con un grupo de datos inmediatamente anteriores, antes de intentar predecir los nuevos valores.

Recursos y Herramientas

Para realizar este trabajo se utilizaron múltiples herramientas y recursos, debido a la amplia gama de tareas por realizar.
Los recursos utilizados en el trabajo fueron:

1) Colab: herramienta colaborativa ger google para programar en python en la nube sin necesidad de correr procesos locales y poder mostrar los resultados
2) Cidetic: Se utilizó el server del cidetic para correr experimentos que demandaban gran cantidad de recursos y no podían ser corridos de forma local ni en colab

Las herramientas utilizadas para el trabajo fueron varias, y algunas fueron descartadas ya que se encontró alguna mejor para realizar la misma tarea.

1) Spoon pentaho: La transformación de los datos extraídos en los sitios web fue tratada con esta herramienta tarea que luego se finalizó con python
2) Python: La mayoría de las traes se realizó con python ya que permitió agilizar el trabajo y dejar plasmado los resultados en las notebooks de colab
3) R studio: En un principio se implementó Random Forest en R, pero luego estas taras se realizaron en colab con sklearn
4) React y firebase: estas herramientas se utilizaron para almacenar los resultados y mostrarlos con mejor calidad y disponibilidad
5) Slides.com: utilizado para la presentacion del trabajo https://slides.com/agustinrodriguez-2/base-de-datos-masias/fullscreen

Fórmulas

Bollinger Bands:

$$BOLU=MA(TP,n)+m*\sigma[TP,n]$$

$BOLD=MA(TP,n)-m*\sigma[TP,n]$
Donde
BOLU=Bollinger Band Superior
BOLD=Bollinger Band Inferior
MA=Moving average
TP =(High+Low+Close)÷3
n=Número de días en un periodo
m= Número de desvíos $\sigma[TP,n]$=Desvío estándar sobre los *n* periodos de TP

RSI

RSI = 100 – [100 / (1 + (Promedio de los precios a la alza / Promedio de los precios a la baja))]

SMA

(N - S) / N

donde:
N = Numeros de dias en un periodo
S = Suma de los precios de cierre en un periodo de tiempo

EMA

$Precio(t)\times k+ EMA(y)\times(1-k)$
donde:
t=Hoy
y=Ayer
N= número de días en EMA
$k=2\div(N+1)$

Z-score

z = (x - u) / s
Donde:
u = media de la muestra
s = desvío estándar de la muestra
x = valor a transformar

Referencias

[1] Amit, Y. & Geman, D. (1997). Shape quantization and recognition with randomized trees. Neural Computation, 9, 1545–1588.
[2] Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.