Práctica Python

Máster en Inteligencia Artificial aplicada a los Mercados Financieros (mIA-X)
fcalle@grupobme.es

1. Trading con bandas de Bollinger. (3pt)

En este ejercicio vamos a tratar de implementar un sistema de trading automático usando las famosas bandas Bollinger. Si quieres conocer más sobre este indicador puedes consultar [1].

1. ☆☆☆ Utilizando la API de IEX vista en clase descarga los precios de cierre del ETF del índice SP500 con ticker SPY. Guarda los datos en un csv para no tener que realizar la descarga cada vez (recuerda que tienes un límite).
2. ★★☆ Calcula las bandas de Bollinger. Recuerda que las bandas Bollinger se calculan como $MA - K\sigma$ y $MA + K\sigma$, donde MA es la media móvil de 30 muestras sobre el precio, σ es la desviación típica móvil de 30 muestras sobre el precio y $K = 2$.
3. ☆☆☆ Realiza una figura del apartado anterior.
4. ★★★ Usando las bandas Bollinger, calcula cuándo comprar y vender el activo. Compramos cuando el precio del activo sale y vuelve a entrar en la banda inferior, y vendemos cuando toca la banda superior. Por simplificación, solo operamos a largo y si estamos comprados no podemos volver a comprar otra vez.
5. ★★☆ Realiza una figura del apartado anterior. Intenta obtener algo parecido a la Figura 1.
6. ★★★ Calcula la rentabilidad media por operación.
7. ★★★ Intenta optimizar los parámetros que quieras del cálculo de las bandas. ¿Qué resultados obtienes?, ¿tiene sentido esta optimización?, ¿crees que los resultados son realistas?. Si usas para esto la librería *multiprocesing* obtendrás puntuación extra.

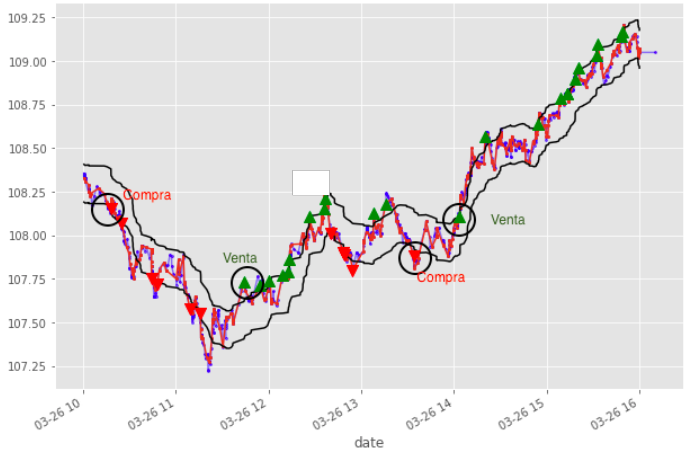


Figura 1: Bandas Bollinger, con los puntos de compra y venta.

2. Montecarlo y Bootstrapping (2 pt)

En este ejercicio vamos a realizar una serie de proyecciones de los retornos que podemos esperar, invirtiendo en el IBEX con dividendos durante un plazo de 5 años. El objetivo final es obtener un análisis que diga al inversor las situaciones media, peor y mejor que se podrían obtener al realizar la inversión.

Realizaremos estas proyecciones de dos maneras: Montecarlo simple y Bootstrapping simple.

Los pasos serán los siguientes:

1. ☆☆☆ Carga los datos de cierre del Ibex con dividendos que están en el fichero: `ibex_div_data_close.csv`.
2. ☆☆☆ Calcula los retornos logarítmicos de la serie.
3. ☆☆☆ Realiza una figura de la distribución de los retornos (puedes usar la función *distplot* de *seaborn*). ¿Qué observas?.
4. ★☆☆ Método I: Simulación de Montecarlo.
   - Calcula la media y la desviación típica de los retornos obtenidos en el apartado 2.
   - Genera un dataframe con 1000 columnas, que serán cada una de las simulaciones y donde las filas se extenderán un plazo de 5 años, desde el último de día de la serie original. Los datos del dataframe se generan de forma aleatoria siguiendo una distribución normal con la media y la desviación típica obtenida anteriormente.
5. ★★☆ Método II: Bootstrapping simple.
   - En el Bootstrapping simple en vez de generar los retornos de forma aleatoria, realizamos un muestreo aleatorio de los retornos de la serie original.
   - Genera el dataframe con las 1000 simulaciones, muestreando aleatoriamente con remplazo los retornos de la serie original.
6. ★☆☆ Para cada método, usando los retornos de una única simulación (una sola columna), realiza una figura de la distribución de los retornos (puedes usar la función `distplot` de `seaborn`). Compara la figura para cada método. ¿Qué concusiones puedes extraer?.
7. ★★☆ Para cada método utilizando los retornos simulados, calcula la evolución temporal de invertir una unidad monetaria en cada una de las simulaciones generadas.
8. ☆☆☆ Para cada método utilizando el resultado del apartado anterior, obtén una figura donde muestres 100 simulaciones. Deberás obtener una figura parecida a la Figura 2.
9. ★☆☆ Para cada método usando el dataframe de simulaciones, nos quedamos para cada día con los retornos que ocupan los percentiles 0.05, 0.5 y 0.95.
10. ★☆☆ Para cada método utilizando el cálculo anterior, realiza una figura parecida a la Figura 3.
11. ★★☆ Realiza una comparación de los dos métodos y extrae unas conclusiones, ¿cuál crees que es el mejor?.

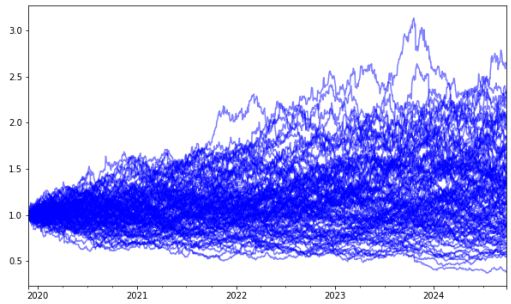


Figura 2: Simulaciones aleatorias.

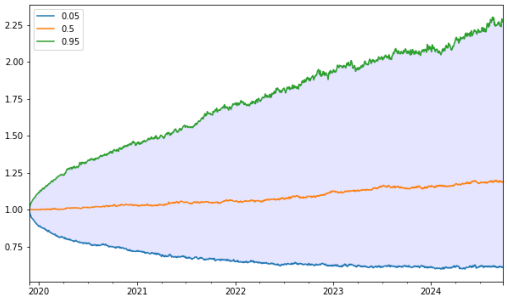


Figura 3: Proyección a 5 años.

3. Tratamiento de series financieras (4 pt)

En este ejercicio vamos a practicar el tratamiento de series temporales financieras. Para ello usaremos datos tick a tick de un ETF del IVE (S&P 500 Value Index).

Calcularemos muestreos sobre estos datos tick a tick que son muy comunes en finanzas, en concreto desarrollaremos las llamadas *ticks bars* y *dollar bars*. Si estas interesado en conocer más detalles sobre esto puedes consultar [2].

Los datos los obtendremos de `http://www.kibot.com/free_historical_data.aspx`, disponibles en el apartado *"free tick intraday data"* subapartado IVE: *"Tick with bid ask data"*.

Realiza los siguientes apartados:

1. ★☆☆ Carga los datos en pandas (el formato de los datos lo puedes ver en `http://www.kibot.com/support.aspx#data_format`) y pon todas las columnas en el tipo correspondiente. Dado que el fichero de datos es muy grande, trabaja solo una parte del mismo.

2. ★☆☆ Calcula las velas (open, high, low, close, vol) horarias, diarias, mensuales y anuales.

3. ★☆☆ Haz un gráfico de velas para cada apartado anterior. Puedes encontrar la función `plot_candle` para hacerlo en el módulo `utils.py` adjunto a esta práctica (es la misma que usamos en clase).

4. ★★★ Haz una función que muestre el precio cada vez que se produce una determinada cantidad de negociaciones, pasada por parámetro. Estas son las denominadas *ticks bars*. Prueba la función con 10000 negociaciones. Realiza una figura.

5. ★★★ Haz una función que muestre el precio cada vez que se negocia una determinada cantidad de dólares, pasada por parámetro. Estas son las denominadas *dollar bars*.Prueba la función con 100.000 dólares. Realiza una figura.

6. ★★★ Calcula los retornos diarios en el caso de las velas diarias, los retornos cada 10.000 negociaciones y los retornos cada 1.000.000$ negociados.

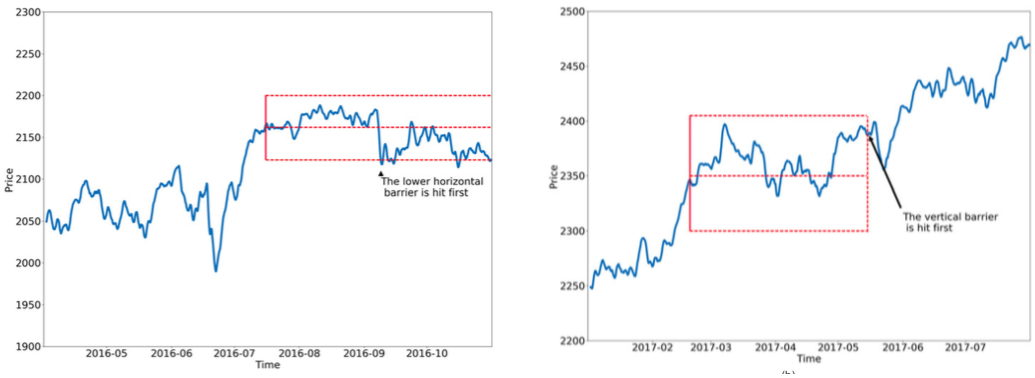
7. ★★☆ Haz una figura que compare la distribución de los tres retornos. ¿Qué diferencias observas?.

4. Triple-Barrier Method (1pt)

En este ejercicio vamos a calcular una serie de etiquetas para cada dato basándonos en el *Triple-Barrier Method* presentado en [2, 3].

1. ☆☆☆ Usando las dollar bars calculadas en el apartado anterior. Programa una función que calcule para cada muestra una etiqueta, que tendrá valor 0, 1 o -1. Estos valores están determinados dependiendo de que barrera se toque antes: 0 para la barrera vertical, 1 para la horizontal superior y -1 para la inferior. Puedes ver dos ejemplos en la Figura 4. La distancia en porcentaje a las barreras verticales y el número de muestras a la barrera horizontal son parámetros de la función.

2. ☆☆☆ Realiza una estudio de la distancia en porcentaje a las barreras verticales y el número de muestras a la horizontal. Busca la manera de que obtengas el número de etiquetas de cada clase (0, -1 y 1) lo más balanceada posible.



(a) La salida para esa muestra es -1 (b) La salida para esa muestra es 0

Figura 4: Distintos ejemplos del *Triple-Barrier Method*. Figuras tomadas de [2]

5. Comentarios adicionales

- Para ambos ejercicios puedes usar notebooks, spyder, vscode o una combinación de los mismos. Si usas un editor guarda todas las figuras que generes en ficheros.
- Se valorarán de manera adicional los siguientes puntos:
  - La calidad y claridad del código.
  - Las eficiencia de la solución obtenida.
  - El uso de funciones y módulos. Se recomienda tener el máximo número de funciones posibles.
  - La documentación de las mismas. Se recomienda usar docstrings para cada función.
  - La escritura del código según PEP8.
  - La calidad de las figuras generadas.

Referencias

[1] J. Bollinger, *Bollinger on Bollinger Bands*. McGraw-Hill Education, 2001.

[2] M. L. de Prado, *Advances in Financial Machine Learning*. Wiley Publishing, 1st ed., 2018.

[3] M. M. López de Prado, *Machine Learning for Asset Managers*. Elements in Quantitative Finance, Cambridge University Press, 2020.