Práctica Python

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

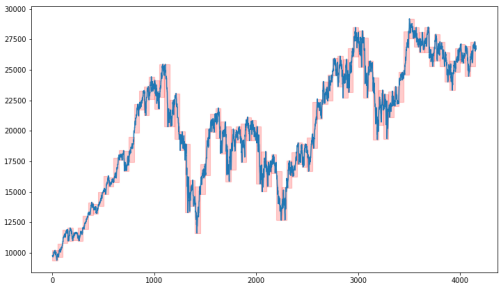
1. Montecarlo y Bootstrapping (5 pt)

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.

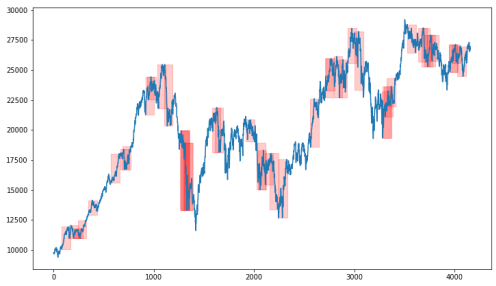
Realizaremos estas proyecciones de cuatro maneras: Montecarlo simple, Bootstrapping simple, Block Bootstrapping y Random Block Bootstrapping.

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. ★★★ Método III: Block Bootstrapping.
   - El método de Block Bootstrapping es una evolución del anterior, donde la serie original se divide en bloques contiguos como se puede observar en la Figura 1a.
   - Genera el dataframe con las 1000 simulaciones muestreando aleatoriamente los bloques. Es decir, obtenemos los retornos necesarios de los bloques elegidos de forma aleatoria con remplazamiento para generar las simulaciones.
   - El tamaño de los bloques se puede configurar. En este caso utilizaremos bloques de un tamaño de 20 días.
7. ★★★ Método IV: Random Block Bootstrapping.



(a) Block Bootstrapping: Bloques a usar para todas las simulaciones.



(b) Random Block Bootstrapping: bloques a usar para una única simulación.

Figura 1: Block Bootstrapping

- El método Random Block Bootstrapping es una evolución del anterior donde los bloques no son contiguos, sino que el tiempo de inicio del bloque es determinado de forma aleatoria como se puede observar en la Figura 1b.
- Genera un dataframe de la misma forma que el método anterior, con la diferencia de que para cada simulación los bloques de donde obtenemos los retornos serán distintos.
- El tamaño de los bloques se puede configurar. En este caso utilizaremos bloques de un tamaño de 20 días.

8. ★☆☆ 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?.

9. ★★☆ Para cada método utilizando los retornos simulados, calcula la evolución temporal de invertir una unidad monetaria en cada una de las simulaciones generadas.

10. ☆☆☆ 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.

11. ★☆☆ 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.

12. ★☆☆ Para cada método utilizando el cálculo anterior, realiza una figura parecida a la Figura 3.

13. ★★☆ Realiza una comparación de los cuatro métodos y extrae unas conclusiones, ¿cuál crees que es el mejor?. En el caso de los dos últimos métodos, ¿qué conclusiones sacas al variar el tamaño de los bloques?.

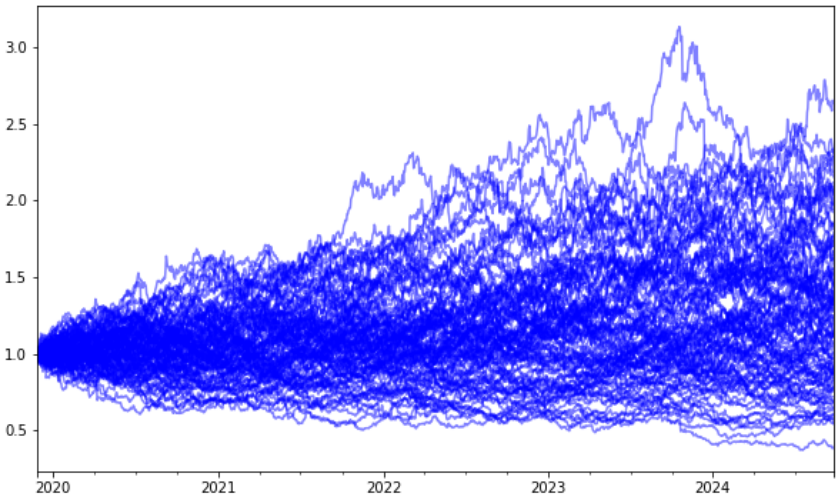


Figura 2: Simulaciones aleatorias.

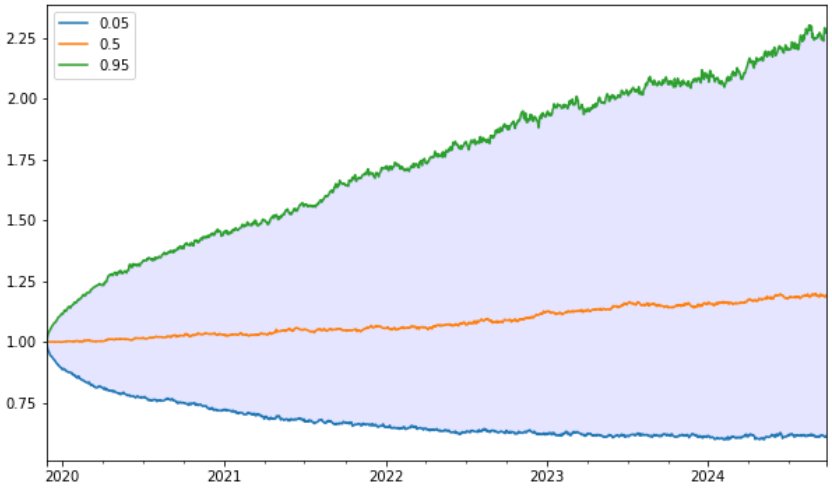


Figura 3: Proyección a 5 años.

2. Tratamiento de series financieras (5 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).

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. Puedes usar la cantidad de datos que quieras para realizar el ejercicio, según la capacidad de tu ordenador.

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.

4. ★★★ Haz una función que muestre el precio cada vez que se produce una determinada cantidad de negociaciones, pasada por parámetro. 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. Prueba la función con 100.000 dólares. Realiza una figura.

6. ★★★ Utilizando las funciones anteriores y las velas diarias, 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 de la distribución de los tres retornos. ¿Qué diferencias observas?

8. ★★☆ Con los precios muestreados cada 100.000 dólares calcula las bandas de bollinger. Recuerda que las bandas bollinger se suelen calcular como $MA - K\sigma$ y $MA + K\sigma$, donde MA es la media móvil de 20 muestras, σ es la desviación típica móvil de 20 muestras y K es dos.

9. ☆☆☆ Realiza una figura del apartado anterior.

10. ★★★ 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. Cambia los parámetros que quieras tanto para el cálculo de las bandas, como para el efectivo negociado.

11. ★★☆ Realiza una figura del apartado anterior. Intenta obtener algo parecido a la Figura 4.

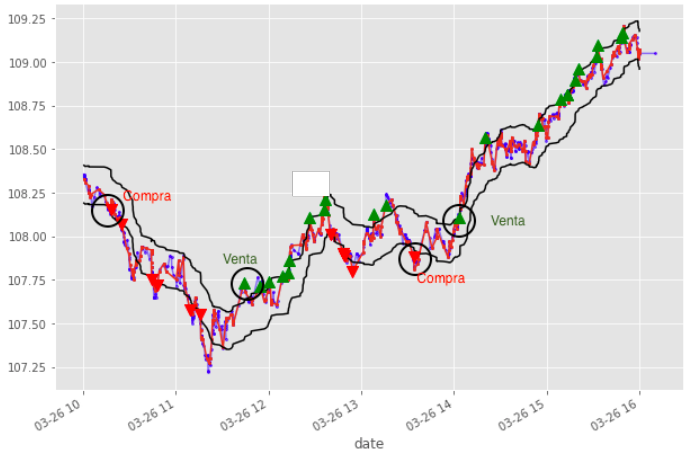


Figura 4: Bandas Bollinger, con los puntos de compra y venta.

Nota: Dado que el fichero de datos es muy grande, trabaja solo una parte del mismo.

3. Comentarios adicionales

- Para ambos ejercicios puedes usar notebooks, spyder, vscode o una combinación de los mismos. Si usas spyder o otro 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.