Práctica Final: Market Making Algorítmico en Polymarket - Pitch a un Venture Capital

Asignatura: Técnicas de Trading Algorítmico

El Escenario: Vendedme vuestro Algoritmo

Queridos alumnos,

Para esta práctica final, vamos a cambiar las reglas del juego. Olvidaos de la entrega tradicional. Vais a presentar vuestro proyecto final como un *pitch* a un inversor.

Imaginad que soy un socio en un Venture Capital. Entiendo mucho de tecnología, he invertido en SaaS, en IA y en infraestructuras cloud, pero sé poco de los mercados financieros y sus tripas. Estoy fascinado con Polymarket y quiero invertir en un equipo que lo domine, pero no encuentro perfiles que combinen tres cosas clave:

- Conocimiento de blockchain
- Algoritmos de trading
- **Sangre joven** que tan bien vende en tiempos de expansión de masa monetaria.

Vuestro objetivo es convencerme. Tenéis que presentarme una PPT/Html vendiéndome vuestro sistema de trading algorítmico. No necesito una presentación excesivamente técnica a nivel matemático; no me aburráis con demostraciones complejas. Quiero que me deis pinceladas de alto nivel sobre lo que hace vuestro algoritmo: cómo se conecta a Polymarket, cómo estima un precio justo (*mid-price*), cómo añade un *spread* y cómo cotiza (ya sea en *streaming* o bajo ciertas condiciones). (15’ de presentación, cada minuto que os paséis resta)

Tenéis que venderme el *hype* de vuestra tecnología. Quiero ver un PPT pulido y, al final, tendréis que defender vuestro proyecto en una ronda de preguntas (10 minutos). Estas preguntas irán más al hueso de la arquitectura de vuestro sistema y, con una probabilidad del 100%, os lanzaré alguna pregunta a pillar para ver cómo reaccionáis bajo presión.

Ah, y una cosa más. Haremos un examen rápido tipo test de 20 minutos justo antes de que empiecen las presentaciones, por lo que pueda pasar más adelante. Sin presión. Estamos en un máster exigente y la asignatura no es fácil. No puedo aceptar un mal proyecto, por eso os he quitado los deberes y prácticas sin sentido. Quiero que me entreguéis un proyecto que os va a aportar mucho más que cualquier ejercicio guiado en un entorno controlado que no se adecúa con la realidad.

Con respecto a la ejecución del proyecto, os recomiendo que creéis una wallet específica para este proyecto, no metáis muchos tokens. Os pido mínimo un día de operativa hasta

que gastéis tokens o al menos una serie de operaciones que luego podéis quitar, pero enseñadme que se han mandado a mercado. Enseñad la patita y escondedla vía algo y hacer screenshots de que habéis mandado bien a mercado si no os atrevéis. Si alguno gana pela id pensando el post en linkedin. El pitch lo tenéis.

Qué Espero de Vosotros en esta Práctica

1. **Github. Aprender a trabajar en equipo en un entorno profesional.** Utilizad **Github**. Este proyecto será una carta de presentación potentísima en futuras entrevistas de trabajo, no es una tarea para cubrir el expediente. Cuando en una empresa escuchéis "GIT" o "repo", sabréis perfectamente de qué va la historia. Hay mil vídeos en YouTube y tenéis muchas IAs a vuestra disposición. Al final, todo se resume en `git clone`, `git pull`, `git checkout`, `git commit` y `git push`. Ved un par de vídeos y aprended haciendo. No guardéis código en vuestro local. Generaros ramas dev e id haciendo push sobre la rama master que me debéis presentar.

2. **Utilizar IDEs Agénticos.** Es hora de que sepáis qué es un agente de código. Descargaos VS Code con la extensión de **GitHub Copilot** o buscad otras herramientas como Cursor. Esto os va a facilitar la tarea una barbaridad, y precisamente por eso os pido un proyecto duro, porque con estas herramientas podéis (y debéis) aspirar a mucho más. **Se viene llorado de casa, toca currar y producir resultados.**

3. **Programar Orientado a Objetos (OOP) con cabeza.** Muchas veces los agentes os van a generar código de esta manera. Os recomiendo que en vuestros *prompts* les pidáis que no hagan *overkilling* y que mantengan el código sencillo. **Diseñad vosotros la estructura de clases**: cómo queréis que sean los "planos" de vuestros objetos, sus atributos y sus funciones. No le digáis a un agente que os programe todo de primeras. Dadle contexto, explicadle lo que queréis que haga. No dejéis que os gobierne, recordad la frase: "*if you don't design your own life plans, chances are you'll fall into someone else's plan. And guess what they have planned for you? Not much.*". Si no, acabaréis con un repo complicadísimo de entender y poco escalable.

4. **Entended la complejidad de un proyecto real y disruptivo.** Quiero que os enfrentéis a los problemas que surgen cuando se intenta innovar. Que entendáis los retos técnicos de un proyecto que se conecta en *streaming* a un mercado real y todo lo que vais a tener que aprender por el camino. El lunes pasado tuve la suerte de comer con el CTO de BBVA, Álvaro Martín, y me lo dejó claro: los grandes profesionales del futuro serán o súper especialistas en un nicho o generalistas que entiendan de todo. Saber de negocio y vender humo ya no sirve para cobrar bien. Tenéis que entender de tecnología, y si no sabéis de algo, tenéis que buscar la forma de aprenderlo o saber utilizarlo.

5. **Construir algo con valor real.** Si sois unos cracks, podéis construir algo de lo que perfectamente podríais vivir como un *side project*, una startup personal o para trabajar en otras empresas que se están montando sobre Polymarket. Esta

experiencia os servirá para venderla como un proyecto complejo y relevante que solucionasteis en vuestras futuras entrevistas. Fijaos en todos los "palabros" que toca este proyecto: **Git, Kalman Filter, Polymarket, Market Making, Trading Algorítmico, Agentes de IA...** Además, tendréis el proyecto en GitHub para compartirlo en LinkedIn y en vuestro CV. Vais a diferenciaros una barbaridad.

Formato del Proyecto y Criterios de Evaluación

Esto es lo que tenéis que construir y lo que voy a valorar en vuestra presentación y código:

1. **Conexión al Mercado:** Debéis conectaros a Polymarket. Podéis transferir tokens a través de un DEX (como Uniswap, usando su API o librerías) o directamente interactuando con la blockchain a través de una wallet como MetaMask. En la presentación, tendréis que explicar claramente qué tipo de participante sois:
    - **Liquidity Providers (Market Makers):** Si vuestro algoritmo cotiza precios de compra y venta de forma continua en *streaming*.
    - **Liquidity Takers:** Si vuestro algoritmo ejecuta órdenes solo cuando se cumplen vuestras condiciones de entrada al mercado.

2. **Clase de Estimación de `FairPrice`:** Tenéis que crear una clase específica para estimar el precio "justo" o teórico de un mercado. Tenéis dos opciones:
    - **Opción A (Recomendada):** Utilizar un **Filtro de Kalman**. Si elegís esta vía, tendréis que explicarme qué habéis entendido del filtro, para qué sirve y cómo lo habéis implementado (¿Single-Asset, Multi-Asset?).
    - **Opción B (Plan B):** Si no lo conseguís, podéis simplemente inferir que el `FairPrice` es el `MidPrice` (el punto medio entre la mejor oferta de compra y de venta). Si hacéis esto, justificad por qué y cuáles son sus limitaciones.

3. **Clase para Calcular el `Spread`:** Necesitaréis otra clase para calcular el *spread* que añadiréis a vuestro `FairPrice` para generar las cotizaciones de compra y venta. Recordad que este *spread* es vuestra ganancia teórica. Podéis usar:
    - Un enfoque estadístico simple: por ejemplo, 2 desviaciones estándar de la estimación del Filtro de Kalman.
    - Un modelo más avanzado como el de **Avellaneda-Stoikov** (si os veis con fuerzas).
    - *Nota: Os introduciré en clase el modelo de Gueant, pero no lo metáis en la práctica que os volveréis locos. Eso es material de TFM.*

4. **`main` y Arquitectura:** Quiero ver un fichero principal (`main.py` o similar) donde se instancien y orquesten todas las clases de forma organizada y limpia. En la presentación, explicadme:
    - Por qué vuestro diseño cumple (o no) los 4 principios de la Programación Orientada a Objetos (Encapsulación, Abstracción, Herencia y Polimorfismo).
    - Qué herramientas habéis utilizado: ¿VS Code, Cursor…?

    - ¿Cómo habéis usado los agentes de IA? ¿ChatGPT, Copilot, otros? ¿Habéis pagado alguna suscripción? Sed transparentes.

5. **Resultados y Análisis Crítico:**
    - **Mercado elegido:** ¿Qué mercado de Polymarket habéis escogido y por qué? (¿Volatilidad, interés personal, liquidez?).
    - **Performance:** ¿Cómo os ha ido? No pasa absolutamente nada si habéis perdido dinero. Una simulación o una ejecución de un solo día puede salir muy mal. Lo que quiero es que me expliquéis **qué ha pasado y por qué creéis que fue mal**.
    - **Próximos Pasos:** ¿Qué mejoras haríais? ¿Cómo evolucionaría vuestro proyecto si tuvierais más tiempo?

¡Mucho ánimo y a por ello! Este proyecto es un reto, pero el aprendizaje y la recompensa merecen la pena.

Propuesta de cómo yo organizaría el equipo:

No se os suele ayudar a potenciar el trabajo en equipo y se nota mucho la diferencia entre equipos corporativos normales y equipos de alto rendimiento (No más de 4 personas). Suelen sacar más trabajo y de forma más eficiente 4 personas bien organizadas con herramientas que optimicen sus funciones.

En este caso el tener Github (Controlador de versiones online) y programando utilizando OOP váis a ser capaces de trabajar en paralelo, que cada uno sepa cuál es su función y qué se espera de él. Tener un rol específico no te exime de saber cómo funcionan todas las partes del proyecto, pero si te permite acotar la expectativa sobre tu trabajo.

Roles y Responsabilidades para el Proyecto

1. Arquitecto de Software y Git Master (El Director de Orquesta)

- **Misión:** Es la persona que tiene el "mapa" completo del proyecto en la cabeza. Su principal responsabilidad es que todas las piezas encajen de forma coherente.
- **Tareas:**
    - Diseñar la estructura inicial de las clases (`main`, `Connection`, `FairPrice`, `SpreadCalculator`) y definir cómo se comunican entre ellas.
    - Configurar el repositorio en **GitHub**, establecer las ramas de trabajo (`develop`, `feature/…`) y gestionar los *pull requests* para integrar el código de los demás.
    - Asegurarse de que el proyecto cumple los principios de la **Programación Orientada a Objetos**.
    - Integrar el trabajo de todos en el `main` y resolver los conflictos que puedan surgir.
    - Liderar la preparación de la PPT, asegurando que la narrativa sea coherente.

2. Especialista en Conectividad y API

- **Misión:** Es el encargado de abrir el camino al mercado. Su foco es conseguir que el algoritmo "hable" con Polymarket, ya sea como *Market Maker* o *Taker*.
- **Tareas:**
    - Investigar y programar la **conexión con Polymarket**, ya sea a través de la API de un DEX o interactuando directamente con la blockchain vía MetaMask/ethers.js.
    - Gestionar la wallet, las claves y la firma de transacciones.
    - Crear la clase que se encarga de recibir los datos del mercado en *streaming* y de enviar las órdenes.
    - Documentar para el equipo cómo obtener y usar los datos que él provee.

3. Analista Cuantitativo (El Cerebro del Precio)

- **Misión:** Es el corazón matemático del proyecto. Su responsabilidad es desarrollar la lógica para estimar el precio "justo" del activo.
- **Tareas:**
    - Investigar a fondo y programar la clase de estimación de **`FairPrice`**.
    - Implementar el **Filtro de Kalman** (Opción A). Si es muy complejo, desarrollar y justificar el modelo de `MidPrice` (Opción B).
    - Hacer pruebas para asegurarse de que la estimación del precio es lógica y estable.
    - Preparar una explicación clara y de alto nivel sobre el modelo elegido para la presentación.

4. Ingeniero de Estrategia y Riesgo (El Gestor de Beneficios)

- **Misión:** Convertir el precio justo en una estrategia de trading rentable y controlada. Decide "cuánto" se quiere ganar en cada operación y gestiona el riesgo.
- **Tareas:**
    - Diseñar y programar la clase para **Calcular el `Spread`**.
    - Utilizar la salida del `FairPrice` del Analista Cuantitativo como input para su modelo (ej. usando la desviación estándar de la estimación de Kalman).
    - Si el equipo es ambicioso, investigar y aplicar modelos como el de **Avellaneda-Stoikov**.
    - Analizar el **rendimiento del algoritmo** (`Performance`), preparar los resultados y la justificación de por qué fue bien o mal.