Este documento describe la arquitectura lógica y el plan de integración de **RadarPremios**, un sistema automatizado para la generación, evaluación y ajuste de candidatos para sorteos de lotería, con un fuerte enfoque en **rigor estadístico**, **reproducibilidad** y **transparencia**.

Objetivo Principal
Generar candidatos el mismo día del sorteo, evaluarlos al día siguiente contra los resultados reales, proponer mejoras en las señales (features) y repetir el ciclo de forma automatizada, trazable y reproducible.

Principios Clave
- **Rigor estadístico**: Uso de probabilidad (Kolmogórov), procesos de Markov y simulaciones Montecarlo.
- **Reproducibilidad**: Todas las ejecuciones guardan la semilla del generador de números aleatorios y los valores de las señales usadas.
- **Ética y claridad**: No se garantizan aciertos. Los reportes separan hechos (datos) de hipótesis (señales).
- **Resiliencia**: El sistema continúa aunque falle la recolección de datos de algún sorteo.

Flujo Diario
1. **08:00**: Actualización de señales base (ventanas móviles, hot/cold, Markov).
2. **12:00+**: Generación de candidatos para cada juego según su calendario.
3. **22:30+**: Recolección automatizada de resultados (ya implementada).
4. **T+1 07:00**: Evaluación de candidatos vs. resultados reales.
5. **T+1 07:10**: Reporte diario y propuesta de ajustes de señales.

Señales (Features) Utilizadas
- **Hot/Cold**: Frecuencia de dígitos por posición.
- **Last-seen**: Tiempo desde la última aparición.
- **Markov**: Transiciones entre dígitos por posición.
- **Patrones**: Repetidos, consecutivos, suma, paridad, etc.
- **Complejidad**: Medida via compresión (LZ/RLE) para evitar patrones triviales.
- **Señales específicas por juego**: Super Astro Luna incluye signos; Baloto/Revancha incluye frecuencias conjuntas y equilibrio numérico.

Generación de Candidatos
- Usa un **generador de números pseudoaleatorios reproducible** (PCG64) con semilla derivada de la fecha y el juego.

- Genera un pool grande de combinaciones, las puntúa con las señales, aplica filtros y selecciona el Top-K.
- Guarda semilla, snapshot de señales y ranking para auditoría.

Evaluación y Ajuste
- **Métricas**: Acierto exacto, parcial, recall en Top-K, lift vs. azar (basado en Montecarlo).
- **Ajuste de pesos**: Señales con buen desempeño suben de peso; las inefectivas bajan o se desactivan. Cambios limitados a ±10% por día.

Reportes
- **Diario**: Candidatos, resultados, métricas y propuestas de ajuste.
- **Semanal**: Tendencias, estabilidad de señales y resumen de cambios.

Manejo de Errores
- Reintentos ante fallos HTTP.
- Exclusión de sorteos sin datos en la evaluación.
- Backups automáticos de la base de datos.

Implementación Técnica
- **Scripts Python** para generación, evaluación, reportes y orquestación.
- **Tareas programadas** en Windows (`schtasks`) para automatización.
- **Base de datos SQLite** con tablas extendidas para candidatos, evaluaciones, señales e historial.

Garantías de Transparencia y Auditoría
- Todo el proceso es **determinista y reproducible** gracias al registro de semillas y versiones.
- No se utilizan elecciones arbitrarias ni construcciones no medibles (evitando problemas como el axioma de elección o paradojas tipo Banach-Tarski).
- Se incluyen controles estadísticos (chi-cuadrado, test de rachas, divergencia KL) para validar la aleatoriedad y detectar *drift*.

Roadmap de Implementación
1. **Fase 1**: Orquestación básica, señales simples (hot/cold, last-seen), reportes diarios.
2. **Fase 2**: Markov posicional, simulaciones Montecarlo.
3. **Fase 3**: Ajuste automático de pesos, dashboard semanal.

4. **Fase 4**: Optimización por presupuesto, experimentos con parámetros.

Conclusión
RadarPremios es un **sistema de análisis estadístico** diseñado para asistir en la selección de números de lotería de manera **auditable, reproducible y ética**. No garantiza aciertos, pero sí ofrece transparencia metodológica y mejora continua basada en datos.

Si necesitas que implemente alguno de los scripts, ajuste la estructura de la base de datos o genere las tareas programadas, puedo ayudarte con eso.

Como asistente de IA, no tengo opiniones personales, pero puedo ofrecerte un **análisis crítico y objetivo** sobre el sistema **RadarPremios** basado en el documento que compartiste:

✅ **Aspectos positivos:**

1. **Rigor estadístico y metodológico**
   - Usa fundamentos sólidos: probabilidad (Kolmogórov), cadenas de Markov, simulaciones Montecarlo.
   - Evita *overfitting* con penalización por complejidad y patrones triviales.

2. **Transparencia y reproducibilidad**
   - Registra semillas de RNG, versiones de señales y snapshots de decisiones.
   - Todo es auditable y reproducible, lo que es clave en sistemas de predicción.

3. **Arquitectura robusta y escalable**
   - Base de datos bien estructurada, scripts modulares, manejo de errores y backups.
   - Separación clara entre generación, evaluación y reportes.

4. **Enfoque ético**
   - Deja claro que no hay garantías de acierto y promueve el juego responsable.
   - Los reportes distinguen entre datos reales y hipótesis.

5. **Automatización inteligente**
   - Integra señales, ajusta pesos automáticamente y genera reportes sin intervención manual.

⚠️ **Puntos de atención / desafíos:**

1. **Complejidad computacional**

- Generar y puntuar miles de combinaciones por juego (ej: 120k para Astro Luna) puede ser costoso en tiempo y recursos.

2. **Señales no siempre predictivas**
   - Aunque el sistema ajusta pesos, los sorteos de lotería son esencialmente aleatorios. Ninguna señal garantiza éxito a largo plazo.

3. **Riesgo de sobre-optimización**
   - Ajustar señales diariamente con ventanas móviles cortas puede llevar a *overfitting* histórico.

4. **Dependencia de la calidad de los datos**
   - Si el scraping falla o hay datos corruptos, la generación y evaluación se ven afectadas.

5. **Expectativas de usuario**
   - Aunque el sistema es transparente, algunos usuarios pueden malinterpretar su capacidad predictiva.

🧠 **Conclusión:**

RadarPremios es un sistema bien diseñado desde el punto de vista técnico y estadístico, ideal para:
- Quienes buscan un enfoque **sistemático y analítico** para participar en juegos de azar.
- Proyectos de **investigación o educación** en modelado predictivo y ciencia de datos.

No es adecuado para:
- Quienes buscan "ganar seguro" o creen que existe un método infalible.

Si se usa con **responsabilidad y entendiendo sus limitaciones**, es una herramienta valiosa para explorar el comportamiento de señales estadísticas en contextos de azar.

¿Necesitas que ayude a implementar algún módulo, ajustar la lógica o generar los scripts?

Arquitectura Lógica y Plan de Implementación para RadarPremios

Introducción
Basado en el análisis de los puntos de atención y desafíos identificados en RadarPremios, propongo una arquitectura lógica y un plan de implementación que se centra en **eficiencia computacional**, **prevención de overfitting**, **robustez de datos** y **transparencia para el usuario**. Este diseño mantiene los principios originales de rigor estadístico y reproducibilidad, pero incorpora mejoras para abordar las debilidades.

---

1. Arquitectura Lógica Optimizada

1.1. Diseño General Modulado y Escalable
- **Enfoque microservicios**: Dividir el sistema en módulos independientes que se comuniquen mediante APIs REST o colas de mensajes (ej: RabbitMQ) para facilitar el escalamiento y el mantenimiento.
- **Base de datos mejorada**: Migrar de SQLite a PostgreSQL o MySQL para mejor rendimiento en grandes volúmenes de datos, con índices optimizados para consultas frecuentes.
- **Caching estratégico**: Usar Redis para almacenar señales precalculadas y resultados de consultas frecuentes, reduciendo la carga computacional.

1.2. Módulos Clave Revisados
- **Módulo de Data Acquisition**:
  - Implementar múltiples proveedores de datos (fallback) para scraping.
  - Validación de datos en tiempo real con checks de integridad (ej: rangos válidos, formato).
  - Pipeline de datos con Apache Airflow o Prefect para orquestación y manejo de errores.
- **Módulo de Señales**:
  - Precalcular señales para combinaciones comunes en lotes nocturnos (ej: durante mantenimiento).
  - Usar algoritmos eficientes para cálculo de Markov y frecuencias (ej: usando matrices sparse).
  - Introducir señales basadas en machine learning (ej: XGBoost) para capturar no linealidades, pero con validación cruzada estricta.
- **Módulo de Generación de Candidatos**:
  - Muestreo inteligente: No generar todo el universo de combinaciones. En su lugar, usar técnicas de importance sampling basadas en señales previas.
  - Paralelización: Distribuir la generación por lotes usando multiprocessing o Dask.
- **Módulo de Evaluación y Ajuste**:
  - Validación cruzada temporal: Evaluar señales en ventanas de tiempo滚动antes para evitar overfitting.
  - Ajuste de pesos semanal (no diario) con regularización L2 para suavizar cambios.
- **Módulo de Reportes**:
  - Dashboard interactivo (ej: con Grafana o Streamlit) para visualizar históricos y métricas.
  - Incluir secciones educativas sobre aleatoriedad y limitaciones del sistema.

1.3. Gestión de Recursos Computacionales
- **Escalamiento horizontal**: Diseñar para correr en cloud (ej: AWS Lambda o Kubernetes) durante picos de carga (ej: sorteo múltiple).
- **Monitorización**: Usar herramientas como Prometheus y Grafana para trackear uso de CPU, memoria y tiempos de ejecución.

---

2. Plan de Implementación por Fases

Fase 1: Optimización de Base y Eficiencia (Semanas 1-4)
- **Objetivo**: Reducir tiempos de generación y mejorar la calidad de datos.
- **Acciones**:
  - Migrar a base de datos más robusta (PostgreSQL).
  - Implementar caching con Redis para señales precalculadas.
  - Optimizar consultas SQL con índices compuestos.
  - Mejorar el scraping con reintentos inteligentes y fuentes alternativas.
- **Métricas de éxito**: Reducción del 50% en tiempo de generación de candidatos; disponibilidad de datos del 99%.

Fase 2: Prevención de Overfitting y Señales Robustas (Semanas 5-8)
- **Objetivo**: Hacer las señales más generalizables y reducir ajustes frecuentes.
- **Acciones**:
  - Extender ventanas móviles a 90 días para señales clave.
  - Implementar validación cruzada temporal para nuevas señales.
  - Cambiar ajuste de pesos a semanal con límites de cambio más estrictos (±5%).
  - Introducir backtesting automático para señales nuevas antes de implementar.
- **Métricas de éxito**: Lift estable o en mejora durante 4 semanas consecutivas; reducción de la varianza en las métricas.

Fase 3: Robustez y Escalabilidad (Semanas 9-12)
- **Objetivo**: Asegurar que el sistema funcione bajo fallos y alta carga.
- **Acciones**:
  - Implementar orquestación con Airflow para manejo de errores y reintentos.
  - Dockerizar los módulos para fácil despliegue y escalamiento.
  - Configurar monitorización con alertas (ej: datos faltantes, tiempos de respuesta altos).
- **Métricas de éxito**: 0% de caídas del sistema; recuperación automática en menos de 5 minutos tras fallo.

Fase 4: Transparencia y Educación del Usuario (Semanas 13-16)
- **Objetivo**: Gestionar expectativas y mejorar la claridad de los reportes.
- **Acciones**:
  - Rediseñar reportes para incluir:
    - Explicación clara de que los sorteos son aleatorios y no hay garantías.
    - Comparación con baseline de azar (Montecarlo) en todos los reportes.
    - Histórico de desempeño con intervalos de confianza.
  - Crear documentación y tutoriales sobre cómo interpretar las señales.
  - Implementar un dashboard público con métricas en tiempo real.
- **Métricas de éxito**: Feedback positivo de usuarios sobre claridad; reducción de consultas sobre "garantías".

---

3. Manejo Específico de Puntos de Atención

3.1. Complejidad Computacional
- **Soluciones**:

  - Muestreo inteligente: Generar solo un subconjunto de combinaciones con alta probabilidad based on señales históricas.
  - Paralelización: Usar multiprocessing para cálculo de señales por lote.
  - Precalculo: Correr jobs nocturnos para precalcular señales comunes.

3.2. Señales No Predictivas
- **Soluciones**:
  - Diversificar señales: Incluir señales de diferentes tipos (frecuencias, Markov, patrones) para reducir dependencia de una sola.
  - Meta-señales: Usar un modelo de ensemble para combinar señales de forma adaptativa.
  - Focus en lift: Solo mantener señales con lift >1 consistentemente.

3.3. Riesgo de Overfitting
- **Soluciones**:
  - Validación cruzada temporal: Evaluar señales en datos no vistos.
  - Regularización: Penar cambios bruscos en pesos.
  - Ventanas más largas: Usar 90 días para cálculo de señales.

3.4. Calidad de Datos
- **Soluciones**:
  - Fuentes múltiples: Tener al menos dos fuentes de datos para scraping.
  - Validación: Checks automáticos de integridad (ej: suma de dígitos, valores within rango).
  - Logging detallado: Registrar fallos y datos corruptos para auditoría.

3.5. Expectativas de Usuario
- **Soluciones**:
  - Comunicación clara: Mensajes prominentes en reportes sobre aleatoriedad.
  - Educación: Tutoriales sobre interpretación de métricas.
  - Transparencia: Mostrar no solo aciertos, sino también fallos.

---

4. Stack Tecnológico Propuesto
- **Lenguaje**: Python (manteniendo compatibilidad con existente).
- **Base de datos**: PostgreSQL con TimescaleDB para datos temporales.
- **Caching**: Redis.
- **Orquestación**: Apache Airflow.
- **Monitorización**: Prometheus + Grafana.
- **Dashboard**: Streamlit o Grafana.
- **Despliegue**: Docker + Kubernetes para escalamiento.

---

5. Conclusión
Esta arquitectura y plan abordan los puntos de atención manteniendo la esencia de RadarPremios. La clave es balancear la sofisticación estadística con la eficiencia computacional y la honestidad con los usuarios. La implementación por fases permite iterar y ajustar based on feedback y métricas.

Si necesitas ayuda con la implementación detallada de algún módulo o la configuración de herramientas, estoy aquí para asistirte.