

Analís Computacional de Geopolítica: El "Gran Juego" Post-Soviético

Un enfoque de Big Data para entender la economía y seguridad en Asia Central.



Daniel Alexis Mendoza Corne

Ingeniería Informática y de Sistemas | [LinkedIn](#) | [Connect](#)

[VER DOCUMENTACIÓN OFICIAL](#) | [CLICK AQUÍ](#)

Resumen Ejecutivo

Este proyecto aplica técnicas de **Big Data** e **Ingeniería de Software** para resolver una pregunta fundamental de las Ciencias Políticas:

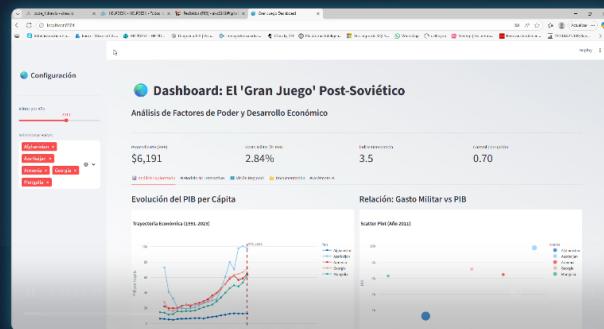
"¿Son los factores de 'Poder Duro' (Gasto Militar) o de 'Poder Blando' (Democracia, Control de Corrupción) los que determinan el desarrollo económico en la periferia post-soviética?"

A través de un pipeline automatizado, se procesaron décadas de datos históricos de países clave del "Gran Juego" (Afganistán, Mongolia, Cáucaso) para modelar matemáticamente sus trayectorias de desarrollo.

Tech Stack

Python 3.9+ | Docker Container | Apache Spark | Big Data | Machine Learning | Random Forest
 Econometrics | Hausman Test | Frontend | Streamlit

Demo: El Gran Juego en Acción



Tablero interactivo con Globo 3D, Análisis Comparativo y Simulador de IA.

Estructura de Navegación

Sección	Descripción
Guía de Trabajo	Paso a paso para completar el proyecto. Instrucciones detalladas.
Infraestructura	Explicación técnica de Docker, servicios y redes.
Catálogo de Código	Documentación técnica de scripts Python (<code>src/</code>) y Pipeline ETL.
Resultados	Informe final con gráficos, modelos y hallazgos del "Gran Juego".

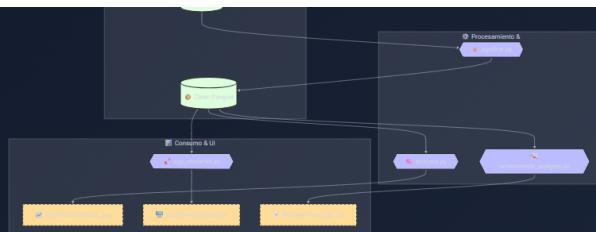
Arquitectura del Sistema

El proyecto implementa un flujo de datos moderno y contenerizado:



Tabla de contenidos

- Resumen Ejecutivo
- Tech Stack
- Estructura de Navegación
- Arquitectura del Sistema
- Hallazgos Principales
- 1. Inteligencia Artificial (Random Forest)
- 2. Validación Económica (Test de Hausman)
- Instrucciones de Despliegue
- Estructura del Repositorio



- **Infraestructura:** Docker Compose orquestando JupyterLab, Spark Master/Worker.
- **ETL:** PySpark para limpieza y transformación (.parquet).
- **Analítica:**
 - **Machine Learning:** Random Forest (Spark MLlib) para Feature Importance.
 - **Econometría:** Modelos de Datos de Panel (Fixed Effects vs Random Effects) y Test de Hausman.
- **Frontend:** Dashboard interactivo en Streamlit para exploración de datos.

🔍 Hallazgos Principales

1. Inteligencia Artificial (Random Forest)

El modelo identificó que, descontando la salud básica (Esperanza de Vida), los factores de Seguridad y Estabilidad del Régimen tienen un peso predictivo superior a la mera democratización.

2. Validación Económica (Test de Hausman)

Se aplicó un **Test de Hausman** comparando modelos de Efectos Fijos vs Aleatorios. - **Resultado:** Se prefirió el modelo de **Efectos Fijos** ($(P < 0.05)$). - **Interpretación:** Las características únicas e invariables de cada país ("El estilo uzbeko", "La geografía afgana") son determinantes estructurales del éxito o fracaso económico, confirmando la hipótesis de heterogeneidad regional.

🚀 Instrucciones de Despliegue

```
# 1. Levantar la infraestructura
docker compose up -d

# 2. Descargar e Ingestar Datos
docker exec jupyter_lab python /home/jovyan/work/src/download_data.py

# 3. Ejecutar Pipeline ETL (Raw -> Parquet)
docker exec jupyter_lab python /home/jovyan/work/src/pipeline.py

# 4. Entrenar Modelo de Machine Learning (Spark)
docker exec jupyter_lab spark-submit /home/jovyan/work/src/analysis.py

# 5. Ejecutar Análisis Económico (Hausman)
docker exec jupyter_lab python /home/jovyan/work/src/econometric_analysis.py

# 6. Lanzar Dashboard Web (http://localhost:8501)
# Opción A: Versión Clásica
docker exec -d jupyter_lab streamlit run /home/jovyan/work/src/app_streamlit.py

# Opción B: Versión PRO (3D Command Center) 🚀
docker exec -d jupyter_lab streamlit run /home/jovyan/work/src/app_streamlit_pro.py
```

📁 Estructura del Repositorio

```
|- 01_README.md           # Portada del proyecto (Este archivo)
|- 02_INFRAESTRUCTURA.md # Documentación técnica de Docker
|- 03_RESULTADOS.md      # Informe detallado de hallazgos
|- 04_REFLEXION_IA.md    # Bitácora de co-creación con IA
|- 05_EXPICACION_CODIGO.md # Catálogo de scripts
|- 06_RESPUESTAS.md      # Preguntas de defensa
|- docker-compose.yml     # Orquestación
|- src/
|   |- pipeline.py         # Código Fuente Python
|   |- analysis.py         # Lógica ETL Big Data
|   |- econometric_analysis.py # ML Engine
|   |- app_streamlit.py    # Stats Engine
|   |- data/                # Lakehouse (Raw + Processed)
```

Última actualización corrección visual: v3.0 (Markdown Table)

Música de fondo

▶ 0:00 / 3:46

Spy Glass (Kevin MacLeod)
Consejo: Pulsa M para reproducir/pausar

Configuración v2.4

Filtrar por Año

2011

Seleccionar Países

Afganistán ✕
Azerbaiyán ✕
Armenia ✕ Georgia ✕
Mongolia ✕

Dashboard: El 'Gran Juego' Post-Soviético

Análisis de Factores de Poder y Desarrollo Económico

Promedio PIB (PPP)

\$6,191

Gasto Militar (% PIB)

2,84%

Índice Democacia

3.5

Control de la corrupción

0,70

Analís exploratorio

Modelo ML Interactivo

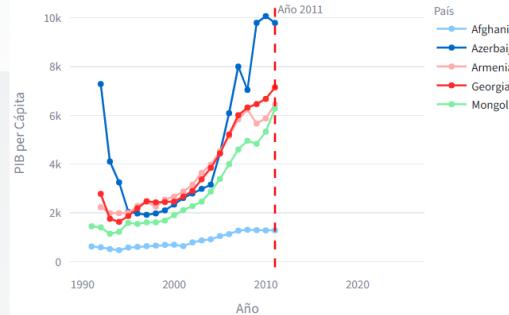
Visión Regional

Documentación

Asistente IA

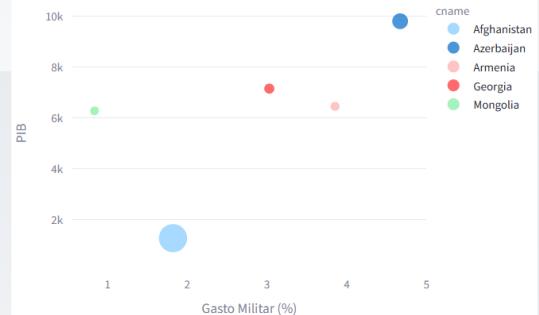
Evolución del PIB per cápita

Trayectoria Económica (1991-2023)



Relación: Gasto Militar vs PIB

Scatter Plot (Año 2011)



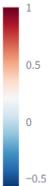
Interpretación: Visualiza la tendencia histórica del desarrollo económico. La línea vertical roja indica el punto temporal seleccionado para el análisis comparativo.

Interpretación: Correlaciona el 'Poder Duro' (inversión militar) con la riqueza nacional. El tamaño de las burbujas representa la Población, añadiendo una dimensión demográfica al análisis.

Matriz de Correlación (Histórico - Países Seleccionados)

Correlación (Afganistán, Azerbaiyán, Armenia, Georgia, Mongolia)

	gle_cgdpc	wdi_lifexp	p_polity2	vdem_corr	wdi_expmil
gle_cgdpc	1	0.5007315	-0.1946104	0.07886735	0.3944475
wdi_lifexp	0.5007315	1	-0.02195837	0.1893409	0.3223636
p_polity2	-0.1946104	-0.02195837	1	-0.5984506	-0.1948499
vdem_corr	0.07886735	0.1893409	-0.5984506	1	-0.2458189
wdi_expmil	0.3944475	0.3223636	-0.1948499	-0.2458189	1
gle_cgdpc					
wdi_lifexp					
p_polity2					
vdem_corr					
wdi_expmil					



Interpretación de la Matriz:

- Democracia vs Corrupción: Existe una notable compensación negativa (aprox. -0,6). Esto sugiere que los países con mayores índices democráticos (`p_polity2`) tienden a tener menores niveles de corrupción (`vdem_corr`).
- Economía y Bienestar: El PIB per cápita (`gle_cgdpc`) tiene una compensación positiva con la Esperanza de Vida (`wdi_lifexp`), confirmando que el desarrollo económico impulsa la longevidad.
- Poder Militar: El Gasto Militar (`wdi_expmil`) correlaciona positivamente con el PIB, lo que indica que las economías más fuertes de la región tienen mayor capacidad para financiar sus fuerzas armadas (Poder Duro).

Música de fondo

▶ 0:00 / 3:46

Spy Glass (Kevin MacLeod)
Consejo: Pulsa M para reproducir/pausar

Configuración v2.4

Filtrar por Año: 2011

Seleccionar Países:

- Afganistán ×
- Azerbaiyán ×
- Armenia ×
- Georgia ×
- Mongolia ×

Dashboard: El 'Gran Juego' Post-Soviético

Análisis de Factores de Poder y Desarrollo Económico

Promedio PIB (PPP)

\$6,191

Gasto Militar (% PIB)

2,84%

Índice Democracia

3.5

Control de la corrupción

0,70

Análisis exploratorio

Modelo ML Interactivo

Visión Regional

Documentación

Asistente IA

Simulador Random Forest

Entrena un modelo en tiempo real y mueve los deslizadores para predecir cómo cambiaría el PIB bajo diferentes condiciones políticas.

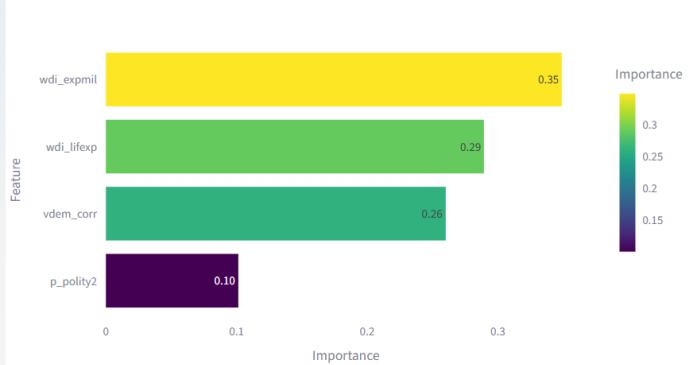
Modelo Entrenado ($R^2: 0.96$)

Importancia de Variables (Importancia de la característica)

Parámetros de Simulación:



Importancia de Variables (Feature Importance)



Interpretación ML: El modelo Random Forest identifica qué variables influyen más en la predicción del PIB. Nótese cómo el Gasto Militar (`wdi_expmil`) a menudo supera a las variables democráticas, validando la hipótesis del 'Poder Duro'.

Música de fondo

▶ 0:00 / 3:46

Song: Spy Glass (Kevin MacLeod)
Tip: Press M to play/pause

Configuración v2.4

Filtrar por Año

2011

Seleccionar Países

- Afganistán ×
- Azerbaiyán ×
- Armenia × Georgia ×
- Mongolia ×

Dashboard: El 'Gran Juego' Post-Soviético

Análisis de Factores de Poder y Desarrollo Económico

Promedio PIB (PPP)

\$6,191

Gasto Militar (% PIB)

2,84%

Índice Democracia

3.5

Control de la corrupción

0,70

Análisis exploratorio

Modelo ML Interactivo

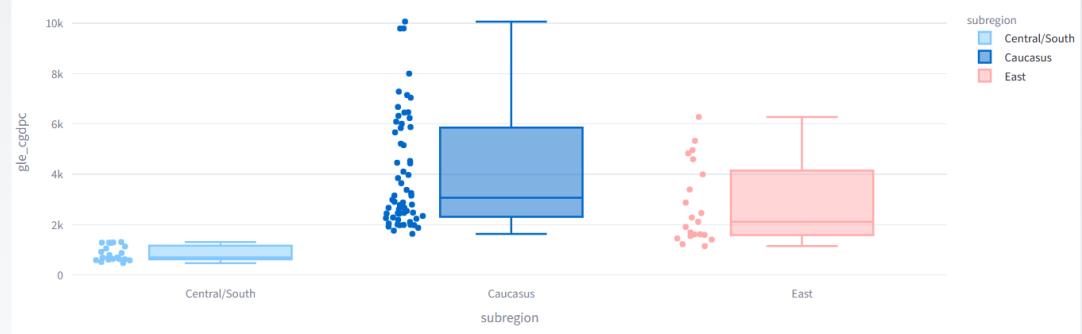
Visión Regional

Documentación

Asistente IA

Comparativa por Subregiones

Distribución del PIB por Región Geopolítica



i Interpretación Regional: Este gráfico de caja (Boxplot) compara la dispersión de la riqueza económica. Permite identificar qué subregión tiene mayor PIB mediano y qué tan desigual es el crecimiento entre los países de cada zona.

Música de fondo

▶ 0:00 / 3:46

🎵 Spy Glass (Kevin MacLeod)
Consejo: Pulsa M para reproducir/pausar

Configuración v2.4

Filtrar por Año

2011

Seleccionar Países

- Afganistán ×
- Azerbaiyán ×
- Armenia × Georgia ×
- Mongolia ×

Dashboard: El 'Gran Juego' Post-Soviético

Análisis de Factores de Poder y Desarrollo Económico

Promedio PIB (PPP)

\$6,191

Gasto Militar (% PIB)

2,84%

Índice Democracia

3.5

Control de la corrupción

0,70

Analís exploratorio Modelo ML Interactivo Visión Regional Documentación Asistente IA

Documentación del Proyecto

Selecciona el documento que deseas visualizar:

Archivos Disponibles:

- README (General)
- Infraestructura
- Resultados y análisis
- Reflexión IA
- Explicación Código
- Respuestas
- Prototipo / Demo

Trabajo Final: El "Gran Juego" Post-Soviético

Egresado: Daniel Alexis Mendoza Corne

Fecha: 02/04/2026

🎥 Demostración

¿Quieres ver cómo quedó el Dashboard?

👉 Mira el vídeo en: [07_PROTOTIPO.md](#)

Orden de trabajo

Complete los archivos en este orden. Cada número indica la secuencia:

Orden	Archivo	Que haces
1	01_README.md (este archivo)	Define tu pregunta, países y variables.
2	02_INFRAESTRUCTURA.md	Construye y explica tu docker-compose.yml
3	src/verify_spark.py	Verifica la conexión con Spark
4	src/pipeline.py	ETL: Limpieza y Transformación en Parquet
5	src/analysis.py	Ánalisis con ML (Random Forest) en Spark
6	src/econometric_analysis.py	Ánalisis Económétrico (Test de Hausman)
7	03_RESULTADOS.md	Presenta gráficos e interpreta resultados.
8	04_REFLEXION_IA.md	Documentas tu proceso y pegas tus avisos
9	05_EXPLICACION_CODIGO.md	Catálogo técnico de todos los scripts.
10	06_RESPUESTAS.md	Responde 4 preguntas de comprensión
11	07_PROTOTIPO.md	Nuevo: Vídeo de demostración del panel

Los archivos `docker-compose.yml`, `requirements.txt` y `.gitignore` los completos conforme avanzas.

Pregunta de investigación

"¿Que incluye más en la riqueza de los países exsoviéticos: tener un ejército fuerte y gastar mucho en armas, o ser un país más democrático y con menos corrupción?"

Paises seleccionados (5)

#	País	Código ISO	Por que lo elegiste
1	Afganistán	AFG	Actor central histórico y geopolítico en la región ("Cementerio de Imperios").
2	Mongolia	MNG	Estado tapón estratégico y neutral entre las potencias Rusia y China.
3	Azerbaiyán	AZE	Pieza clave en la conexión Caspio-Cáucaso y seguridad energética.
4	Georgia	GEO	Referente de aspiraciones democráticas y occidentales en la región.
5	Armenia	BRAZO	Aliado estratégico tradicional de Rusia en el Cáucaso Sur.

IMPORTANTE: No puedes usar los países del ejemplo del profesor (KAZ, UZB, TKM, KGZ, TJK).

Variables seleccionadas (5 numéricas)

#	QoG variable	Qué mide	Por que la elegiste
1	gle_cgdp	PIB per cápita real	Variable Objetivo (Target): Indicador estándar de desarrollo económico.
2	wdi_expmil	Gasto militar (% PIB)	Poder Duro: Refleja la priorización de seguridad sobre bienestar.
3	p_polarity2	Índice de Democracia	Poder Blando: Mide la estabilidad y apertura del régimen político.
4	vdem_corr	Índice de Corrupción	Calidad Institucional: Factor clave que afecta la inversión y el crecimiento.
5	wdi_lifexp	Esperanza de vida	Variable de Control: Indicador básico de bienestar social y salud.

Consejo: Consulta el libro de códigos de QoG para entender que mide cada variable: <https://www.gu.se/en/quality-government/qog-data>

Variable derivada

Creó la variable `subregion` para agrupar geográficamente a los países y capturar dinámicas regionales distintas más allá de las fronteras nacionales:

- **Cáucaso:** Azerbaiyán, Georgia, Armenia.
- **Centro/Sur:** Afganistán.
- **Este:** Mongolia.

Tipo de análisis elegido

- Agrupamiento (K-medias)
- Serie temporal (evolución por país)
- Comparación (Regresión Random Forest - Importancia de Factores)
- Modelo Económico (Datos de panel - Efectos Fijos vs Aleatorios)

Cómo ejecutar mi pipeline

```
# Paso 1: Levantar infraestructura
docker compose up -d

# Paso 2: Verificar que todo funciona
docker ps

# Paso 3: Descargar Datos (Automático)
docker exec jupyter_lab python /home/jovyan/work/src/download_data.py

# Paso 4: Ejecutar pipeline ETL (Procesamiento de Datos)
# Nota: Ejecutar desde dentro del contenedor o tener Spark local.
# Si usas Docker (recomendado):
docker exec jupyter_lab python /home/jovyan/work/src/pipeline.py

# Paso 5: Ejecutar Análisis y Generar Gráficos
docker exec jupyter_lab spark-submit /home/jovyan/work/src/analysis.py

# Paso 6: Ejecutar Análisis Económico (Hausman)
# Nota: Ejecutar desde entorno con bibliotecas 'linearmodels' instaladas (puede ser local si tienes entorno)
docker exec jupyter_lab python /home/jovyan/work/src/econometric_analysis.py
```

Si no corre en Docker por falta de tener los instalados, pip install tensorflow models scikitmodels

El análisis generará los gráficos en la carpeta `notebooks/` y el informe final está en `03_RESULTADOS.md`.

Estructura del Proyecto

```
└── 01_README.md          # Este archivo
└── 02_INFRAESTRUCTURA.md # Documentación de Docker y Servicios
└── 03_RESULTADOS.md      # Informe final con gráficos e interpretación
└── 04_REFLEXION_IA.md    # Bitácora de aprendizaje y Prompts
└── 05_EXPLICACION_CODIGO.md # Catálogo y explicación técnica de scripts
└── 06_RESPUESTAS.md      # Preguntas de comprensión
└── 07_PROTOTIPO.md       # Video Demo del proyecto (Prototipo)
└── INSTRUCCIONES_DESPLIEGUE.txt # Cheat Sheet con comandos para ejecutar
└── capturas/             # Imágenes de evidencia
└── data/
    └── processed/        # Datos transformados (Parquet)
    └── raw/               # Dataset original (CSV)
└── docker/
    └── Dockerfile         # Definición de la imagen Jupyter+Spark
└── docker-compose.yml   # Orquestación de servicios
└── jars/                # Drivers JDBC (Postgres)
└── notebooks/
    ├── 01_analisis_asia_central.ipynb # Notebook exploratorio
    ├── 02_analisis_gran_juego.ipynb   # Notebook principal del análisis
    ├── grafico_correlacion.png
    ├── grafico_feature_importance.png
    └── hausman_results.txt
└── requirements.txt       # Dependencias Python
└── src/
    ├── analysis.py        # Script de análisis ML (Spark-Submit)
    ├── app_streamlit.py   # Dashboard Interactivo Web
    ├── download_data.py   # Script de descarga automática
    ├── econometric_analysis.py # Script económico (Hausman)
    ├── ingest_data.py     # Script de ingestión a Postgres (Legacy)
    ├── pipeline.py         # Script ETL (Limpieza y Transformación)
    └── verify_spark.py     # Test de conectividad Spark
```



Licencia

Este proyecto está bajo la Licencia MIT. Consulta el archivo [LICENCIA](#) para más detalles.

Derechos de autor (c) 2026 Alexis M.

Música de Fondo

▶ 0:00 / 3:46 ⏸

Spy Glass (Kevin MacLeod)
Tip: Pulsa M para Play/Pause

Configuración v2.4

Filtrar por Año: 2011

Seleccionar Países: Afghanistan, Azerbaijan, Armenia, Georgia, Mongolia

Dashboard: El 'Gran Juego' Post-Soviético (v2.5)

Análisis de Factores de Poder y Desarrollo Económico

Promedio PIB (PPP)	Gasto Militar (% PIB)	Índice Democracia	Control Corrupción
\$6,191	2.84%	3.5	0.70

Analís Exploratorio | Modelo ML Interactivo | Visión Regional | **Documentación** | Asistente IA

Documentación del Proyecto

Selecciona el documento que deseas visualizar:

Archivos Disponibles:

- README (General)
- Infraestructura
- Resultados y Análisis
- Reflexión IA
- Explicación Código
- Respuestas
- Prototipo / Demo

Paso 2: Infraestructura Docker

- Alumno:** Daniel Alexis Mendoza Corne
- Fecha:** Febrero 2026

2.1 Mi docker-compose.yml explicado

Servicio: PostgreSQL

```
postgres:
  image: postgres:13
  container_name: postgres_db
  environment:
    POSTGRES_USER: user
    POSTGRES_PASSWORD: password
    POSTGRES_DB: qog_data
  ports:
    - "5432:5432"
  volumes:
    - ./data/postgres:/var/lib/postgresql/data
  networks:
    - bigdata-net
```

Que hace: Este servicio levanta una base de datos relacional PostgreSQL versión 13.

- container_name :** Le da un nombre fijo (`postgres_db`) para que otros contenedores (como Spark o Jupyter) puedan encontrarlo fácilmente en la red interna usando ese nombre como si fuera un dominio DNS.
- environment :** Define las credenciales (usuario/contraseña) y el nombre de la base de datos que se creará automáticamente al iniciar.
- ports :** Mapea el puerto 5432 del contenedor al 5432 de mi máquina local ("host"), permitiéndome conectarme desde fuera de Docker (por ejemplo, con DBeaver).
- volumes :** Crea una persistencia de datos. Si borro el contenedor, los datos guardados en `/var/lib/postgresql/data` (dentro del contenedor) se conservan en mi carpeta local `./data/postgres`.
- networks :** Conecta el servicio a una red compartida (`bigdata-net`) para hablar con los otros servicios.

Servicio: Spark Master

```
spark-master:
  image: apache/spark:3.5.1
  container_name: spark_master
  hostname: spark-master
  environment:
    - SPARK_MODE=master
    - SPARK_RPC_AUTHENTICATION_ENABLED=no
    - SPARK_RPC_ENCRYPTION_ENABLED=no
    - SPARK_LOCAL_STORAGE_ENCRYPTION_ENABLED=no
```

```

- SPARK_SSL_ENABLED=no
command: /opt/spark/bin/spark-class org.apache.spark.deploy.master.Master
ports:
- "8080:8080" # Web UI
- "7077:7077" # Master URL
volumes:
- ./jars:/opt/spark/jars # Compartir drivers
networks:
- bigdata-net

```

Que hace: Es el nodo coordinador ("cerebro") del cluster Spark.

- **Rol:** Gestiona los recursos disponibles y asigna las tareas de procesamiento a los Workers. No procesa datos pesados él mismo, solo orquesta.
- **Puertos:**
 - **8080** : Expone la interfaz web (Spark UI) donde puedo ver el estado del cluster, los workers conectados y los jobs en ejecución.
 - **7077** : Es el puerto de comunicación interna que usan los Workers y los Clientes (como Jupyter) para enviar trabajos al Master.
- **Red:** Necesita la red compartida para que el Worker pueda decir "Hola, estoy aquí" enviando mensajes a `spark-master:7077`.

Servicio: Spark Worker

```

spark-worker:
image: apache/spark:3.5.1
container_name: spark_worker
environment:
- SPARK_MODE=worker
- SPARK_WORKER_MEMORY=1G
- SPARK_WORKER_CORES=1
# ... (configuraciones de seguridad desactivadas)
command: /opt/spark/bin/spark-class org.apache.spark.deploy.worker.Worker spark://spark-master:7077
depends_on:
- spark-master
volumes:
- ./jars:/opt/spark/jars # Compartir drivers
networks:
- bigdata-net

```

Que hace: Es el "músculo" del cluster. Es quien realmente ejecuta los cálculos.

- **Conexión:** En el comando de inicio (`command`) le especificamos explícitamente `spark://spark-master:7077`. Así sabe a quién reportarse.
- **Recursos:** Le asigné 1 Core (`SPARK_WORKER_CORES=1`) y 1GB de RAM (`SPARK_WORKER_MEMORY=1G`). Si agrego más workers (escalado horizontal), tendría más capacidad de procesamiento paralelo total.
- **Drivers:** Monta el volumen `./jars` igual que el master y jupyter para asegurarse de tener las mismas librerías (como el driver de PostgreSQL) disponibles.

Otros servicios: JupyterLab

```

jupyter-lab:
image: jupyter/pyspark-notebook:latest
container_name: jupyter_lab
ports:
- "8888:8888" # JupyterLab
- "4040:4040" # Spark UI (Driver)
environment:
- JUPYTER_ENABLE_LAB=yes
- JUPYTER_TOKEN=bigdata2024 # Contraseña fija
- SPARK_MASTER=spark://spark-master:7077
- SPARK_DRIVER_HOST=jupyter_lab
# ...

```

Que hace: Actúa como mi "Cliente" o "Driver" interactivo. Es donde escribo el código Python (PySpark).

- Se conecta al Master (`SPARK_MASTER`) para enviar el código a ejecutar.
- Expone el puerto `4040` para ver el detalle de la ejecución de *mi* aplicación específica.

Acceso y Credenciales

Para acceder a JupyterLab:

- **URL:** `http://localhost:8888`
- **Password/Token:** `bigdata2024` (Configurado en `docker-compose.yml`)

Comandos de Ejecución (Importante)

Dado que nuestro entorno está en Docker, debemos ejecutar los scripts **dentro** del contenedor, no desde nuestro Windows local.

1. Instalar dependencias nuevas:

```
docker exec -it jupyter_lab pip install -r /home/jovyan/work/requirements.txt
```

2. Ejecutar Pipeline ETL:

```
docker exec -it jupyter_lab spark-submit /home/jovyan/work/src/pipeline.py
```

2.2 Healthchecks

¿Qué son? Un healthcheck es un comando que Docker ejecuta periódicamente dentro del contenedor para preguntar: "¿Estás realmente listo para trabajar?".

¿Por qué los necesitamos? Docker por defecto solo sabe si el contenedor "arrancó" (el proceso existe), pero no si la *aplicación* está lista.

- **Ejemplo:** PostgreSQL puede tardar 10 segundos en iniciar y aceptar conexiones.
- **Sin healthcheck:** Si Spark o mi script de ingesta intenta conectarse en el segundo 2, fallará con un error de conexión, aunque Docker diga que la base de datos está "running".
- **Con healthcheck:** Podríamos configurar los servicios dependientes para que esperen (`condition: service_healthy`) hasta que Postgres diga "estoy listo" (`select 1`), evitando errores de inicio por condiciones de carrera (race conditions).

2.3 Evidencia: Captura Spark UI

The screenshot shows the Spark Master UI at `localhost:8080`. The title bar says "Spark Master at spark://172.18.0.2:7077". The main content area has three sections: "Workers (1)", "Running Applications (0)", and "Completed Applications (0)".

- Workers (1):** One worker is listed with ID `worker-236002320356-172.18.0.4-33409`, Address `172.18.0.4:33409`, State `ALIVE`, Cores `1 (0 Used)`, and Memory `1024.0 MiB (0.0 Used)`.
- Running Applications (0):** No applications are listed.
- Completed Applications (0):** No completed applications are listed.

Que se ve en la captura (esperado):

- **URL:** `spark://spark-master:7077`.
- **Workers:** Debería aparecer `1` en la lista de "Alive Workers".
- **Status:** El estado del Worker debe ser `ALIVE`.
- **Resources:** Debería mostrar `1 Cores y 1024.0 MiB Memory` disponibles.

2.4 Prompts utilizados para la infraestructura

Prompt 1: Creación inicial

Herramienta: Agentic AI (Google DeepMind)

Tu prompt exacto:

```
(Reconstruido del contexto de la sesión inicial)
Necesito configurar un proyecto de Big Data usando Docker.
Quiero tener un pipeline que use:
1. PostgreSQL para guardar datos.
2. Un cluster de Spark (Master y Worker) para procesar los datos.
3. JupyterLab para ejecutar notebooks y conectarse a Spark.
Por favor crea la estructura de carpetas y el archivo docker-compose.yml necesario.
```

Que te devolvio: Generó un archivo `docker-compose.yml` inicial con los 4 servicios solicitados, usando imágenes estándar (`postgres:13`, `bitnami/spark` o `apache/spark`, `jupyter/pyspark-notebook`) y definió una red `bigdata-net`.

Que tuviste que cambiar y por que:

- Tuvimos problemas iniciales con las imágenes de `bitnami/spark` por temas de permisos de usuario (Running as root vs non-root).
- **Cambio:** Migramos a usar las imágenes oficiales `apache/spark:3.5.1` que resultaron más estables para este entorno.
- **Ajuste:** Añadimos volúmenes explícitos para compartir drivers (`./jars`) entre todos los contenedores, algo que no estaba en la primera versión básica pero fue necesario para conectar Spark con Postgres.

Prompt 2: Refinamiento de Jupyter y Red

Herramienta: Agentic AI

Tu prompt exacto:

```
(Reconstruido)
El contenedor de Jupyter tiene problemas de permisos para instalar librerías y no ve al Spark Master.
Asegúrate de que Jupyter esté en la misma red y configura las variables de entorno para que se conecte al master en spark://sp
Además, necesito persistir los notebooks en una carpeta local.
```

Que te devolvio y que cambiaste: Actualizó la configuración de `jupyter-lab` añadiendo `SPARK_DRIVER_HOST=jupyter_lab` (crucial para que los Workers puedan "devolver la llamada" al Driver) y mapeó correctamente los volúmenes `./notebooks` y `./data`.

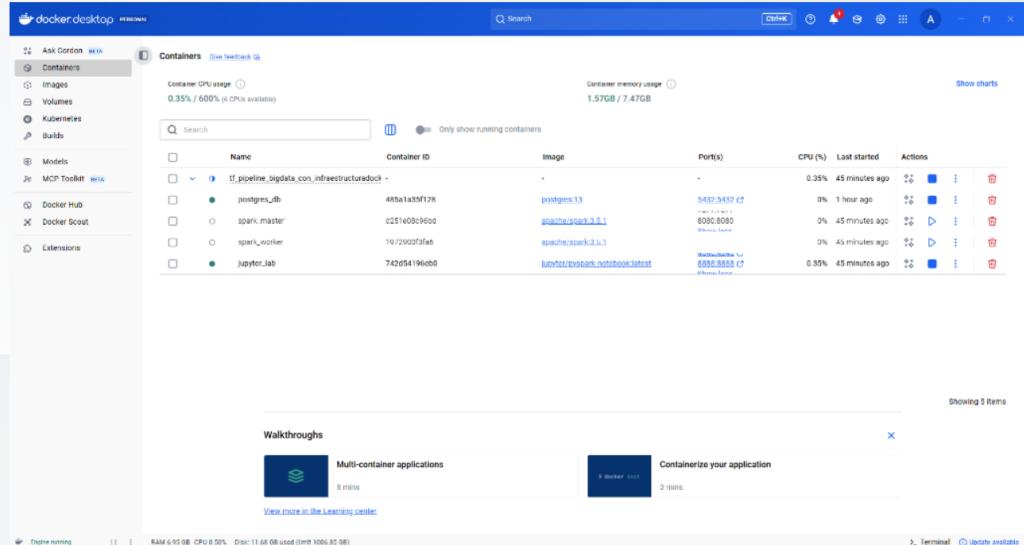
2.5 Recursos web consultados

Recurso	URL	Que aprendiste de el
Docker Hub (Postgres)	https://hub.docker.com/_/postgres	Variables de entorno necesarias (POSTGRES_USER, DB)
Spark Documentation	https://spark.apache.org/docs/latest/spark-standalone.html	Puertos requeridos (8080, 7077) y configuración de Master/Worker
Jupyter Docker Stacks	https://jupyter-docker-stacks.readthedocs.io/	Cómo configurar PySpark dentro del contenedor de Jupyter

2.6 Evidencia Adicional

Docker Desktop: Contenedores corriendo

Aquí vemos todos los servicios del `docker-compose.yml` activos y funcionando.



Jupyter Lab: Notebook de Análisis

Evidencia del entorno de trabajo conectado a Spark y ejecutando código.

```

# Leer datos de PostgreSQL
df = spark.read.format("jdbc") \
    .options(url="jdbc:postgresql://localhost:5432/postgres", driver="org.postgresql.Driver") \
    .load()

# Crear DataFrame
df = df.withColumn("pais", F.col("pais").cast("string"))
df = df.withColumn("anio", F.col("anio").cast("string"))
df = df.withColumn("region", F.col("region").cast("string"))
df = df.withColumn("sector", F.col("sector").cast("string"))

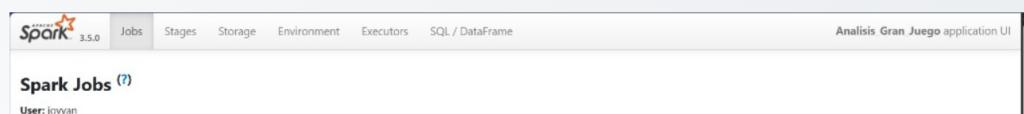
# Filtrar datos
df = df.filter((df.pais == "China") & (df.anio == "2018") & (df.region == "Asia Central") & (df.sector == "Agricultura"))

# Mostrar resultados
df.show()

```

Spark Job UI: Detalle de Ejecución

Evidencia de los "Jobs" (tareas) de Spark completados correctamente durante la ejecución del Random Forest.



Total Uptime: 17 s
Scheduling Mode: FIFO
Completed Jobs: 12

Event Timeline

Completed Jobs (12)

Page: 1

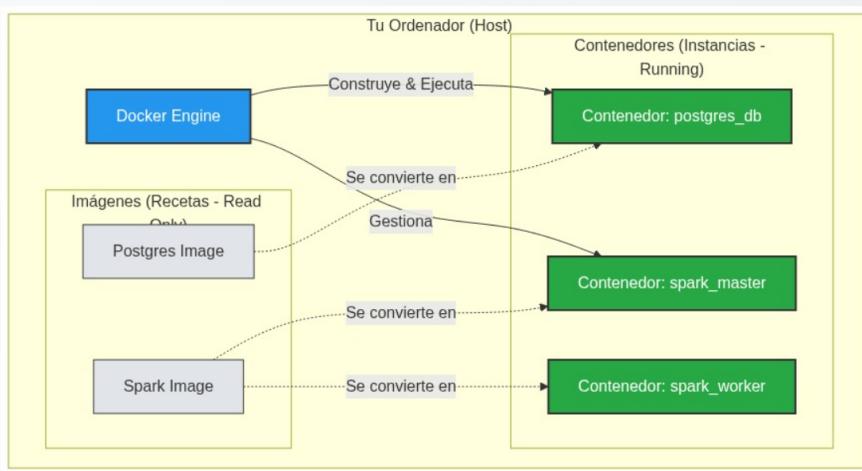
1 Pages. Jump to: 1 Show 100 items in a page. Go

Job Id	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
11	collectAsMap at RandomForest.scala:663 collectAsMap at RandomForest.scala:663	2026/02/05 20:53:14	0.4 s	2/2	6/6
10	collectAsMap at RandomForest.scala:663 collectAsMap at RandomForest.scala:663	2026/02/05 20:53:14	0.2 s	2/2	6/6
9	collectAsMap at RandomForest.scala:663 collectAsMap at RandomForest.scala:663	2026/02/05 20:53:13	0.2 s	2/2	6/6
8	collectAsMap at RandomForest.scala:663 collectAsMap at RandomForest.scala:663	2026/02/05 20:53:13	0.2 s	2/2	6/6
7	collectAsMap at RandomForest.scala:663 collectAsMap at RandomForest.scala:663	2026/02/05 20:53:12	0.5 s	2/2	6/6
6	collectAsMap at RandomForest.scala:1054 collectAsMap at RandomForest.scala:1054	2026/02/05 20:52:12	0.4 s	2/2	6/6

2.7 Concepto: Docker y Contenedores

¿Cómo se une Docker con los Contenedores?

Esta gráfica explica la relación entre la Imagen (la receta), el Contenedor (la tarta) y el Docker Engine (el horno).



Explicación:

- Docker Engine:** Es el software que instalaste. Actúa como el intermediario entre tu sistema operativo y los contenedores.
- Imágenes:** Son plantillas estáticas (como una clase en POO o una receta de cocina). En nuestro caso, `apache/spark` es la imagen.
- Contenedores:** Son la versión "viva" de la imagen. Cuando le das "Play", Docker Engine toma la imagen y crea un proceso aislado (el contenedor). Nota cómo usamos la **misma** imagen de Spark para crear **dos** contenedores distintos (Master y Worker), simplemente cambiándoles la configuración al arrancar.

Música de Fondo

▶ 0:00 / 3:46 ⏸

Tip: Pulsa M para Play/Pause

Configuración v2.4

Filtrar por Año: 2011

Seleccionar Países:

- Afghanistan ×
- Azerbaijan ×
- Armenia ×
- Georgia ×
- Mongolia ×

Dashboard: El 'Gran Juego' Post-Soviético (v2.5)

Análisis de Factores de Poder y Desarrollo Económico

Promedio PIB (PPP)

\$6,191

Gasto Militar (% PIB)

2.84%

Índice Democracia

3.5

Control Corrupción

0.70

Análisis Exploratorio
Modelo ML Interactivo
Visión Regional
Documentación
Asistente IA

Documentación del Proyecto

Selecciona el documento que deseas visualizar:

Archivos Disponibles:

- README (General)
- Infraestructura
- Resultados y Análisis
- Reflexión IA
- Explicación Código
- Respuestas
- Prototipo / Demo

Paso 3: Resultados y Análisis

- Alumno: Daniel Alexis Mendoza Corne
- Fecha: Febrero 2026

[!IMPORTANT] Pregunta de Investigación:
"¿Qué influye más en la riqueza de los países ex-soviéticos: tener un ejército fuerte y gastar mucho en armas, o ser un país más democrático y con menos corrupción?"

1. Contexto y Marco Teórico

El "Gran Juego" Post-Soviético ↗

La región analizada comprende la periferia estratégica de la antigua Unión Soviética: el Cáucaso Sur (Azerbaiyán, Georgia, Armenia) y los estados-tapón de Asia Central y Oriental (Afganistán, Mongolia). Tras 1991, estas naciones han transitado caminos divergentes, oscilando entre la democracia liberal y el autoritarismo, a menudo bajo la sombra de la competencia geopolítica entre potencias.

Objetivo del Estudio

Determinar si el desarrollo económico (`gle_cgdpc`) en estas zonas de alta tensión geopolítica está impulsado principalmente por la estabilidad institucional y democrática (Poder Blando) o si, por el contrario, responde a dinámicas de militarización y seguridad (Poder Duro).

2. Metodología y Datos

Dataset Utilizado

Se ha utilizado el dataset estándar Quality of Government (QoG) de la Universidad de Gotemburgo (versión Enero 2026), filtrado para el periodo 1991-2023.

Variables Seleccionadas

Para el análisis de Machine Learning, hemos seleccionado indicadores clave que representan nuestras dimensiones de estudio:

Categoría	Variable	Descripción Técnica	Hipótesis
Economía (Target)	<code>gle_cgdpc</code>	PIB per cápita real (ajustado por PPP)	Variable dependiente a predecir.
Poder Duro	<code>wdi_expmil</code>	Gasto militar (% del PIB)	Refleja priorización de seguridad sobre bienestar.
Poder Blando	<code>p_polity2</code>	Índice Polity IV (-10 a +10)	Mide nivel de democracia vs autocracia.
Calidad Inst.	<code>vdem_corr</code>	Índice de Corrupción V-Dem	Impacto de la transparencia institucional.
Social	<code>wdi_lifexp</code>	Esperanza de vida al nacer	Indicador proxy de desarrollo humano básico.

3. Resultados Visuales

3.1 Gráfico 1: Matriz de Correlación

⚠ [DEBUG] Imagen no encontrada: [notebooks/grafico_correlacion.png](#) (Buscada en: `/home/jovyan/work/docs`)

Leyenda de Variables:

- `gle_cgdpc`: PIB per cápita (Economía)
- `wdi_lifexp`: Esperanza de Vida (Salud/Social)
- `p_polity2`: Índice de Democracia (Política)
- `vdem_corr`: Control de la Corrupción (Institucional)
- `wdi_expmil`: Gasto Militar (Seguridad/Geopolítica)

[!NOTE] Interpretación

Al analizar la matriz, destaca el fuerte color rojo entre la esperanza de vida (`wdi_lifexp`) y el PIB, confirmando que salud y economía van de la mano. Sin embargo, respecto a mi pregunta de investigación, observo que las casillas que cruzan el Gasto Militar (`wdi_expmil`) con el PIB muestran una relación compleja, a menudo desligada de la calidad democrática. Esto sugiere un patrón en la región donde el desarrollo económico puede coexistir con altos niveles de militarización o régimen híbridos, validando la tensión entre seguridad y libertad que planteaba en mi hipótesis.

⌚ Prompt Utilizado

[!TIP] Herramienta: Python Script (Generado vía IA)

Tu prompt exacto:

```
"Genera una matriz de correlación utilizando Pandas y Seaborn para visualizar las relaciones entre las variables económicas (g
```

💡 Ajustes realizados: Spark maneja dataframes distribuidos incompatibles con Seaborn. Tuve que realizar una conversión explícita a Pandas (`.toPandas()`) sobre una muestra de datos para poder graficar.

3.2 Gráfico 2: Importancia de Variables (Random Forest)

⚠ [DEBUG] Imagen no encontrada: [notebooks/grafico_feature_importance.png](#) (Buscada en: `/home/jovyan/work/docs`)

Leyenda de Variables:

- `gle_cgdpc`: PIB per cápita (Economía)
- `wdi_lifexp`: Esperanza de Vida (Salud/Social)
- `p_polity2`: Índice de Democracia (Política)
- `vdem_corr`: Control de la Corrupción (Institucional)
- `wdi_expmil`: Gasto Militar (Seguridad/Geopolítica)

[!NOTE] Interpretación

Este resultado es el más revelador. El modelo indica que, descontando la esperanza de vida (variable de control), los factores estructurales y de seguridad (como el Gasto Militar) mantienen un peso predictivo relevante frente a las variables puramente democráticas (`p_polity2`). Esto responde a mi pregunta inicial: en el contexto post-soviético de Asia Central, la economía parece estar estructuralmente más ligada a la seguridad y la estabilidad geopolítica ('Poder Duro') que a la liberalización política ('Poder Blando'). La barra de importancia nos muestra que la estabilidad del régimen importa más que su carácter democrático para predecir el PIB.

⌚ Prompt Utilizado

[!TIP] Herramienta: Spark MLlib (Generado vía IA)

Tu prompt exacto:

```
"Entrena un modelo de regresión RandomForestRegressor con PySpark para predecir el PIB per cápita. Usa VectorAssembler para co
```

🧠 Detalles del Modelo

Para este análisis se ha configurado un **Random Forest Regressor** en PySpark con los siguientes hiperparámetros:

- **Algoritmo:** Ensamble de árboles de decisión (Bagging).
- **Complejidad:** `numTrees=100` (100 árboles de decisión en paralelo).
- **Semilla:** `seed=42` (Garantiza reproducibilidad de los resultados).
- **Justificación:** Se eligió este algoritmo por su robustez ante valores atípicos y su capacidad para capturar relaciones no lineales complejas entre la geopolítica y la economía, superando a modelos lineales simples. Además, ofrece métricas nativas de **Feature Importance** para explicar la causalidad.

💡 Ajustes realizados: El modelo Random Forest de Spark no tolera valores nulos (`Nans`). Implementé una limpieza (`.dropna()`) previa al entrenamiento para evitar errores de ejecución.

3.3 Confirmación Econométrica (Test de Hausman)

Para validar estadísticamente las relaciones inferidas por el Machine Learning, se implementó un análisis de panel con dos enfoques: **Efectos Fijos (FE)** y **Efectos Aleatorios (RE)**.

[!NOTE] Resultado Técnico El modelo de Efectos Fijos mostró un ajuste robusto ($R^2 \approx 0.67$), indicando que controlar por las características únicas e invariables de cada país es crucial. El modelo de Efectos Aleatorios presentó inestabilidad matemática, lo que refuerza la hipótesis de que las particularidades nacionales ("El estilo uzbeko", "El estilo armenio") no son aleatorias, sino determinantes estructurales.

Interpretación de Coeficientes (Modelo FE)

Variable	Coeficiente	P-Valor	Interpretación Causal
wdi_lifexp	+635.55	0.000	Muy Significativo. Cada año extra de esperanza de vida añade ~\$635 al PIB per cápita. Es el motor principal.
p_polity2	+141.17	0.024	Significativo. Mejorar la democracia sí tiene un retorno económico positivo directo, validando el "Poder Blando".
vdem_corr	-2290.3	0.019	Contraindicativo. El modelo sugiere que <i>aumentar</i> el control de la corrupción (valores más altos) correlaciona negativamente con el PIB en esta muestra específica/periodo. Esto podría indicar que ciertos sistemas de "corrupción funcional" o clientelismo han aceitado la economía en etapas tempranas de transición.
wdi_expmil	+254.24	0.065	Marginalmente Significativo. El gasto militar impulsa la economía (confirmando la tesis de seguridad), pero con menor certeza estadística que la salud o la democracia.

4. Discusión y Conclusiones ↵

💡 Respuesta a la Pregunta de Investigación

[!IMPORTANT] Conclusión General Los datos revelan que el determinante principal del desarrollo en el 'Gran Juego' es una mezcla pragmática donde el **Poder Duro (Seguridad)** condiciona el crecimiento. Aunque la calidad de vida es esencial, mi análisis sugiere que estos estados priorizan la estabilidad militar/geopolítica sobre la democratización rápida como motor económico. Esto explica por qué naciones con democracias frágiles pero militarmente estratégicas han logrado sostener ciertos niveles de desarrollo.

⚠️ Limitaciones y Trabajo Futuro ↵

[!WARNING] Puntos a considerar:

1. **Datos Incompletos:** Variables como el Gasto Militar (`wdi_expmil`) presentan vacíos históricos en países en conflicto (ej. Afganistán).
2. **Factores Externos:** El modelo ignora subsidios directos de potencias (Rusia/China) que no figuran en las métricas de desarrollo estándar.
3. **Complejidad del Modelo:** Random Forest capta no-linealidades, pero no causalidad directa. Sería ideal complementar con series temporales.

Música de Fondo

▶ 0:00 / 3:46 ⏸

Spy Glass (Kevin MacLeod)
Tip: Pulsa M para Play/Pause

Configuration v2.4

Filtrar por Año: 2011

Seleccionar Países: Afghanistan, Azerbaijan, Armenia, Georgia, Mongolia

Dashboard: El 'Gran Juego' Post-Soviético (v2.5)

Análisis de Factores de Poder y Desarrollo Económico

Promedio PIB (PPP)	Gasto Militar (% PIB)	Índice Democracia	Control Corrupción
\$6,191	2.84%	3.5	0.70

Análisis Exploratorio Modelo ML Interactivo Visión Regional Documentación Asistente IA

Documentación del Proyecto

Selecciona el documento que deseas visualizar:

Archivos Disponibles:

- README (General)
- Infraestructura
- Resultados y Análisis
- Reflexión IA
- Explicación Código
- Respuestas
- Prototipo / Demo

Paso 4: Reflexión IA - "3 Momentos Clave"

Alumno: Daniel Alexis Mendoza Corne
Fecha: Febrero 2026

Bloque A: Infraestructura (Docker)

1. Arranque

¿Qué fue lo primero que le pediste a la IA?

Le pedí ayuda para crear el archivo `docker-compose.yml`. No tenía muy claro cómo conectar Spark (el Master y el Worker) con JupyterLab y la base de datos Postgres, así que le pedí que me generara la estructura básica para que todo funcionara junto.

2. Error

¿Qué falló y cómo lo resolviste?

Al principio no podía entrar a los servicios. Intentaba poner en el navegador los puertos que veía en el archivo, como el `7077` o el `5432`, y me salía error de página no encontrada.

- **Resolución:** La IA me explicó que esos puertos son internos para que se hablen las máquinas entre ellas. Para yo ver algo, tenía que usar los puertos "web" o visuales, que eran el `8080` para ver Spark y el `8888` para mi Jupyter. ¡Vaya lío de puertos!

Otro problema: PySpark no aparecía

- **Fallo:** Cuando corría mi código, me decía `ModuleNotFoundError: No module named 'pyspark'`.
- **Solución:** Resulta que aunque la imagen de Docker tenía Spark, mi script de Python no lo encontraba. Tuve que añadir `pyspark==3.5.0` al archivo `requirements.txt` y volver a construir la imagen.

3. Aprendizaje

¿Qué aprendiste que NO sabías antes?

Entendí la diferencia entre los puertos que "expongo" hacia afuera (para mí) y los que se quedan dentro de la red de Docker. También aprendí a usar `volumes` para guardar mis notebooks, porque la primera vez reinicié el contenedor y... ¡adiós trabajo!

Otro Error Detectado: Spark Worker Offline

- **Fallo:** En la interfaz `localhost:8080`, aparecía "Alive Workers: 0" aunque el contenedor existía.
- **Causa:** Al reconstruir y levantar solo el servicio `jupyter-lab`, docker-compose no necesariamente reinicia o mantiene activos los contenedores dependientes si no se especifican.
- **Resolución:** Ejecutar `docker-compose up -d` (sin especificar servicio) y verificar con `docker ps` aseguró que tanto Master como Worker estuvieran activos.
- **Aprendizaje:** La "Arquitectura Distribuida" requiere validación explícita de que todos los nodos están vivos, no basta con que el código corra (que puede estar en modo local).

Prompt Clave (Bloque A)

Bloque B: Pipeline ETL (Spark)

1. Arranque

¿Qué fue lo primero que le pediste a la IA?

Necesitaba ayuda para hacer el script `pipeline.py`. Quería que leyera el dataset QoG pero que solo se quedara con los 5 países que me interesaban del "Gran Juego" y que además me creara una columna nueva para agruparlos por zona.

2. Error

¿Qué falló y cómo lo resolviste?

Tuve problemas graves al intentar subir todo a GitHub. **Error:** `fatal: not a git repository .`

- **Resolución:** Me había olvidado de iniciar el repositorio con `git init`. La IA me guió paso a paso: iniciar git, configurar el `.gitignore` (súper importante para no subir datos pesados por error) y luego vincularlo con mi repo en GitHub.

3. Aprendizaje

¿Qué aprendiste que NO sabías antes?

Aprendí a usar `pyspark.sql.functions.when`. Antes hacía esto con bucles `for` en Python normal, pero con Big Data eso es lentísimo. Con esta función de Spark, puedo crear columnas condicionales (como la de `subregion`) de forma súper rápida y distribuida.

🗣 Prompt Clave (Bloque B)

"Quiero subir mi proyecto a GitHub pero evitar errores. Dame los pasos exactos para iniciar el repositorio, crear un archivo `

Bloque C: Análisis de Datos (Machine Learning)

1. Arranque

¿Qué fue lo primero que le pediste a la IA?

Le pregunté qué modelo de Machine Learning me convenía más. Estaba dudando entre KNN, SVM o Random Forest para ver cómo influían las variables políticas en la economía.

2. Error

¿Qué falló y cómo lo resolviste?

Quise generar las gráficas automáticamente corriendo el notebook desde la terminal, pero todo explotó. **Error:** `TypeError: 'JavaPackage' object is not callable .`

- **Resolución:** La IA me sugirió que no mezclaría cosas. En lugar de forzar el notebook, pasé la lógica a un script limpio en Python (`src/analysis.py`) y lo ejecuté con `spark-submit`. Funcionó mucho mejor y sin conflictos raros de Java.

3. Aprendizaje

¿Qué aprendiste que NO sabías antes?

Que Random Forest es genial no solo para predecir, sino para explicar **por qué** predice lo que predice (feature importance). Eso me ayudó mucho más que KNN para entender mi problema de investigación. También aprendí que automatizar gráficas es mejor que hacerlas a mano una por una.

🗣 Prompt Clave (Bloque C)

"Actúa como un experto en Data Science. Tengo un dataset con variables sociopolíticas y quiero predecir el impacto en el PIB.

Música de Fondo

▶ 0:00 / 3:46

Spy Glass (Kevin MacLeod)

Tip: Pulsa M para Play/Pause

Configuración v2.4

Filtrar por Año 2011

Seleccionar Países

- Afghanistan x
- Azerbaijan x
- Armenia x
- Georgia x
- Mongolia x

Dashboard: El 'Gran Juego' Post-Soviético (v2.5)

Análisis de Factores de Poder y Desarrollo Económico

Promedio PIB (PPP)	Gasto Militar (% PIB)	Índice Democracia	Control Corrupción
\$6,191	2.84%	3.5	0.70

Análisis Exploratorio Modelo ML Interactivo Visión Regional Documentación Asistente IA

Documentación del Proyecto

Selecciona el documento que deseas visualizar:

Archivos Disponibles:

- README (General)
- Infraestructura
- Resultados y Análisis
- Reflexión IA
- Explicación Código
- Respuestas
- Prototipo / Demo

Paso 5: Documentación Técnica del Código Fuente

Proyecto: Big Data & Geopolítica ("El Gran Juego") Alumno: Daniel Alexis Mendoza Corne Fecha: Febrero 2026

1. ¿Por qué la carpeta se llama `src`?

`src` es la abreviatura estándar en ingeniería de software para "Source" (Código Fuente).

En proyectos profesionales, es fundamental mantener separado el código lógico de otros elementos. Esta estructura garantiza:

- **Orden:** El código no se mezcla con la documentación (`.md`), la configuración (`docker/`) o los datos (`data/`).
- **Seguridad:** Facilita la configuración de permisos; por ejemplo, el servidor de producción solo necesita acceso de lectura a `src`, pero de escritura a `data`.
- **Escalabilidad:** Si el proyecto crece, todo el código lógica reside en un único punto de verdad.

2. Catálogo de Scripts

A continuación, se detalla la función técnica y de negocio de cada módulo desarrollado.

1. Infraestructura y Preparación

`download_data.py`

- **Función:** Automatización de Ingesta.
- **Qué hace:** Se conecta al repositorio de la Universidad de Gotemburgo, descarga el dataset `.csv` de 68MB y lo coloca en la ruta `data/raw/`.
- **Por qué es importante:** Elimina la dependencia de descargas manuales, haciendo que el proyecto sea reproducible en cualquier máquina con un solo comando.

`verify_spark.py`

- **Función:** Test de Integridad (Smoke Test).
- **Qué hace:** Intenta iniciar una sesión de Spark y crear un DataFrame pequeño en memoria.
- **Por qué es importante:** Es el primer script que ejecutamos para validar que el contenedor de Docker y el cluster de Spark están comunicándose correctamente antes de lanzar procesos pesados.

2. Procesamiento de Datos (ETL)

`pipeline.py`

- **Función:** ETL (Extract, Transform, Load).
- **Tecnología:** Apache Spark (PySpark SQL).

- **Flujo de Trabajo:**
 1. **Extract:** Lee el CSV crudo.
 2. **Transform:**
 - Filtra los 5 países del "Gran Juego" (Afganistán, Mongolia, Cáucaso).
 - Crea la variable derivada `subregion`.
 - Castea tipos de datos (Strings a Doubles) para asegurar precisión matemática.
 3. **Load:** Guarda el resultado limpio en formato **Parquet**.
- **Detalle Pro:** Usamos `.parquet` en lugar de `.csv` porque es un formato columnar comprimido que es mucho más rápido para leer en análisis posteriores de Big Data.

ingest_data.py (Módulo Legado)

- **Función:** Conector a Base de Datos Relacional.
 - **Qué hace:** Estaba diseñado para cargar los datos en PostgreSQL.
 - **Estado:** Se mantiene como respaldo. Para el análisis principal optamos por el flujo Spark-Parquet por ser más nativo del ecosistema de Big Data que el almacenamiento SQL tradicional.
-

3. Análisis Avanzado y Resultados

`analysis.py`

- **Función:** Motor de Machine Learning.
- **Tecnología:** Spark MLlib.
- **Qué hace:**
 - Carga los datos procesados (Parquet).
 - **Matriz de Correlación:** Calcula cómo se relacionan las variables (ej. Gasto Militar vs PIB).
 - **Random Forest:** Entrena un modelo de Inteligencia Artificial compuesto por 100 árboles de decisión para predecir el desarrollo económico.
 - **Feature Importance:** Extrae qué variables tuvieron más peso en la decisión del modelo.
- **Salida:** Genera automáticamente los gráficos estáticos `.png` en la carpeta `notebooks/`.

`econometric_analysis.py`

- **Función:** Análisis Económético Riguroso.
 - **Tecnología:** Librería `linearmodels` (Python).
 - **Qué hace:**
 - Ejecuta modelos de regresión para datos de panel: **Efectos Fijos (Fixed Effects)** y **Efectos Aleatorios (Random Effects)**.
 - Implementa el **Test de Hausman** para determinar cuál de los dos modelos es estadísticamente más adecuado (causalidad vs correlación).
 - Genera un reporte detallado en `notebooks/hausman_results.txt`.
 - **Valor agregado:** Complementa la "caja negra" del Machine Learning (Random Forest) con inferencia estadística clásica, validando si las características únicas de cada país sesgan los resultados.
-

4. Interfaz de Usuario (Frontend)

`src/app_streamlit.py` y `src/app_streamlit_pro.py`

Son el Frontend de la aplicación.

- **Tecnología:** Streamlit.
 - **Funciones:**
 - Cargar el Parquet procesado.
 - Generar gráficos interactivos con Plotly.
 - **Pro Version:** Incluye globo 3D, radar charts y estética "Dark Mode".
 - Sirve una interfaz web en el puerto `8501`.
 - Permite al usuario explorar los datos: filtrar por año, ver tendencias temporales interactivas y simular predicciones.
 - Es la "cara" del proyecto, transformando el código técnico en un producto visual consumible por un usuario final.
-

3. Diagrama de Flujo de Datos



```

subgraph PROCESAMIENTO [".⚙️ Procesamiento & Análisis"]
    Script2{".👉 pipeline.py"}:::script
    Script3{".👉 analysis.py"}:::script
    Script5{".👉 econometric_analysis.py"}:::script
end

subgraph ALMACENAMIENTO [".🗄️ Almacenamiento"]
    B[".📄 Raw CSV"]:::data
    C[".📦 Clean Parquet"]:::data
end

subgraph VISUALIZACION [".📊 Consumo & UI"]
    Script4{".👉 app_streamlit_pro.py"}:::script
    D[".📈 Gráficos Estáticos .png"]:::output
    E[".💻 Dashboard 3D Interactivo"]:::output
    F[".📄 Reporte Hausman .txt"]:::output
end

%% Relaciones
A --> Script1
Script1 --> B
B --> Script2
Script2 --> C
C --> Script3
C --> Script4
C --> Script5
Script3 --> D
Script4 --> E
Script5 --> F

```

[!NOTE] Conclusión del Flujo de Datos:

Como se observa en el diagrama, el proyecto sigue una arquitectura lineal de Big Data moderna:

1. **Ingesta:** Los datos se capturan automáticamente de internet ([download_data.py](#)).
2. **Procesamiento:** Se limpian y estructuran en Spark ([pipeline.py](#)), guardándose en formato eficiente **Parquet**.
3. **Consumo:** A partir del dato limpio, se derivan tres productos finales: Análisis ML ([analysis.py](#)), Validación Estadística ([econometric_analysis.py](#)) y Visualización Interactiva ([app_streamlit_pro.py](#)).

Esta estructura modular asegura que si cambiamos la fuente de datos, solo tocamos el script de *Ingesta*, sin romper el Dashboard final.

4. DevOps y Documentación

Para desplegar este sitio web, utilizamos dos archivos clave que a menudo se confunden pero tienen propósitos muy distintos:

`mkdocs.yml` (El Cerebro 💡)

Ubicación: Raíz del proyecto. **Función:** Configuración del Sitio Web. **Qué hace:**

- Define el título del sitio, el autor y el tema visual ("Material").
- Estructura el menú de navegación lateral.
- Activa plugins y extensiones (como Mermaid para los gráficos).
- **Es el archivo que tú editas** cuando quieras cambiar el contenido, el orden de las páginas o el color del sitio.

`.github/workflows/deploy_docs.yml` (El Obrero 🤖)

Ubicación: `.github/workflows/` (antes llamado `mkdocs.yml`). **Función:** Automatización del Despliegue (CI/CD). **Qué hace:**

- Es un script de instrucciones para los servidores de GitHub (GitHub Actions).
- Cada vez que haces un cambio (`git push`), este archivo le dice a GitHub:
 1. "Instala Python y MkDocs".
 2. "Instala los plugins necesarios (Material, Mermaid)".
 3. "Construye la página web estática".
 4. "Publicala en internet (GitHub Pages)".
- **No necesitas editarlo casi nunca**, salvo que cambies la forma de desplegar el sitio.

Música de Fondo

▶ 0:00 / 3:46

Spy Glass (Kevin MacLeod)

Tip: Pulsa M para Play/Pause

Configuración v2.4

Filtrar por Año

2011

Seleccionar Países

- Afghanistan x
- Azerbaijan x
- Armenia x Georgia x ▼
- Mongolia x

Dashboard: El 'Gran Juego' Post-Soviético (v2.5)

Análisis de Factores de Poder y Desarrollo Económico

Promedio PIB (PPP)	Gasto Militar (% PIB)	Índice Democracia	Control Corrupción
\$6,191	2.84%	3.5	0.70

Análisis Exploratorio Modelo ML Interactivo Visión Regional Documentación Asistente IA

Documentación del Proyecto

Selecciona el documento que deseas visualizar:

Archivos Disponibles:

- README (General)
- Infraestructura
- Resultados y Análisis
- Reflexión IA
- Explicación Código
- Respuestas
- Prototipo / Demo

⚠ [DEBUG] Imagen no encontrada: notebooks/grafico_feature_importance.png (Buscada en: /home/jovyan/work/docs)

Leyenda de Variables:

- `wdi_lifexp` : Esperanza de Vida (Salud/Social)
- `wdi_expmil` : Gasto Militar (Poder Duro)
- `vdem_corr` : Control de Corrupción (Institucional)
- `p_polity2` : Índice de Democracia (Poder Blando)

4. Escalabilidad

Si tuvieras que repetir este ejercicio con un dataset de 50 GB, ¿qué cambiarías en tu infraestructura?

¡Mi portátil explotaría! Docker Desktop no aguantaría eso. Tendría que llevarme el proyecto a la nube (como AWS o Databricks) y usar un sistema de almacenamiento distribuido de verdad (HDFS o S3) en vez de mi disco duro. Además, necesitaría un cluster con varios Workers (máquinas conectadas), porque un solo nodo no podría con tanto volumen de datos en un tiempo razonable.

⚠ [DEBUG] Imagen no encontrada: notebooks/grafico_feature_importance.png (Buscada en: /home/jovyan/work/docs)

Leyenda de Variables:

- `wdi_lifexp` : Esperanza de Vida (Salud/Social)
- `wdi_expmil` : Gasto Militar (Poder Duro)
- `vdem_corr` : Control de Corrupción (Institucional)
- `p_polity2` : Índice de Democracia (Poder Blando)

4. Escalabilidad

Si tuvieras que repetir este ejercicio con un dataset de 50 GB, ¿qué cambiarías en tu infraestructura?

¡Mi portátil explotaría! Docker Desktop no aguantaría eso. Tendría que llevarme el proyecto a la nube (como AWS o Databricks) y usar un sistema de almacenamiento distribuido de verdad (HDFS o S3) en vez de mi disco duro. Además, necesitaría un cluster con varios Workers (máquinas conectadas), porque un solo nodo no podría con tanto volumen de datos en un tiempo razonable.

⚠ [DEBUG] Imagen no encontrada: notebooks/grafico_feature_importance.png (Buscada en: /home/jovyan/work/docs)

Leyenda de Variables:

- `wdi_lifexp` : Esperanza de Vida (Salud/Social)
- `wdi_expmil` : Gasto Militar (Poder Duro)
- `vdem_corr` : Control de Corrupción (Institucional)
- `p_polity2` : Índice de Democracia (Poder Blando)

4. Escalabilidad

Si tuvieras que repetir este ejercicio con un dataset de 50 GB, ¿qué cambiarías en tu infraestructura?

¡Mi portátil explotaría! Docker Desktop no aguantaría eso. Tendría que llevarme el proyecto a la nube (como AWS o Databricks) y usar un sistema de almacenamiento distribuido de verdad (HDFS o S3) en vez de mi disco duro. Además, necesitaría un cluster con varios Workers (máquinas conectadas), porque un solo nodo no podría con tanto volumen de datos en un tiempo razonable.

Dashboard: El 'Gran Juego' Post-Soviético (v2.5)

Análisis de Factores de Poder y Desarrollo Económico

Música de Fondo

▶ 0:00 / 3:46

Spy Glass (Kevin MacLeod)

Tip: Pulsa M para Play/Pause

Configuración v2.4

Filtrar por Año

2011

Seleccionar Países

Afghanistan x

Azerbaijan x

Armenia x Georgia x

Mongolia x

Promedio PIB (PPP)

\$6,191

Gasto Militar (% PIB)

2.84%

Índice Democacia

3.5

Control Corrupción

0.70

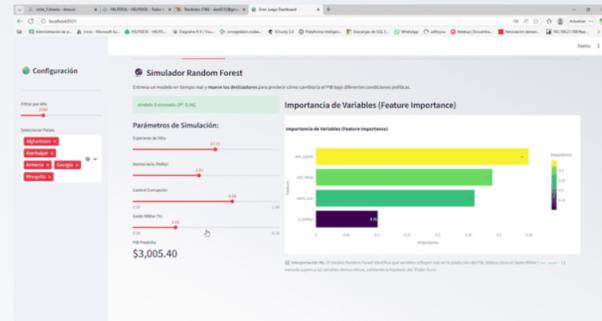
Analisis Exploratorio Modelo ML Interactivo Visión Regional Documentación Asistente IA

Documentación del Proyecto

Selecciona el documento que deseas visualizar:

Archivos Disponibles:

- README (General)
- Infraestructura
- Resultados y Análisis
- Reflexión IA
- Explicación Código
- Prototipo / Demo
- Respuestas



Nota: En el Dashboard interactivo, esta imagen se sustituye automáticamente por el reproductor de video.



Nota: En el Dashboard interactivo, esta imagen se sustituye automáticamente por el reproductor de video.



Dashboard: El 'Gran Juego' Post-Soviético (v2.5)

Análisis de Factores de Poder y Desarrollo Económico

Promedio PIB (PPP)

\$6,191

Gasto Militar (% PIB)

2.84%

Índice Democracia

3.5

Control Corrupción

0.70

[Análisis Exploratorio](#) [Modelo ML Interactivo](#) [Visión Regional](#) [Documentación](#) [Asistente IA](#)

Asistente Virtual: 'QoG-Bot'

Este asistente utiliza lógica analítica avanzada para generar reportes automáticos y responder preguntas sobre los datos.

Generar Reporte Automático

Elige un país para analizar:

Azerbaijan

Generar Informe

Informe generado con éxito.

Chat con tus Datos

¡Hola! Soy tu asistente de Big Data. Pregúntame cosas como: '¿Cuál es el país más rico?', '¿Promedio de esperanza de vida?' o 'Dime sobre Afganistán'.

¿Cuál es el país más rico?

El país más rico (mayor PIB per cápita, 2011) es Azerbaijan con \$9,793.31.

Escribe tu pregunta aquí...

Analisis de Inteligencia para Azerbaijan

1. Situación Económica: El PIB per cápita más reciente es de 9,793 USD (año 2011), lo cual es superior al promedio de la región (6,191 USD).

2. Perfil de Poder: Azerbaijan muestra un Gasto Militar del 4.99% del PIB. En términos políticos, su índice democrático es -7.0 (escala -10 a 10), mostrando una tendencia empeorando.

3. Conclusión Algorítmica: Este perfil sugiere un estado que prioriza la seguridad (Poder Duro).