

🌱 MllancamanA


first commit

36daab1 · 31 minutes ago🕒

151 lines (85 loc) · 11.3 KB

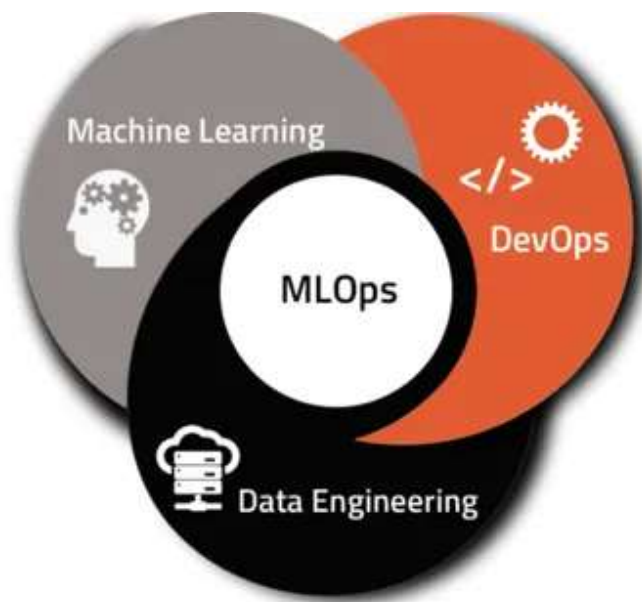
PreviewCodeBlame

Raw📄📥✎▾☰



# PROYECTO INDIVIDUAL N°1

## Machine Learning Operations (MLOps)



¡Bienvenidos al primer proyecto individual de la etapa de labs! En esta ocasión, deberán hacer un trabajo situándose en el rol de un *MLOps Engineer*.

## Descripción del problema (Contexto y rol a desarrollar)

---

### Contexto

---

Tienes tu modelo de recomendación dando unas buenas métricas 😊, y ahora, cómo lo llevas al mundo real? 🙄

El ciclo de vida de un proyecto de Machine Learning debe contemplar desde el tratamiento y recolección de los datos (Data Engineer stuff) hasta el entrenamiento y mantenimiento del modelo de ML según llegan nuevos datos.

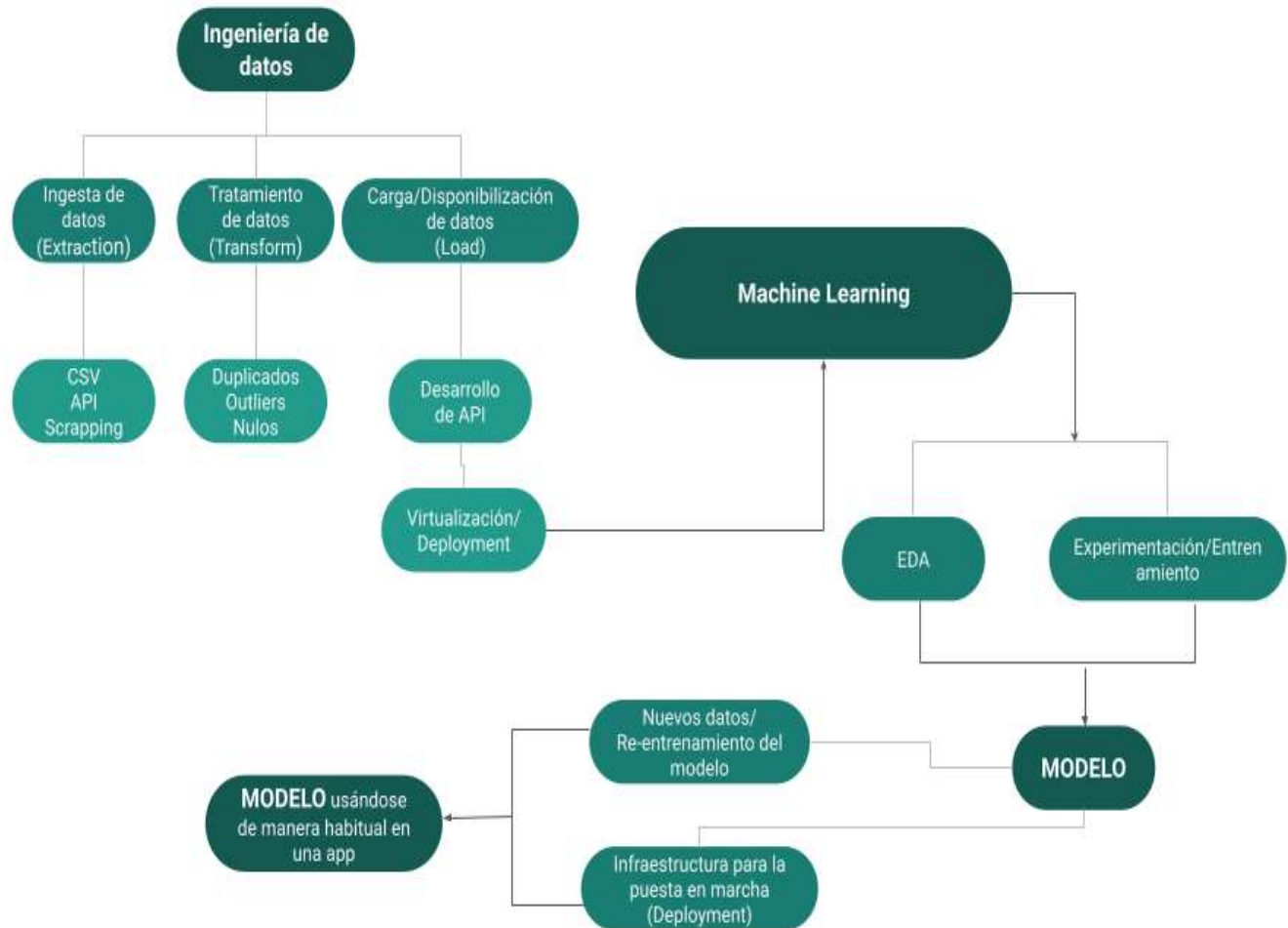
### Rol a desarrollar

---

Empezaste a trabajar como **Data Scientist** en Steam, una plataforma multinacional de videojuegos. El mundo es bello y vas a crear tu primer modelo de ML que soluciona un problema de negocio: Steam pide que te encargues de crear un sistema de recomendación de videojuegos para usuarios. 😞

Vas a sus datos y te das cuenta que la madurez de los mismos es poca (ok, es nula 🤖): Datos anidados, de tipo raw, no hay procesos automatizados para la actualización de nuevos productos, entre otras cosas... haciendo tu trabajo imposible 😫.

Debes empezar desde 0, haciendo un trabajo rápido de **Data Engineer** y tener un **MVP** (*Minimum Viable Product*) para el cierre del proyecto! Tu cabeza va a explotar 🤖, pero al menos sabes cual es, conceptualmente, el camino que debes de seguir !. Así que espantas los miedos y pones manos a la obra 🙌



Nota que aquí se reflejan procesos, no herramientas tecnológicas. Haz el ejercicio de entender qué herramienta del stack corresponde a cada parte del proceso

## Propuesta de trabajo (requerimientos de aprobación)

**Transformaciones** : Para este MVP no se te pide transformaciones de datos( aunque encuentres una motivo para hacerlo ) pero trabajaremos en leer el dataset con el formato correcto. Puedes eliminar las columnas que no necesitan para responder las consultas o preparar los modelos de aprendizaje automático, y de esa manera optimizar el rendimiento de la API y el entrenamiento del modelo.

**Feature Engineering** : En el dataset *user\_reviews* se incluyen reseñas de juegos hechos por distintos usuarios. Debes crear la columna '**sentiment\_analysis**' aplicando análisis de sentimiento con NLP con la siguiente escala: debe tomar el valor '0' si es malo, '1' si es neutral y '2' si es positivo. Esta nueva columna debe reemplazar la de *user\_reviews.review* para facilitar el trabajo de los modelos de machine learning y el análisis de datos. De no ser posible este análisis por estar ausente la reseña escrita, debe tomar el valor de 1 .

**Desarrollo API** : Propones disponibilizar los datos de la empresa usando el framework **FastAPI**. Las consultas que propones son las siguientes:

Debes crear las siguientes funciones para los endpoints que se consumirán en la API, recuerden que deben tener un decorador por cada una (`@app.get('/')`).

- **def developer( desarrollador : str )**: Cantidad de items y porcentaje de contenido Free por año según empresa desarrolladora. Ejemplo de retorno:

Año	Cantidad de Items	Contenido Free
2023	50	27%
2022	45	25%
xxxx	xx	xx%

- **def userdata( user\_id : str )**: Debe devolver cantidad de dinero gastado por el usuario, el porcentaje de recomendación en base a `reviews.recommend` y cantidad de items .

Ejemplo de retorno: {"Usuario X" : us213ndjss09sdf, "Dinero gastado": 200 USD, "% de recomendación": 20%, "cantidad de items": 5}

- **def UserForGenre( genero : str )**: Debe devolver el usuario que acumula más horas jugadas para el género dado y una lista de la acumulación de horas jugadas por año de lanzamiento.

Ejemplo de retorno: {"Usuario con más horas jugadas para Género X" : us213ndjss09sdf, "Horas jugadas": [{Año: 2013, Horas: 203}, {Año: 2012, Horas: 100}, {Año: 2011, Horas: 23}]}

- **def best\_developer\_year( año : int )**: Devuelve el top 3 de desarrolladores con juegos MÁS recomendados por usuarios para el año dado. (`reviews.recommend = True` y comentarios positivos)

Ejemplo de retorno: [{"Puesto 1" : X}, {"Puesto 2" : Y}, {"Puesto 3" : Z}]

- **def developer\_reviews\_analysis( desarrolladora : str )**: Según el desarrollador, se devuelve un diccionario con el nombre del desarrollador como llave y una lista con la cantidad total de registros de reseñas de usuarios que se encuentren categorizados con un análisis de sentimiento como valor positivo o negativo.

Ejemplo de retorno: {'Valve' : [Negative = 182, Positive = 278]}

### Importante

El MVP *tiene* que ser una API que pueda ser consumida segun los criterios de [API REST o RESTful](#) desde cualquier dispositivo conectado a internet. Algunas herramientas como por ejemplo, Streamlit, si bien pueden brindar una interfaz de consulta, no cumplen con las condiciones para ser consideradas una API, sin workarounds.

**Deployment** : Conoces sobre [Render](#) y tienes un [tutorial de Render](#) que te hace la vida mas fácil 😊 . También podrías usar [Railway](#), o cualquier otro servicio que permita que la API pueda ser consumida desde la web.

### Análisis exploratorio de los datos : (Exploratory Data Analysis-EDA)

Ya los datos están limpios, ahora es tiempo de investigar las relaciones que hay entre las variables del dataset, ver si hay outliers o anomalías (que no tienen que ser errores necesariamente 🙄 ), y ver si hay algún patrón interesante que valga la pena explorar en un análisis posterior. Las nubes de palabras dan una buena idea de cuáles palabras son más frecuentes en los títulos, ¡podría ayudar al sistema de predicción! En esta ocasión vamos a pedirte que no uses librerías para hacer EDA automático ya que queremos que pongas en práctica los conceptos y tareas involucrados en el mismo. Puedes leer un poco más sobre EDA en [este artículo](#)

### Modelo de aprendizaje automático :

Una vez que toda la data es consumible por la API, está lista para consumir por los departamentos de Analytics y Machine Learning, y nuestro EDA nos permite entender bien los datos a los que tenemos acceso, es hora de entrenar nuestro modelo de machine learning para armar un **sistema de recomendación**. Para ello, te ofrecen dos propuestas de trabajo: En la primera, el modelo deberá tener una relación ítem-ítem, esto es se toma un item, en base a que tan similar esa ese ítem al resto, se recomiendan similares. Aquí el input es un juego y el output es una lista de juegos recomendados, para ello recomendamos aplicar la *similitud del coseno*. La otra propuesta para el sistema de recomendación debe aplicar el filtro user-item, esto es tomar un usuario, se encuentran usuarios similares y se recomiendan ítems que a esos usuarios similares les gustaron. En este caso el input es un usuario y el output es una lista de juegos que se le recomienda a ese usuario, en general se explican como "A usuarios que son similares a tí también les gustó...". Deben crear al menos **uno** de los dos sistemas de recomendación (Si se atreven a tomar el desafío, para mostrar su capacidad al equipo, ¡pueden hacer ambos!). Tu líder pide que el modelo derive obligatoriamente en un GET/POST en la API símil al siguiente formato:

Si es un sistema de recomendación item-item:

- **def recomendacion\_juego( id de producto )**: Ingresando el id de producto, deberíamos recibir una lista con 5 juegos recomendados similares al ingresado.

Si es un sistema de recomendación user-item:

- **def recomendacion\_usuario( *id de usuario* )**: Ingresando el id de un usuario, deberíamos recibir una lista con 5 juegos recomendados para dicho usuario.

**Video** : Necesitas que al equipo le quede claro que tus herramientas funcionan realmente! Haces un video mostrando el resultado de las consultas propuestas y de tu modelo de ML entrenado! Recuerda presentarte, contar muy brevemente de que trata el proyecto y lo que vas a estar mostrando en el video. Para grabarlo, puedes usar la herramienta Zoom, haciendo una videollamada y grabando la pantalla, aunque seguramente buscando, encuentres muchas formas más. 😊

**Spoiler**: El video NO DEBE durar mas de **7 minutos** y DEBE mostrar las consultas requeridas en funcionamiento desde la API y una breve explicación del modelo utilizado para el sistema de recomendación. En caso de que te sobre tiempo luego de grabarlo, puedes mostrar/explicar tu EDA, ETL e incluso cómo desarrollaste la API.

## Criterios de evaluación

---

**código** : Prolijidad de código, uso de clases y/o funciones, en caso de ser necesario, código comentado. Se tendrá en cuenta el trato de los valores str como `COUNTER-strike` / `COUNTER-STRIKE` / `counter-strike` .

**Repositorio** : Nombres de archivo adecuados, uso de carpetas para ordenar los archivos, README.md presentando el proyecto y el trabajo realizado. Recuerda que este último corresponde a la guía de tu proyecto, no importa que tan corto/largo sea siempre y cuando tu 'yo' + 1.5 AÑOS pueda entenderlo con facilidad.

**Cumplimiento** de los requerimientos de aprobación indicados en el apartado Propuesta de trabajo

NOTA: Recuerde entregar el link de acceso al video. Puede alojarse en YouTube, Drive o cualquier plataforma de almacenamiento. **Verificar que sea de acceso público, recomendamos usar modo incógnito en tu navegador para confirmarlo.**

Aquí te sintetizamos que es lo que consideramos un MVP aprobatorio, y la diferencia con un producto completo.

## MVP Ingeniería de datos

Esperamos que tengan todo el ETL realizado y la API deployada en cualquier servicio, lista para consumir. Pueden faltar endpoints o no funcionar correctamente, pero debe haber una API consumible.

Debe estar el deployment realizado y el video que muestra que funciona correctamente

Al menos 1 endpoint es funcional en el deploy

## MVP Machine Learning

Esperamos que haya un EDA analizando un poco los datos respecto al problema que deben resolver. Deberia haber un endpoint en la API para poder consumir el modelo. Puede no devolver lo esperado, pero debe estar el endpoint accesible.

Debe estar modelo funcionando al menos en el video.

## Producto Completo

Deben tener EDA, readme completo, detallado, con instrucciones, repositorio ordenado, código bien comentado, deben funcionar los 6 endpoints de la API que esperamos con las respuestas que esperamos, puede haber alguna interfaz gráfica adicional.

Con el producto completo, no es necesario realizar el video, ¡pero suma como un plus!

## Fuente de datos

- [Dataset](#): Carpeta con el archivo que requieren ser procesados, tengan en cuenta que hay datos que estan anidados (un diccionario o una lista como valores en la fila).
- [Diccionario de datos](#): Diccionario con algunas descripciones de las columnas disponibles en el dataset.

## Material de apoyo

En este mismo repositorio podrás encontrar algunos (hay repositorios con distintos sistemas de recomendación) [links de ayuda](#). Recuerda que no son los unicos recursos que puedes utilizar!