PROYECTO INDIVIDUAL Nº1

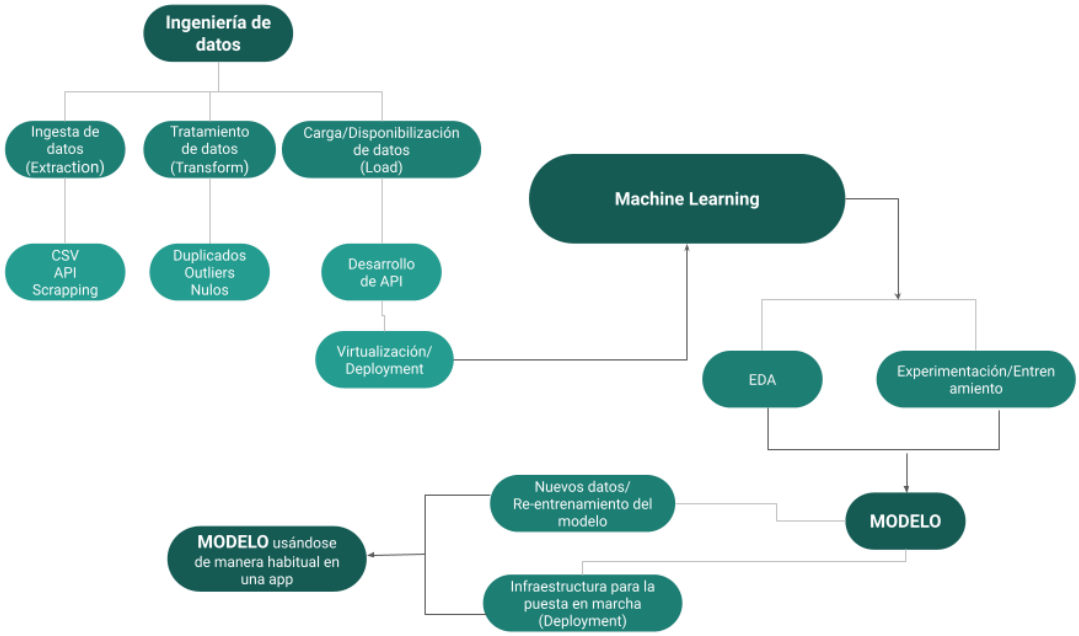
Machine Learning Operations (MLOps)

Juegos de video en línea STEAM [Juegos (steampowered.com)]
Amelia Herrera Briceño
melinnicri@gmail.com
Enero, 2024

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

Se debe empezar desde cero y obtener finalmente un Producto viable mínimo MVP (Minimum viable product).

Imagen del proceso de transformación de los datos hacia el modelo de ML:



Propuesta de trabajo (requerimientos de aprobación):

Transformaciones: Para este MVP no se te pide transformaciones de datos (aunque encuentres un 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.

- Se realizaron cambios sutiles en las columnas que eran necesarias, cambiando los NaNs por 0, o el tipo de dato.
- Habían 88.000 NaNs, por lo que se extirpó de raíz (todas las filas).
- Desanidado de algunas columnas, como genres o tags.

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.

- Se hizo un análisis de sentimiento VADER.

Desarrollo API: Propones disponibilizar los datos de la empresa usando el framework ***FastAPI***.

Los 3 archivos originales se encuentran en “DataBase”, como también el diccionario de datos.

1) games.ipynb guarda First_Game_genres.csv y First_Game_tags.csv
2) items.ipynb guarda Third_Users.csv
3) reviews.ipynb guarda Second_Reviews.csv

Para armar las consultas, se hicieron las nuevas dataframes desde:

- tags.ipynb abre First_Game_tags.csv y lo guarda como First_Game_destag.csv
- Desa_tag.ipynb abre First_Game_desTag.csv y lo guarda como First_Game_desanidadoTag.csv
- Consultas.ipynb abre First_Game_genres.csv, Third_Users.csv, First_Game_tags.csv, First_Game_desanidadoTag.csv y Second_Reviews.csv

Las consultas que propones son las siguientes:

- def PlayTimeGenre(*genero: str*): Debe devolver año con más horas jugadas para dicho género.

Ejemplo de retorno: {"Año de lanzamiento con más horas jugadas para Género X": 2013}

- 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.

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 UsersRecommend(*año : int*): Devuelve el top 3 de juegos MÁS recomendados por usuarios para el año dado. (reviews.recommend = True y comentarios positivos/neutrales)

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

- def UsersWorstDeveloper(*año: int*): Devuelve el top 3 de desarrolladoras con juegos MENOS recomendados por usuarios para el año dado. (reviews.recommend = False y comentarios negativos)

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

- def sentiment_analysis(*empresa desarrolladora: str*): Según la empresa desarrolladora, se devuelve un diccionario con el nombre de la desarrolladora 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.

Ejemplo de retorno: {'Valve': [Negative = 182, Neutral = 120, Positive = 278]}

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.

- En este caso, de los playtime_forever, hay una media de 9,9 x 10 a la 2 minutos, 990 minutos (16,5 horas) +/- 5,4 x 10 a la 3, 5400 minutos (90 horas) de juego.
- Bajo el 75% de los usuarios, juegan 3,5 x 10 a la 2 minutos, 350 minutos (5,83 horas).

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:

- Archivo: ML.ipynb abre Third_Users.csv

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.

//FIN … _@v