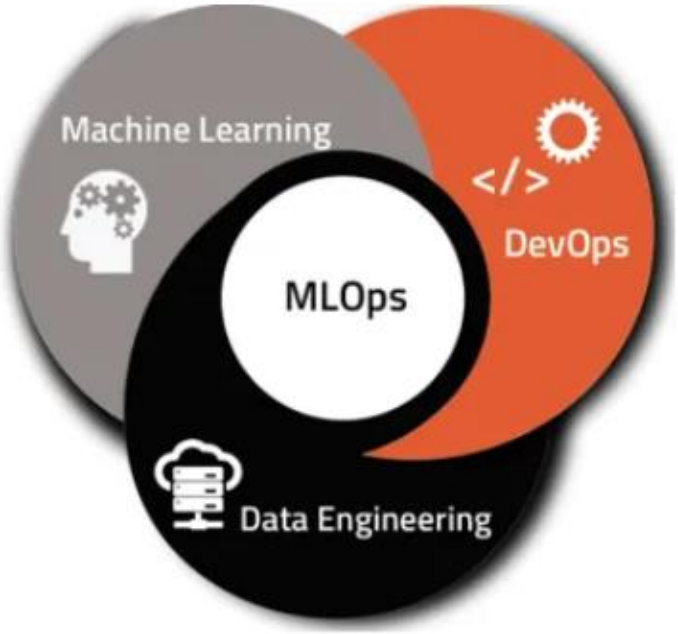
PROYECTO INDIVIDUAL Nº1

Machine Learning Operations (MLOps)



Name: Eduardo Bursese

Cohorte: Dataft16

Overview of the project:

From the database of a game dealer, the task is to develop an API which should give the outputs of the following requests:

- ✓ For a given 'game developer', inform quantity of items and free content per year.
- ✓ For a given 'game genre', it should give the user who played more time, and split the hours he played by years.
- ✓ For a given 'year', the output must be the top three developers recommended by users.
- ✓ For a given 'user', the output should be the total of money the user spent, and percentage of recommended items on the total reviews the user made.
- ✓ For a given 'developer', the API should give as an output, the total quantity of users' reviews with both positive and negative values.
- ✓ And the last one, a recommendation system with 2 alternatives:
  - ➢ For a given 'game', recommend 5 games to play, or, according to a 'user's recommendation', give him 5 new items as recommendation to play.

Source to do the development:

Three data frames were provided to make the functions asked. Below is the explanation about how they were transformed.

The idea is to have at the end, only one table with the data of the 3 given data frames joined.

User_reviews table {users_reviwes.ipynb}: The main task done on this table was to explode the data on the column of reviews to separate the data, and keep just the necessary columns like item_id, recommend and review. On joined_id was created from user_id and item_id to link user with the item.

Also that new id lets to join this table with the table out of steam_games file.

Steam.games table {steam_games.ipynb}: The data on genres columns were exploded to separate the data, and take the first genre as the reference for that item.

User_items table {user_items.ipynb}: The column item was exploded to create also here the joined_id mentioned above, to join this table with user-review.

Finally this table is joined with steam_games table on item_id, giving as a result only one table to have the output for the 4 queries where machine learning was not necessary.

For ML 2 other data frame were created, game_steam.csv and games_id.csv.

On all the tables, duplicated values were delete, also rows with whole zero values. Besides that, columns that were not necessary for the queries were also deleted.

Below it is the dictionary of final tables.

Dictionary	
File = merged_users_items_reviews_games	
Column name	
item_name	name of the game
playtime_forever	time in hours played by user
items_count	the total quantity of items played by user
joined_id	a combination of the item_id and user_id
item_id	the id of the game
recommend	boolean data type according to user opinion
review	comments written by user
user_id	idetification of user
publisher	who published the game
genres	a category of the game
title	the name of game
release_date	when the game was published
price	price of the game
developer	who created
year	year when was published
price2	price in float type
sentiment_analysis	values 0, 1, or 2 according to sentiment analyst

File = games_id	
item_name	name of the game
items_count	the total quantity of items played by user
item_id	the id of the game
review	comments written by user
publisher	who published the game
title	the name of game

File = games_steam	
item_name	name of the game
items_count	the total quantity of items played by user
item_id	the id of the game
review	comments written by user
publisher	who published the game

The API is running on this link https://main-py-m4mj.onrender.com/docs#/ .

The repository of the rest of the data in on this link https://github.com/EBjithub/Informes.git .

The video showing the API working is shared on this link
https://drive.google.com/drive/folders/1XAtdgRpqlX7qRarOpBPmcgyuDeunhdkM?usp=sharing

https://drive.google.com/file/d/1UApf8GB84KpnM6iO3JQOeT3o0H5-dMt8/view?usp=drive_link

<end of the document>
