**Procedimiento - Definición del problema (Contexto y papel a desempeñar)**

En este proyecto, llevaré a cabo un proceso de ETL (Extracción, Transformación y Carga), la creación de una API (Interfaz de Programación de Aplicaciones), EDA (Análisis Exploratorio de Datos) y concluiremos con un modelo de aprendizaje automático (ML) para la recomendación de películas.

**Contexto**

¿Cómo llevarás tu modelo de recomendación, que produce buenas métricas, al mundo real?

El ciclo de vida de un proyecto de aprendizaje automático debe considerar desde el manejo y recopilación de datos (tareas de ingeniería de datos) hasta la formación y mantenimiento del modelo de ML a medida que llegan nuevos datos.

**Papel a desempeñar**

Acabas de empezar a trabajar como Científico de Datos en una start-up que ofrece servicios de agregación de plataformas de streaming. Vas a crear tu primer modelo de ML que resuelve un problema de negocio: un sistema de recomendación que aún no ha sido implementado!

Al revisar los datos, te das cuenta de que la madurez de los mismos es baja: los datos están anidados, sin transformar, y no existen procesos automatizados para la actualización de nuevas películas o series, entre otras cosas, lo que hace tu trabajo muy difícil.

Tendrás que empezar desde cero, realizando un rápido trabajo de ingeniería de datos y tener un Producto Mínimo Viable (MVP) para el cierre del proyecto. Aunque la tarea parece abrumadora, al menos tienes una idea clara del camino a seguir. Así que superas tus miedos y te pones manos a la obra.

**ETL (Extracción, Transformación y Carga)**

Descripción de mis Datos: Característica\Descripción

* adult: Indica si la película es clasificada como X, exclusiva para adultos.
* belongs\_to\_collection: Un diccionario que muestra a qué franquicia o serie de películas pertenece la película.
* budget: El presupuesto de la película, en dólares.
* genres: Un diccionario que muestra todos los géneros asociados a la película.
* homepage: La página web oficial de la película.
* id: ID de la película.
* imdb\_id: ID de IMDB de la película.
* original\_language: Idioma original de la película.
* original\_title: Título original de la película.
* overview: Breve resumen de la película.
* popularity: Puntuación de popularidad de la película, asignada por TMDB (TheMoviesDataBase).
* poster\_path: URL del póster de la película.
* production\_companies: Lista de las compañías productoras asociadas a la película.
* production\_countries: Lista de los países donde se produjo la película.
* release\_date: Fecha de lanzamiento de la película.
* revenue: Recaudación de la película, en dólares.
* runtime: Duración de la película, en minutos.
* spoken\_languages: Lista de los idiomas que se hablan en la película.
* status: Estado actual de la película (si fue anunciada, si ya se estrenó, etc).
* tagline: Frase célebre asociada a la película.
* title: Título de la película.
* video: Indica si hay o no un tráiler en video disponible en TMDB.
* vote\_average: Puntuación promedio de las reseñas de la película.
* vote\_count: Número de votos recibidos por la película, en TMDB.

Transformaciones: Para este MVP no necesitas perfección, ¡necesitas rapidez! ⏩ Harás estas, y solo estas, transformaciones a los datos:

Eliminar las columnas que no serán utilizadas, video, imdb\_id, adult, original\_title, poster\_path y homepage.

Los valores nulos de los campos revenue, budget deben ser rellenados con el número 0.

Crear la columna con el retorno de inversión, llamada return con los campos revenue y budget, dividiendo estas dos últimas revenue / budget, cuando no hay datos disponibles para calcularlo, deberá tomar el valor 0.

Los valores nulos del campo release date deben eliminarse. Si hay fechas, deben tener el formato AAAA-mm-dd, además deberán crear la columna release\_year donde extraerán el año de la fecha de estreno.

Algunos campos, como belongs\_to\_collection, production\_companies y otros (ver diccionario de datos) están anidados, esto es o bien tienen un diccionario o una lista como valores en cada fila, ¡deberán desanidarlos para poder y unirlos al dataset de nuevo hacer alguna de las consultas de la API! O bien buscar la manera de acceder a esos datos sin desanidarlos.

encontraremos este proceso en el archivo Datos-ETL.ipynb

API (Interfaz de Programación de Aplicaciones)

Desarrollo: Sugieres hacer disponibles los datos de la empresa utilizando el marco de trabajo FastAPI. Las consultas que sugieres son las siguientes:

Deben crear 6 funciones para los endpoints que se consumirán en la API, recuerden que deben tener un decorador por cada una (@app.get(‘/’)).

#FUNCION 1

@app.get("/score\_titulo/{titulo\_de\_la\_filmación}")

def peliculas\_idioma(idioma:str):

idioma\_filtro = df[df['original\_language'] == idioma]

cantida\_pelis = idioma\_filtro['original\_language'].shape[0]

return {'idioma:':idioma, 'cantidad peliculas:':cantida\_pelis}

#FUNCION 2

@app.get("/peliculas\_duracion/{pelicula}")

def peliculas\_duracion(pelicula:str):

filtro = df[df["title"] == pelicula]

duracion = filtro["runtime"].values[0] if len(pelicula) > 0 else None

year = filtro["year"].values[0] if len(pelicula) > 0 else None

return {'PELICULA:':pelicula,'duracion en minutos:':duracion, 'año de estreno:':year }

#FUNCION 3

@app.get("/obtener\_informacion\_franquicia/{franquicia}")

def obtener\_informacion\_franquicia(franquicia):

franquicia\_df = df[df["name"].str.contains(franquicia, na=False)]

cantidad\_peliculas = len(franquicia\_df)

ganancia\_total = franquicia\_df["revenue"].sum()

ganancia\_promedio = franquicia\_df["revenue"].mean()

return f"La franquicia {franquicia} posee {cantidad\_peliculas} películas, una ganancia total de {ganancia\_total} y una ganancia promedio de {ganancia\_promedio}."

#FUNCION 4

@app.get("/peliculas\_pais/{Pais}")

def peliculas\_pais( Pais: str ):

pais = df[df["production\_countries"].str.contains(Pais, na=False)]

cantidad\_peliculas = len(pais)

return f"Se produjeron ", {cantidad\_peliculas}, "películas en el país ", {Pais}

#FUNCION 5

@app.get("/productora\_exitosa/{productora}")

def productora\_exitosa(productora:str):

productora\_filtro = df[df['production\_companies'] == productora]

cantidad = productora\_filtro['revenue'].sum()

cantidad\_peliculas = productora\_filtro['production\_companies'].shape[0]

return{'productora:':productora, 'ganancias totales:':cantidad, 'cantidad de peliculas generadas:':cantidad\_peliculas}

productora\_exitosa('Universal Pictures')

#FUNCION 6

@app.get("/get\_director/{nombre\_director}")

def get\_director(nombre\_director):

director\_films = df[df['directors'].notnull() & df['directors'].str.contains(nombre\_director)]

director\_titles = df[df['directors'].notnull() & df['directors'].str.contains(nombre\_director, na=False)]['title'].tolist()

director\_return = df[df['directors'].notnull() & df['directors'].str.contains(nombre\_director, na=False)]['return'].sum()

for movie in director\_titles:

revenue = df.loc[df['title'] == movie, 'revenue'].values[0]

print('Las películas y sus ganancias:', movie, revenue)

directors\_count = len(director\_films)

print('El número de películas que ha dirigido es:', directors\_count)

print('Sus películas fueron:', director\_titles)

print('Sus ganancias totales son:', director\_return)

@app.get("/recomendacion/{titulo}")