Data Science - Proyecto de Productivización

¡Bienvenidos al proyecto! Como sabréis os acaban de pedir una PoC (proof of concept) para un proyecto de Data Science. Al ser una PoC no se busca que esté perfecta, sino que mostréis el potencial del proyecto.

Vais a trabajar con bases de datos, Flask, git, cloud y PySpark.

Se pide:

Debéis tener un modelo que haga predicciones. NO se va a valorar que el modelo tenga un buen scoring (si lo tiene, ¡pues genial!). Podéis apoyaros en un modelo que ya tengáis hecho previamente.

Flask

Debéis definir las rutas de una aplicación flask, como mínimo para hacer predicciones. Puede haber rutas para entrenar el modelo, para leer datos de una base de datos, para almacenar predicciones en una base de datos, distintas rutas para distintas predicciones… Todo lo que se os ocurra cuando tenga sentido. Debe haber rutas que acepten parámetros en la URL, rutas que acepten parámetros en el cuerpo de la petición HTTP y rutas que acepten ambas opciones.

Git

Debéis utilizar git como sistema de control de versiones distribuido. Debe existir la rama main, hotfix, release, develop, feature A, feature B, feature C.
Main es la rama de producción. Hotfix es para cambios muy pequeños en producción. Release es la rama con los cambios listos para llevar a producción. Develop es la rama donde se trabajan los cambios. Feature A es la rama de trabajo desarrollado por el desarrollador A, Feature B es la rama de trabajo desarrollado por la desarrolladora B y de forma equivalente para Feature C.

Debéis tener un repo en GitHub del proyecto. Ese repo pertenece al desarrollador senior del proyecto. Jugaréis distintos roles. Debéis hacer fork a ese repo (desde GitHub) los tres miembros juniors (A, B y C).Cada miembro clonará el repo forkeado en su GitHub. El repo original del desarrollador senior tendrá las ramas Main, HotFix y Develop. Cada desarrollador tras forkear y clonar, creará su rama Feature A (o la oportuna suya) donde realizará su trabajo (sus commits). Cuando los commits tengan la funcionalidad buscada, la rama feature se mergeará con Develop.

El repo original al que git le hace seguimiento debe contener a su vez más carpetas y ficheros. Se valorará utilizar correctamente y justificado comandos como: checkout, revert, reset, pull, push, merge, rebase…

Podéis jugar con las ramas para tener conflictos al mergear y quedaros con un código, con otro, con una parte de cada uno…

Podéis jugar un poco a https://learngitbranching.js.org/?locale=es_ES y a https://ohmygit.org/ para entender bien git.
Debéis hacer una captura de pantalla de los comandos que vais ejecutando en la terminal/VSCode y el resultado de los ficheros.

Al final debe haber tres pull request. Ojo, antes de hacer cada pull request, aseguraos de que vuestro repo en local está lo más actualizado posible con el repo en upstream (no en origin).
Aceptaréis algún pull request y rechazaréis algún otro.

Cloud/Endpoint

Debéis realizar un despliegue de un endpoint en la nube deforma que se puedan hacer consultas en esa API para obtener predicciones. Podéis apoyaros en un ec2 de AWS, gunicorn y nginx.

PySpark/Bases de datos

Adicionalmente, los datos que usa el modelo para entrenarse deben estar guardados en una base de datos, ya sea relacional o NoSQL. Debe elegirse una base de datos, la query para almacenar los datos y la query para leer los datos. Dicha base de datos se puede leer desde Spark en Databricks Community Edition, donde además de poder hacer la lectura de los datos, también puede hacer el preprocesado (podéis conectaros a un s3, a un MongoDB…). Se adjunta un notebook para que tengáis una referencia de reprocesado en Spark.

Equipos

Trabajaréis toda la clase juntos y os dividiréis en 3 grupos. Los grupos empezarán a trabajar a la vez en paralelo.

GRUPO A: Flask
Debéis diseñar y codificar los scripts para la app.

GRUPO B: Git
Debéis diseñar qué estructura de ramas y commits queréis generar en esa simulación para demostrar un trabajo en equipo de desarrolladores trabajando a la vez. Usaréis los scripts del grupo A.

GRUPO C: Cloud/Endpoint
Despliegue de un endpoint funcional en la nube.

GRUPO D: PySpark/Databases
Las bases de datos donde se almacenan los datos previos al modelado (en crudo, o también tras algún tipo de procesado). El procesado de esas tablas en un cluster de Spark. La opción de guardar esas tablas dentro de Databricks o en un s3 (¡free tier!)/MongoDB (¡la versión gratuita compartida!) u otra base de datos relacional o NoSQL en la nube (usando free tier o gratuito).

IMPORTANTE: No hay que incurrir en ningún coste para practicar los conceptos vistos en clase.No almacenes GB de datos, no entrenes modelos con GPUs o TPUs durante mucho tiempo, no te salgas del free tier, NO subas contraseñas a GitHub, ten alarmas que te avisen del fin de free tier y de predecir la cantidad de un gasto.

Plan de trabajo

El diseño de los grupos es para que avancéis todas las partes desde el principio, pero no todos los bloques tendrán la misma carga en los mismos momentos. Aquellos alumnos que finalicen su parte antes ayudarán en los otros grupos donde la carga de trabajo sea mayor.

Entregables

- PDF explicando el caso de uso con capturas de pantalla y los pasos de todo explicado de forma sencilla.

- Notebooks, scripts.
- Presentación tipo PowerPoint/Streamlit/Prezi…

- Vídeo de la app de flask desplegada haciendo predicciones.

¡A por todas! :-)