

Seminario de Programación Funcional en Scala

Trabajo Práctico

Prof. Pedro Carossi

5 de mayo de 2019

Consideraciones

Este trabajo práctico consta de dos instancias: una primera de desarrollo y otra posterior de revisión y feedback. La nota final del mismo estará compuesta en un **80 %** por la nota de la primera instancia y en un **20 %** por la de la segunda.

El trabajo se desarrollará en equipos de **2** integrantes.

- Fecha de entrega primera instancia: 12/06/2019
- Fecha de entrega segunda instancia: 21/06/2019

Los trabajos se aceptarán hasta las 23:59hs con links a los repositorios (git públicos) donde estén. Se evaluará todo aquello pushado hasta dicho horario inclusive y no más tarde. Los reportes escritos deberán ser presentados en latex.

Consigna

Primera Parte

La primera instancia consta de un desarrollo en el lenguaje de programación *Scala* haciendo uso exclusivamente de herramientas del paradigma funcional y comentando el mismo utilizando el estilo *Scaladoc*. Es decir que el código producido deberá ser *Funcional Puro*. El mismo se puede dividir en tres módulos o etapas:

- Construir un pipeline de datos de manera puramente funcional para llenar una base de datos (recomendación, usar *Postgres*) a partir del csv *train.csv* provisto.
- Desarrollar otro pipeline de datos que ingiera los datos de la base de datos generada en el punto anterior, produzca un ETL y con la data manipulada entrene un xgboost utilizando spark (local) y typelevel/frameless contra la etiqueta “APOCRYPHA”. Finalmente persistir el modelo en formato *pmml*.

Nota 0: Para este punto se deberá tener en cuenta que los datos tienen distintos formatos o tipos, v.gr. Int, String, Json. Es por esto que el ETL será fundamental para la performance del xgboost. Se deberán identificar y distinguir las variables numéricas de las categóricas o mixtas pues a estas últimas será necesario encodearlas para embeberlas en datos numéricos (requerido por xgboost). A su vez, se deberán procesar los jsons y aplanarlos, en algunos casos se podrían usar técnicas simples de NLP para extraer información.

Nota 1: Para entrenar el xgboost deberán tener un dataset de entrenamiento y otro de validación. Para construirlos elijan de manera aleatoria a partir del índice con el que armaron la base del primer punto los registros que formarán parte de cada conjunto. Hecho esto, hagan las consultas a la base a partir de los índices ya separados.

- A partir del modelo obtenido en el punto anterior construir un servicio *REST* utilizando *http4s* que reciba como entrada (en el body del GET) un json con los mismos campos que ten´[U+FFFD]a el csv original salvo por el campo “APOCRYPHA”. Recibido el dato, se deberá verificar si el mismo ya se encuentra en la base, en cuyo caso devolverá la etiqueta “APOCRYPHA” persistida en ese registro. De lo contrario, se procesarán los datos con el pipeline del punto anterior, se responderá a la solicitud con la evaluación del modelo en estos y se persistirán los datos con la nueva etiqueta obtenida a la base.

Se deberá utilizar *docker compose* para entregar imágenes que contengan todo el desarrollo ya “cableado” y que permitan rápidamente probarlo en tiempo de ejecución, siempre acompañando al código fuente.

Concluido el servicio anterior, se evaluará el otro dataset provisto, *test.csv* utilizando la api *REST* desarrollada y se comparará el valor obtenido contra la etiqueta real. A partir de esto se escribirá un pequeño reporte con los resultados observados y conclusiones.

Segunda Parte

La segunda instancia consiste en hacer una revisión del trabajo de un equipo distinto del propio y escribir un reporte de feedback tanto con observaciones y propuestas de mejora como con puntos destacados del desarrollo que podrían mejorar el trabajo propio o que fueron claves también en el desarrollo propio.
