



TRABAJO INTEGRADOR FINAL

Refactorización de código para poner en producción un modelo de ML

Una de las principales tareas que debe abordar un machine learning engineer (MLE) es interpretar el código desarrollado por los científicos de datos (DS) y ser capaz de refactorizarlo para convertirlo en una pieza de software confiable y bien estructurado.

Para poner en práctica estas habilidades suponga que se ha incorporado a un equipo de datos que se encuentra desarrollando un modelo de ML para predecir precios de propiedades. El equipo ya entrenó su primer modelo de regresión simple para cumplir con las métricas definidas en conjunto con el negocio, y le han pedido que refactorice el código para poder comenzar a realizar las pruebas del modelo.

Para que pueda comenzar con la tarea solicitada el DS le ha enviado un archivo comprimido que contiene:

- El dataset con datos sobre las transacciones utilizadas para definir y entrenar el modelo.
- La notebook de EDA desarrollada por él con el análisis exploratorio de datos, las transformaciones realizadas, las pruebas de distintas arquitecturas de modelos y la selección del modelo final.

Dentro del notebook de EDA encontrará los siguientes pasos que deberá ordenar en distintos bloques/módulos de código para poder crear un pipeline que procese nuevos datos de entrada:

- ANALISIS UNIVARIADO: para cada una de las features numéricas y categóricas.
- ANALISIS BI-VARIADO: para detectar correlaciones e interacción entre las variables.

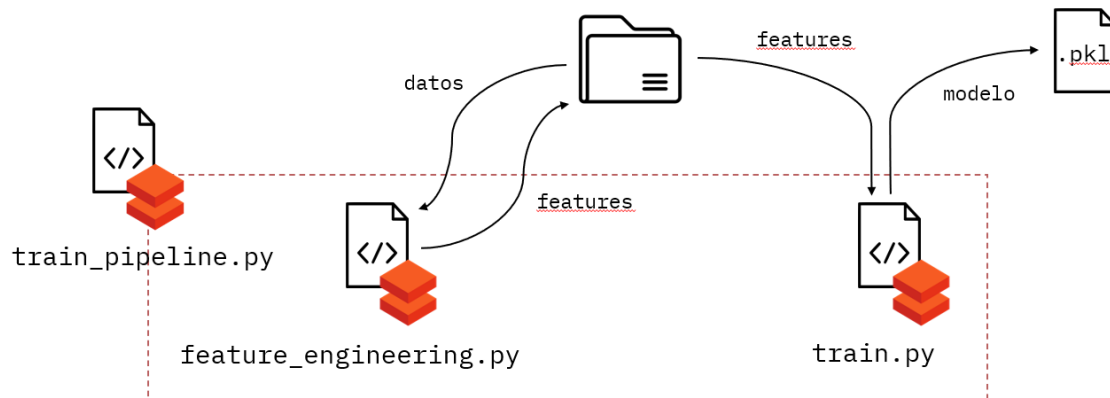
Dentro de cada una de ellas se realizan transformaciones y limpieza de datos que deberán ser replicadas en las tareas de feature engineering para obtener las mismas transformaciones de datos.

Dentro de esa misma notebook se encuentran las pruebas de varias arquitecturas. Se deberá tomar la arquitectura que mejor performó y agregarla al pipeline de inferencia del proceso.

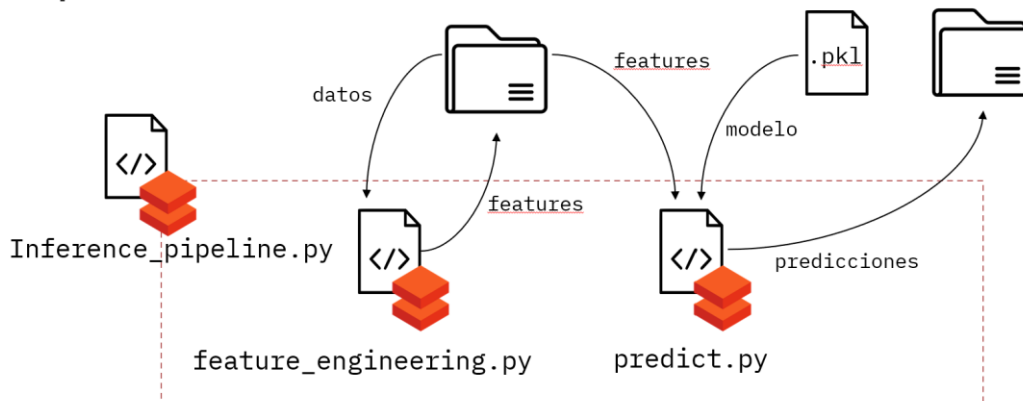


Se propone, aunque no es mandatorio, un esquema de scripts como el siguiente:

Pipeline de entrenamiento



Pipeline de inferencia



La implementación en Databricks es **opcional**.

Una vez desarrollada y probada la implementación elegida, utilizar alguno de los frameworks vistos para optimización de hiperparámetros (HPs) y realizar un ajuste fino del modelo. También se puede proponer una arquitectura que mejore las métricas.

El ajuste de HPs puede ser realizado en una notebook aparte, **no es necesario que ese nuevo modelo se incorpore a los pipelines**. La notebook con el ajuste de HPs se debe guardar en la carpeta "Notebook".

Entregables esperados:

Desarrollo de pipeline de procesamiento: script/s de Python que permita/n procesar nuevos datos. Dentro de estos scripts se deben respetar las transformaciones y los procesos implementados por el DS.

Pruebas: la ejecución del pipeline debe poder realizarse con una llamada desde consola. Se debería poder pasarle la ruta del dataset de test y que el proceso corra de punta a punta.



Calidad del producto: se deben aplicar la mayor cantidad de buenas prácticas posibles siguiendo los estándares vistos en el curso.

Documentación: cada función y módulo de Python debe tener su docstring correspondiente. Agregar comentarios in-line según se considere necesario para aclarar la ejecución del código. Agregar un readme sencillo para aclarar el modo de uso del código.

Formato de entrega:

Opc. 1: el trabajo deberá ser subido a su repositorio personal y agregar como colaborador del proyecto al usuario AlexBarria. Deberán ir haciendo los commits a medida que vayan avanzando y al final del proceso realizar un PR. La aprobación del PR quedará a cargo del docente una vez que se resuelvan todas correcciones del PR.

Opc. 2: comprimir todos los archivos del proyecto en un .zip/.rar y enviarlo por mail al docente (alexbarria_14@hotmail.com).

Lineamientos y recomendaciones:

- Prestar especial atención a las buenas prácticas de programación vistas a lo largo de la materia.
- Usar las herramientas pylint y autopep8 para lograr scripts prolijos y ordenados.
- Agregar loggings puede favorecer la tarea de depurar el código.
- Para verificar que las tareas de refactorización no afectaron el desarrollo del DS, se pueden calcular las métricas del modelo antes y después de la refactorización. Estas métricas deberían ser iguales.
- En la medida de lo posible, mantener los comentarios y la documentación en inglés.
- Para el manejo de argumentos entre los scripts se puede utilizar el módulo argparse de Python.
- Generar un archivo requirements.txt puede ayudar a que otro usuario pueda ejecutar su código con todas las dependencias necesarias.

Evaluación:

El trabajo puede ser realizado en grupos de hasta 3 personas.

La nota del TP representa un %60 de la nota total para la aprobación de la materia. El resto de la nota de la materia será dividida en partes iguales entre los formularios de Google de cada clase. Es requisito obligatorio tener aprobado el TP para aprobar la materia.

La fecha máxima de entrega será la última clase del bimestre donde haremos una breve puesta en común del trabajo (no es necesario preparar ningún tipo de presentación).