OBSERVACIONES IMPORTANTES PARA BLOG DETALLE

Transformaciones de los datos en pipeline.

Gurobi acepta varias transformaciones de los datos en un pipeline de entrenamiento de sklearn que luego se conecta como restricción en gurobi.

Dentro de las transformaciones aceptadas por gurobi se tiene:

- standarscaler

- polynomial features (max grade 2). Gurobi no accept order 3

- Transformaciones en las columnas

Dentro de las transformaciones interesantes que se pueden hacer que gurobi no acepta se tienen:

- minmaxscaler

- custom transformation of columns. Es decir, aplicar una transformación custom a una columna, definido por una función custom creada por el usuario def transform\_good\_data():

Entrenamiento modelo por segmentos

Al analizar los datos y entender el problema, se pueden indentificar variables que afectan más la predicción y que segmentar los datos bajo dicha variable ofrece claramente una división en los datos de forma de operación diferente. Luego, para abordar cómo predecir con datos segmentados justamente se entrenan más modelos con las diferentes condiciones de operación

Por ejemplo

Feature\_A <= 100: entrenar modelo A con dichos datos

Feature A > 100: entrenar modelo B con dichos datos

Al hacer esto se tiene un modelo por tramos, una especie de “pair\_wise” model.

Importante, en el ejemplo desarrollado se hace el Split por una variable Observada (variable no controlable, variable que es feature de un modelo y no está conectado con modelos de etapas previas). Se podría hacer con variables que son controlables o target de un proceso, pero tener en cuenta que el optimizador mueve los valores de dichas variables lo que implica tener cuidado cuando los valores se muevan entre el threshold porque podría pasar de un modelo a otro sin que se desee dicho efecto.

(fotos códigos)

Para implementar el ejemplo desarrollado como una constraint de gurobi, en el momento en que se cargan los modelos se evalúa la variable observada y ver en qué segmento se encuentra y en base a eso cargar un modelo u otro. Luego, como esta variable observada es un valor fijo que no cambia no hay problema en implementar un modelo “pairwise” divido en diferentes modelos independientes.

OPTIMIZADOR

1. Mejoras en el optimizador. Seguir con el optimizador sin parametrizar (hardcodeado) obtenido en el repo anterior y agregar las mejoras desarrolladas previamente

Improve the optimization engine development in the previous repo: <https://github.com/joseortegalabra/Optimization-Industrial-Process/blob/main/6_optimization/2_optimization_3_models-v2.ipynb>

Update optimization engine with the updates in the training models:

* using piecewise models (to improve the performance of the models)
* more complex models (using gradient boosting, but gurobi support a lot of models)
* transformations in columns (standarscaler, polynomial features)
* operations in columns (logaritm, euler, etc)
* adding more models (in the stage D0EOP with the same input fetures predict two differents target, used as features in the next stage D1)

1. Luego de tener el optimizador actualizado y hardcodeado, parametrizarlo para funcionar con cualquier modelo de machine learning entrenado y con cualquier feature de decisión