ITPAero

Marco Pérez González
David Cano Rosillo

Noviembre 2024

Contents

1 Introducción

Las principales características del reto son las siguientes:

1. En el taller contamos con la información de los anteriores brochados que ha realizado nuestra máquina y broca. Esto es importante porque podemos modelizar el problema como una serie temporal, donde debemos usar la información de filas anteriores para predecir la actual.

2. Cada broca, usada en cada máquina y con un utillaje específico, se comporta de una forma concreta. Por tanto, las variables categóricas cobran una importancia mayor.

2 Creación de Variables

Sabiendo estos puntos, hemos creado variables artificiales a raíz de las básicas que intentan modelar todo esto. Especialmente variables que tratan de resumir como se ha comportado cada broca o máquina anteriormente. Estas han sido algunas de las más importantes:

1. **BrochaSNtarget_lag1**: Es el último error que se obtuvo en la misma brochaSN que queremos predecir.

2. **Utillajetarget_lag1**: Es el último error que se obtuvo en el mismo utillaje que queremos predecir.

3. **BrochaSNtarget_rolling_max7**: El máximo error de esa brocha de las últimas 7 veces que se usó.

Hay que tener cuidado de no hacer trampa tomando en cuenta el valor de la fila a predecir, por lo que se excluye este siempre. En el código creamos muchas variables, pero luego nos quedamos con las k mejores usando SHAP para medir su importancia.

3 Modelos y Explicabilidad

Hemos utilizado los siguientes modelos: CatBoost, LGBM y XGBoost, todos basados en ensembles de árboles de decisión de Boosting por gradiente. La razón por la que optamos por estos modelos es principalmente por su rendimiento, aunque también son muy útiles por su explicabilidad usando

los SHAP values. Esta herramienta nos ha permitido ver cuáles de las variables artificiales están mejorando más nuestro modelo, permitiendo un mejor Feature Engineering.

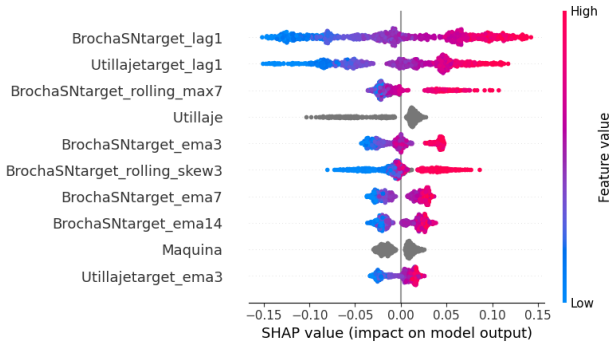

Figure 1: Análisis de importancias utilizando SHAP. Las variables más importantes tienen que ver con el histórico de la brocha.

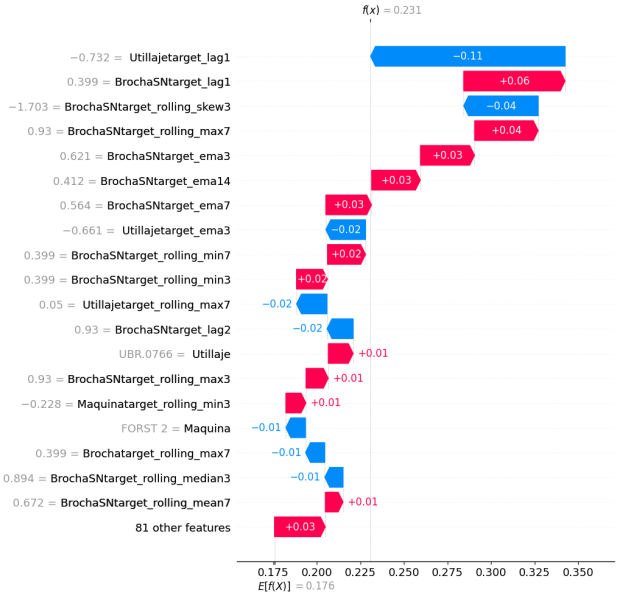

Figure 2: Visualización de como cada variable movió la predicción del modelo usando SHAP. Observamos que algunas variables mueven la predicción hacia la izquierda, mientras que otras la aumentan.

4 Optimización de hiperparámetros

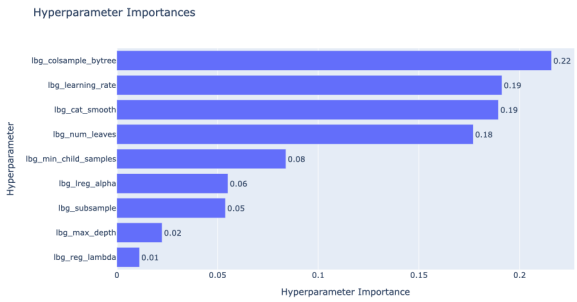

Figure 3: Los hiperparámetros más importantes según Optuna.

Para encontrar los mejores hiperparámetros usamos optimización bayesiana. Esta técnica entrena varios modelos y toma en cuenta los valores que obtuvo con cada hiperparámetro para explorar las zonas más prometedoras. Los hiperparámetros se buscan utilizando los resultados que se obtienen en la validación. Estos son algunas gráficas de qué regiones de hiperparámetros fueron más importantes.

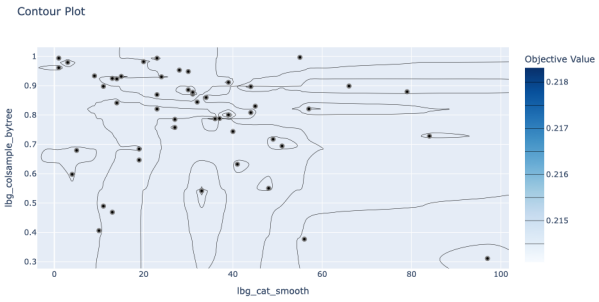

Figure 4: Optuna trata de aprender la relación entre los hiperparámetros y el error. En este caso la métrica enseñada es RMSE.

5 Resultados y Validación Cruzada

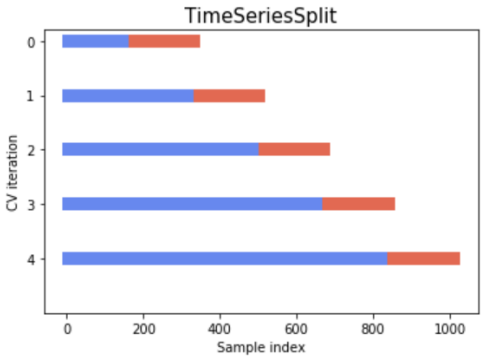

Figure 5: Validación en serie temporal

La validación cruzada se realizó utilizando técnicas específicas para series temporales, asegurando que los conjuntos de entrenamiento y prueba no se solapen. Para cada uno de los conjuntos de validación creados, se entrena un modelo y se guardan los modelos entrenados.
Empleando esta técnica hemos entrenado un modelo diferente para cada uno de los 10 Folds que hemos usado para validar, pesando cada modelo según un peso polinómico que da más importancia a los modelos más nuevos. Esto tiene sentido, ya que estos han sido entrenados en datos más recientes y, por tanto, más similares a los actuales.

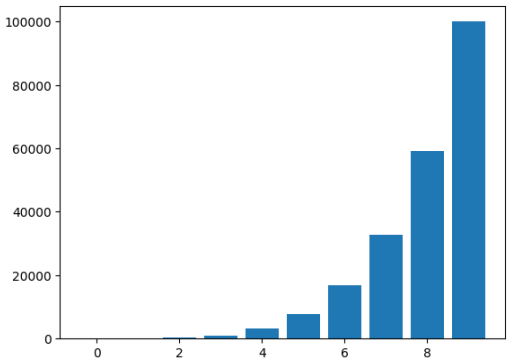

Figure 6: Peso polinómico para cada uno de los 10 Folds

Para realizar las predicciones en el conjunto de prueba final, seguimos el siguiente procedimiento:

1. Para cada tipo de modelo (por ejemplo, LGBM), calculamos la media de las predicciones en el conjunto de prueba de todos los modelos entrenados.

2. Finalmente, combinamos las predicciones de los tres tipos de modelos (LGBM, CatBoost y XGBoost) haciendo una media nuevamente.

Con estos modelos, obtenemos los siguientes resultados:

Table 1: Resultados obtenidos para el eje X

Métricas	**LGBM**	**CatBoost**	**XGBoost**	**Ensemble**
RMSE (Validación)	0.2364	0.2220	0.2228	**0.2205**
RMSE (Test)	0.2329	0.2275	0.2270	**0.2273**
Max err. (Validación)	0.8222	0.7516	0.7514	**0.7468**
Max err. (Test)	1.0791	1.1917	1.0542	**1.0666**

Table 2: Resultados obtenidos para el eje Z

Métricas	**LGBM**	**CatBoost**	**XGBoost**	**Ensemble**
RMSE (Validación)	0.0331	0.0332	0.0321	**0.0321**
RMSE (Test)	0.0434	0.0466	0.0488	**0.0453**
Max err. (Validación)	0.0994	0.1039	0.0998	**0.0995**
Max err. (Test)	0.2407	0.2553	0.2426	**0.2416**

Table 3: Resultados obtenidos para el eje B

Métricas	**LGBM**	**CatBoost**	**XGBoost**	**Ensemble**
RMSE (Validación)	0.0348	0.0260	0.0262	**0.0273**
RMSE (Test)	0.0391	0.0313	0.0312	**0.0341**
Max err. (Validación)	0.1145	0.0959	0.0962	**0.0977**
Max err. (Test)	0.1737	0.1655	0.1693	**0.1641**

Table 4: Resultados obtenidos para el eje C

Métricas	**LGBM**	**CatBoost**	**XGBoost**	**Ensemble**
RMSE (Validación)	0.0212	0.0198	0.0211	**0.0202**
RMSE (Test)	0.01822	0.0228	0.0183	**0.0180**
Max err. (Validación)	0.1084	0.1000	0.1065	**0.1038**
Max err. (Test)	0.1003	0.2380	0.1104	**0.1046**

6 Repositorio

Todo el código se puede encontrar en el repositorio de GitHub:
`https://github.com/marcoperg/indiesIA-ITPAero`