Modelo de predicción de fallecimientos en accidentes de carretera en Canadá

Puesta en producción del modelo

Práctica II asignatura Machine Learning

Máster en Data Science

Autores:

Pablo Carretero Collado

Ramón Guerrero Morales

Objetivo:

El objetivo de esta práctica final de la asignatura "Machine Learning "del máster en Data Science, es poner en producción el modelo generado en la anterior práctica mediante una API.

Por tanto, pondremos en producción la predicción de muertes por siniestro en Canadá, dadas unas características concretas con el fin de que, como aseguradora, fuese posible el cálculo de la provisión necesaria.

Contexto:

En una primera práctica mediante datos públicos (https://www.kaggle.com/tbsteal/canadian-car-accidents-19942014?select=drivingLegend.pdf) procedimos a analizar la predicción de siniestros en Canadá. A continuación, mencionamos los pasos que seguimos:

1. **Análisis Exploratorio de Datos (EDA):**

En una primera instancia, iniciamos el análisis del conjunto de datos para obtener una visión clara de estos y proceder con la transformación y limpieza para posteriormente modelizar y realizar las predicciones adecuadas.

En este caso hemos modificado el EDA introduciendo la función joblib para exportar el conjunto de datos que necesitamos en cada momento para en un futuro utilizarlo en la aplicación Flask para construir la API.

En el EDA se encuentran las transformaciones necesarias de las variables, eliminando la información innecesaria de cada elemento. Además, realizamos la codificación de las variables categóricas y de las temporales. Esta codificación de variables se ha tenido que realizar de manera manual, pues se han tenido algunos problemas para hacerlo con un pipe. Una modificación que hemos tenido que realizar ha sido el Mean Encoding, que explicamos a continuación:

Mean Encoding: para la ejecución del trabajo hemos intentado agrupar la codificación mediante un pipeline, pero no se ha conseguido con éxito, pues se eliminaba siempre la variable objetivo y por ello no se podía realizar el Mean Encoding. Para conseguirlo hemos realizado el mismo procedimiento de manera manual. Esto se ha realizado calculando la media agrupada (como si fueran pesos) y posteriormente guardando dichos "pesos" en un diccionario, que guardaremos en un joblib, para posteriormente poder cargarlo en la aplicación.

Posteriormente realizamos la debida separación en Train y Test, tratamos los valores missing y finalmente procedemos con el escalado de variables y el rebalanceo, quedando los datos en perfectas condiciones para proceder con los modelos.

2. **Modelización:**

En el proceso de modelización se ponen en marcha varios modelos con el fin de encontrar el que mejor prediga mediante los datos que hemos transformado. Esta fase la iniciamos con un modelo Lasso mediante el cual hacemos una selección de variables para introducir en el modelo, dejando fuera aquellas que aportan menos información.

Entre los modelos que desarrollamos se encuentran el Random Forest, GLM, Support Vector Machine, XGBoost, LightGBoost. De todos ellos, el que mejor métricas nos ofrecía era el Random Forest, por lo que optimizamos sus hiperparámetros.

Una vez se optimizaron los hiperparámetros, se guardó el modelo en un joblib para su posterior carga en la aplicación.

Desarrollo API

La creación de un modelo de machine learning cobra sentido cuando se implementa como una aplicación web para que los usuarios puedan interactuar con ella. Para ello, el objetivo final es brindar el modelo como un servicio, a través de una interfaz. Esta interfaz de programación de aplicaciones se denomina API (Aplicación Programming Interfaces). Las API permiten que los sistemas informáticos se comuniquen entre sí, enviando información al servidor y devolviendo la respuesta. Para poder llevar esto a cabo hemos utilizado FLASK, que es un framework de Python que permite crear aplicaciones web, API, etc.

Los archivos de Python se llaman módulos, y se pueden identificar con la extensión.py. Estos módulos definen funciones, clases y variables como las que nombraremos a continuación.

Pasos que seguimos durante el desarrollo de la aplicación

Los pasos que hemos realizado para la construcción de la app son los siguientes:

1. Importamos las debidas **librerías**.
2. Definimos el nombre de la app **Flask** como la variable **__name__**.
3. Cargamos mediante **joblib** las variables elegidas por el modelo Lasso, sin transformaciones previas como por ejemplo las transformaciones que se aplicarían a las variables temporales. Estas son las variables que recibirá la API mediante un **json**.
4. Cargamos nuestro modelo final, **Random Forest** con hiperparámetros.
5. En el **@app.route("/")**, **route()** es un decorador que le dice a Flask qué URL debe activar la función definida como **predict()** al cual le añadimos los métodos **GET** y **POST** para recibir los datos del fichero json y enviar los datos para la predicción.
6. Definimos la función **predict()** que va a incluir todas las transformaciones necesarias, así como la predicción final.
7. Recibimos el fichero **json** y lo convertimos en dataframe para poder realizar las próximas transformaciones.
8. Cargamos los diccionarios del M**ean Encoding** para poder convertir los pesos del fichero recibido a los que aceptará nuestro modelo.
9. Realizamos el **Cyclical Encoding** para convertir las variables temporales a las variables que recibirá nuestro modelo (seno y coseno).
10. Cargamos nuestro escalado realizado en el EDA.
11. Finalmente obtenemos las mismas variables que se introducen en el modelo de predicción final.
12. La predicción que obtenemos será un porcentaje con las probabilidades de morir en un siniestro mediante la función **predict_proba()**.
13. Establecemos la variable **__name__** como **__main__** porque el módulo que se está ejecutando es el programa principal. El módulo **sys** proporciona funciones y variables utilizadas para manipular partes del entorno de Python.
14. Posteriormente, escogemos el puerto donde se encontrará la API, en nuestro caso **12345**.
15. Seguidamente cargaremos el modelo y las columnas que queremos que aparezcan en el **json** mediante **joblib**.
16. Una vez corremos la aplicación y copiamos el link creado en **Postman**, podremos introducir los datos mediante un fichero **json** y que se produzca las respuestas.

Ejemplos

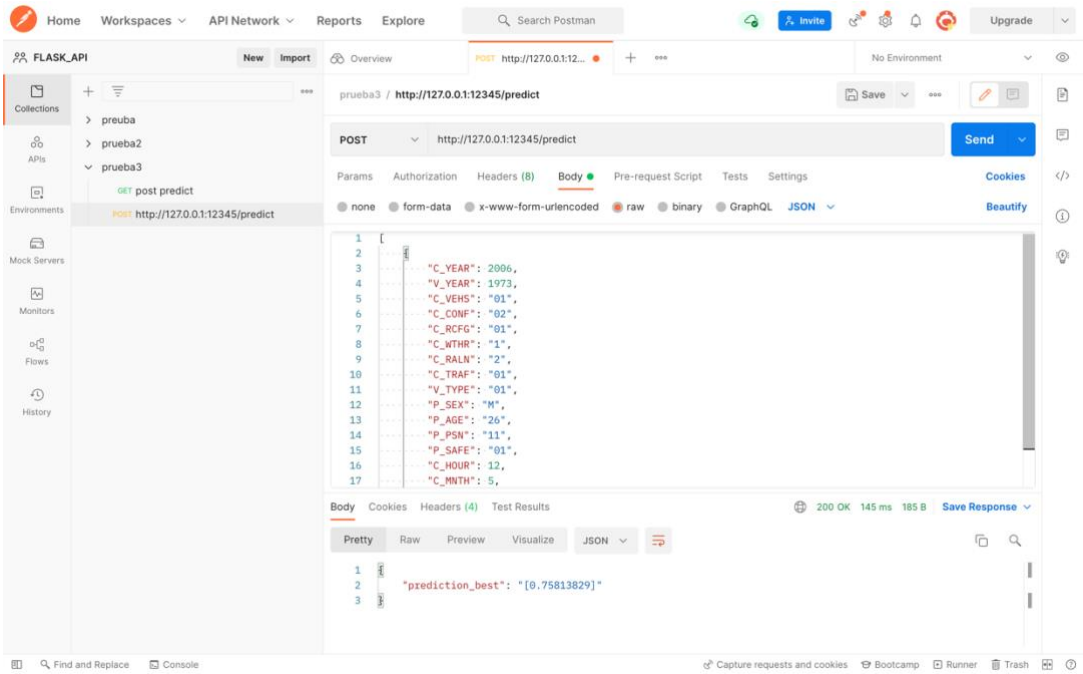
A continuación, se observan dos ejemplos con distintos valores para las variables:

EJEMPLO 1

En este primer ejemplo los datos introducidos han sido los siguientes:

Variables	Datos
C_YEAR	2006
V_YEAR	1973
C_VEHS	01
C_CONF	02
C_RCFG	01
C_WTHR	1
C_RALN	2
C_TRAF	01
V_TYPE	01
P_SEX	M
P_AGE	26
P_PSN	11
P_SAFE	01
C_HOUR	12
C_MNTH	5
C_WDAY	2

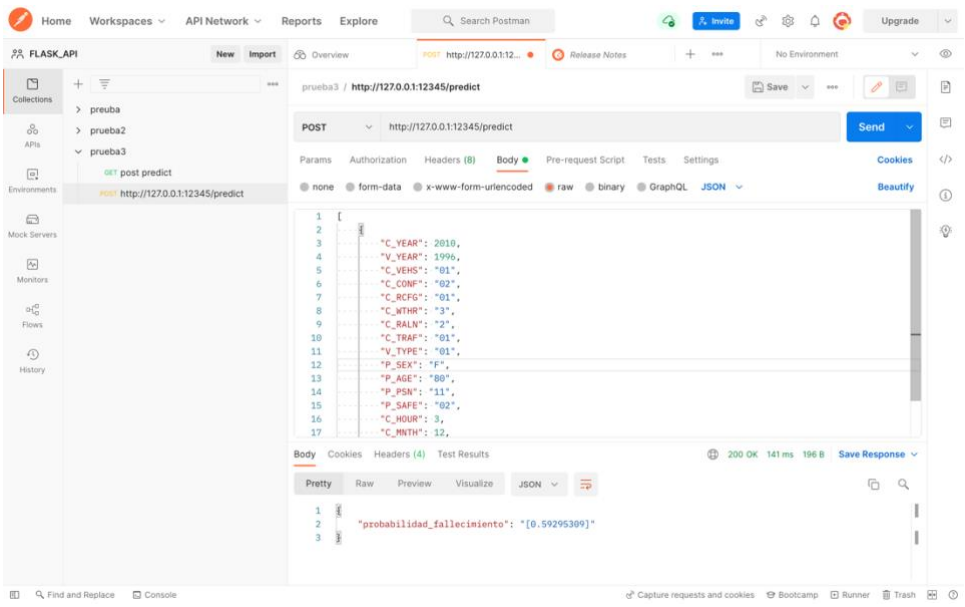
Obteniendo una predicción de fallecimiento del 75.8%.



EJEMPLO 2

Variables	Datos
C_YEAR	2010
V_YEAR	1996
C_VEHS	01
C_CONF	02
C_RCFG	01
C_WTHR	3
C_RALN	2
C_TRAF	01
V_TYPE	01
P_SEX	F
P_AGE	80
P_PSN	11
P_SAFE	02
C_HOUR	3
C_MNTH	12
C_WDAY	5

Se obtuvo una predicción del 59.29% de fallecer.



Los datos introducidos en las variables son los mismos datos del diccionario que se introdujo en el propio proyecto del modelo anterior. Para no hacer compleja la búsqueda, el diccionario será introducido en la misma carpeta que la aplicación y el informe desarrollado.