

PUESTA EN PRODUCCIÓN DEL MODELO DE SINIESTRALIDAD EN

ACCIDENTES DE TRÁFICO

Aprendizaje Automático

Colegio Universitario de Estudios Financieros (CUNEF)

Autores:

Marisa Alonso (marisa.alonso@cunef.edu)

Ignacio Sigüenza (ignacio.siguenza@cunef.edu)

Información general:

En la presente práctica se implementa el modelo de predicción de mortalidad en accidentes de tráfico desarrollado en la práctica anterior por medio de una REST API. Para ello, se emplea *Flask*, un *framework* de desarrollo *back-end* para Python, peticiones de prueba, tanto válidas como erróneas, y el propio modelo ya entrenado y almacenado en un archivo binario.

Archivos/Componentes de código requeridos:

- Código del funcionamiento del servidor, local en este caso, que soporta el funcionamiento de la API
- Peticiones en formato JSON con datos para poner a prueba el funcionamiento de la API
- Plantilla HTML básica para permitir la entrada manual de datos al usuario e imprimir por pantalla las respuestas de la API correspondiente a las predicciones en formato de probabilidad

Desarrollo:

En primer lugar, se crea un script de Python (extensión .py), para poder ser ejecutado. Posteriormente, importan las librerías y funciones de *Flask* necesarias, para finalmente inicializar dentro de dicho archivo una aplicación de *Flask*. El contenido de dicho objeto hace referencia a las distintas rutas/URLs que componen la aplicación web y que, en nuestro caso, siguen la siguiente estructura:

- **Ruta /:** Contiene la página de inicio a la que se accede una vez conectado, mostrando un mensaje de bienvenida y la interfaz gráfica (GUI) a través de la cual el usuario puede introducir los datos de un modo más sencillo y manual:

Ilustración 1: Página de inicio de la aplicación web

Bienvenido a la API de predicción de mortalidad en accidentes de coche

Proyecto desarrollado por:

Marisa Alonso (marisa.alonso@cunef.edu)

Ignacio Sigüenza (ignacio.siguenza@cunef.edu)

Canadian accidents

C_YEAR	C_MNTH	C_WDAY	C_HOUR	C_VEHS	C_CONF	C_TRAF
C_RCFG	C_WTHR	C_RSUR	C_RALN	V_TYPE	V_ANTIG	P_SEX
P_AGE	P_PSN	P_SAFE	P_USER	Predict		

- **Ruta /predictGUI:** En esta ruta se procesa de manera interna la información introducida en la GUI de la página de inicio, tras accionar un disparador en forma de botón HTML, y realiza la predicción de similar modo al caso posterior, con la única diferencia del origen y formato de ingesta de los datos:

Ilustración 2: Introducción de datos (interfaz gráfica)

Bienvenido a la API de predicción de mortalidad en accidentes de coche

Proyecto desarrollado por:

Marisa Alonso (marisa.alonso@cunef.edu)

Ignacio Sigüenza (ignacio.siguenza@cunef.edu)

Canadian accidents

2007	4	5	12	2	3	F
02	4	2	54	13	11	22
03	06	02	1	Predict		

Ilustración 3: Obtención de predicción (interfaz gráfica)

Bienvenido a la API de predicción de mortalidad en accidentes de coche

Proyecto desarrollado por:

Marisa Alonso (marisa.alonso@cunef.edu)

Ignacio Sigüenza (ignacio.siguenza@cunef.edu)

Canadian accidents

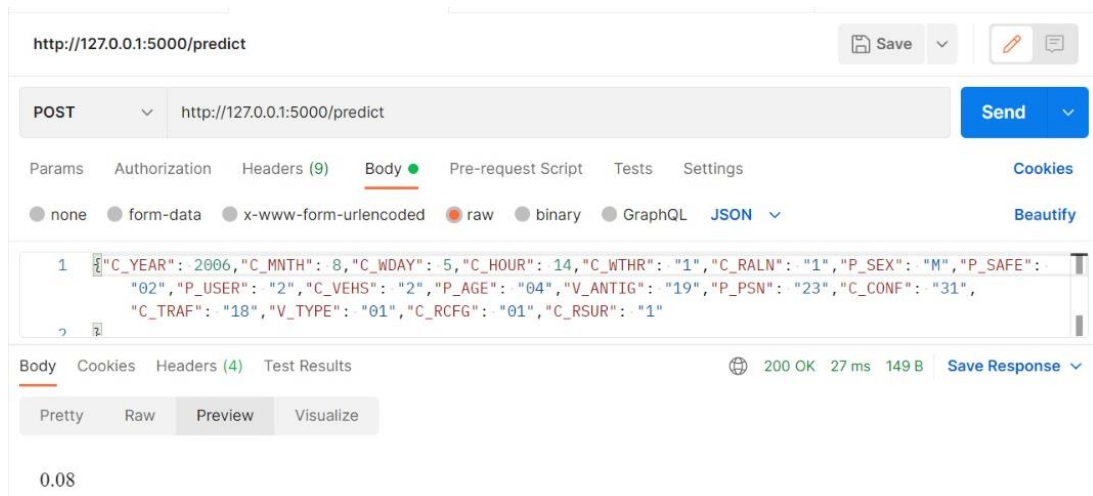
C_YEAR	C_MNTH	C_WDAY	C_HOUR	C_WTHR	C_RALN	P_SEX
P_SAFE	P_USER	C_VEHS	P_AGE	V_ANTIG	P_PSN	C_CONF
C_TRAF	V_TYPE	C_RCFG	C_RSUR	Predict		

Accident probability of fatality is 0.0

- **Ruta /predict:** Esta ruta permite al usuario hacer uso del modelo, prediciendo en base a unos datos que se facilitan por medio de un archivo JSON y a través de una petición de tipo POST. Este método para llevar a cabo la predicción puede realizarse desde un script de Python u otros medios, cómo es el caso de la plataforma de gestión de APIs, Postman. En nuestro caso, hemos utilizado ejemplos de ambas vías.

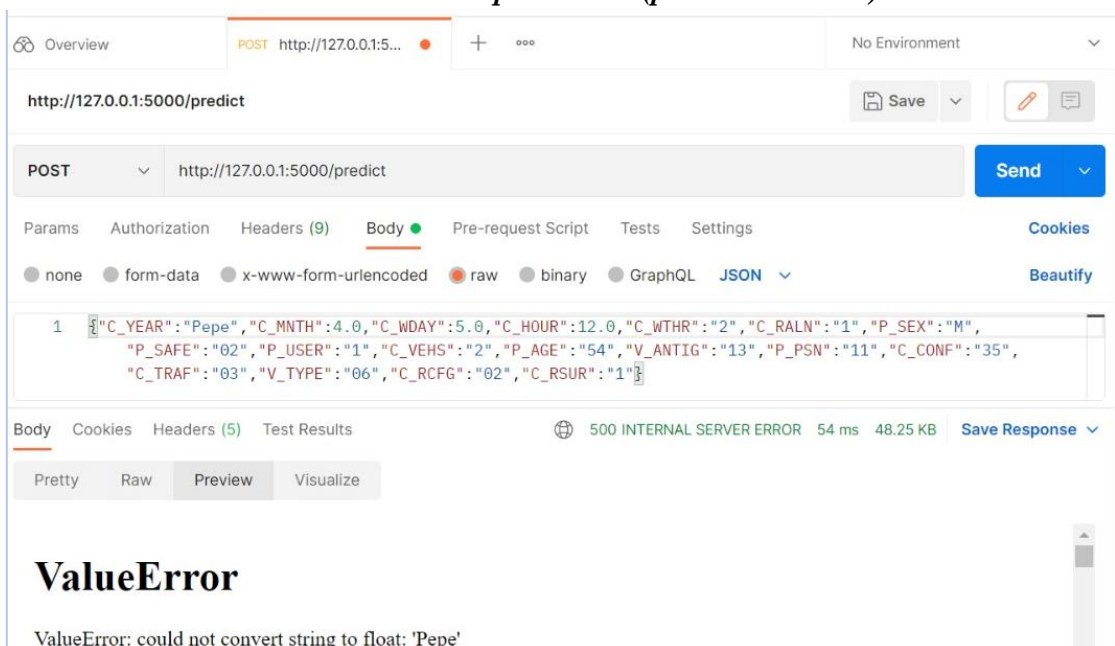
En el caso de Postman, hemos introducido un perfil aleatorio de datos en formato JSON, coincidiendo con el del diccionario de Python en este caso, y hemos obtenido la misma predicción que hubiésemos obtenido ingresando dicha entrada en el documento del modelo original:

Ilustración 4: Obtención de predicción (petición válida) Postman



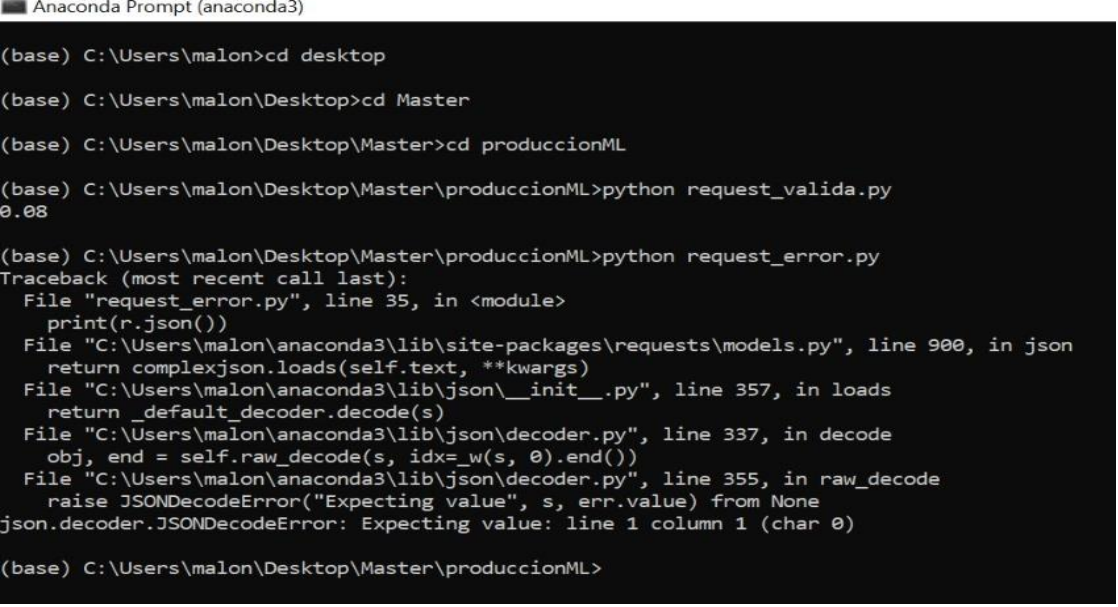
Adicionalmente, hemos probado a introducir valores erróneos en una variable numérica continua, para comprobar el tipo de error que se manifiesta en estos casos:

Ilustración 5: Obtención de predicción (petición errónea) Postman



Hemos comprobado con los mismos datos que en las peticiones, pero en este caso utilizando dos archivos request_valida.py y request_error.py y ejecutándolos desde el terminal.

Ilustración 6: Obtención de predicciones terminal



```
Anaconda Prompt (anaconda3)

(base) C:\Users\malon>cd desktop

(base) C:\Users\malon\Desktop>cd Master

(base) C:\Users\malon\Desktop\Master>cd produccionML

(base) C:\Users\malon\Desktop\Master\produccionML>python request_valida.py
0.08

(base) C:\Users\malon\Desktop\Master\produccionML>python request_error.py
Traceback (most recent call last):
  File "request_error.py", line 35, in <module>
    print(r.json())
  File "C:\Users\malon\anaconda3\lib\site-packages\requests\models.py", line 900, in json
    return complexjson.loads(self.text, **kwargs)
  File "C:\Users\malon\anaconda3\lib\json\__init__.py", line 357, in loads
    return _default_decoder.decode(s)
  File "C:\Users\malon\anaconda3\lib\json\decoder.py", line 337, in decode
    obj, end = self.raw_decode(s, idx=_w(s, 0).end())
  File "C:\Users\malon\anaconda3\lib\json\decoder.py", line 355, in raw_decode
    raise JSONDecodeError("Expecting value", s, err.value) from None
json.decoder.JSONDecodeError: Expecting value: line 1 column 1 (char 0)

(base) C:\Users\malon\Desktop\Master\produccionML>
```

En el caso de las peticiones erróneas, el modelo tan solo y levanta mensaje de error cuando la inserción de datos no válidos se produce en aquellas variables que no son de tipo categórico, cómo en nuestro caso la variable que mide el año del accidente (C_YEAR). En el caso de las columnas con información categórica, los valores erróneos son interpretados como categorías adicionales y no invalidan el proceso de predicción provocando mensaje de error.

Además, es importante resaltar que, tanto en el método dentro de /predict como /predictGUI, es necesario realizar una transformación del formato de entrada de los datos para que el modelo pueda utilizarlos y, adicionalmente, también hemos optado por mantener las probabilidades de pertenecer a la clase 1, en vez de clasificarlas, puesto que esto puede dar más información al usuario y da lugar a que genere su propia interpretabilidad.

Dicho esto, en nuestro caso no se requiere de un preprocesado de datos (ej.-preprocesado de datos categóricos), puesto que nuestro modelo ya incorpora de manera nativa un preprocesador de los mismos.

Finalmente, debido a la forma que hemos construido el dataframe con los datos que recibimos en la request, el orden que deben de seguir las variables es el siguiente: 'C_YEAR', 'C_MNTH', 'C_WDAY', 'C_HOUR', 'C_WTHR', 'C_RALN', 'P_SEX', 'P_SAFE', 'P_USER', 'C_VEHS', 'P_AGE', 'V_ANTIG', 'P_PSN', 'C_CONF', 'C_TRAF', 'V_TYPE', 'C_RCFG', 'C_RSUR'. En caso de introducir información de una variable en otra (es decir, en distinto orden), esto puede dar lugar o no a un error de código, cómo hemos explicado antes en el caso de las entradas incoherentes en columnas categóricas y continuas, pero en la totalidad de las ocasiones invalida la predicción obtenida.