UNIVERSIDAD DE EXTREMADURA

Escuela Politécnica

Ingeniería Informática en Ingeniería del Software

Trabajo Fin de Grado

Herramienta para la aplicación de algoritmos de aprendizaje automático a través de APIs

Adrián Ruiz Parra

Julio, 2022

UNIVERSIDAD DE EXTREMADURA

Escuela Politécnica

Ingeniería Informática en Ingeniería del Software

Trabajo Fin de Grado

Herramienta para la aplicación de algoritmos de aprendizaje automático a través de APIs

Autor: Adrián Ruiz Parra

Tutor: José María Conejero Manzano

Cotutor: Francisco Javier Melchor González

ÍNDICE GENERAL DE CONTENIDOS

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

RESUMEN

En este trabajo se documenta la creación y desarrollo de una herramienta web (*Python Machine Learning REST API Generator*, abreviado como *PyMLAPIGen*).

Esta herramienta tiene el objetivo de permitir a cualquier usuario no experto en el dominio de *Machine Learning*, entrenar algoritmos a partir de un dataset de entrada y poder aplicar dichos algoritmos entrenados a nuevos datasets.

De esta forma, cualquier usuario podrá utilizar la herramienta para poder probar distintos algoritmos con diversos parámetros y así pulir y encontrar la mejor configuración para su dataset.

La herramienta ofrece una interfaz de usuario a través de una aplicación web hospedada en la máquina del usuario. Además, cuenta con una API de programación REST a través de la cual se puede realizar experimentos.

Un experimento genera una API REST con la que se puede interactuar a través de la herramienta web o los diversos *endpoints* de la API. Para generar un experimento, es necesario introducir un dataset, la categoría *Machine Learning* (clasificación, regresión o *clustering*), la columna objetivo del dataset para aprendizajes supervisados, el algoritmo y sus parámetros, los atributos para tener en cuenta del dataset y el tamaño del conjunto de entrenamiento.

Una vez introducido los parámetros requeridos, un modelo será entrenado y evaluado de forma que pueda ofrecer distintas métricas, gráficas y ser utilizado para nuevas predicciones.

Palabras clave: ciencia de datos, aprendizaje automático, *Machine Learning*, generación de APIs REST.

1. INTRODUCCIÓN

El aprendizaje automático [1] (del inglés, *Machine Learning*) es un subcampo de las ciencias de la computación y una rama de la inteligencia artificial, cuyo objetivo es desarrollar técnicas que permitan que las computadoras aprendan.

Lo que se desea conseguir con el aprendizaje automático es que las máquinas aprendan sin ser concretamente programadas para ello. Esto es posible gracias a que los ordenadores son capaces de identificar patrones entre los datos y así realizar predicciones.

Esta técnica es muy utilizada dentro del campo de la ciencia de datos [2] junto a otras como el análisis exploratorio, el aprendizaje profundo (*deep learning*), el procesamiento del lenguaje natural, la visualización de datos y el diseño experimental.

A través de este proyecto se ha tratado de facilitar a los usuarios la realización de experimentos de aprendizaje automático a través de una herramienta web. A partir de esta herramienta, los usuarios podrán realizar experimentos, compararlos y realizar nuevas predicciones.

Así, distintos usuarios podrán realizar sus propios experimentos sin necesidad de tener conocimientos avanzados en programación o aprendizaje automático.

Uno de los propósitos principales de la herramienta es que pueda ser genérica, de modo que permita poder realizar experimentos de aprendizaje automático a partir de cualquier dataset proporcionado.

1.1 Estructura del documento

Este documento está estructurado según los siguientes capítulos:

- En el Capítulo 2 se introducirán los distintos objetivos buscados con el proyecto.
- En el Capítulo 3 se describe el Estado del Arte. En él se introducirá más en detalle los diversos conceptos de la ciencia de datos y del aprendizaje automático, se mostrarán las aplicaciones similares a partir de las cuales tomar referencias y se valorarán las distintas alternativas tecnológicas para la construcción de la herramienta.
- En el Capítulo 4 se explicará la metodología definida llevada a cabo durante la elaboración del proyecto.
- En el Capítulo 5 se detallará el proceso de implementación y desarrollo de la aplicación final.
- En el Capítulo 6 se mostrarán los resultados del proyecto.
- En el Capítulo 7, se repasarán las conclusiones obtenidas durante la consecución del proyecto, así como los trabajos futuros y agradecimientos.
- Finalmente, se incluirán las referencias bibliográficas, un anexo referente al repositorio de la herramienta y un anexo correspondiente al manual de usuario de la aplicación.

2. OBJETIVOS

El principal objetivo del proyecto es la construcción de una herramienta capaz de generar APIs REST a partir de experimentos de aprendizaje automático realizados sobre cualquier dataset destinado a ello.

2.1 Objetivos parciales

Para la consecución de este objetivo principal, se establecieron los siguientes objetivos parciales:

- Conseguir ejecutar un experimento de aprendizaje automático vía web y API REST.
- Aplicar el modelo matemático entrenado del experimento a nuevos conjuntos de datos.
- Generar métricas a partir de los resultados del experimento.
- Desplegar una API REST a partir del experimento con diversos *endpoints* para interactuar con él.
- Ofrecer un conjunto de algoritmos diferentes para los experimentos.
- Poder realizar experimentos a partir de conjuntos de datos genéricos.
- Permitir al usuario diversas opciones de ajuste del experimento para el algoritmo del modelo matemático, el conjunto de datos y el propio experimento.

2.2 Pasos

Los pasos seguidos para completar los objetivos citados previamente son los siguientes:

- Lectura, documentación y realización de cursos acerca del campo de aprendizaje automático.
- Estudio de herramientas con propósito similar al planteado para este proyecto.

- Valoración de tecnologías y *frameworks* para la construcción de la herramienta, así como la metodología optada para llevar a cabo durante esta.
- Una vez seleccionada la tecnología, elaboración de un experimento sencillo para probar la viabilidad del proyecto.
- En base al experimento, comienzo de la construcción de la herramienta.
- Realizar una fase de pruebas para comprobar el buen funcionamiento de la herramienta. Para ello, se ha recopilado y utilizado una serie de datasets de prueba.
- Redacción de un manual de usuario y documentación del código escrito de la herramienta.
- Extracción de conclusiones acerca de la elaboración del proyecto.
- Elaboración de este presente documento y la defensa del proyecto.

3. ESTADO DEL ARTE

3.1 Definiciones

Tal y como se comentó en las anteriores secciones, el dominio del proyecto gira en torno al aprendizaje automático. Este concepto, además, forma parte de un campo interdisciplinario conocido como ciencia de datos.

Para comenzar esta sección de definiciones, se detallará el concepto de ciencia de datos.

3.1.1 Ciencia de datos

La ciencia de datos [3] (proveniente del término inglés, *Data Science)*, es un campo interdisciplinario que involucra métodos científicos, procesos y sistemas para extraer conocimiento o un mejor entendimiento de datos. Entre los campos que comprende la ciencia de datos [4], se incluyen estadísticas, métodos científicos, inteligencia artificial (abreviado como *IA*) y análisis de datos.

Extraer valor de los datos analizados tiene múltiples beneficios [5]. Por ejemplo, para las empresas, supone un apoyo para la toma de decisiones comerciales, planificación estratégica, mejoras de marketing, aumento de la eficiencia entre otros. Además, la ciencia de datos tiene beneficios en más sectores como el sanitario (ayudando al diagnóstico de enfermedades), el académico (monitorización del desempeño de estudiantes), el deportivo (planificación de estrategias de juego), el político (predicción de votos), entre otros.

Además, dentro de un proyecto de ciencia de datos, se incluyen técnicas [3] [5] como preparación de datos, minería de datos, modelado predictivo, análisis estadístico, analítica predictiva, aprendizaje automático, entre otros.

3.1.2 Inteligencia Artificial

La Inteligencia Artificial (también conocido como IA, *artificial intelligence* o *AI*), [6] [7] es una disciplina perteneciente a las ciencias de la computación que intenta replicar e imitar la inteligencia humana para realizar tareas.

La Inteligencia Artificial comprende un amplio conjunto de subcampos, como el aprendizaje, percepción, demostración de teoremas matemáticos, escritura de poesía… Una de las ramas principales es el mencionado Aprendizaje Automático.

3.1.3 Aprendizaje Automático

El Aprendizaje Automático (también conocido en inglés como *Machine Learning*) [1], es una rama de la Inteligencia Artificial cuyo objetivo es desarrollar técnicas que permitan que las computadoras aprendan. Es el [8] proceso a partir del cual se entrenan modelos matemáticos de datos para aprender y realizar predicciones sobre nuevos datos.

El aprendizaje automático [8] usa algoritmos para identificar patrones en los datos, y esos patrones luego se usan para crear un modelo de datos que pueda hacer predicciones.

Los algoritmos de aprendizaje automático [9] pueden clasificarse en función del tipo de aprendizaje que usen, formando así cinco grupos:

- **Aprendizaje supervisado**. En este tipo de aprendizaje se utiliza un grupo de ejemplos o datos etiquetados. Por etiquetados, [10] se entiende que, para cada entrada del conjunto de datos, se conozca el valor de su atributo objetivo o clase. Esto permitirá al algoritmo deducir una función capaz de predecir el atributo objetivo para nuevas entradas.

  Dentro del aprendizaje supervisado, existen dos grandes familias de algoritmos en función del tipo de dato del atributo objetivo:

  - **Clasificación**. En estos algoritmos el atributo objetivo es discreto o categórico.

    - **Regresión**. En estos algoritmos el atributo objetivo es continuo o numérico.
- **Aprendizaje no supervisado**. Los algoritmos de este tipo de aprendizaje, a diferencia del anterior, no disponen de etiquetas o atributos objetivo en los conjuntos de datos. El aprendizaje no supervisado [10] está dedicado a tareas de agrupamiento (***clustering***) cuyo objetivo es encontrar grupos similares en los conjuntos de datos.
- **Aprendizaje semisupervisado**. Este aprendizaje combina los algoritmos de los aprendizajes anteriores. Es decir, utilizan datos tanto etiquetados como no etiquetados para entrenar el modelo matemático. Normalmente, una pequeña cantidad de datos etiquetados frente a una gran cantidad de datos no etiquetados.
- **Aprendizaje por refuerzo**. En este aprendizaje los algoritmos aprenden a base de ensayo y error. Su información de entrada es el *feedback* o retroalimentación que obtiene del mundo exterior como respuesta a sus acciones. Esta retroalimentación puede ser positiva en el caso de éxitos (refuerzos o recompensas) o negativas en el caso de fracasos (castigos). Este aprendizaje forma parte de lo que se conoce como *Deep Learning* o aprendizaje profundo.

3.1.4 API

Una interfaz de programación de aplicaciones [11] (conocida por la sigla *API*, del inglés *application programming interface*), es un conjunto de subrutinas, funciones y procedimientos que ofrece una biblioteca para ser utilizada por otro software a través de una capa de abstracción. Permite [12] la integración del software de aplicaciones gracias a que habilita la comunicación entre componentes [13] a través de la definición de un conjunto de reglas y protocolos.

Usualmente la arquitectura de las API suele explicarse en términos de cliente y servidor. La aplicación que envía la solicitud se conoce como cliente mientras que la que responde se conoce como servidor.

Existen diversos tipos de API [13] en función de su arquitectura y su manera de funcionar. Entre ellas se encuentran:

- **API SOAP**. Estas API utilizan el protocolo simple de acceso a objetos. El cliente y servidor intercambian mensajes mediante *XML*.
- **API RPC**. También se conocen como llamadas a procedimientos remotos. El cliente completa una función (o procedimiento) en el servidor, y el servidor devuelve el resultado al cliente.
- **API WebSocket**. Utilizan objetos JSON para pasar datos. Estas API tienen un tipo de comunicación bidireccional entre cliente y servidor, de forma que el servidor puede enviar mensajes de devolución de llamada (*callbacks*) a los clientes conectados.
- **API REST**. Su nombre significa transferencia de estado representacional (del inglés, *representational state transfer*). Las API REST definen un conjunto de funciones *GET*, *POST, PUT*, *DELETE,* que los clientes utilizan para comunicarse con el servidor. De esta forma, los clientes y servidores intercambian datos mediante mensajes *HTTP*. Los servidores no guardan datos del cliente entre solicitudes, es decir, las API REST poseen ausencia de estado.

3.2 Aplicaciones similares

Con el fin de tomar referencias y conocer el estado actual de herramientas con propósito relacionado al de este proyecto, se realizó una búsqueda de aplicaciones similares y se analizó cada una de ellas.

3.2.1 Retool [14]

Retool es una plataforma *low-code* que facilita la creación de herramientas internas.

Es utilizado para crear aplicaciones mediante un sistema de bloques *drag-and-drop*. Es decir, es posible ensamblar aplicaciones mediante la combinación de bloques en cuestión de minutos.

Estas aplicaciones cuentan con conexiones a bases de datos y API, interfaces gráficas, consultas relacionales…

Además, Retool cuenta con utilidades. Estas utilidades son herramientas adicionales de distintos propósitos disponibles para los usuarios.

Con relación al objetivo del proyecto, se encuentran relacionados las siguientes utilidades:

- ***Generate API from CSV*** [15]. Esta herramienta permite subir un conjunto de datos en formato *CSV* y generar una API REST a partir de él. La API generada cuenta con métodos *GET*, *POST*, *PUT*, *DELETE*, y *PATCH*.
- ***Generate app from CSV*** [16]. A través de esta herramienta es posible generar una aplicación *CRUD* a partir de un conjunto de datos en formato *CSV*. Al tratarse de una aplicación *CRUD*, permite operaciones de inserción, consulta, actualización y borrado de datos.

3.2.2 Datasette [17]

Datasette es una multi-herramienta de código abierto utilizada para explorar y publicar datos. Permite a los usuarios tomar datos de cualquier tipo, analizarlos, explorarlos y publicarlos como un sitio web interactivo junto a una API.

Está enfocado a periodistas, conservadores de museos, archiveros, gobiernos, científicos, investigadores y cualquiera que desee compartir datos con el resto del mundo.

Para ponerlo en funcionamiento, es necesario proveerle de una base de datos *SQLite*.

La herramienta está escrita en el lenguaje de programación *Python*.

3.2.3 PythonOnWheels [18]

PythonOnWheels es un *framework* web del lenguaje de programación *Python* basado en los principios de *Ruby on Rails*.

Este *framework* ofrece un desarrollo rápido de aplicación no intrusivo. No necesita mucha configuración y evita el código repetitivo. Se limita a generar un modelo, un controlador y renderizar vistas.

Resulta interesante respecto al proyecto debido a su capacidad de generar servicios web, incluyendo entre ellos la generación de API REST a partir de bases de datos.

3.2.4 Eve [19]

Eve es un *framework* web de código abierto para el lenguaje de programación *Python*. Está sustentado sobre el *framework Flask* y la librería de validación *Cerberus*. Ofrece soporte nativo para bases de datos *MongoDB*, aunque también es compatible con bases de datos relacionales, *Elasticsearch* y *Neo4js* a través de extensiones de la comunidad.

Eve permite construir y desplegar servicios web RESTful personalizables a partir de la base de datos a la que esté conectada.

3.2.5 Tabla resumen

Una vez repasadas y estudiadas las herramientas con propósito similar al planteado para el proyecto, se introduce al lector una tabla resumiendo las principales características de cada herramienta.

Herramienta	Propósito	Relación
Retool: Generate API from CSV	Generar API REST a partir de un conjunto de datos CSV.	Poder generar una API REST completamente a partir de un conjunto de datos
Retool: Generate app from CSV	Generar una aplicación CRUD a partir de un conjunto de datos CSV.	Poder generar una aplicación completamente a partir de un conjunto de datos
Datasette	Exploración, análisis y publicación de datos a partir de una base de datos.	Generación de sitio web interactivo y API a partir de una base de datos.
PythonOnWheels	Desarrollo rápido de aplicación no intrusivo.	Generación de servicios web, incluyendo una API REST a partir de un conjunto de datos.
Eve	Construir y desplegar servicios web RESTful personalizables a partir de una base de datos.	Generación de API REST a partir de un conjunto de datos.

Tabla 1: Herramientas Similares

3.3 Alternativas tecnológicas de aprendizaje automático

Tras haber tomado referencias acerca de herramientas similares, se valoró las distintas alternativas tecnológicas para llevar a cabo los experimentos de aprendizaje automático.

Para ello, se comenzará introduciendo las distintas tecnologías que permiten realizar experimentos de aprendizaje automático.

3.3.1 IBM Machine Learning [20]

De mano de la empresa multinacional estadounidense de tecnología y consultoría International Business Machines (abreviado como IBM) existen varias alternativas para la realización de experimentos de aprendizaje automático.

Gracias al suite de herramientas de Machine Learning de IBM es posible combinar distintos productos, como IBM Watson Studio, IBM Watson Machine Learning, IBM Watson OpenScale e IBM Cloud Pak for Data.

IBM Machine Learning tiene un coste de $0.50/CUH (horas de unidad de capacidad consumidas) junto a un plan lite gratuito de 20 horas de unidad de capacidad.

3.3.2 Google Cloud [21]

La plataforma de computación en la nube Google Cloud ofrece diversas alternativas para realizar experimentos de aprendizaje automático a través de sus productos.

Los principales productos que ofrece Google Cloud para aprendizaje automático son Google Cloud AutoML, Google Cloud AI Platform, Google Cloud Vision AI, Google Cloud Text-to-Speech, Google Cloud Natural Language entre otros.

Además, su plataforma escala en función de la experiencia de usuario, de forma que es posible crear experimentos a partir de simples *point and click* y también pulirlos hasta conseguir una avanzada optimización de modelos. Proveen diversas herramientas basadas en código y no basadas en código para elegir la que más se ajuste al usuario.

Google Cloud AI Platform cuesta alrededor de $0.19/hora y ofrece un crédito gratuito de 300$ para los primeros 90 días.

3.3.3 Microsoft Azure Machine Learning [22]

Microsoft Azure Machine Learning permite a los usuarios construir, entrenar y desplegar modelos de aprendizaje automático.

Es posible crear un modelo desde cero en Azure Machine Learning o utilizar otro creado desde una plataforma de código como Pytorch, TensorFlow o Scikit-Learn. También ofrece herramientas de MLOps que ayudan a supervisar, volver a entrenar y volver a implementar modelos.

Azure Machine Learning cuesta sobre $0.333/hora y ofrece 12 meses gratis junto a un crédito de 200$ para los primeros 30 días.

3.3.4 Amazon Machine Learning [23]

Amazon Machine Learning da la capacidad a sus usuarios de construir, desplegar y poner en ejecución aplicaciones de aprendizaje automático en la nube a través de AWS.

Ofrece distintos tipos de aprendizaje, como el aprendizaje automático, procesado de lenguaje natural (NLP) o aprendizaje profundo basado en el reconocimiento de imágenes.

Amazon Machine Learning tiene un coste de $0.42/hora y ofrece un período de un año gratuito de prueba.

3.3.5 TensorFlow [24]

Tensorflow es una plataforma de código abierto de extremo a extremo para el aprendizaje automático. Fue creado por Google Brain team y publicado en 2015.

Es multiplataforma ya que TensorFlow ofrece una colección de flujos de trabajo para desarrollar y entrenar modelos mediante Python o Javascript. Además, puede implementar los modelos con facilidad en la nube, de forma local, en el navegador o en el dispositivo.

Tensorflow es gratuito.

3.3.6 Scikit-Learn [25]

Scikit-learn es una librería de análisis de datos para *Python* de código abierto el cual ofrece varias alternativas para llevar a cabo experimentos de aprendizaje automático.

El proyecto fue comenzado en un evento de Google Summer of Code por el científico de datos David Cournapeau. Su nombre proviene de SciPy Toolkit, de la librería de herramientas y algoritmos matemáticos SciPy.

Es una librería muy popular, contando con una extensa documentación y una gran comunidad detrás. Además, al tratarse de una librería de alto nivel, permite crear modelos matemáticos de aprendizaje automático y utilizarlos en apenas pocas líneas de código.

También destaca que es una librería versátil e integrable con otras librerías de Python conocidas dentro del campo de ciencia de datos, como pueden ser Matplotlib [26] (para gráficos y visualización de datos), NumPy [27] (para vectorización) y pandas [28] (análisis y manipulación de datos).

Scikit-Learn es gratuito.

3.3.7 Tabla resumen

Tecnología	Autor	Precio	Ventajas	Desventajas
IBM Machine Learning	IBM	$0.50 / CUH	• Creación de modelos simples con drag-and-drop. • Extensa documentación junto una API fácil de usar. • Fácil de utilizar.	• Cada servicio debe lanzarse en pestañas distintas. • Restricciones a la hora de ajustar parámetros durante el entrenamiento de modelos.
Google Cloud	Google Cloud	$0.19 / hora	• Interfaz de usuario fácil de utilizar. • Compatibilidad con TPUs y TensorFlow.	• Limitado a 25 modelos máximos ejecutándose concurrentemente. • Difícil personalización.
Microsoft Azure Machine Learning	Microsoft	$0.333 / hora	• Adaptable gracias a la inclusión de código. • Cuenta con modelos de ejemplo ya entrenados. • Periodo de prueba extenso sumado a créditos extra.	• Inclusión de código difícil de entender. • La ejecución de modelos pequeños es lenta. • Número de modelos disponibles limitado.

Amazon Machine Learning	Amazon	$0.42 / hora	• Manejo de conjuntos de datos grandes gracias al uso de múltiples servidores. • Trackeo visual del desarrollo del modelo. • Muy personalizable.	• Más apropiado para aquellos que se encuentren en el ecosistema de Amazon. • Requiere de conocimientos de programación avanzados.
TensorFlow	Google	Gratuito	• Es gratuito. • Compatible con muchos lenguajes (C++, JavaScript, Python…)	• Algunos mensajes de error son difíciles de comprender. • No tiene el mejor rendimiento ni la mayor usabilidad.
Scikit-Learn	David Cournapeau	Gratuito	• Es gratuito. • Fácil de usar y versátil. • Compatible con otras librerías de ciencia de datos populares de Python. • Cuenta con una extensa documentación y gran comunidad	• No es la mejor opción para el aprendizaje profundo. • Baja flexibilidad. • No ofrece soporte para la aceleración de hardware.

Tabla 2: Tecnologías de Aprendizaje Automático

3.3.8 Tecnología escogida

La tecnología seleccionada finalmente fue **Scikit-Learn**.

Las razones de su elección son:

- Al tratarse de una librería de Python, es muy compatible con la utilización de frameworks web, herramientas y librerías del mismo lenguaje. Gracias a ello, se simplifica significativamente el desarrollo de la herramienta a la vez que supone un gran potencial.
- Es una librería de código abierto y gratuita, por lo que no existirán problemas referentes a pagos.
- Scikit-Learn es una librería de alto nivel. Gracias a ello, aporta un nivel de abstracción en cuanto a la construcción, entrenamiento y ejecución de modelos matemáticos de aprendizaje automático.
- Tiene alta compatibilidad con las librerías Matplotlib, NumPy y pandas, librerías muy interesantes de utilizar en la herramienta a construir.
- Cuenta con una extensa documentación y una gran comunidad al tratarse de una de las principales alternativas de aprendizaje automático.
- Además de modelos, Scikit-Learn ofrece funcionalidades interesantes relacionadas al aprendizaje automático como preprocesado de los datos, métricas, pipelines…

3.4 Alternativas tecnológicas de aplicación web

Una vez elegida la tecnología que permita realizar experimentos de aprendizaje automático, es necesario seleccionar bajo qué tecnología comenzar la construcción de la aplicación.

Al haber optado por la librería Scikit-Learn como alternativa tecnológica para el aprendizaje automático, es recomendable que la herramienta web se construya utilizando *Python* como lenguaje de programación para tener así una alta compatibilidad con la librería, ya que se trata de una librería construida bajo el mismo lenguaje.

Dentro del lenguaje de programación *Python*, existen diversas tecnologías y *frameworks* que permiten la construcción de aplicaciones web. En esta sección se repasarán los principales candidatos para la elaboración de la herramienta.

3.4.1 Jupyter Kernel Gateway [29]

Jupyter Kernel Gateway es un servidor web que ofrece accesos sin cabecera a kernels de Jupyter.

Jupyter (Jupyter Notebook), a su vez, es una aplicación web de código abierto a partir de la cual puede ser usada para crear y compartir documentos con código en vivo, ecuaciones, visualizaciones, texto… Soporta muchos lenguajes de programación, como R, Julia, Scala y Python.

Jupyter Kernel Gateway permite desplegar una API REST a partir de un *notebook* de Jupyter. De esta forma, los usuarios pueden comunicarse con la API en forma de llamadas REST.

3.4.2 Flask [30]

Flask es un *microframework* web del lenguaje de programación *Python* que permite desarrollar aplicaciones web de manera sencilla.

Utiliza el motor de plantillas Jinja [31] para renderizar documentos HTML y es compatible con WSGI [32] gracias a la librería Werkzeug [33].

Está diseñado para ser minimalista. Es decir, el *framework* se limita al núcleo de la aplicación, aportando menos funcionalidades que el resto de *frameworks* disponibles.

Para suplir lo citado en el anterior párrafo, Flask da soporte a extensiones. Estas extensiones aportan a Flask numerosas funcionalidades convirtiendo este *framework* en uno muy versátil y escalable.

Está muy bien documentado, es gratuito, tiene una curva baja de aprendizaje y se encuentra apoyado por una gran comunidad que publica diariamente extensiones aumentando las funcionalidades de Flask.

Algunas de las páginas web sustentadas con Flask son LinkedIn y Pinterest.

3.4.3 Django [34]

Django es un *framework* web del lenguaje de programación *Python* de alto nivel que alienta el desarrollo rápido y limpio junto a un diseño pragmático.

Permite desarrollar sitios web completos, seguros, versátiles, escalables, mantenibles, portables y rápidos. Django se encarga de gran parte de las funciones del desarrollo web, relevando al usuario únicamente a desarrollar su idea sin complicaciones.

Es gratuito y de código abierto. Cuenta con una vasta comunidad activa, una gran documentación, extensiones y opciones de soporte gratuitos y de pago.

Algunas de las páginas web sustentadas con Django son Instagram y Spotify.

3.4.4 FastAPI [35]

FastAPI es un *framework* web del lenguaje de programación *Python* diseñado para construir y desplegar API REST.

Está basado en Pydantic [36] y es compatible con programación asíncrona. Puede ejecutarse con Uvicorn [37] y Gunicorn [38].

Algunos de los puntos fuertes de FastAPI es que tiene un buen rendimiento, un desarrollo ágil, reduce errores, intuitivo, fácil de utilizar, minimiza la duplicación de código, robusto y está basado en estándares como OpenAPI [39] (conocido previamente como Swagger), OAuth 2.0 [40] y JSON Schema [41].

Algunas de las compañías más reconocidas que utilizan FastAPI son Netflix, Uber y Microsoft.

3.4.5 Tabla resumen

Tecnología	Ventajas	Desventajas
Jupyter Kernel Gateway	• Compatibilidad con Jupyter Notebook. • Fácil de utilizar.	• No ofrece interfaz gráfica. • Desarrollo lento.
Flask	• Flexibilidad gracias a que cuenta con una gran cantidad de extensiones. • Es posible modificar y ajustar la mayoría de partes de Flask. • Curva de aprendizaje baja. • Cuenta con una gran comunidad y amplia documentación.	• Depende de las extensiones, las cuales pueden involucrar fallos de seguridad. • No sigue ningún estándar, por lo que emigrar de Flask a otro framework puede resultar difícil. • Maneja las peticiones una a una por turnos, por lo que puede tener un rendimiento inferior.
Django	• Estructura robusta. • Comunidad activa y buena documentación.	• No es apropiado para proyectos pequeños. • Curva de aprendizaje alta. • Es considerado monolítico.
FastAPI	• Construido bajo estándares JSON Schema, OAuth 2.0 y OpenAPI. • Fácil despliegue de API.	• Tecnología poco madura. • La comunidad es pequeña comparado al resto de tecnologías.

Tabla 3: Tecnologías de Aplicación Web

3.4.6 Tecnología escogida

Tras la elaboración de demostraciones prácticas con las mencionadas tecnologías y reuniones con los tutores, la tecnología seleccionada finalmente para la herramienta fue **Flask**.

Las razones de su elección son:

- Es un *framework* minimalista y muy flexible. Esto permitirá incorporar los experimentos de aprendizaje automático fácilmente.
- Tiene una gran documentación y comunidad detrás. Gracias a ello, se cuenta con muchas referencias externas y soluciones a posibles problemas que puedan surgir durante el desarrollo.
- Cuenta con una gran cantidad de extensiones que ayudarán a expandir las funcionalidades de la aplicación según sean requeridas.
- Ofrece facilidades para realizar tests, tanto unitarios como de integración y funcionales.
- Incluye un servidor web de desarrollo, por lo que no requiere de una infraestructura para probar la aplicación.
- Gracias a extensiones, Flask tiene la capacidad de desplegar API REST.
- Utiliza el motor de plantillas Jinja [31], lo cual resulta beneficioso a la hora de renderizar documentos HTML con parámetros de entrada. Además, presenta compatibilidad con WSGI [32].
- Es un *framework* gratuito y de código abierto amparado bajo una licencia BSD.

4. METODOLOGÍA

Una vez seleccionado el entorno de trabajo base (lenguaje *Python* y *framework* Flask para la arquitectura de la herramienta y librería Scikit-Learn para experimentos de aprendizaje automático), dio comienzo el desarrollo de la herramienta para este proyecto y con ello la metodología a utilizar.

4.1 Funciones núcleo de la aplicación

En primer lugar, se enfocó en las funciones núcleo de la aplicación, que corresponden a la generación de API REST a partir de experimentos de aprendizaje automático. Estos experimentos, además, debían poder realizarse a partir de cualquier dataset válido (en formato *CSV*) y ofrecer un considerable surtido de algoritmos de aprendizaje supervisado (clasificación y regresión) y aprendizaje no supervisado (agrupamiento) para entrenar, evaluar y utilizar el modelo matemático entrenado para realizar predicciones sobre nuevos conjuntos de datos.

Estas funciones núcleo fueron desarrolladas utilizando Jupyter Notebook y la librería Scikit-Learn y fueron desplegadas como API REST gracias a la tecnología comentada en el apartado 3.4.1 Jupyter Kernel Gateway [29]. Se realizo así debido a que, en este punto, no era necesario ninguna interfaz, por lo que se agilizó el desarrollo.

4.2 Metodología de la construcción de la aplicación

Ya definidas, analizadas, implementadas, probadas y documentadas las funciones núcleo, se inició el desarrollo de la herramienta final con el *framework* Flask.

La metodología utilizada para su construcción fue una metodología ágil [42]. Está basada en un desarrollo iterativo e incremental en la cual se divide el ciclo de vida del desarrollo del software en iteraciones. Estas iteraciones, a su vez [43], se dividen en planificación, análisis, diseño, implementación, pruebas y documentación.

Siguiendo esta metodología, durante el desarrollo, se fueron planificando, analizando y diseñando requisitos de la aplicación durante reuniones con los tutores del proyecto. Los requisitos finales se encuentran descritos en el apartado 5.6 Funcionalidades.

Una vez definidos estos requisitos, se fueron implementando, probando y documentando de forma que, una vez finalizados la mayor parte de requisitos pendientes, se volviese a tener una reunión con los tutores para revisar, confirmar y validar los requisitos completados y planificar requisitos nuevos o modificar los actuales.

Tras construir la primera versión estable de la herramienta, tuvo lugar una fase de pruebas en la cual la aplicación fue probada con un gran conjunto de datasets diferentes. Además, para aumentar la fiabilidad de su estructura, se construyeron pruebas unitarias, de integración y funcionales. A partir de esta fase de pruebas, también surgieron nuevos requisitos que fueron implementados.

En las fases finales del proyecto, se redactó la documentación y manual de usuario de la herramienta para orientar a cualquier usuario que quiera hacer uso de ella.

4.3 Tiempo empleado

El tiempo empleado para el desarrollo de este proyecto se puede consultar en la siguiente tabla:

Motivo	**Tiempo (h)**
Documentación acerca de aprendizaje automático	40
Documentación acerca de aplicaciones similares	12
Documentación acerca de tecnologías de desarrollo de aplicaciones web	26
Desarrollo de las funciones núcleo de la aplicación	20
Construcción herramienta	113
Pruebas de la herramienta	15
Documentación y manual de usuario de la herramienta	24
Redacción de memoria	45
Reuniones con los tutores	7
Total	**302**

Tabla 4: Tiempo empleado en el proyecto

5. IMPLEMENTACIÓN Y DESARROLLO

En este apartado se destacarán las condiciones de desarrollo de la herramienta final y el transcurso de esta a lo largo del tiempo.

5.1 Entorno de trabajo

El entorno de trabajo del desarrollo de la aplicación fue el siguiente:

- **IDE:** *Visual Studio Code* [44].
- **Prueba de API REST:** *Postman* [45]
- **Repositorio de la herramienta:** Repositorio privado en *GitHub* [46].
- **Lenguajes utilizados.**
    - **Python**. (Lógica de dominio o *backend*, paquete herramienta)
    - **HTML, JavaScript y CSS**. (Presentación o *frontend*).
    - **Dockerfile**. (Para contenedores).
    - **Docstring**. (Documentación de Código Python)
    - **ReStructuredText**. (Lenguaje utilizado por el motor Sphinx para la generación del manual de usuario).
- **Framework backend:** *Flask* [30].
- **Framework frontend:** *Bootstrap* [47].
- **Aprendizaje automático:** Librería *Scikit-Learn* [25].
- **Generación del manual de usuario:** *Sphinx* [48].
- **Comunicación con tutores.**
    - **Escrito:** Correo electrónico y Trello [49].
    - **Reuniones:** Zoom [50].

5.2 Arquitectura

La aplicación final está estructurada como un paquete de Python instalable [51] [52].

5.2.1 Estructura de directorios del paquete

La estructura de directorios es la siguiente:

- */docs/*. Directorio correspondiente a todo aquello relacionado con la documentación y manual de usuario. Sphinx [48] es ejecutado en este directorio.
- */pymlapigen/*. Aplicación web desarrollada.
- */tests/*. Directorio correspondiente a los diversos tests desarrollados para la herramienta.
- *Dockerfile*. Fichero Docker para la ejecución de la aplicación en contenedores.
- *MANIFEST.in*. Fichero de comandos adicionales para la instalación de la herramienta.
- *requirements.txt*. Fichero con las librerías de *Python* necesarias para ejecutar la aplicación.
- *run.py*. Script de *Python* que manda a ejecución la aplicación.
- *setup.py.* Fichero de instalación del paquete de la aplicación.

5.2.1.1 Estructura de directorios de la aplicación

Dentro de la aplicación, en el directorio */pymlapigen/*, se encuentra la estructura típica de las aplicaciones creadas con el *framework* Flask [51].

- */static/.* Ficheros estáticos de la aplicación web. Contiene ficheros *CSS*, *JavaScript*, *assets* y documentos subidos por el usuario.
- */templates/*. Plantillas *HTML* listas para ser renderizadas utilizando el motor *Jinja* [31].
- *__init__.py*. Fichero principal del módulo.
- *api_generator.py*. Implementa la clase *API_Generator*. Representa un experimento de aprendizaje automático.
- *cli.py*. Componente de *Python* encargado de ejecutar la aplicación desde consola.
- *config.py.* Fichero de configuración del *framework* Flask.

- *routes.py*. Se encarga de la lógica de las vistas.

5.2.2 Componentes de la aplicación

Los componentes desarrollados para la herramienta son los siguientes:

- *API_Generator*: este componente representa un experimento de aprendizaje automático. Aúna las funciones núcleo de la aplicación desarrolladas previamente a la construcción de la herramienta. Este componente ofrece diferentes funciones y métodos para ajustar, entrenar, evaluar un modelo matemático y realizar nuevas predicciones.
- *Routes*: este componente, por otro lado, se encarga de procesar las peticiones del usuario devolviendo, o bien documentos *HTML* a través del motor de renderización de plantillas *Jinja* o bien objetos *JSON* a través de los distintos *endpoints* de la aplicación. Implementa el *framework* Flask [30] para gestionar las peticiones.

Estos componentes utilizan una serie de librerías externas para llevar a cabo sus funciones. En la Figura 1, se muestran qué librerías utiliza cada uno. Por un lado, el componente *Routes* usa los módulos de Flask [51] para atender a las peticiones del usuario. Por el otro lado, el componente *API_Generator* emplea la librería Scikit-Learn para llevar a cabo los experimentos de aprendizaje automático. Las funciones del resto de librerías usadas se pueden consultar en el apartado 5.4 Librerías utilizadas.

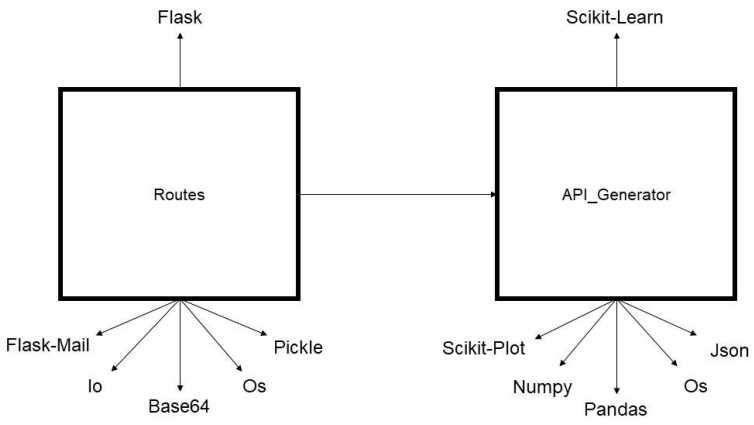


Figura 1: Librerías externas usadas por componentes

5.2.3 Experimentos de aprendizaje automático

Tal y como se mencionó previamente, el componente *API_Generator* es el responsable de llevar a cabo y gestionar los experimentos de aprendizaje automático. De esta forma, una instancia de este componente representa un experimento.

Un experimento está formado por los siguientes elementos:

- **Conjunto de datos**. Utilizado para definir el dominio del experimento y que el modelo matemático aprenda. Este conjunto es dividido a su vez en dos subconjuntos, un subconjunto de entrenamiento para entrenar el modelo y otro de test para evaluar su rendimiento. Este elemento es implementado como un *DataFrame* (estructura de datos de dos dimensiones como tablas), clase importada de la librería de análisis y manipulación de datos, pandas [28].
- **Modelo matemático**. Empleado para realizar predicciones sobre nuevos datos del dominio establecido, utilizando el algoritmo seleccionado por el usuario. Para poder desempeñar esta función, previamente es entrenado a partir del subconjunto de entrenamiento del conjunto de datos y evaluado con el subconjunto de test. Los

modelos y los algoritmos son provistos por la librería de aprendizaje automático, Scikit-Learn [25].

5.2.4 Relaciones

En la Figura 2 se introduce un esquema con las relaciones existen que entre los componentes de la aplicación citados anteriormente y como interactúan con las peticiones del usuario.

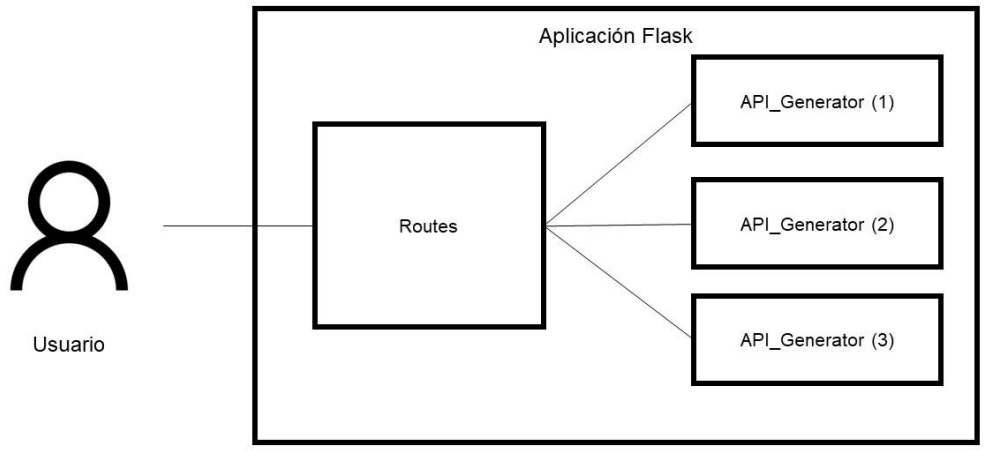


Figura 2: Esquema de relaciones

Se pueden apreciar las siguientes relaciones:

- El usuario manda sus peticiones a la aplicación y estas son procesadas por el componente *Routes*.
- El componente *Routes* recibe las peticiones del usuario y las procesa. Además, cuenta con una referencia a cada experimento de aprendizaje automático que se encuentre en memoria (componentes *API_Generator*). Normalmente, una petición de usuario involucra la

ejecución de algún método de *API_Generator* o la creación/destrucción de una instancia.

- Una vez el componente *Routes* finaliza el procesamiento de la petición, esta le devuelve una respuesta al usuario en forma de:
    - Documento *HTML* si fue realizada a partir de la aplicación web.
    - Objeto *JSON* si, en cambio, fue realizada a partir de un *endpoint* de alguna API REST.

5.2.5 Principales flujos

En esta sección, se introducirán los flujos de trabajo de las principales funcionalidades de la aplicación.

- **Consultar experimento**

  El usuario puede consultar los distintos experimentos de aprendizaje automático realizados tanto a través de un navegador como una petición *HTTP GET* a la API REST.

  La consulta puede estar relacionada acerca del conjunto de datos, las métricas, parámetros o gráficos del experimento.

  1. El usuario realiza la consulta acerca del experimento que desea.
  2. El componente *Routes* busca el experimento solicitado dentro de los que se encuentran en memoria.
     a. En caso de encontrarlo, pide la información que quiere consultar el usuario a la instancia *API_Generator* y se la devuelve a éste, ya sea en forma de página web o como objeto JSON en el cuerpo del resultado de la petición *HTTP GET*.
     b. En caso de no encontrarlo, informa al usuario de que el experimento no existe.

  En la Figura 3 se introduce un diagrama de secuencia que representa el flujo descrito.

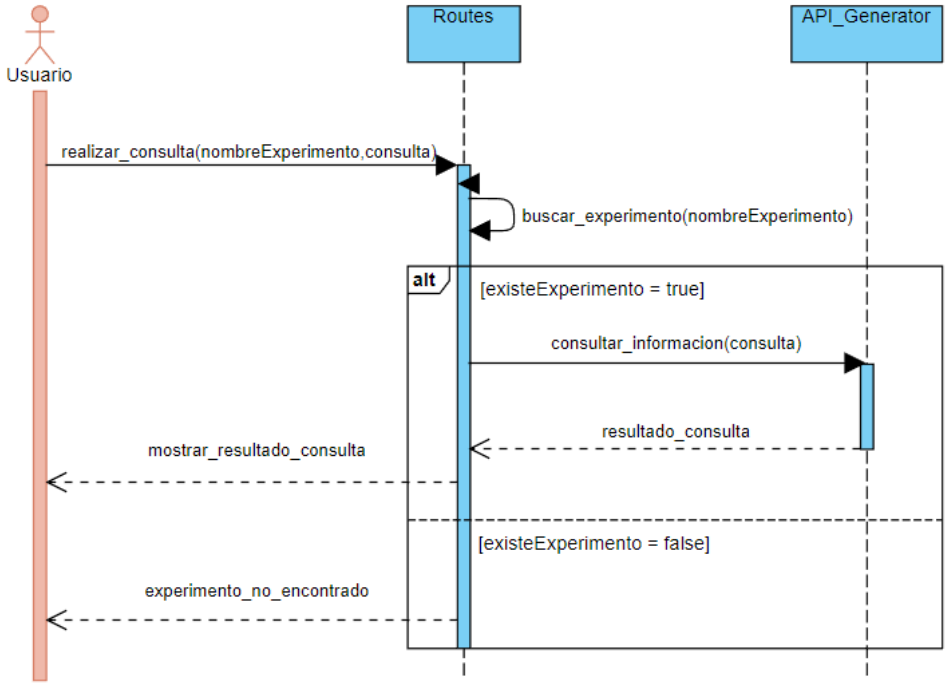


Figura 3: Diagrama de secuencia de consultar experimento

- **Realizar experimento**

  El usuario puede realizar un experimento de aprendizaje automático tanto a través de un navegador como una petición *HTTP POST* a la API REST.

  Para llevar a cabo el experimento, es necesario introducir el nombre, proveer un conjunto de datos, seleccionar el algoritmo del modelo matemático y ajustar los parámetros. Estos elementos son provistos a partir de un formulario en la aplicación web o como JSON en el cuerpo de la petición *HTTP*.

  1. El usuario provee los elementos requeridos para el experimento y solicita su realización.
  2. El componente *Routes* crea una instancia de *API_Generator* utilizando los parámetros introducidos por el usuario.

3. Una vez instanciado, el componente *API_Generator* entrena y evalúa el modelo matemático del experimento. Cuando finaliza, notifica a *Routes*.
4. Si el usuario marcó la opción de recibir un correo electrónico de notificación, el componente *Routes* enviará un correo al email proporcionado utilizando la extensión de Flask, Flask-Mail [53].
5. Finalmente, el componente *Routes* genera una página *Home* del experimento y se la envía al usuario o responde a la petición *HTTP* con los *endpoints* de la API desplegada del experimento.

En la Figura 4 se introduce un diagrama de secuencia que representa el flujo descrito.

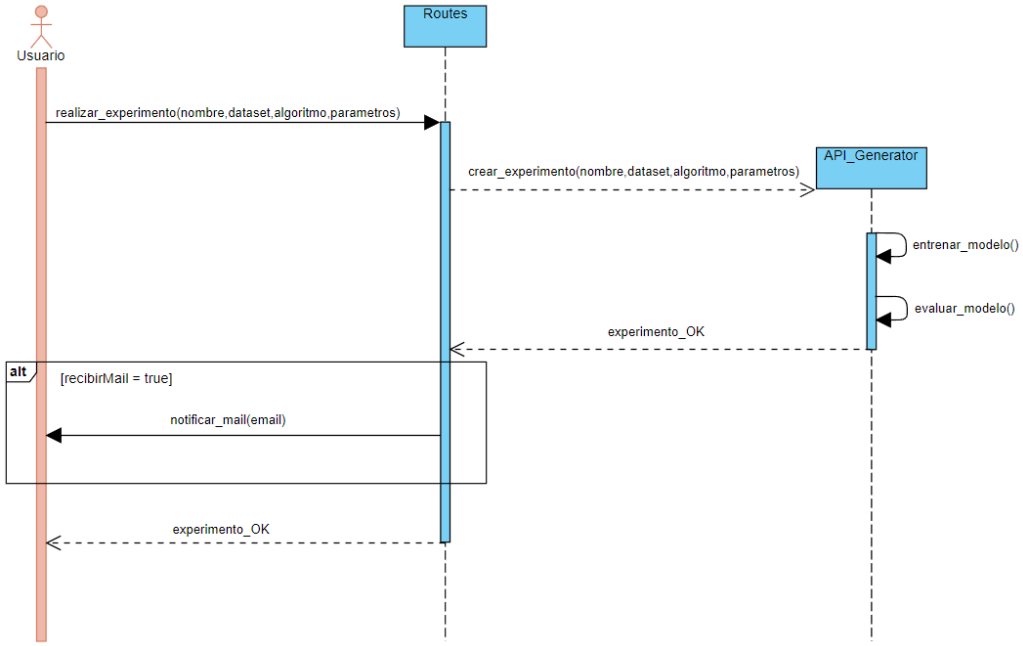


Figura 4: Diagrama de secuencia de realizar experimento de aprendizaje automático

- **Predecir nuevas entradas de datos**

  Una vez un experimento completa el entrenamiento y evaluación de su modelo matemático, el usuario puede utilizarlo para realizar predicciones en entradas de datos nuevas.

  Para ello, el usuario debe introducir las entradas de datos a través de un formulario en la aplicación web o como cuerpo de una petición *HTTP POST*.

  1. El usuario provee las entradas de datos nuevas al experimento al que desea realizar las predicciones.
  2. El componente *Routes* busca el experimento solicitado dentro de los que se encuentran en memoria.
     a. En caso de encontrarlo:
        i. *Routes* pide a la instancia *API_Generator* que realice la predicción sobre los datos provistos por el usuario utilizando su modelo matemático.
        ii. *API_Generator* lleva a cabo la predicción y comunica al componente *Routes* el resultado.
        iii. Finalmente, *Routes* transmite al usuario el resultado de la predicción.
     b. En caso de no encontrarlo, informa al usuario de que el experimento no existe.

  En la Figura 5 se introduce un diagrama de secuencia que representa el flujo descrito.

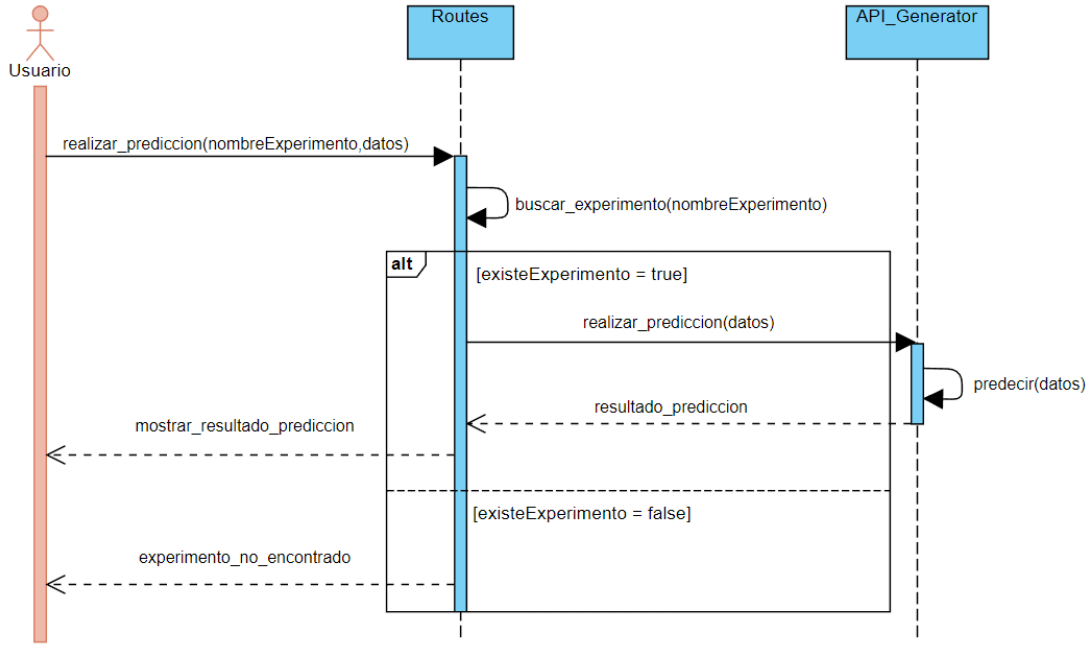


Figura 5: Diagrama de secuencia de realizar predicciones

5.3 Interfaz gráfica

La interfaz gráfica de la aplicación web se lleva a cabo gracias al motor de plantillas que utiliza el *framework* utilizado Flask, Jinja [31].

Jinja es un motor de plantillas rápido, expresivo y extensible. Permite escribir código *Python* a través de ciertas palabras reservadas de su sintaxis.

Dentro de sus funcionalidades, destaca la herencia de plantillas a través de bloques, flujos condicionales, bucles, depuración de errores, filtros, tests, funciones…

Las plantillas que utilizan tienen la extensión *HTML*. Consisten en documentos *HTML* los cuales incluyen fragmentos específicos cuya sintaxis concreta es reconocida por el motor.

Una vez una petición de usuario es procesada en la aplicación web, cuando se va a realizar una respuesta, Flask introduce los parámetros de entrada necesarios a Jinja (entre ellos plantillas y variables) para que éste renderice

un documento HTML válido para que sea finalmente entregado al usuario como respuesta a su petición inicial.

Mencionar también el uso del *framework* de *frontend* Bootstrap [47] para maquetar el diseño visual de la aplicación, incrementar la velocidad de su desarrollo y otorgarle la propiedad de ser adaptable (*Responsive Design*) [54].

5.4 Librerías utilizadas

Python, el lenguaje de programación seleccionado para construir la herramienta, es conocido por su gran cantidad de librerías y módulos públicos de todo tipo.

Existe una biblioteca estándar de módulos incluida en la instalación base del lenguaje *Python*, aunque también hay disponible una amplia colección en la web por parte de la comunidad.

Además, el *framework* Flask cuenta con una gran cantidad de extensiones en forma de librerías.

Para la construcción de la herramienta, se ha hecho uso de una variedad de librerías con el objetivo de implementar las funcionalidades propuestas.

En la siguiente tabla se repasarán todas las librerías utilizadas por la aplicación y su función dentro de ella.

Librería	**Función**
Api-Generator	Módulo local. Manipular y gestionar un experimento de aprendizaje automático.
Flask [30]	Framework web utilizado para la aplicación.
Flask-Mail [53]	Extensión de Flask. Aporta la funcionalidad de poder enviar correos electrónicos.
Io [55]**, Base64** [56]	Guardar los gráficos generados en un búfer temporal.
Json [57]	Operaciones relacionadas a objetos JSON.
Numpy [27]	Vectorización, casting y algunas operaciones matemáticas.
Os [58]	Operaciones relacionados al sistema operativo.
Pickle [59]	Exportar e importar experimentos de aprendizaje automático serializados.
Pandas [28]	Manipular los conjuntos de datos gracias a sus conocidos *DataFrame*.
Scikit-plot [60]	Generar gráficos de los experimentos de aprendizaje automático.
Sklearn [25]	Experimentos de aprendizaje automático.
Sys [61]	Obtener parámetros de entrada

Tabla 5: Librerías Python utilizadas

5.4.1 Librería Scikit-Learn

Gracias a la librería seleccionada Scikit-Learn [25], ha sido posible implementar la lógica necesaria para poder realizar experimentos de aprendizaje automático.

La librería incluye diversos recursos que permiten realizar experimentos. Además, provee de clases, funciones y métodos complementarios que enriquecen el aprendizaje.

Exceptuando los algoritmos, se han utilizado los siguientes módulos de Scikit-Learn:

Módulo	**Función**
sklearn.model_selection	Dividir el conjunto de datos de entrada en un conjunto de entrenamiento y otro de evaluación.
sklearn.metrics	Evaluar el modelo entrenado de forma que pueda mostrar métricas al usuario para así valorar el éxito del experimento.
sklearn.preprocessing	Aplicar la función *One-Hot-Encoding* a los datos categóricos para transformarlos en valores numéricos aptos para entrenar un modelo matemático.

Tabla 6: Módulos utilizados de Scikit-Learn (exceptuando algoritmos)

Por otra parte, de los algoritmos de aprendizaje automático que ofrece Scikit-Learn, se han incluido en la herramienta los siguientes:

Algoritmo	**Tipo de aprendizaje**	**Módulo**	**Clase**
Gaussian Naive Bayes	Supervisado (Clasificación)	*sklearn.naive_bayes*	GaussianNB
C-Support Vector Classification	Supervisado (Clasificación)	*sklearn.neighbors*	SVC
K-nearest neighbors	Supervisado (Clasificación)	*sklearn.svm*	KNeighborsClassifier
Decision Tree	Supervisado (Clasificación)	*sklearn.tree*	DecisionTreeClassifier
Random Forest	Supervisado (Clasificación)	*sklearn.ensemble*	RandomForestClassifier
Linear Regression	Supervisado (Regresión)	*sklearn.linear_model*	LinearRegression
Epsilon-Support Vector Regression	Supervisado (Regresión)	*sklearn.svm*	SVR
Stochastic Gradient Descent	Supervisado (Regresión)	*sklearn.linear_model*	SGDRegressor
Kernel Ridge	Supervisado (Regresión)	*sklearn.kernel_ridge*	KernelRidge
Gradient Boosting	Supervisado (Regresión)	*sklearn.ensemble*	GradientBoostingRegressor
K-Means	No supervisado (Agrupamiento)	*sklearn.cluster*	KMeans
Affinity Propagation	No supervisado (Agrupamiento)	*sklearn.cluster*	AffinityPropagation
Mean Shift	No supervisado (Agrupamiento)	*sklearn.cluster*	MeanShift
Mini-Batch K-Means	No supervisado (Agrupamiento)	*sklearn.cluster*	MiniBatchKMeans

Tabla 7: Algoritmos de aprendizaje automático importados de Scikit-Learn

5.5 Conjuntos de datos

Para realizar un experimento de aprendizaje automático, es necesario partir de un conjunto de datos (en inglés, *dataset*). Es a partir de estos datos de los cuales un modelo matemático aprende utilizando un algoritmo.

En proyectos de ciencias de datos, la importancia del conjunto de datos es alta, ya que es uno de los factores más importantes a la hora de decantar el éxito del experimento.

Estos conjuntos de datos consisten en tablas. Cada columna representa un atributo de la información y cada fila representa una entrada de información. En experimentos de aprendizaje supervisado, una de las columnas será considerada el atributo objetivo, y será la que el modelo deberá predecir en nuevas entradas.

En la aplicación web, se utilizan conjuntos de datos en formato CSV, de forma que la primera fila corresponda a los nombres de los atributos y a partir de la segunda fila en adelante se encuentren los datos. Estos datos se encontrarán delimitados por un separador que será introducido a la hora de subir los datos.

Por otro lado, al crear el experimento utilizando la API REST, el conjunto de datos debe ser introducido utilizando el formato JSON [62] (acrónimo de JavaScript Object Notation). El conjunto de datos consistirá en un vector de objetos representando las entradas de información. Estos objetos consistirán en objetos JSON cuyo contenido estará basado en pares clave-valor de los atributos de la entrada de información junto a su valor.

Durante el desarrollo de la aplicación, se estudió el uso del formato JSON-stat [63] para los conjuntos de datos. Sin embargo, tras múltiples experimentos utilizándolo con distintas tecnologías, se decantó por descartar este formato. Esto fue debido a que, por la naturaleza del formato, este exige una entrada por cada combinación de datos posible (siguiendo un producto

escalar de dimensiones), lo que desemboca en objetos JSON llenos de información en blanco con una relación de información útil muy baja.

5.6 Funcionalidades

5.6.1 Realizar experimentos de aprendizaje automático

La mayor funcionalidad de la aplicación consiste en permitir al usuario poder realizar sus propios experimentos de aprendizaje automático.

Para ello, debe provisionar los siguientes datos.

- Conjunto de datos (en formato CSV o JSON) del experimento.
- Forma de tratar los valores omitidos o *NaN* (*Not a number*).
- Tipo de aprendizaje, algoritmo del modelo y el atributo objetivo (este último solo es necesario en experimentos de aprendizaje supervisado).
- Tamaño del conjunto de test, ajustes del algoritmo y parámetros del experimento.

Una vez provisionados los datos se entrenará un modelo matemático con ellos y será evaluado para obtener métricas referentes a su desempeño. El algoritmo entrenado podrá ser utilizado para realizar nuevas predicciones.

El usuario tiene la posibilidad de hacer tres tipos de experimentos en función del tipo de aprendizaje usado:

- Experimentos de clasificación (aprendizaje supervisado) en los cuales el modelo será entrenado para predecir un atributo objetivo discreto o categórico de una entrada de datos.
- Experimentos de regresión (aprendizaje supervisado) cuyos modelos son entrenados para predecir un atributo objetivo numérico y continuo de una entrada de datos.
- Experimentos de agrupamiento (aprendizaje no supervisado). En estos experimentos, el modelo es entrenado para agrupar las entradas de datos en conjuntos similares.

Los experimentos de aprendizaje automático pueden ser realizados tanto desde un navegador a través de la aplicación web como desde una petición *HTTP POST* a través de la API REST de la herramienta.

5.6.2 Consultar experimentos de aprendizaje automático

Cuando el usuario finaliza la realización de un experimento de aprendizaje automático, éste puede consultar:

- El conjunto de datos empleado.
    - El usuario podrá visualizar los datos en una tabla y podrá descargar el conjunto en formato *JSON* y *CSV*.
    - El conjunto de datos puede ser ordenado en función de cualquiera de sus atributos de manera ascendente y descendente.
    - Se podrá aplicar filtros de forma que se visualicen únicamente aquellos datos que cumplan los filtros seleccionados.
- Las métricas resultantes de la evaluación del modelo matemático del experimento. En función del tipo del experimento, se evaluarán unas métricas u otras.
    - Además, el usuario puede consultar los resultados de la fase de test del experimento. Es decir, podrá visualizar los valores predichos para cada entrada de datos del conjunto de test junto al valor original. En experimentos de aprendizaje supervisado se mostrará también si cada predicción es correcta o no.
- Los parámetros del experimento, del modelo matemático y del algoritmo utilizado.
- Gráficas del experimento generadas a partir de la librería externa usada Scikit-plot [60]. Las gráficas mostradas dependen del tipo de experimento llevado a cabo.

Las rutas de las ventanas de la aplicación web correspondientes a un experimento tienen como prefijo el nombre de éste.

5.6.3 Modificar un experimento de aprendizaje automático

Si el usuario, tras finalizar un experimento de aprendizaje automático, desea modificar cualquiera de sus parámetros, tendrá disponible una opción desde la aplicación destinada a ello.

Para ello, basta con acceder al experimento correspondiente y realizar las modificaciones pertinentes.

Modificar un experimento conllevará volver a entrenar y evaluar el modelo.

5.6.4 Eliminar un experimento de aprendizaje automático

El usuario puede eliminar un experimento de aprendizaje automático si ya no le es de utilidad y lo desea.

Si se decide eliminar un experimento, éste será borrado de memoria de forma irreversible.

5.6.5 Exportar e importar experimentos de aprendizaje automático

La aplicación ofrece al usuario la opción de poder exportar experimentos de aprendizaje automáticos ya entrenados y evaluados de forma serializada en documentos.

Estos documentos, a su vez, pueden ser subidos en la aplicación de manera que los experimentos serializados sean importados ya entrenados y evaluados. Si un experimento es importado, no hay necesidad de repetir el entrenamiento ni la evaluación del modelo.

Los documentos exportados e importados tienen la extensión "*.api*".

Esta funcionalidad es llevada a cabo gracias a la librería externa de serialización de objetos Python, Pickle [59].

5.6.6 Realizar predicciones con nuevos datos en un experimento

Una vez un modelo matemático haya sido entrenado y evaluado, éste estará disponible para ser utilizado para realizar predicciones sobre nuevos datos.

El usuario podrá introducir estos nuevos datos de diferentes formas:

- Introduciendo manualmente la entrada de datos a través de un formulario.
- A través de un objeto *JSON*. Este puede consistir en un objeto compuesto de pares clave-valor cuyas claves serán el nombre de los atributos de los datos. También es posible introducir un vector de objetos cuya estructura sea la mencionada anteriormente.
- Mediante un fichero *CSV*.

En experimentos de aprendizaje supervisado, el valor predicho corresponderá al atributo objetivo de la entrada de datos.

Por otro lado, en experimentos de aprendizaje no supervisado, el valor predicho corresponderá al conjunto de datos al que se le asigna la entrada de datos.

5.6.7 Comparar experimentos de aprendizaje automático

Tras realizar diversos experimentos, el usuario podrá comparar cada uno de ellos utilizando las diferentes métricas disponibles tras la evaluación de los modelos.

Para ello, la aplicación permite albergar varios experimentos en memoria, de forma que es posible acceder a ellos fácilmente para su comparación.

5.6.8 Desplegar una API REST por cada experimento de aprendizaje automático

Cada experimento de aprendizaje automático que se lleve a cabo en la aplicación desplegará una API REST con distintos *endpoints* para ser utilizados.

De esta forma, es posible interactuar con el experimento tanto en la aplicación web como en los *endpoints* de la API.

Los *endpoints* de la API REST desplegada tendrán como prefijo el nombre del experimento.

5.6.9 Recibir un correo electrónico como notificación tras finalizar un experimento

En función del tamaño del conjunto de datos de un experimento y del algoritmo seleccionado, los tiempos de cómputo del entrenamiento del modelo matemático pueden elevarse.

Entonces, el usuario puede configurar el servidor de correos electrónicos SMTP (*Simple Mail Transfer Protocol*) para que la aplicación le envíe un correo en cuanto el experimento complete el entrenamiento y evaluación.

La funcionalidad de los correos electrónicos ha sido implementada gracias a la extensión de Flask, Flask-Mail [53].

5.6.10 Manipular el conjunto de entrada del experimento de aprendizaje automático

Durante la realización de un experimento de aprendizaje automático, es posible manipular el conjunto de entrada para buscar el mayor éxito del experimento.

Obligatoriamente, es necesario escoger el modo de procesado de valores nulos y *NaN* (*Not a number*). En caso de ser un conjunto de datos completo,

esta elección no supone ninguna alteración. Los modos disponibles son los siguientes:

- Modo ***drop***: las entradas de información (filas) que contengan algún valor nulo serán desechadas.
- Modo ***fill***: los valores nulos serán rellenados con el valor fijo introducido.
- Modo ***mean***: rellena los valores con el valor numérico medio de la columna o el valor discreto más frecuente.

El modo elegido también se aplica durante la predicción de nuevos conjuntos de datos que cuenten con valores nulos o *NaN* en ellos.

Además, para manipular el conjunto de datos, también es posible seleccionar qué columnas desechar para el experimento. Esto es debido a que ciertas columnas pueden no aportar información útil o pueden estar corruptas.

5.6.11 Ejecución en contenedores Docker

A partir de la inclusión del fichero de instrucciones Dockerfile [64], se facilitó al usuario la ejecución de la aplicación en un contenedor.

Para ello, es necesario crear una imagen Docker de la aplicación utilizando el fichero Dockerfile y ponerla en ejecución en un contenedor.

5.7 Documentación

A medida que se fue escribiendo el código de la aplicación, también se fue documentando, de forma que este fuese más comprensible y mantenible.

Esta documentación se realizó siguiendo la convención docstring [65].

Docstring es un estándar de documentación de programas del lenguaje de programación *Python*. Consiste en cadenas *string* literales utilizados como primera línea de un módulo, función, clase o definición de método. De tal

forma, un docstring se convierte en el atributo *__doc__* del objeto documentado.

Además, gracias a utilizar este estándar, durante la redacción del manual de usuario, Sphinx permitió la generación de un apartado de documentación del código de forma automática a partir de la documentación docstring.

5.8 Manual de usuario

Una vez construida la herramienta, se procedió a redactar un manual de usuario.

Este manual de usuario sería de utilidad para el usuario ya que explicaría tanto la instalación de la herramienta, como todas las funcionalidades que ofrece incluyendo unos ejemplos prácticos.

Para redactar el manual, se utilizó el generador de documentación de *Python*, Sphinx [48].

Sphinx es un generador de documentación escrito por Georg Brandl. Aunque originalmente fue creado para la documentación de programas del lenguaje de programación *Python*, también tiene uso dentro de la documentación de proyectos de software de un vasto rango de lenguajes. Utiliza el lenguaje de etiqueta reStructuredText [66] para generar la documentación y puede generarla en diversos formatos como *HTML*, *LaTeX* (incluyendo versiones *PDF*), *ePub*, *Textinfo*, páginas de *manual* y texto plano.

Todos los documentos relacionados al manual de usuario se encuentran en el directorio */docs/*. Dentro de este directorio se encuentran los ficheros de configuración de Sphinx, las secciones redactadas en reStructuredText y ficheros estáticos.

Para su versión *HTML*, se escogió la plantilla Read the Docs Sphinx Theme [67], plantilla popular utilizada en proyectos de Read the Docs [68], plataforma de documentación de proyectos del lenguaje de programación *Python*.

La documentación de este proyecto está dividida en dos secciones diferenciadas, el manual de usuario y la documentación del código.

El manual de usuario contiene los siguientes apartados:

- **Instalación de la herramienta**. Instrucciones para la instalación de la herramienta.
- **Ejemplos prácticos**. Ejemplos prácticos para probar la herramienta. Se han incluido los siguientes ejemplos:
    - Ejemplo de experimento de aprendizaje automático de clasificación.
    - Ejemplo de experimento de aprendizaje automático de regresión.
    - Ejemplo de experimento de aprendizaje automático de agrupamiento.
- **Opciones de experimentación**. Explicación de todas las posibles opciones a la hora de realizar un experimento de aprendizaje automático.
- **Ventanas**. Descripción en detalle de cada una de las ventanas de la aplicación web.
- **Endpoints**. Descripción en detalle de cada uno de los *endpoints* de la API de la herramienta.
- **Opciones adicionales**. Explicación de funcionalidades extra de la aplicación.

En la Figura 6 se muestra el aspecto del manual de usuario en formato *HTML*.

Figura 6: Manual de usuario en formato HTML

Por otro lado, está la documentación del código fuente. Ésta fue generada automáticamente en su mayor parte gracias al uso del estándar docstring y la extensión autodoc [69] de Sphinx.

En la Figura 7 se muestra el aspecto de la documentación del código fuente en formato *HTML*.

Figura 7: Documentación del código fuente en formato HTML

5.9 Pruebas

Una vez construida la herramienta, tuvo lugar un periodo de pruebas de la aplicación.

Para ello, se llevaron a cabo diversas pruebas de caja negra [70] y de caja blanca [71] en las cuales se comprobó el correcto funcionamiento tanto interno del programa como funcional.

Además, se realizaron pruebas de experimentos de aprendizaje automático utilizando una gran cantidad de conjuntos de datos distintos extraídos del portal de ciencia de datos y aprendizaje automático Kaggle [72].

Gracias a la realización de estas pruebas, se encontraron y arreglaron problemas de funcionamiento y usabilidad.

Para el diseño de pruebas unitarias y funcionales se hizo uso del *framework* de pruebas pytest [73] debido a que es un software de pruebas de código escrito en el lenguaje de programación *Python* muy completo y además cuenta con facilidades para aplicaciones construidas con Flask.

6. RESULTADOS Y DISCUSIÓN

Finalizada la construcción de la herramienta, es posible dividir el resultado en dos secciones. Por un lado, la aplicación web con interfaz navegable a través de un navegador web. Por el otro, el despliegue de una API REST global de la aplicación.

Para poner ambos en ejecución, es necesario inicializar la aplicación. En la Figura 8 se muestra la inicialización de la herramienta desde consola.

```
D:\Universidad\Cuarto\Segundo Semestre\TFG\PyMLAPIGen>pymlapigen
 * Serving Flask app 'pymlapigen' (lazy loading)
 * Environment: production
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Figura 8: Inicialización de la aplicación

6.1 Herramienta Web

La herramienta web es accesible una vez inicializada y desplegada la aplicación a través de un navegador web.

En la Figura 9 se muestra el aspecto de la herramienta web.

Figura 9: Herramienta web

Esta herramienta web ofrece una interfaz gráfica basada en páginas web.

En la página principal de la aplicación, se visualizarán los distintos experimentos de aprendizaje automático que se hayan llevado a cabo y estén en memoria.

En la Figura 10 se muestra la página principal de la herramienta web.

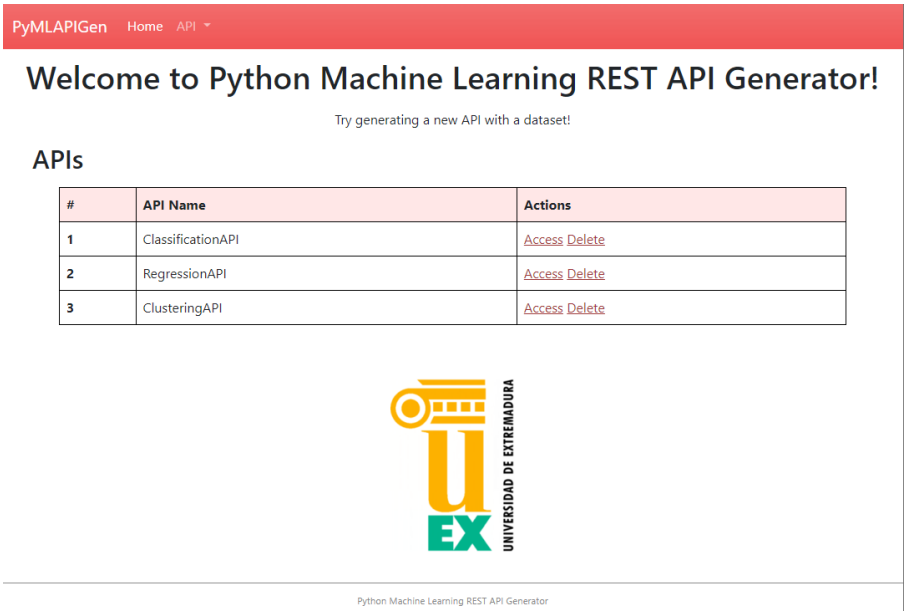


Figura 10: Página principal

Dispone de una barra de navegación en la cual se puede acceder a las distintas partes de la aplicación.

En la Figura 11 se muestra la barra de navegación de la herramienta web.

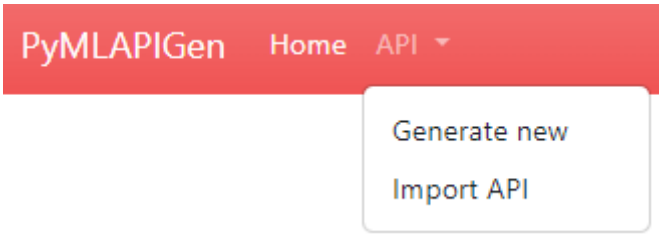


Figura 11: Barra de navegación principal

A la hora de realizar un experimento de aprendizaje automático, el usuario deberá rellenar un formulario de entrada con las características del experimento.

De la Figura 12 a la Figura 15 se muestran los pasos necesarios para la realización de un experimento de aprendizaje automático en la herramienta web.

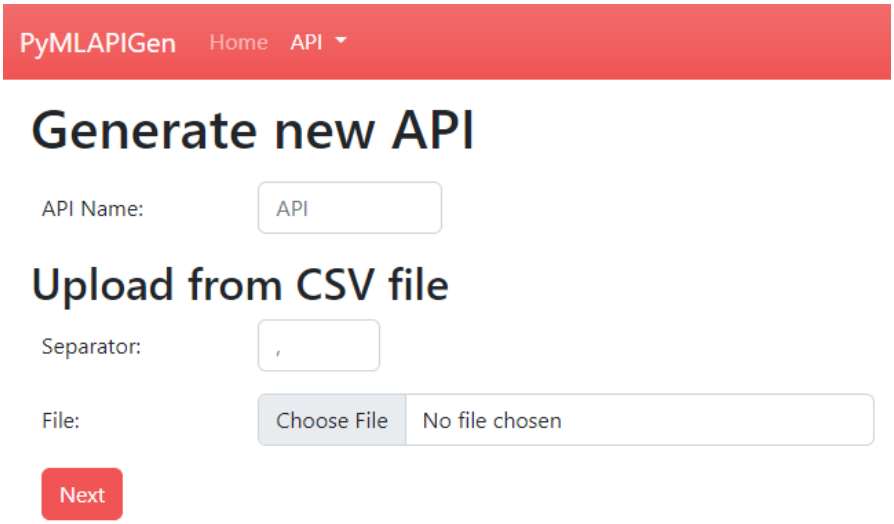


Figura 12: Creación de experimento de aprendizaje automático (conjunto de datos)

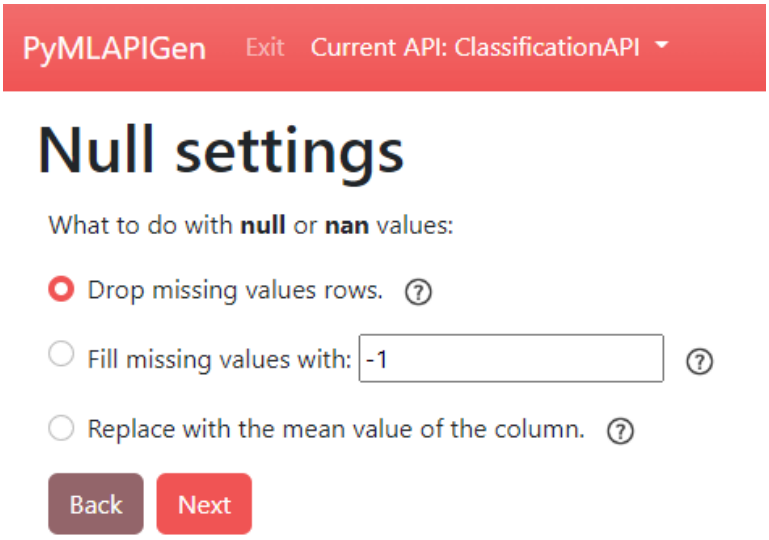


Figura 13: Creación de experimento de aprendizaje automático (valores nulos y NaN)

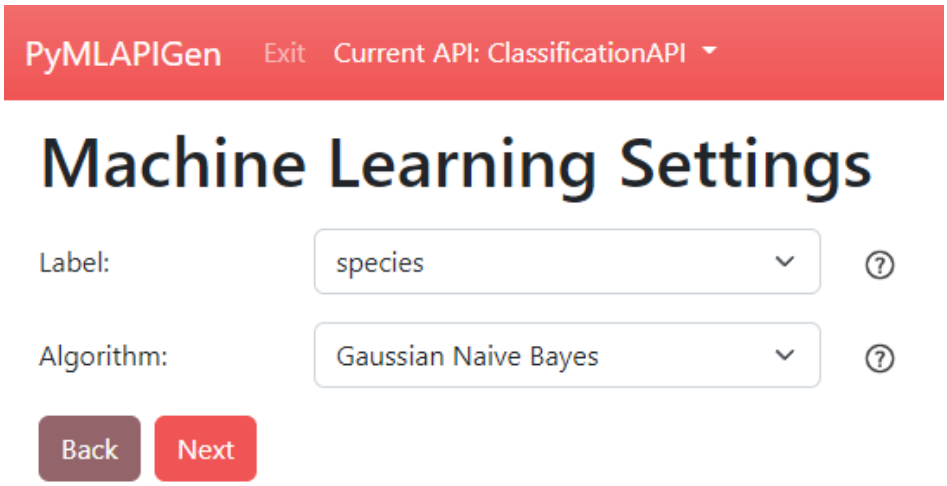


Figura 14: Creación de experimento de aprendizaje automático (ajustes del experimento)

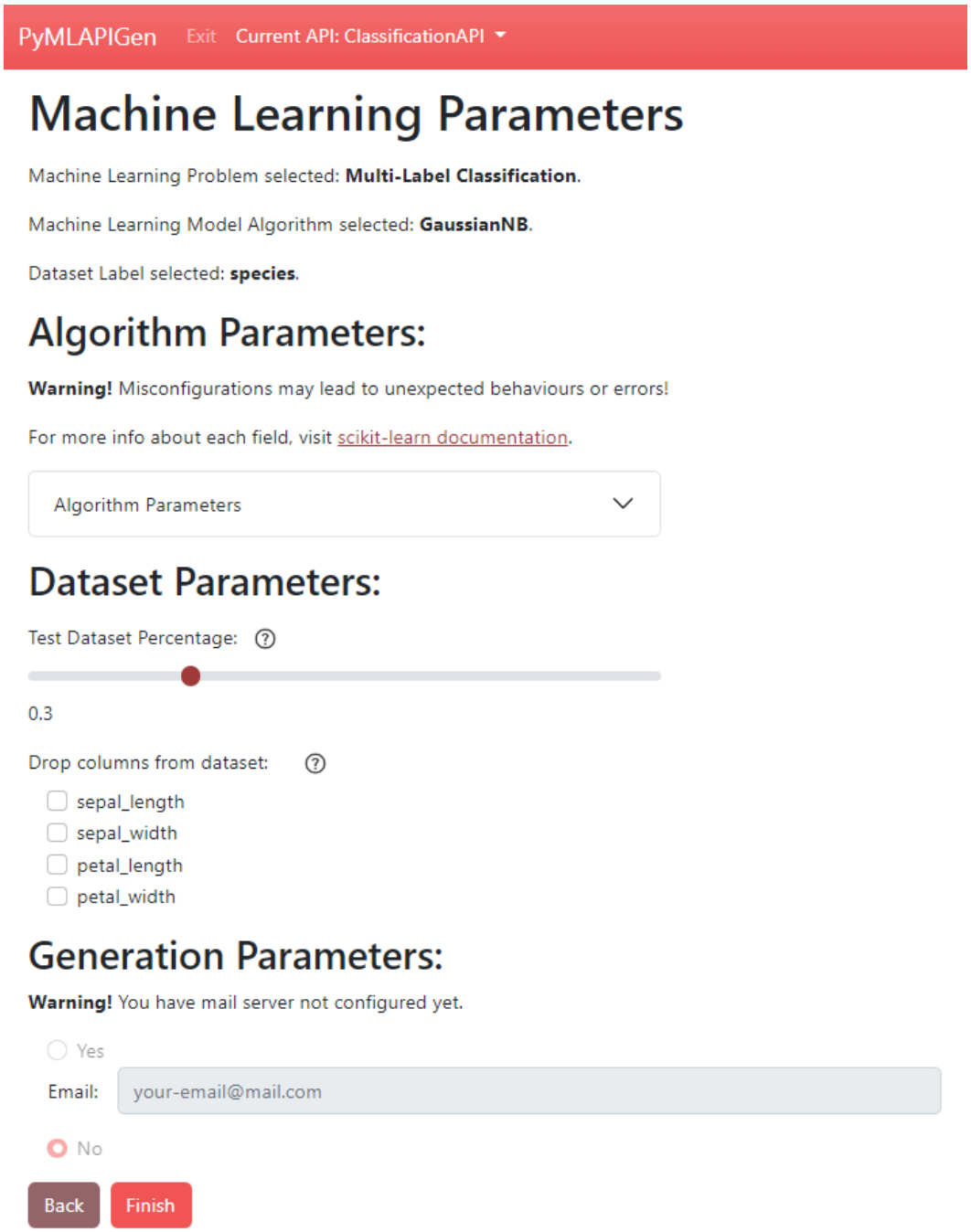


Figura 15: Creación de experimento de aprendizaje automático (parámetros adicionales)

Tras finalizar el formulario del experimento, se lleva a cabo el entrenamiento y evaluación del modelo matemático. Este modelo ejecutará el algoritmo seleccionado en el formulario sobre el conjunto de datos subido.

Desde la página principal se podrá acceder a la vista de un experimento. En la Figura 16 se muestra la vista principal del experimento, en la cual se puede visualizar todas las rutas web y *endpoints* generadas a partir del experimento.

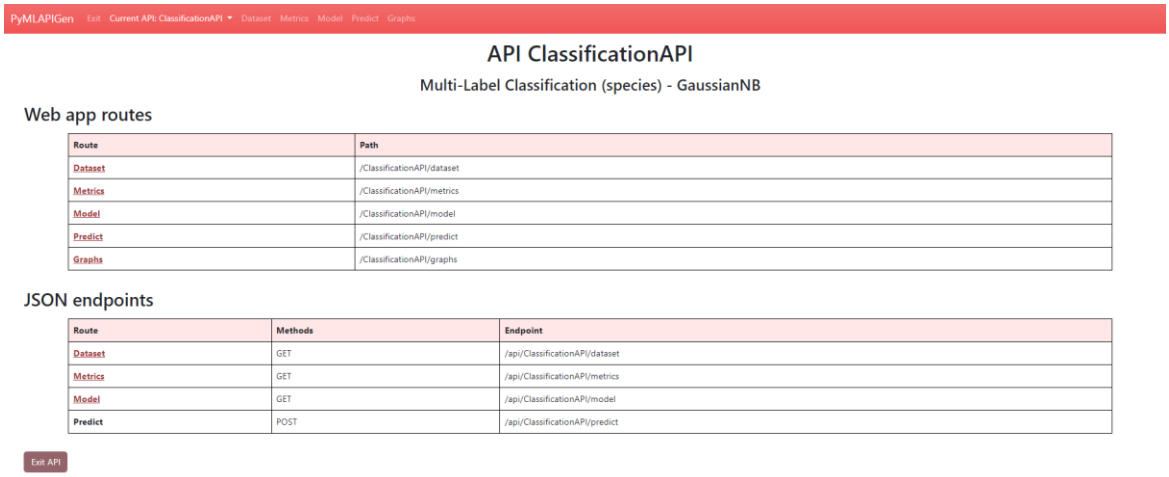


Figura 16: Página principal de un experimento

Esta vista trae consigo nuevas opciones reflejadas en la barra de navegación incluyendo nuevas páginas.

De la Figura 17 a la Figura 23, se muestran la barra de navegación y las páginas pertenecientes a un experimento.

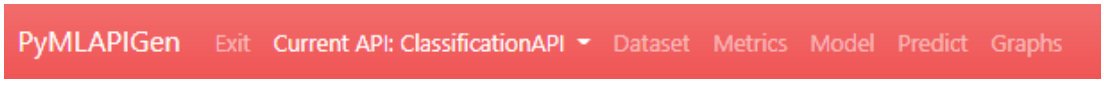


Figura 17: Barra de navegación de un experimento

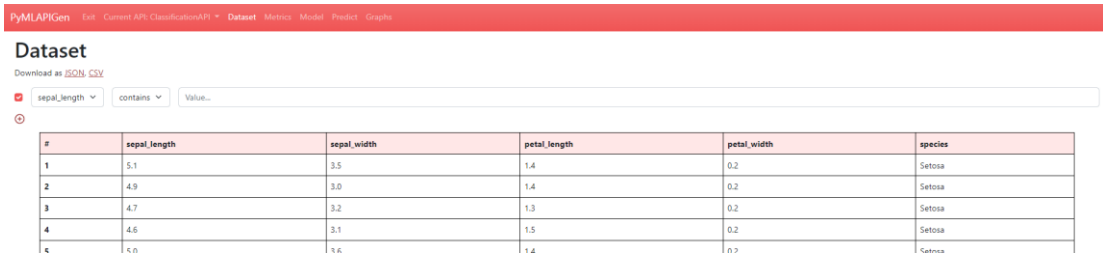


Figura 18: Página dataset de un experimento

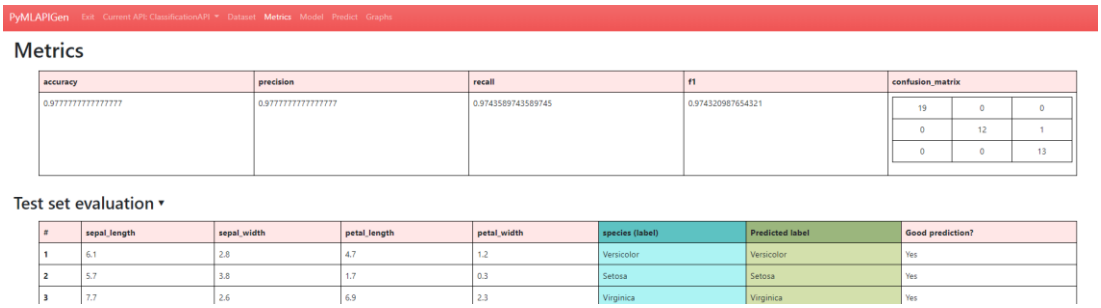


Figura 19: Página métricas de un experimento

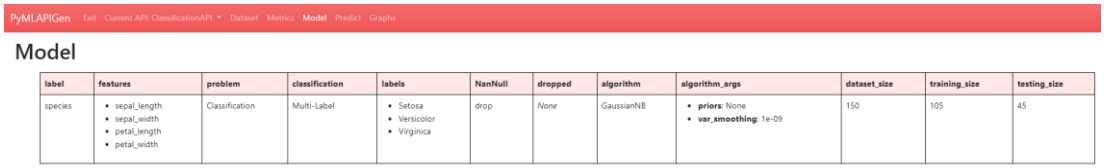


Figura 20: Página modelo de un experimento

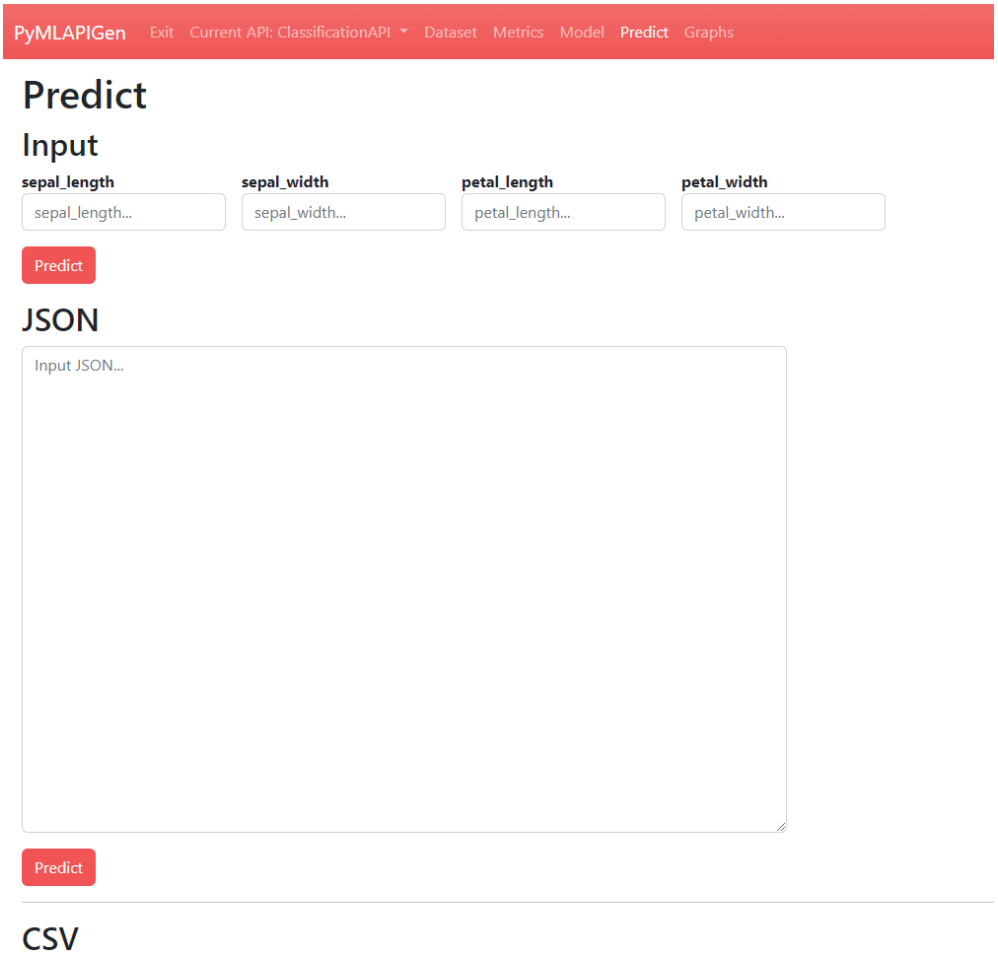


Figura 21: Página predicciones de un experimento

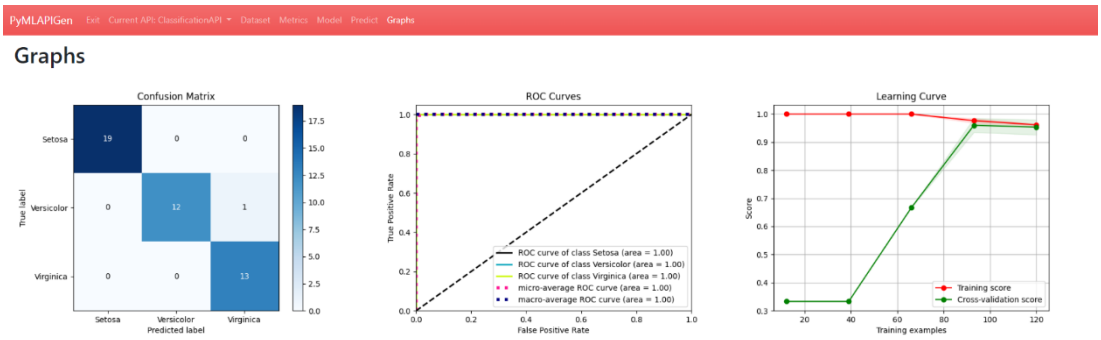
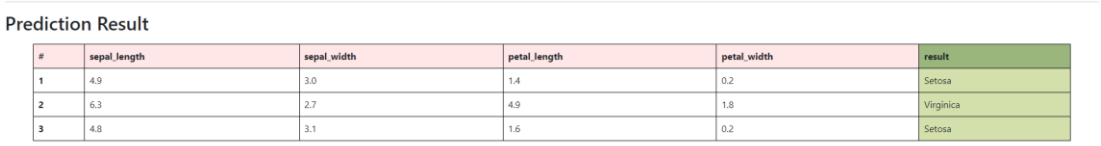


Figura 22: Página gráficos de un experimento



Prediction Result

#	sepal_length	sepal_width	petal_length	petal_width	result
1	4.9	3.0	1.4	0.2	Setosa
2	6.3	2.7	4.9	1.8	Virginica
3	4.8	3.1	1.6	0.2	Setosa

Figura 23: Predicción de nuevos datos de un experimento

En el manual de usuario se encuentra detallado tanto la realización de experimentos de aprendizaje automático como todas las páginas y funcionalidades de la herramienta web.

6.2 API REST

Además de la herramienta web, la aplicación permite la interacción a través de los distintos *endpoints* de una API REST desplegada. Todos los *endpoints* comienzan con el prefijo */api/*

Gracias a peticiones HTTP [74] (*GET*, *POST*…) el usuario será capaz de interactuar con la aplicación, realizar sus experimentos de aprendizaje automático, acceder y consultar los experimentos en memoria y realizar predicciones con nuevos datos.

Cada experimento de aprendizaje automático realizado (tanto por la herramienta web como por la API REST) desplegará una serie de nuevos *endpoints* a la API. Estos tendrán como prefijo el nombre del experimento.

Los *endpoints* principales disponibles son los siguientes:

Endpoint	**Método HTTP**	**Función**
/api	GET	Devuelve el estado de la aplicación y los experimentos en memoria.
/api/load	POST	Realiza un experimento de aprendizaje automático (incluyendo entrenamiento y evaluación).

Tabla 8: Endpoints principales de la API REST

Por otro lado, los *endpoints* correspondientes a un experimento de aprendizaje automático son los citados a continuación:

Endpoint	**Método HTTP**	**Función**
/api/<experimento>	GET	Devuelve los parámetros principales del experimento junto a sus endpoints.
/api/<experimento>/dataset	GET	Devuelve el conjunto de datos del experimento.
/api/<experimento>/metrics	GET	Devuelve las métricas del experimento.
/api/<experimento>/model	GET	Devuelve los parámetros escogidos del experimento.
/api/<experimento>/predict	POST	Realiza predicciones con datos nuevos.

Tabla 9: Endpoints de un experimento

Una descripción más detallada de todos los *endpoints* y sus diferentes opciones se encuentra en el manual de usuario.

7. CONCLUSIONES

7.1 Conclusiones

La realización de este proyecto ha supuesto el aprendizaje de muchos valores y conocimientos dentro del campo de la ingeniería informática del software, del desarrollo web, de la ciencia de datos, inteligencia artificial y aprendizaje automático.

El haber dividido el desarrollo del proyectos en objetivos parciales ha permitido una progresión más efectiva a la par que abrió la posibilidad de reconducir los objetivos en función de los resultados de los anteriores.

He aprendido la importancia de la labor de estudiar las distintas tecnologías disponibles, analizando las ventajas y desventajas de cada una, para elegir así la mejor arquitectura que sustente el software a desarrollar.

También me ha ayudado a valorar el proceso de desarrollo de software en su completitud. Desde el estudio de requisitos a la documentación del código desarrollado.

Además, gracias al desarrollo de este trabajo, he aprendido a utilizar bastantes herramientas, tecnologías y estándares útiles.

En cuanto a lo personal, me considero bastante satisfecho con el resultado del trabajo y del desarrollo de este. Tanto yo como mis tutores hemos llevado un trabajo constante que ha perdurado hasta el final del proyecto.

Finalmente, se ha conseguido lograr el desarrollo de una aplicación, tanto web como API, a partir de la cual cualquier usuario puede realizar sus experimentos de aprendizaje automático sin necesitar de conocimientos profundos en programación ni de *Machine Learning*.

7.2 Trabajos futuros

Algunos de los trabajos futuros relacionados con este proyecto pueden ser relacionados con la extensión de funcionalidades:

- Inclusión de un mayor número de algoritmos para los experimentos de aprendizaje automático.
- Expansión de los tipos de aprendizaje automático disponibles (aprendizaje semisupervisado, aprendizaje por refuerzo…)
- Implementación de experimentos de aprendizaje profundo utilizando redes neuronales artificiales.
- Implementación de experimentos de procesamiento del Lenguaje Natural (*NLP*)
- Aumento de maneras de manipular los datos para incrementar el éxito de los experimentos.
- Incremento del número de gráficos disponibles para los experimentos.
- Pruebas con un mayor número de conjuntos de datos.

Otros trabajos futuros pueden surgir acerca de la implementación de la idea llevada a cabo en el proyecto en otras tecnologías, tanto de aprendizaje automático como de construcción de aplicaciones.

AGRADECIMIENTOS

Me gustaría incluir esta sección de agradecimientos para agradecer a aquellas personas que me han impulsado a seguir adelante, no solo durante en desarrollo del trabajo, sino toda mi carrera universitaria. Sin ellos, nada de esto hubiera sido posible.

A todos los profesores y compañeros de clase que me han acompañado durante estos años de carrera.

A José María Conejero y Francisco Javier Melchor, por haberme guiado durante todo el desarrollo de este proyecto.

A Alberto Gómez por orientarme durante toda mi etapa universitaria y ser mi tutor.

A mi familia, por haberme cuidado, visto crecer, apoyado en todos los momentos, guiarme y ayudarme. Especialmente mi madre y mi padre.

A mis amigos de Don Benito, por haberme acompañado durante esta etapa, transmitido vuestras experiencias y apoyado en todo momento.

A mi pareja, por ser un pilar de mi vida, por haberme hecho creer en mí, apoyarme durante todos los días, haberme escuchado siempre que lo necesitaba, darme la confianza que me falta, por ayudarme a levantar cada vez que he caído.

A todos vosotros, muchísimas gracias.

REFERENCIAS BIBLIOGRÁFICAS

[1] Wikipedia, «Aprendizaje automático - Wikipedia, la enciclopedia libre,» [En línea]. Available: https://es.wikipedia.org/wiki/Aprendizaje_autom%C3%A1tico. [Último acceso: 21 Junio 2022].

[2] datos.gob.es, «Ciencia de datos, machine learning y deep learning | datos.gob.es,» [En línea]. Available: https://datos.gob.es/es/blog/ciencia-de-datos-machine-learning-y-deep-learning. [Último acceso: 21 Junio 2022].

[3] Wikipedia, «Ciencia de datos - Wikipedia, la enciclopedia libre,» [En línea]. Available: https://es.wikipedia.org/wiki/Ciencia_de_datos. [Último acceso: 21 Junio 2022].

[4] Oracle, «¿Qué es la ciencia de datos? | Oracle México,» [En línea]. Available: https://www.oracle.com/mx/data-science/what-is-data-science/. [Último acceso: 21 Junio 2022].

[5] ComputerWeekly, «¿Qué es Ciencia de datos? - Definición en WhatIs.com,» [En línea]. Available: https://www.computerweekly.com/es/definicion/Ciencia-de-datos. [Último acceso: 21 Junio 2022].

[6] Wikipedia, «Inteligencia artificial - Wikipedia, la enciclopedia libre,» [En línea]. Available: https://es.wikipedia.org/wiki/Inteligencia_artificial. [Último acceso: 21 Junio 2022].

[7] Oracle, «¿Qué es la inteligencia artificial (IA)? | Oracle España,» [En línea]. Available: https://www.oracle.com/es/artificial-intelligence/what-is-ai/. [Último acceso: 21 Junio 2022].

[8] M. Azure, «¿Qué es el aprendizaje automático? | Microsoft Azure,» [En línea]. Available: https://azure.microsoft.com/es-es/overview/what-is-machine-learning-platform/#techniques. [Último acceso: 21 Junio 2022].

[9] G. Everywhere, «Machine Learning | Qué es, tipos, ejemplos y cómo implementarlo,» [En línea]. Available: https://www.grapheverywhere.com/machine-learning-que-es-tipos-ejemplos-y-como-implementarlo/. [Último acceso: 21 Junio 2022].

[10] J. C. R. J. M. A. R. C. Q. Jordi Gironés Roig, Minería de datos: modelos y algoritmos, UOC, 2017.

[11] Wikipedia, «Interfaz de programación de aplicaciones - Wikipedia, la enciclopedia libre,» [En línea]. Available: https://es.wikipedia.org/wiki/Interfaz_de_programaci%C3%B3n_de_aplicaciones. [Último acceso: 21 Junio 2022].

[12] Xataka, «API: qué es y para qué sirve,» [En línea]. Available: https://www.xataka.com/basics/api-que-sirve. [Último acceso: 21 Junio 2022].

[13] AWS, «¿Qué es una API? - Guía sobre las API para principiantes - AWS,» [En línea]. Available: https://aws.amazon.com/es/what-is/api/#:~:text=API%20significa%20%E2%80%9Cinterfaz%20de%20programaci%C3%B3n,de%20servicio%20entre%20dos%20aplicaciones.. [Último acceso: 21 Junio 2022].

[14] Retool, «Retool | Build internal tools, remarkably fast.,» [En línea]. Available: https://retool.com/. [Último acceso: 22 Junio 2022].

[15] Retool, «Retool | Generate API from CSV,» [En línea]. Available: https://retool.com/utilities/generate-api-from-csv. [Último acceso: 22 Junio 2022].

[16] Retool, «Retool | Generate app from CSV,» [En línea]. Available: https://retool.com/utilities/generate-app-from-csv. [Último acceso: 22 Junio 2022].

[17] Datasette, «Datasette: An open source multi-tool for exploring and publishing data,» [En línea]. Available: https://datasette.io/. [Último acceso: 21 Junio 2022].

[18] PythonOnWheels, «PythonOnWheels Home,» [En línea]. Available: https://www.pythononwheels.org/article/aa2b4e96-816f-4a2b-a3a8-a195975093ae. [Último acceso: 22 Junio 2022].

[19] E. T. S. W. t. REST, «Python REST API Framework: Eve, the Simple Way to REST. — Eve 2.0 documentation,» [En línea]. Available: https://docs.python-eve.org/en/stable/index.html. [Último acceso: 22 Junio 2022].

[20] IBM, «What is Machine Learning? | IBM,» [En línea]. Available: https://www.ibm.com/cloud/learn/machine-learning. [Último acceso: 22 Junio 2022].

[21] G. Cloud, «Productos de inteligencia artificial y aprendizaje automático | Google Cloud,» [En línea]. Available: https://cloud.google.com/products/ai?hl=es. [Último acceso: 22 Junio 2022].

[22] Microsoft, «Servicios de informática en la nube | Microsoft Azure,» [En línea]. Available: https://azure.microsoft.com/es-es/. [Último acceso: 22 Junio 2022].

[23] AWS, «Servicios gratuitos de Machine Learning: AWS,» [En línea]. Available: https://aws.amazon.com/es/free/machine-learning/?r=qal-mls. [Último acceso: 22 Junio 2022].

[24] TensorFlow, «TensorFlow,» [En línea]. Available: https://www.tensorflow.org/. [Último acceso: 22 Junio 2022].

[25] Scikit-Learn, «scikit-learn: machine learning in Python — scikit-learn 1.1.1 documentation,» [En línea]. Available: https://scikit-learn.org/stable/. [Último acceso: 23 Junio 2022].

[26] matplotlib, «Matplotlib — Visualization with Python,» [En línea]. Available: https://matplotlib.org/. [Último acceso: 23 Junio 2022].

[27] NumPy, «NumPy,» [En línea]. Available: https://numpy.org/. [Último acceso: 23 Junio 2022].

[28] pandas, «pandas - Python Data Analysis Library,» [En línea]. Available: https://pandas.pydata.org/. [Último acceso: 23 Junio 2022].

[29] J. K. Gateway, «Jupyter Kernel Gateway — Jupyter Kernel Gateway 2.6.0.dev0 documentation,» [En línea]. Available: https://jupyter-kernel-gateway.readthedocs.io/en/latest/#. [Último acceso: 23 Junio 2022].

[30] Flask, «Welcome to Flask — Flask Documentation (2.1.x),» [En línea]. Available: https://flask.palletsprojects.com/en/2.1.x/. [Último acceso: 23 Junio 2022].

[31] Jinja, «Jinja — Jinja Documentation (3.1.x),» [En línea]. Available: https://jinja.palletsprojects.com/en/3.1.x/. [Último acceso: 23 Junio 2022].

[32] N. Alonso, «¿Qué es un WSGI?. La palabra WSGI es una de esas palabras… | by Nacho Alonso | Medium,» [En línea]. Available: https://medium.com/@nachoad/que-es-wsgi-be7359c6e001. [Último acceso: 23 Junio 2022].

[33] Werkzeug, «Werkzeug | The Pallets Projects,» [En línea]. Available: https://palletsprojects.com/p/werkzeug/. [Último acceso: 23 Junio 2022].

[34] Django, «The web framework for perfectionists with deadlines | Django,» [En línea]. Available: https://www.djangoproject.com/. [Último acceso: 23 Junio 2022].

[35] FastAPI, «FastAPI,» [En línea]. Available: https://fastapi.tiangolo.com/. [Último acceso: 23 Junio 2022].

[36] pydantic, «pydantic,» [En línea]. Available: https://pydantic-docs.helpmanual.io/. [Último acceso: 23 Junio 2022].

[37] Uvicorn, «Uvicorn,» [En línea]. Available: https://www.uvicorn.org/. [Último acceso: 23 Junio 2022].

[38] Gunicorn, «Gunicorn - Python WSGI HTTP Server for UNIX,» [En línea]. Available: https://gunicorn.org/. [Último acceso: 23 Junio 2022].

[39] OpenAPI, «Home - OpenAPI Initiative,» [En línea]. Available: https://www.openapis.org/. [Último acceso: 23 Junio 2022].

[40] O. 2.0, «OAuth 2.0 — OAuth,» [En línea]. Available: https://oauth.net/2/. [Último acceso: 29 Junio 2022].

[41] J. Schema, «JSON Schema | The home of JSON Schema,» [En línea]. Available: https://json-schema.org/. [Último acceso: 23 Junio 2022].

[42] Wikipedia, «Desarrollo ágil de software - Wikipedia, la enciclopedia libre,» [En línea]. Available: https://es.wikipedia.org/wiki/Desarrollo_%C3%A1gil_de_software. [Último acceso: 24 Junio 2022].

[43] axarnet, «Desarrollo ágil de software. ¿Cómo funciona?,» [En línea]. Available: https://axarnet.es/blog/desarrollo-agil. [Último acceso: 24 Junio 2022].

[44] V. S. Code, «Visual Studio Code - Code Editing. Redefined,» [En línea]. Available: https://code.visualstudio.com/. [Último acceso: 24 Junio 2022].

[45] Postman, «Postman API Platform | Sign Up for Free,» [En línea]. Available: https://www.postman.com/. [Último acceso: 24 Junio 2022].

[46] GitHub, «GitHub,» [En línea]. Available: https://github.com/. [Último acceso: 24 Junio 2022].

[47] Bootstrap, «Bootstrap · The most popular HTML, CSS, and JS library in the world.,» [En línea]. Available: https://getbootstrap.com/. [Último acceso: 24 Junio 2022].

[48] Sphinx, «Overview — Sphinx documentation,» [En línea]. Available: https://www.sphinx-doc.org/en/master/. [Último acceso: 24 Junio 2022].

[49] Trello, «Trello | Gestiona los proyectos de tu equipo desde cualquier lugar | Trello,» [En línea]. Available: https://trello.com/es. [Último acceso: 24 Junio 2022].

[50] Zoom, «Videoconferencia, teléfono en la nube, seminarios web, chat, eventos virtuales | Zoom,» [En línea]. Available: https://zoom.us/. [Último acceso: 24 Junio 2022].

[51] Flask, «Large Applications as Packages — Flask Documentation (2.1.x),» [En línea]. Available: https://flask.palletsprojects.com/en/2.1.x/patterns/packages/. [Último acceso: 24 Junio 2022].

[52] K. R. &. R. Python, «Structuring Your Project — The Hitchhiker's Guide to Python,» [En línea]. Available: https://docs.python-guide.org/writing/structure/. [Último acceso: 24 Junio 2022].

[53] Flask-Mail, «flask-mail — Flask-Mail 0.9.1 documentation,» [En línea]. Available: https://pythonhosted.org/Flask-Mail/. [Último acceso: 24 Junio 2022].

[54] Wikipedia, «Diseño web adaptable - Wikipedia, la enciclopedia libre,» [En línea]. Available: https://es.wikipedia.org/wiki/Dise%C3%B1o_web_adaptable. [Último acceso: 24 Junio 2022].

[55] io, «io — Core tools for working with streams — Python 3.10.5 documentation,» [En línea]. Available: https://docs.python.org/3/library/io.html. [Último acceso: 24 Junio 2022].

[56] Base64, «base64 — Codificaciones de datos Base16, Base32, Base64, y Base85 — documentación de Python - 3.10.5,» [En línea]. Available: https://docs.python.org/es/3/library/base64.html. [Último acceso: 24 Junio 2022].

[57] Json, «json — JSON encoder and decoder — Python 3.10.5 documentation,» [En línea]. Available:

https://docs.python.org/3/library/json.html. [Último acceso: 24 Junio 2022].

[58] Os, «os — Miscellaneous operating system interfaces — Python 3.10.5 documentation,» [En línea]. Available: https://docs.python.org/3/library/os.html. [Último acceso: 24 Junio 2022].

[59] Pickle, «pickle — Python object serialization — Python 3.10.5 documentation,» [En línea]. Available: https://docs.python.org/3/library/pickle.html. [Último acceso: 24 Junio 2022].

[60] Scikit-plot, «Welcome to Scikit-plot's documentation! — Scikit-plot documentation,» [En línea]. Available: https://scikit-plot.readthedocs.io/en/stable/. [Último acceso: 24 Junio 2022].

[61] Sys, «sys — System-specific parameters and functions — Python 3.10.5 documentation,» [En línea]. Available: https://docs.python.org/3/library/sys.html. [Último acceso: 24 Junio 2022].

[62] Wikipedia, «JSON - Wikipedia, la enciclopedia libre,» [En línea]. Available: https://es.wikipedia.org/wiki/JSON. [Último acceso: 24 Junio 2022].

[63] J.-s. A. s. l. s. f. d. d. -. JSON-stat.org, «JSON-stat,» [En línea]. Available: https://json-stat.org/. [Último acceso: 24 Junio 2022].

[64] Docker, «Dockerfile reference | Docker Documentation,» [En línea]. Available: https://docs.docker.com/engine/reference/builder/. [Último acceso: 24 Junio 2022].

[65] P. E. Proposals, «PEP 257 – Docstring Conventions | peps.python.org,» [En línea]. Available: https://peps.python.org/pep-0257/. [Último acceso: 25 Junio 2022].

[66] Wikipedia, «ReStructuredText - Wikipedia, la enciclopedia libre,» [En línea]. Available: https://es.wikipedia.org/wiki/ReStructuredText. [Último acceso: 25 Junio 2022].

[67] R. t. D. S. Theme, «Read the Docs Sphinx Theme — Read the Docs Sphinx Theme 1.0.0 documentation,» [En línea]. Available: https://sphinx-rtd-theme.readthedocs.io/. [Último acceso: 25 Junio 2022].

[68] R. t. Docs, «Inicio | Read the Docs,» [En línea]. Available: https://readthedocs.org/. [Último acceso: 25 Junio 2022].

[69] S. Autodoc, «sphinx.ext.autodoc – Include documentation from docstrings — Sphinx documentation,» [En línea]. Available: https://www.sphinx-doc.org/en/master/usage/extensions/autodoc.html. [Último acceso: 25 Junio 2022].

[70] TestingBaires, «Pruebas de Caja Negra y un enfoque práctico,» [En línea]. Available: https://testingbaires.com/pruebas-caja-negra-enfoque-practico/#:~:text=Las%20Pruebas%20de%20Caja%20Negra,ejecuci%C3%B3n%20internos%20en%20el%20software.. [Último acceso: 25 Junio 2022].

[71] Wikipedia, «Pruebas de caja blanca - Wikipedia, la enciclopedia libre,» [En línea]. Available: https://es.wikipedia.org/wiki/Pruebas_de_caja_blanca. [Último acceso: 25 Junio 2022].

[72] Kaggle, «Kaggle: Your Machine Learning and Data Science Community,» [En línea]. Available: https://www.kaggle.com/. [Último acceso: 25 Junio 2022].

[73] pytest, «pytest: helps you write better programs — pytest documentation,» [En línea]. Available: https://docs.pytest.org/en/7.1.x/. [Último acceso: 25 Junio 2022].

[74] D. Mozilla, «Métodos de petición HTTP - HTTP | MDN,» [En línea]. Available: https://developer.mozilla.org/es/docs/Web/HTTP/Methods. [Último acceso: 26 Junio 2022].

[75] Sphinx, «First steps to document your project using Sphinx — Sphinx documentation,» [En línea]. Available: https://www.sphinx-doc.org/en/master/tutorial/first-steps.html. [Último acceso: 30 Junio 2022].

ANEXOS

ANEXO 1: Repositorio de la herramienta

El repositorio del proyecto puede encontrarse en el siguiente enlace. Las instrucciones de su instalación pueden encontrarse tanto en el repositorio como en el manual de usuario.

ANEXO 2: Manual de usuario

En este anexo se introduce la versión en formato PDF del manual de usuario desarrollado en el motor Sphinx.

Este mismo documento puede ser generado por el usuario utilizando las herramientas disponibles en el repositorio. Para ello, es suficiente con dirigirse al directorio *docs* y utilizar Sphinx para generar el manual en el formato deseado [75].

Python Machine Learning API Generator

Versión 1

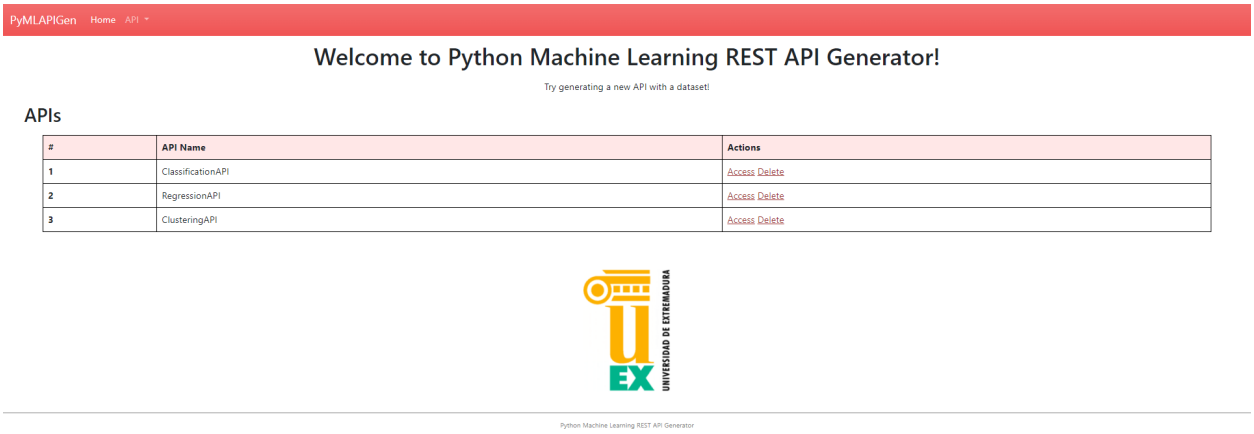
Adrian Ruiz Parra

26 de junio de 2022

Índice general

CAPÍTULO 1

Introducción



PyMLAPIGen es una herramienta que permite a cualquier usuario no experto en el dominio de Machine Learning entrenar algoritmos con un dataset de entrada y poder aplicar dichos algoritmos a nuevos datasets, tanto a través de una herramienta web sencilla como vía API de programación.

Esta herramienta ha sido construida en el lenguaje *Python* utilizando el microframework Flask.

Para la programación de Machine Learning, se ha utilizado la librería Scikit-Learn.

CAPÍTULO 2

Manual de Usuario

A través de este manual podrás **instalar** la herramienta, informarte acerca de cada **opción** en la aplicación y probar algunos **ejemplos**.

2.1 Instalación

En esta sección se te mostrará como instalar la aplicación en diferentes sistemas operativos.

2.1.1 Software necesario

Para poder ejecutar la aplicación, es necesario que tengas en tu equipo instalado el software Python.

2.1.2 Descargar y ejecutar herramienta

Una vez instalado todo el software necesario, es momento de descargar la herramienta e iniciarla.

Esto es posible de dos formas:

- **Instalar** y **utilizar** la aplicación a través de **pip** y la **línea de comandos**.

1. Abrir una terminal/consola.
2. Ejecutar el comando `pip install git+https://github.com/adruizp/PYMLAPIGEN` para descargar la herramienta y las librerías necesarias.
3. Iniciar la aplicación con el comando `pymlapigen`

```
pymlapigen
 * Serving Flask app 'pymlapigen' (lazy loading)
 * Environment: production
```

(continué en la próxima página)

(proviene de la página anterior)

```
 * Debug mode: off
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

4. Acceder al URL http://localhost:5000

- **Clonar**, **instalar** y **ejecutar** el repositorio de la aplicación.

1. Abrir una terminal/consola en el directorio que se desee descargar la aplicación.
2. Ejecutar el comando `git clone https://github.com/adruizp/PYMLAPIGEN` para clonar el repositorio que contiene la herramienta.

```
git clone https://github.com/adruizp/PYMLAPIGEN
  Cloning into 'PYMLAPIGEN'...
  remote: Enumerating objects: 197, done.
  remote: Counting objects: 100% (197/197), done.
  remote: Compressing objects: 100% (131/131), done.
  Receiving objects:  73% (144/197)), reused 145 (delta 61), pack-reused 0
  Receiving objects: 100% (197/197), 250.51 KiB | 1.25 MiB/s, done.
  Resolving deltas: 100% (104/104), done.
```

3. Acceder al directorio de la aplicación con el comando `cd PYMLAPIGEN`
4. Instalar las librerías necesarias con el comando `pip install -r requirements.txt`
5. Iniciar la aplicación con el comando `python run.py`

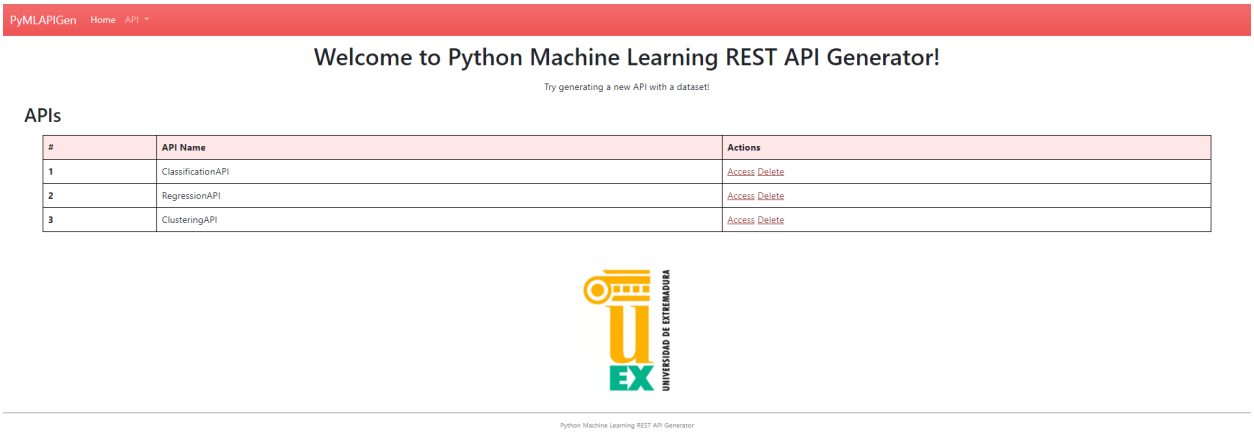
```
python run.py
 * Serving Flask app 'pymlapigen' (lazy loading)
 * Environment: production
 * Debug mode: off
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

6. Acceder al URL http://localhost:5000

2.1.3 Acceder a la herramienta

Una vez inicializada la herramienta, es posible acceder a ella, ya sea:

- A través de un navegador web (ruta http://localhost:5000)



- A través de una petición GET al endpoint http://localhost:5000/api/.

```
curl http://localhost:5000/api/
  {"status":"Api is working"}
```

2.2 Ejemplos

En esta sección encontrarás ejemplos sencillos para que generes tus primeras APIs.

2.2.1 Clasificación

En este apartado se mostrará un ejemplo de uso de la aplicación para **generar una API** con un experimento de **clasificación**.

En el dominio de Machine Learning, la **clasificación** es un tipo de aprendizaje **supervisado**. El valor que se quiere predecir es una clase dentro de un **número discreto** de **clases**.

Si se intenta predecir entre dos clases se trata de una **clasificación binaria** mientras que si el número de clases posibles es superior a dos se trata de una **clasificación multiclase**.

Dataset

El dataset que se va a utilizar para este ejemplo práctico consiste en un conjunto de datos de **flores iris**.

Este dataset es **público** y tiene un formato **CSV**. Puede descargarlo pulsando aqui.

Las columnas de este dataset son las siguientes:

sepal_length **Atributo** que indica la longitud del sépalo. (*Float*)

sepal_width **Atributo** que indica la anchura del sépalo. (*Float*)

petal_length **Atributo** que indica la longitud del petalo. (*Float*)

petal_width **Atributo** que indica la anchura del petalo. (*Float*)

species **Etiqueta/Clase** a predecir. Representa la **especie** de la flor. (*String*)

```
"sepal_length","sepal_width","petal_length","petal_width","species"
5.1,3.5,1.4,.2,"Setosa"
4.9,3,1.4,.2,"Setosa"
4.7,3.2,1.3,.2,"Setosa"
4.6,3.1,1.5,.2,"Setosa"
...
```

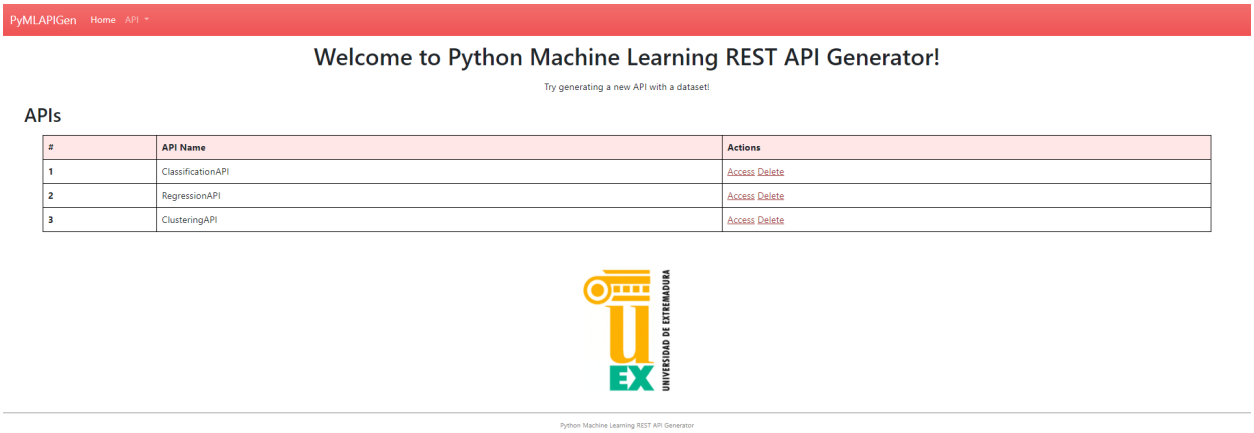
Generar API (Herramienta Web)

En primer lugar, hay que inicializar la herramienta:

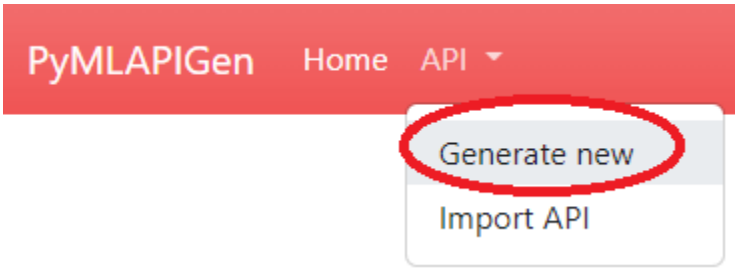
- Si se instaló como paquete usando el comando `pymlapigen`.
- Si se clonó el repositorio con el comando `python run.py`.

```
* Serving Flask app 'pymlapigen' (lazy loading)
* Environment: production
* Debug mode: off
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

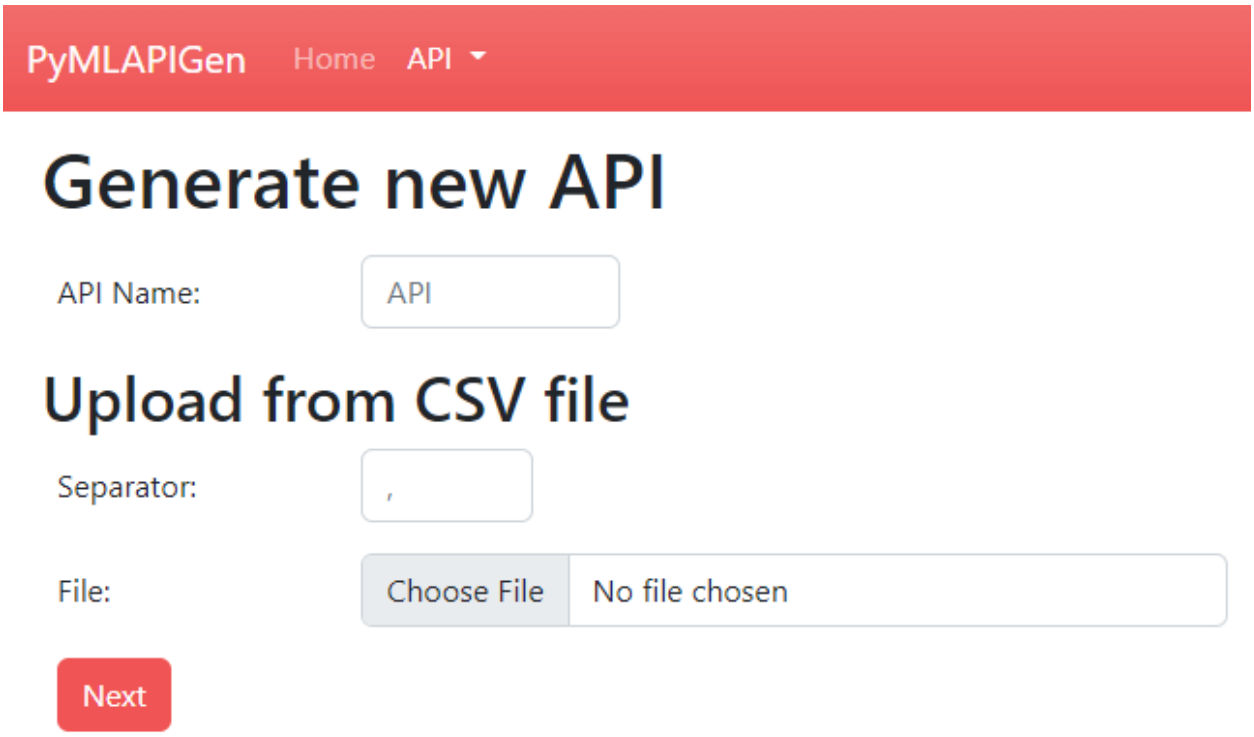
Una vez inicializada la herramienta, acceder a ella a través del URL http://localhost:5000.



Dentro de la aplicación, para comenzar la generación de la API, desplegar el menú *API* y pulsar sobre *Generate New*.



Paso 1



En este paso, se debe escoger el **nombre de la API** a generar y el **dataset** del experimento.

Para este ejemplo, introduciremos en el formulario los siguientes valores:

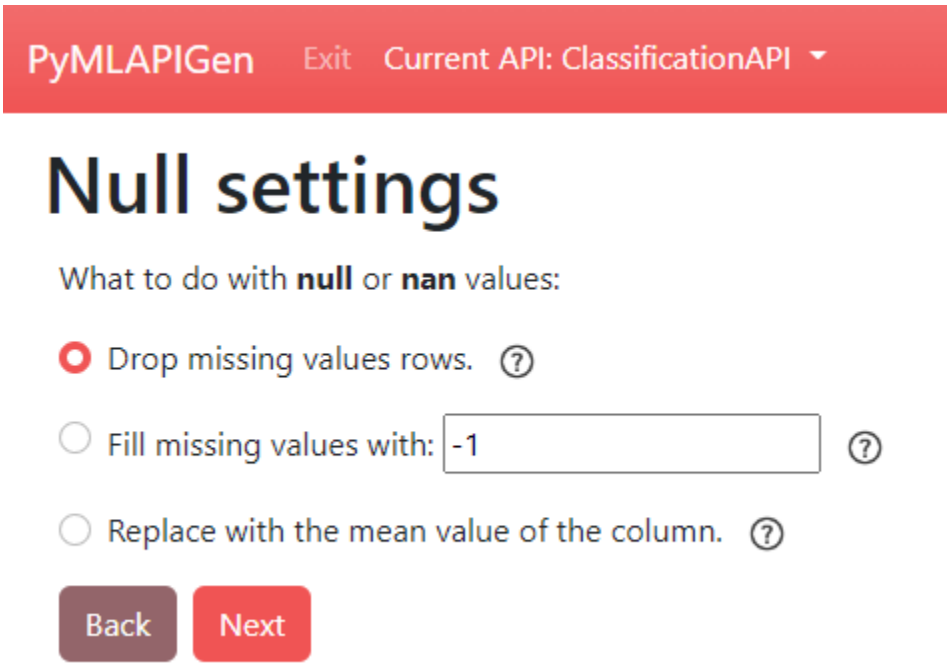
API Name *ClassificationAPI*

Separator ,

File *Fichero descargado del apartado dataset*

Una vez introducidos los parámetros, presionar el botón **Next**.

Paso 2

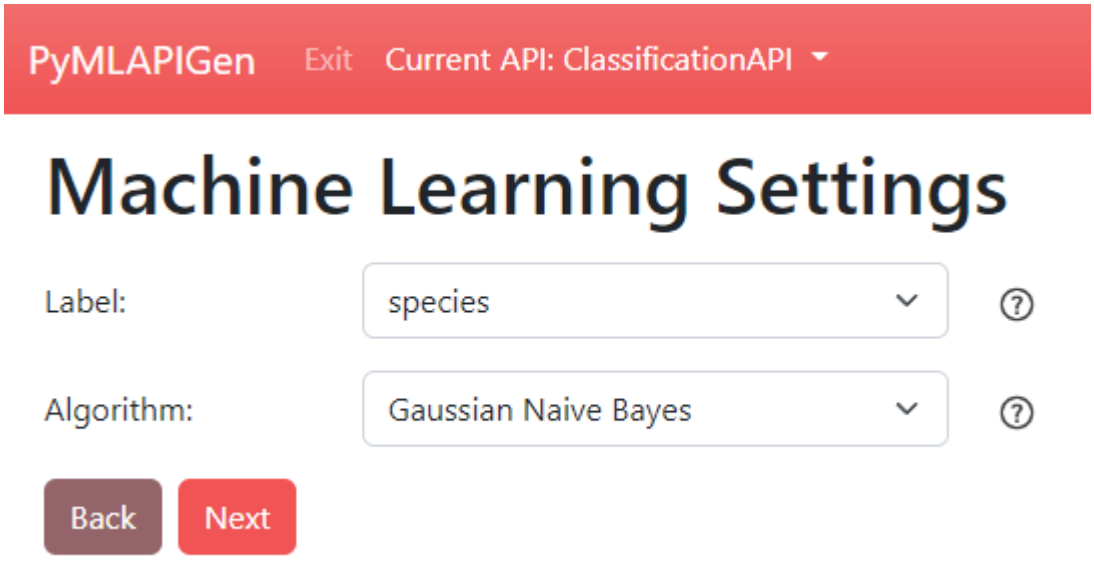


En este paso, se debe escoger **qué hacer** con los **valores que faltan** o **NaN** del dataset.

Como en el dataset de este ejemplo todos los valores **están** y son **válidos**, este paso es irrelevante.

Simplemente presionar el botón **Next**.

Paso 3



En este paso, se debe escoger la **etiqueta/clase** del dataset a partir del cual se entrenará el modelo para predecir en el experimento de clasificación.

También se debe elegir el **algoritmo** de Machine Learning que llevará a cabo el modelo que se entrenará.

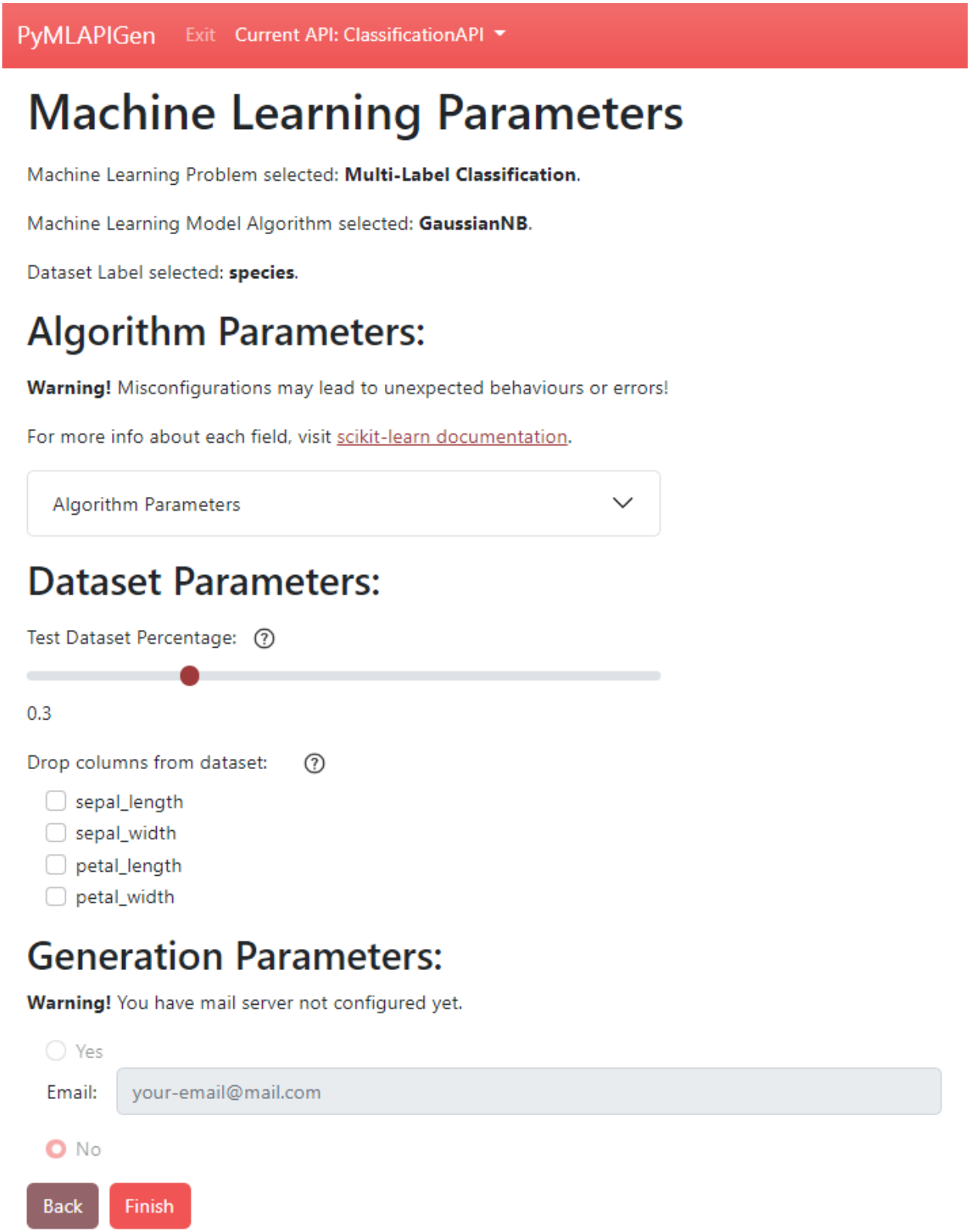
Para este ejemplo, introduciremos en el formulario los siguientes valores:

Label *species*

Algorithm *Classification* > *Gaussian Naive Bayes*

Una vez introducidos los parámetros, presionar el botón **Next**.

Paso 4

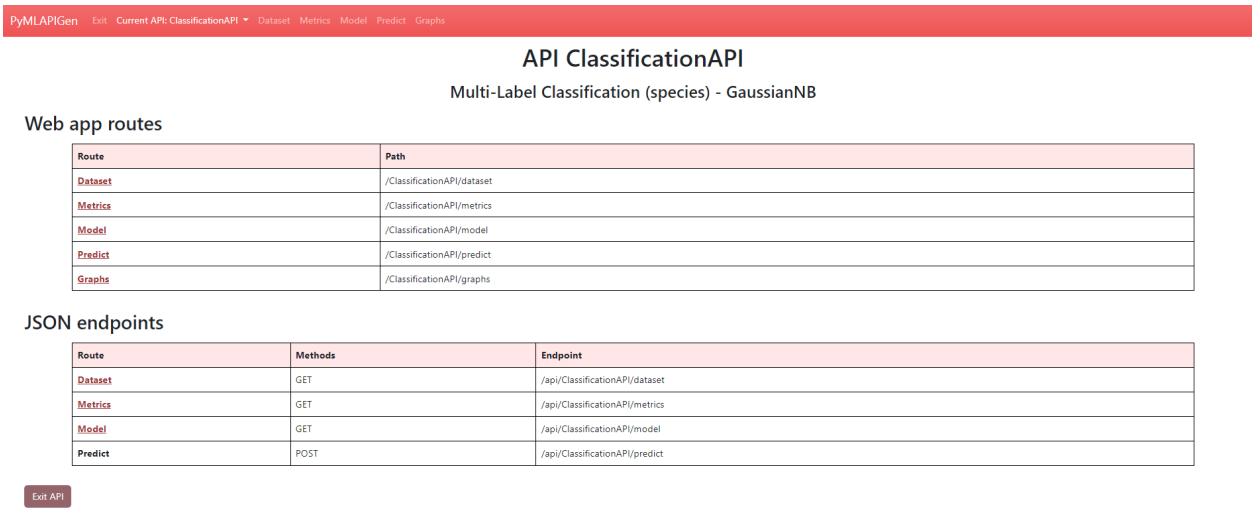

PyMLAPIGen
Exit
Current API: ClassificationAPI
Machine Learning Parameters
Machine Learning Problem selected: Multi-Label Classification.
Machine Learning Model Algorithm selected: GaussianNB.
Dataset Label selected: species.
Algorithm Parameters:
Warning! Misconfigurations may lead to unexpected behaviours or errors!
For more info about each field, visit scikit-learn documentation.
Algorithm Parameters
Dataset Parameters:
Test Dataset Percentage:
0.3
Drop columns from dataset:
sepal_length
sepal_width
petal_length
petal_width
Generation Parameters:
Warning! You have mail server not configured yet.
Yes
Email:
your-email@mail.com
No
Back
Finish

En este último paso, se deben seleccionar los **parámetros adicionales** del experimento.

Para este ejemplo, se dejarán las **opciones por defecto** y se presiona el botón **Finish** para comenzar la **generación de la API**.

Ventanas API (Herramienta Web)

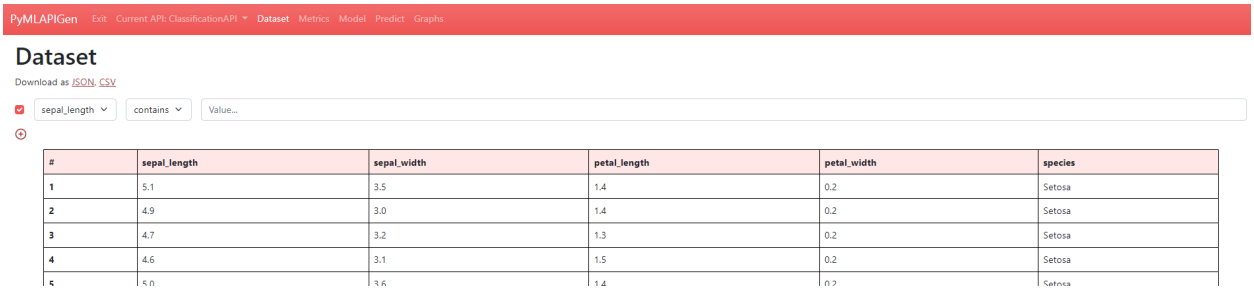
Una vez se genere la API (entrenamiento y evaluación del modelo) se nos redirigirá a la ventana **HOME** de la API (http://localhost:5000/ClassificationAPI).



En esta ventana podrás acceder a las distintas **rutas** y los diferentes **endpoints** de la API.

Además, en la **barra de navegación** podrás navegar entre las distintas ventanas de la API o salir de ella.

Dataset

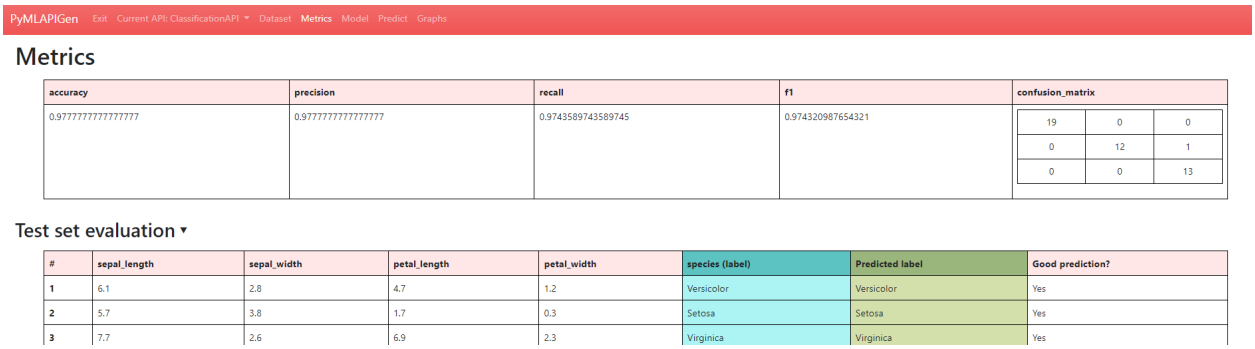


(http://localhost:5000/ClassificationAPI/dataset)

En esta ventana podrás **visualizar** el dataset del experimento de la API generada.

También puedes **ordenar** la tabla pulsando en las cabeceras y aplicar **filtros**.

Métricas

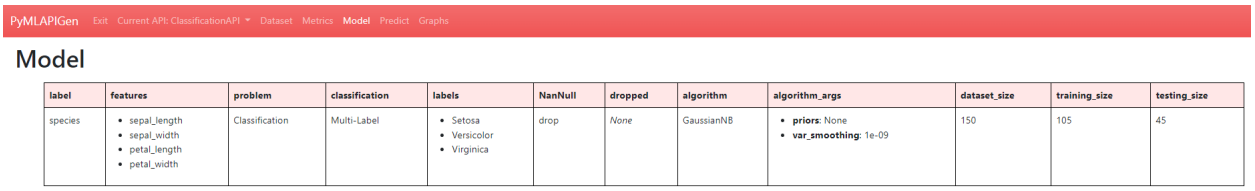


(http://localhost:5000/ClassificationAPI/metrics)

En esta ventana podrás **visualizar** las métricas del experimento de la API generada.

También puedes **desplegar** y **echar un vistazo** al resultado de la **fase de evaluación** del experimento. En él, puedes ver el **valor original** de la etiqueta/clase, el **valor predicho** y si es **correcto o no**.

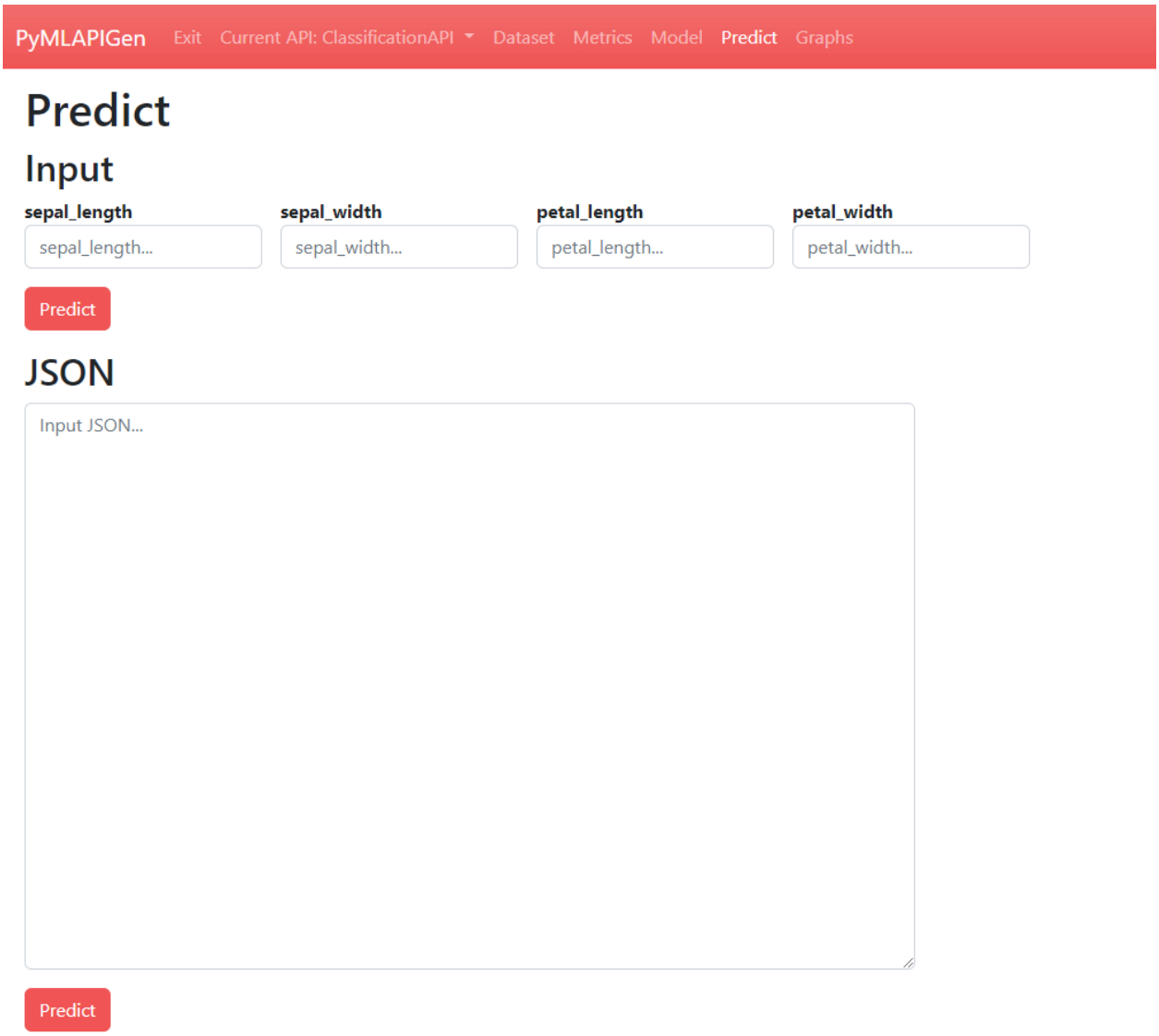
Model



(http://localhost:5000/ClassificationAPI/model)

En esta ventana podrás **visualizar** los parámetros escogidos para el experimento de la API generada.

Predecir



(http://localhost:5000/ClassificationAPI/predict)

En esta ventana podrás realizar **predicciones** utilizando el modelo entrenado del experimento de la API generada.

Puedes realizar las predicciones de diversas formas:

- Introduciendo los datos **manualmente**.
- Utilizando un objeto **JSON** como parámetro de entrada.
- A través de un fichero **CSV** de entrada.

Una vez introducidos los datos y pulsado el botón **Predict**, si los datos son correctos y no hay ningún problema, se nos mostrará abajo el resultado de la predicción.

Por ejemplo, mandar a predecir el siguiente objeto JSON:

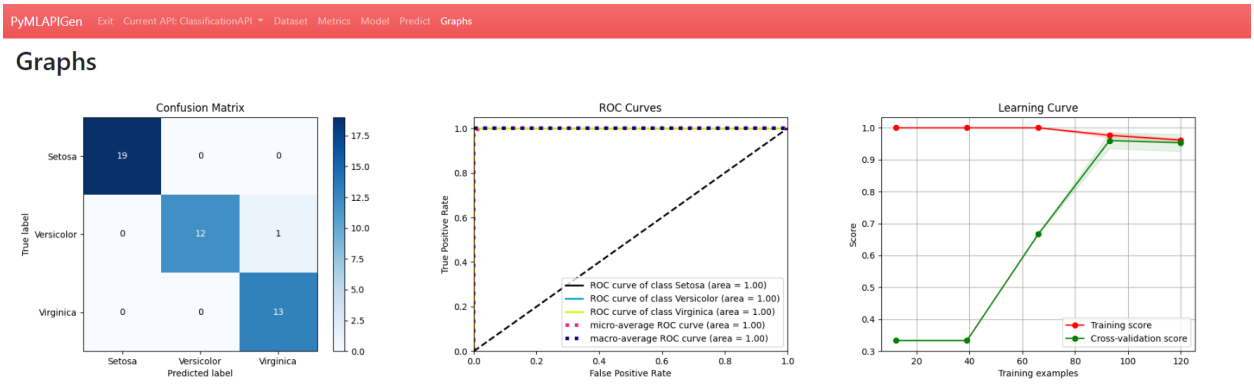
```
[{
   "sepal_length": 4.9,
   "sepal_width": 3.0,
   "petal_length": 1.4,
   "petal_width": 0.2
}, {
   "sepal_length": 6.3,
   "sepal_width": 2.7,
   "petal_length": 4.9,
   "petal_width": 1.8
}, {
   "sepal_length": 4.8,
   "sepal_width": 3.1,
   "petal_length": 1.6,
   "petal_width": 0.2
}]
```

El resultado es el siguiente:

Prediction Result

#	sepal_length	sepal_width	petal_length	petal_width	result
1	4.9	3.0	1.4	0.2	Setosa
2	6.3	2.7	4.9	1.8	Virginica
3	4.8	3.1	1.6	0.2	Setosa

Gráficos



(http://localhost:5000/ClassificationAPI/graphs)

En esta ventana podrás **visualizar** distintos **gráficos** en función del modelo del experimento de la API generada.

Generar API (Petición JSON)

Además de la aplicación web, es posible generar y utilizar la API a partir de **peticiones JSON** a los endpoints de la aplicación.

Petición JSON (/load)

Para generar una API equivalente al generado con la aplicación web, se debe enviar una petición **HTTP POST** al endpoint http://localhost:5000/api/load. El cuerpo de esta petición HTTP POST será este JSON.

Si todo funciona correctamente, se nos debería devolver el siguiente resultado de la operación:

```
{
   "success": "The API has been successfully generated and its now operable.",
   "endpoints": {
      "home": {
         "methods": "GET",
         "endpoint": "/api/ClassificationAPIFromJSON"
      },
      "dataset": {
         "methods": "GET",
         "endpoint": "/api/ClassificationAPIFromJSON/dataset"
      },
      "metrics": {
         "methods": "GET",
         "endpoint": "/api/ClassificationAPIFromJSON/metrics"
      },
      "model": {
         "methods": "GET",
         "endpoint": "/api/ClassificationAPIFromJSON/model"
      },
      "predict": {
         "methods": "POST",
         "endpoint": "/api/ClassificationAPIFromJSON/predict"
      }
   }
}
```

Endpoints API (Peticiones JSON)

GET Dataset

Endpoint: http://localhost:5000/api/ClassificationAPIFromJSON/dataset

En este endpoint podrás **consultar** el **dataset** del experimento de la API generada.

Resultado:

```
[
   {
      "petal_length": 1.4,
      "petal_width": 0.2,
```

(continué en la próxima página)

(proviene de la página anterior)


```
      "sepal_length": 5.1,
      "sepal_width": 3.5,
      "species": "Setosa"
   },
   {
      "petal_length": 1.4,
      "petal_width": 0.2,
      "sepal_length": 4.9,
      "sepal_width": 3.0,
      "species": "Setosa"
   },
   ...
   {
      "petal_length": 5.1,
      "petal_width": 1.8,
      "sepal_length": 5.9,
      "sepal_width": 3.0,
      "species": "Virginica"
   }
]
```


GET Metrics

Endpoint: http://localhost:5000/api/ClassificationAPIFromJSON/metrics

En este endpoint podrás **consultar** la **evaluación** del experimento de la API generada.

Resultado:


```
{
   "accuracy": 0.9777777777777777,
   "precision": 0.9777777777777777,
   "recall": 0.9743589743589745,
   "f1": 0.974320987654321,
   "confusion_matrix": [
         [
            19,
            0,
            0
         ],
         [
            0,
            12,
            1
         ],
         [
            0,
            0,
            13
         ]
      ]
   }
```

GET Model

Endpoint: http://localhost:5000/api/ClassificationAPIFromJSON/model

En este endpoint podrás **consultar** los **parámetros del experimento** de la API generada.

Resultado:


```
{
   "label": "species",
   "features": [
      "petal_length",
      "petal_width",
      "sepal_length",
      "sepal_width"
   ],
   "problem": "Classification",
   "classification": "Multi-Label",
   "labels": [
      "Setosa",
      "Versicolor",
      "Virginica"
   ],
   "NanNull": "drop",
   "dropped": [],
   "algorithm": "GaussianNB",
   "algorithm_args": {},
   "dataset_size": 150,
   "training_size": 105,
   "testing_size": 45
}
```


POST Predict

Endpoint: http://localhost:5000/api/ClassificationAPIFromJSON/predict

En este endpoint podrás realizar **predicciones** al experimento de la API generada.

Para ello, en el cuerpo de la petición HTTP POST se introducirá un objeto JSON con los parámetros de entrada.

Por ejemplo se va a mostrar una petición HTTP POST cuyo cuerpo es:


```
[{
   "sepal_length": 4.9,
   "sepal_width": 3.0,
   "petal_length": 1.4,
   "petal_width": 0.2
}, {
   "sepal_length": 6.3,
   "sepal_width": 2.7,
   "petal_length": 4.9,
   "petal_width": 1.8
}, {
   "sepal_length": 4.8,
```


(continué en la próxima página)

(proviene de la página anterior)

```
   "sepal_width": 3.1,
   "petal_length": 1.6,
   "petal_width": 0.2
}]
```

Resultado:

```
[
 "Setosa",
 "Virginica",
 "Setosa"
]
```

2.2.2 Regresión

En este apartado se mostrará un ejemplo de uso de la aplicación para **generar una API** con un experimento de **regresión**.

En el dominio de Machine Learning, la **regresión** es un tipo de aprendizaje **supervisado**. El valor que se quiere predecir es un **valor numérico** (etiqueta) dentro de un **dominio continuo** a partir del resto de atributos.

Dataset

El dataset que se va a utilizar para este ejemplo práctico consiste en un conjunto de datos de **propinas** de un restaurante.

Este dataset es **público** y tiene un formato **CSV**. Puede descargarlo pulsándo aqui.

Las columnas de este dataset son las siguientes:

total_bill **Atributo** que indica la cuenta total de la comida. (*Float*)

tip **Etiqueta** a predecir. Representa la propina que da el cliente. (*Float*)

sex **Atributo** que indica el sexo del cliente. (*String*)

smoker **Atributo** que indica si el cliente fuma. (*String*)

day **Atributo** que indica el día de la semana en el que tuvo lugar la comida. (*String*)

time **Atributo** que indica si la comida fue almuerzo o cena. (*String*)

size **Atributo** que indica el número de personas que conformaban el cliente. (*Integer*)

```
"total_bill","tip","sex","smoker","day","time","size"
16.99,1.01,"Female","No","Sun","Dinner",2
10.34,1.66,"Male","No","Sun","Dinner",3
21.01,3.5,"Male","No","Sun","Dinner",3
23.68,3.31,"Male","No","Sun","Dinner",2
...
```

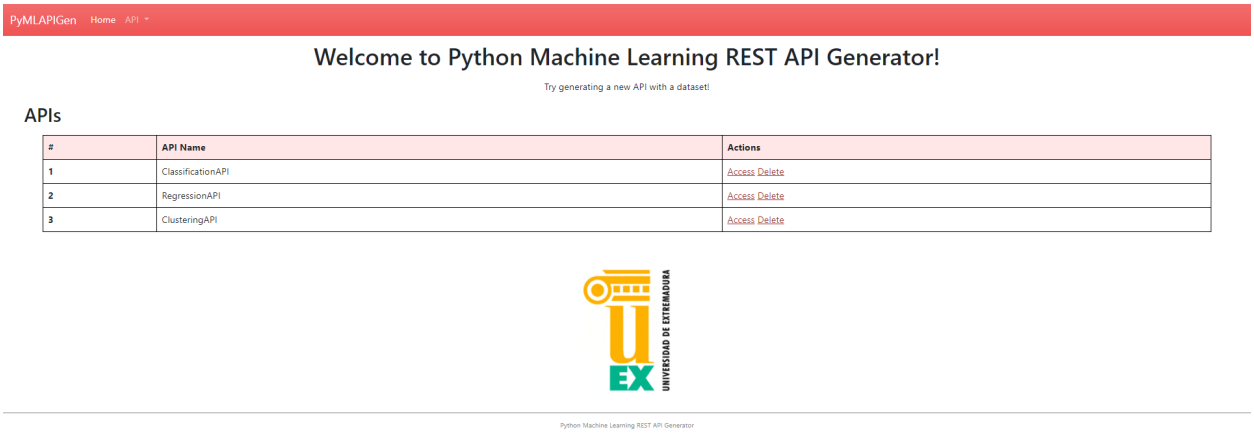
Generar API (Herramienta Web)

En primer lugar, hay que inicializar la herramienta:

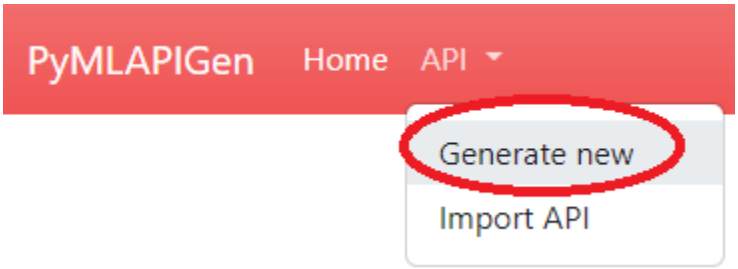
- Si se instaló como paquete usando el comando `pymlapigen`.
- Si se clonó el repositorio con el comando `python run.py`.

```
* Serving Flask app 'pymlapigen' (lazy loading)
* Environment: production
* Debug mode: off
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

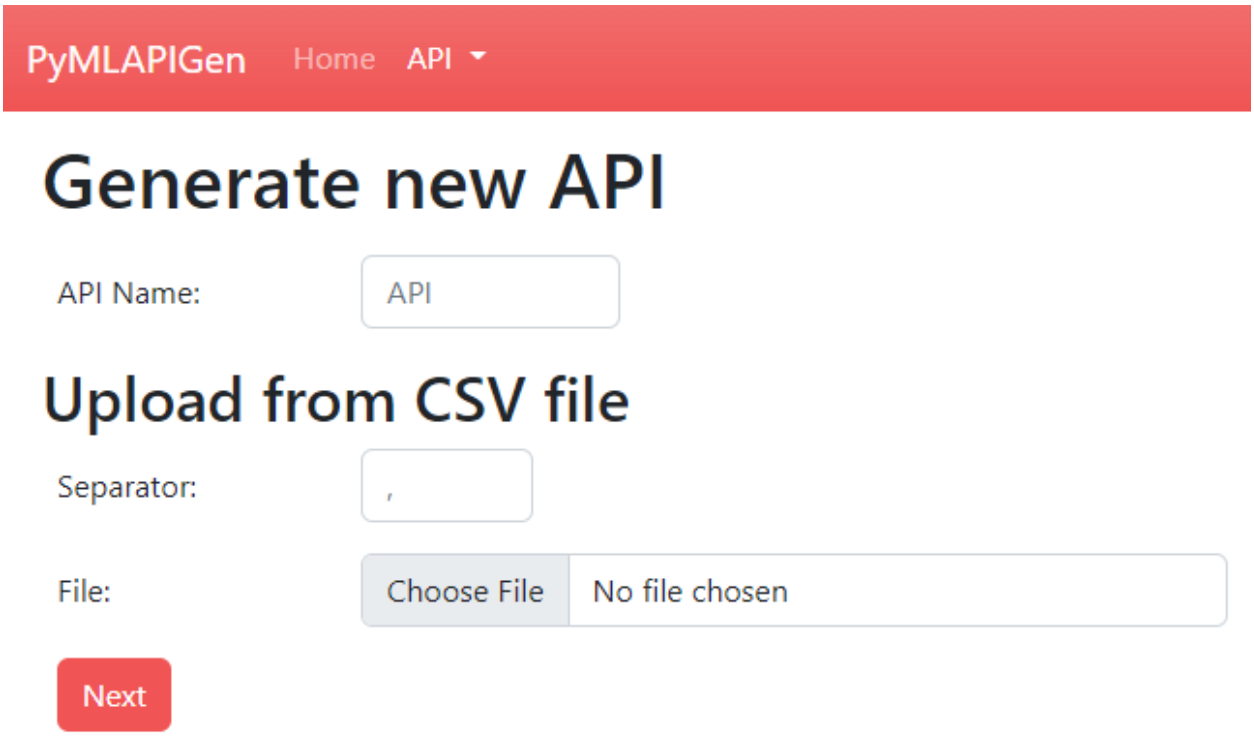
Una vez inicializada la herramienta, acceder a ella a través del URL http://localhost:5000.



Dentro de la aplicación, para comenzar la generación de la API, desplegar el menú *API* y pulsar sobre *Generate New*.



Paso 1



En este paso, se debe escoger el **nombre de la API** a generar y el **dataset** del experimento.

Para este ejemplo, introduciremos en el formulario los siguientes valores:

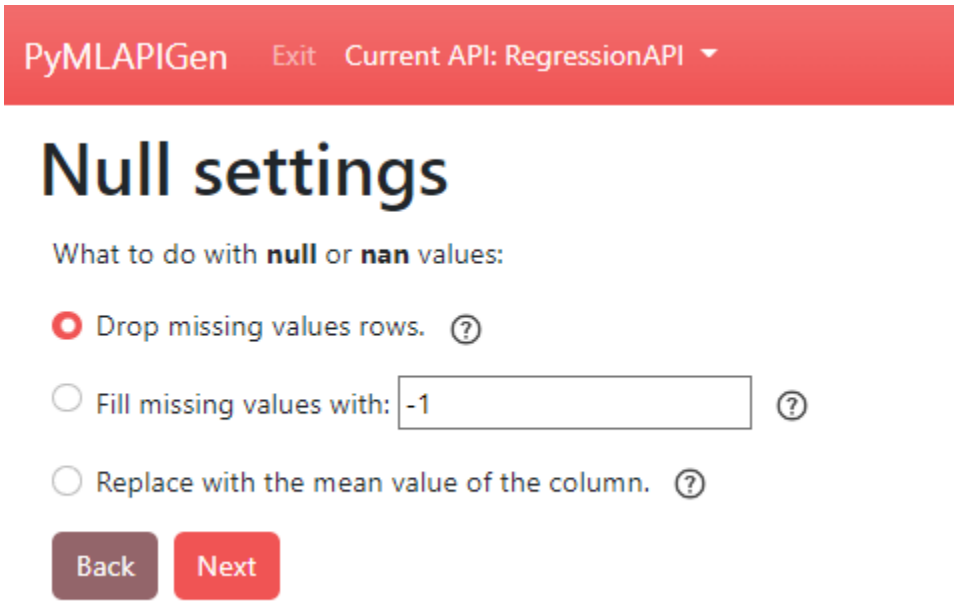
API Name *RegressionAPI*

Separator ,

File *Fichero descargado del apartado dataset*

Una vez introducidos los parámetros, presionar el botón **Next**.

Paso 2

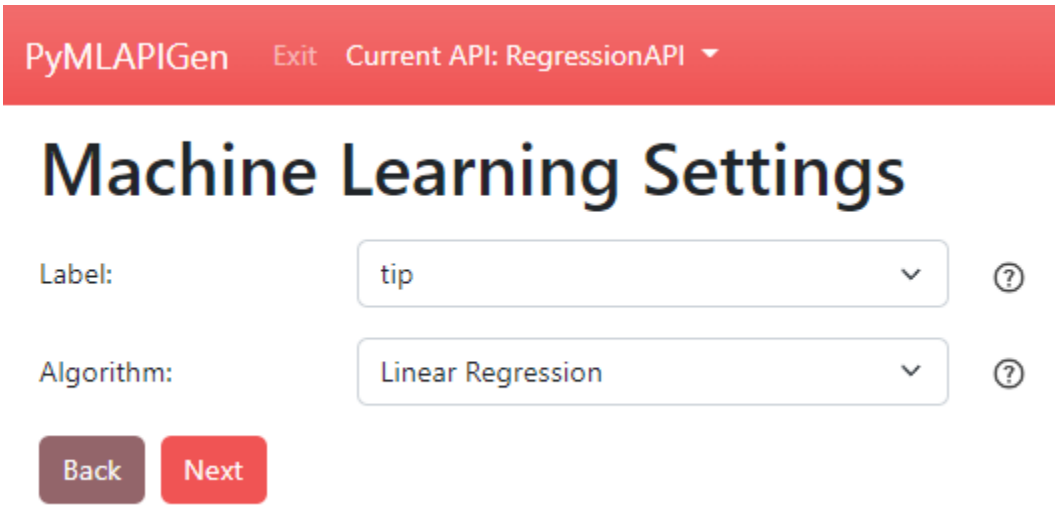


En este paso, se debe escoger **qué hacer** con los **valores que faltan** o **NaN** del dataset.

Como en el dataset de este ejemplo todos los valores **están** y son **válidos**, este paso es irrelevante.

Simplemente presionar el botón **Next**.

Paso 3



En este paso, se debe escoger la **etiqueta** del dataset a partir del cual se entrenará el modelo para predecir en el experimento de regresión.

También se debe elegir el **algoritmo** de Machine Learning que llevará a cabo el modelo que se entrenará.

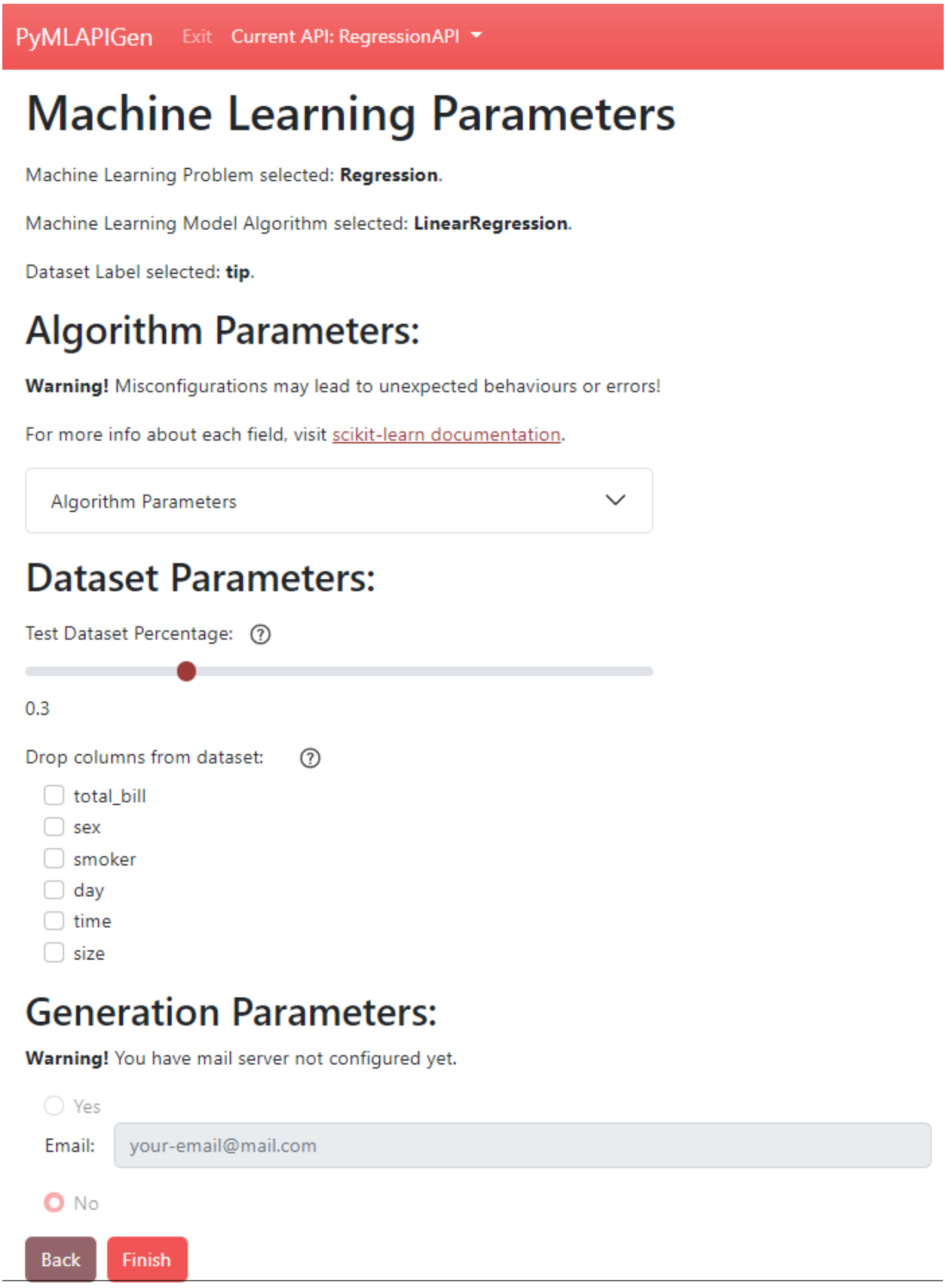
Para este ejemplo, introduciremos en el formulario los siguientes valores:

Label *tips*

Algorithm *Regression > Linear Regression*

Una vez introducidos los parámetros, presionar el botón **Next**.

Paso 4

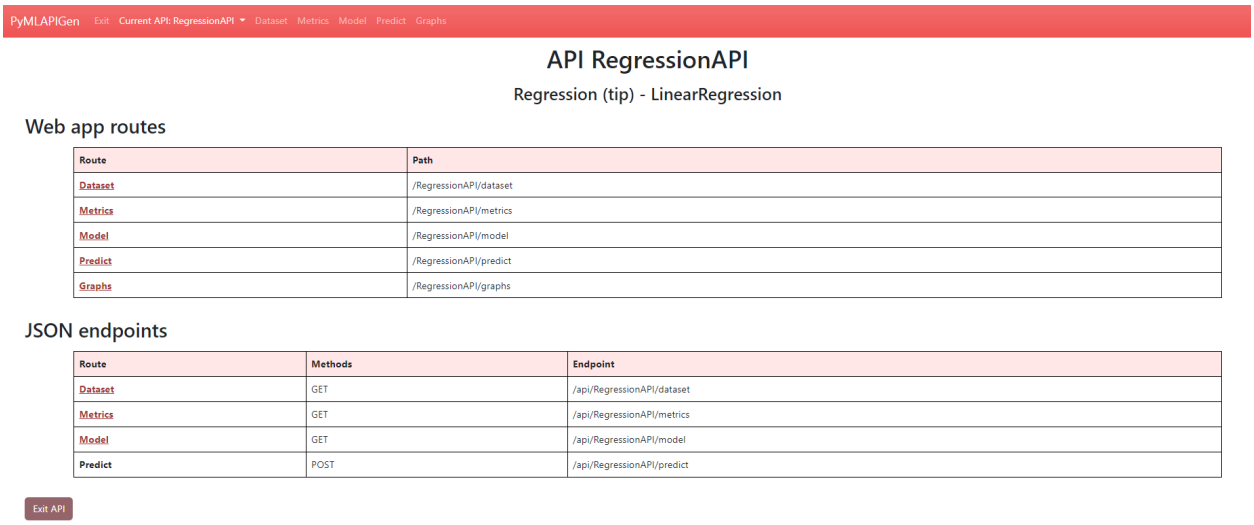

PyMLAPIGen
Exit
Current API: RegressionAPI
Machine Learning Parameters
Machine Learning Problem selected: Regression.
Machine Learning Model Algorithm selected: LinearRegression.
Dataset Label selected: tip.
Algorithm Parameters:
Warning! Misconfigurations may lead to unexpected behaviours or errors!
For more info about each field, visit scikit-learn documentation.
Algorithm Parameters
Dataset Parameters:
Test Dataset Percentage:
0.3
Drop columns from dataset:
total_bill
sex
smoker
day
time
size
Generation Parameters:
Warning! You have mail server not configured yet.
Yes
Email:
your-email@mail.com
No
Back
Finish

En este último paso, se deben seleccionar los **parámetros adicionales** del experimento.

Para este ejemplo, se dejarán las **opciones por defecto** y se presiona el botón **Finish** para comenzar la **generación de la API**.

Ventanas API (Herramienta Web)

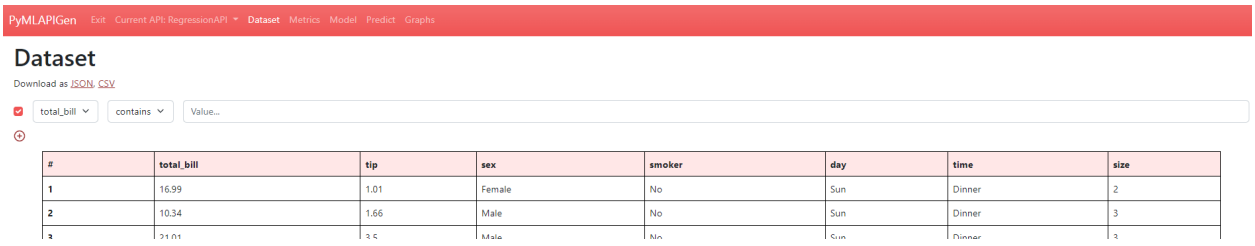
Una vez se genere la API (entrenamiento y evaluación del modelo) se nos redirigirá a la ventana **HOME** de la API (http://localhost:5000/RegressionAPI).



En esta ventana podrás acceder a las distintas **rutas** y los diferentes **endpoints** de la API.

Además, en la **barra de navegación** podrás navegar entre las distintas ventanas de la API o salir de ella.

Dataset

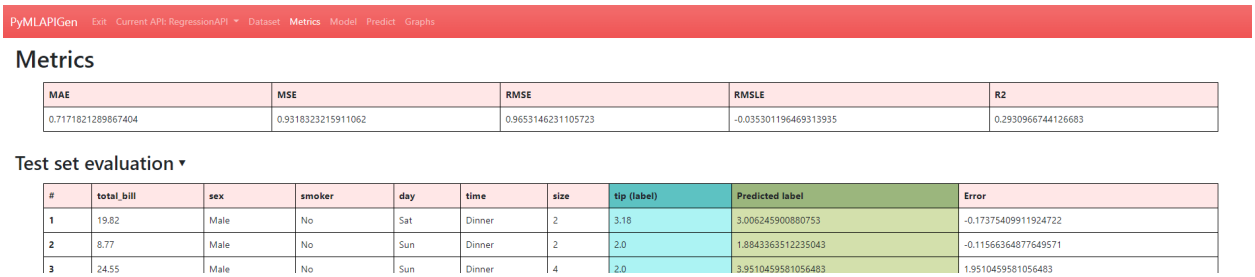


(http://localhost:5000/RegressionAPI/dataset)

En esta ventana podrás **visualizar** el dataset del experimento de la API generada.

También puedes **ordenar** la tabla pulsando en las cabeceras y aplicar **filtros**.

Métricas

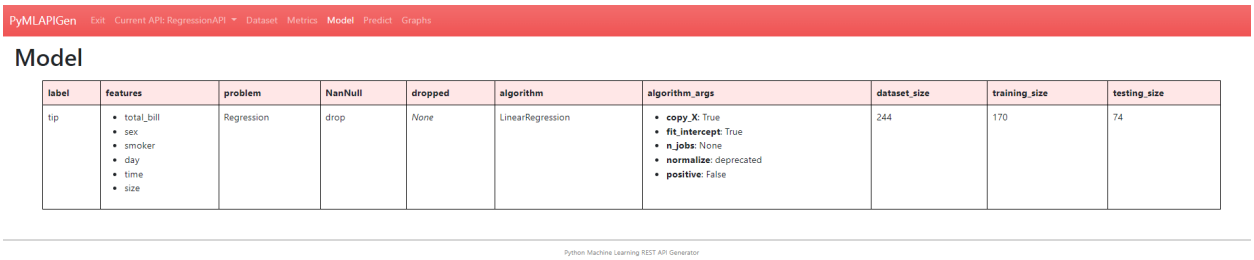


(http://localhost:5000/RegressionAPI/metrics)

En esta ventana podrás **visualizar** las métricas del experimento de la API generada.

También puedes **desplegar** y **echar un vistazo** al resultado de la **fase de evaluación** del experimento. En él, puedes ver el **valor original** de la etiqueta, el **valor predicho** y el error cometido.

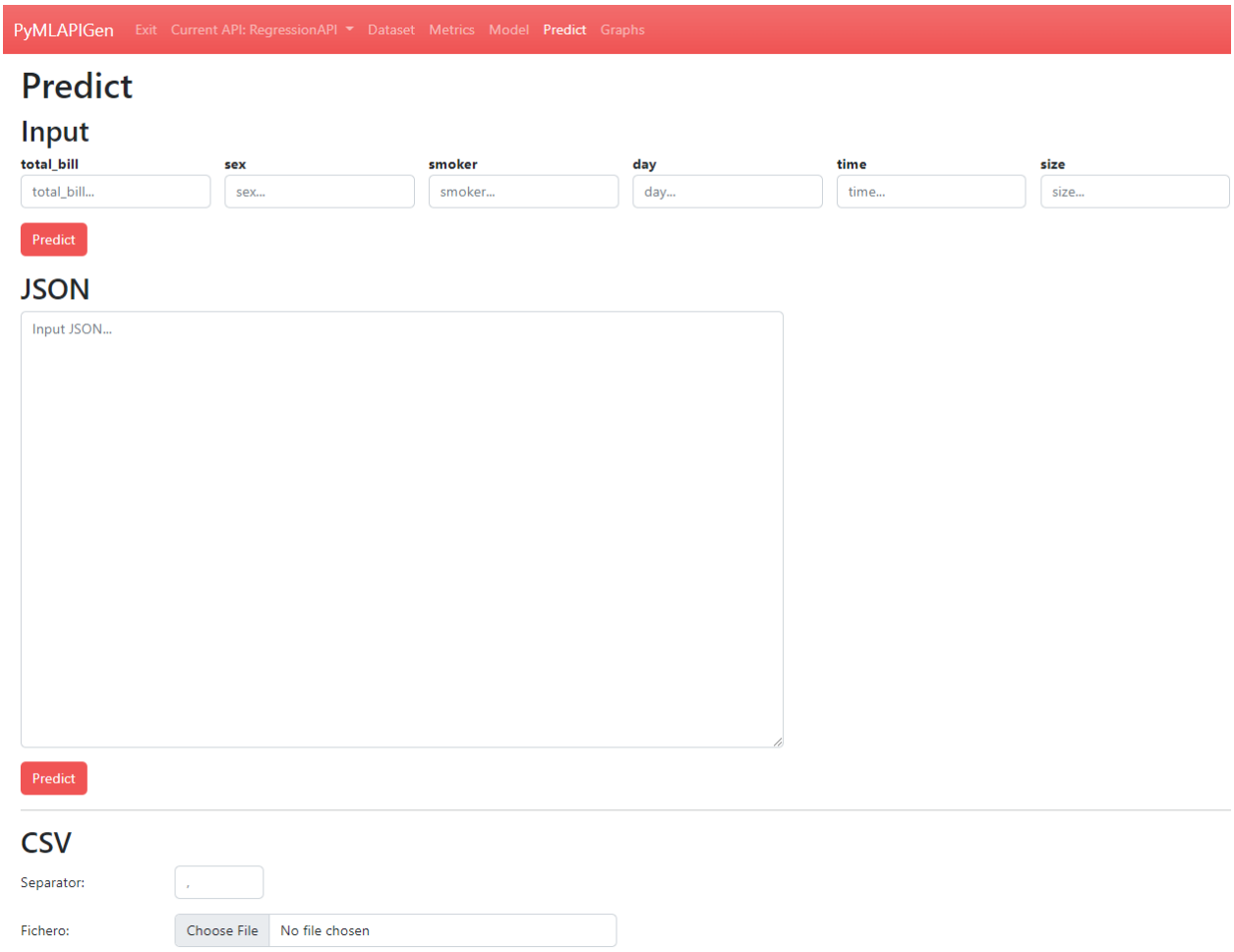
Model



(http://localhost:5000/RegressionAPI/model)

En esta ventana podrás **visualizar** los parámetros escogidos para el experimento de la API generada.

Predecir



(http://localhost:5000/RegressionAPI/predict)

En esta ventana podrás realizar **predicciones** utilizando el modelo entrenado del experimento de la API generada.

Puedes realizar las predicciones de diversas formas:

- Introduciendo los datos **manualmente**.
- Utilizando un objeto **JSON** como parámetro de entrada.
- A través de un fichero **CSV** de entrada.

Una vez introducidos los datos y pulsado el botón **Predict**, si los datos son correctos y no hay ningún problema, se nos mostrará abajo el resultado de la predicción.

Por ejemplo, mandar a predecir el siguiente objeto JSON:

```
[{
    "total_bill": 24.08,
    "sex": "Female",
    "smoker": "No",
    "day": "Thur",
    "time": "Lunch",
```

(continué en la próxima página)

(proviene de la página anterior)

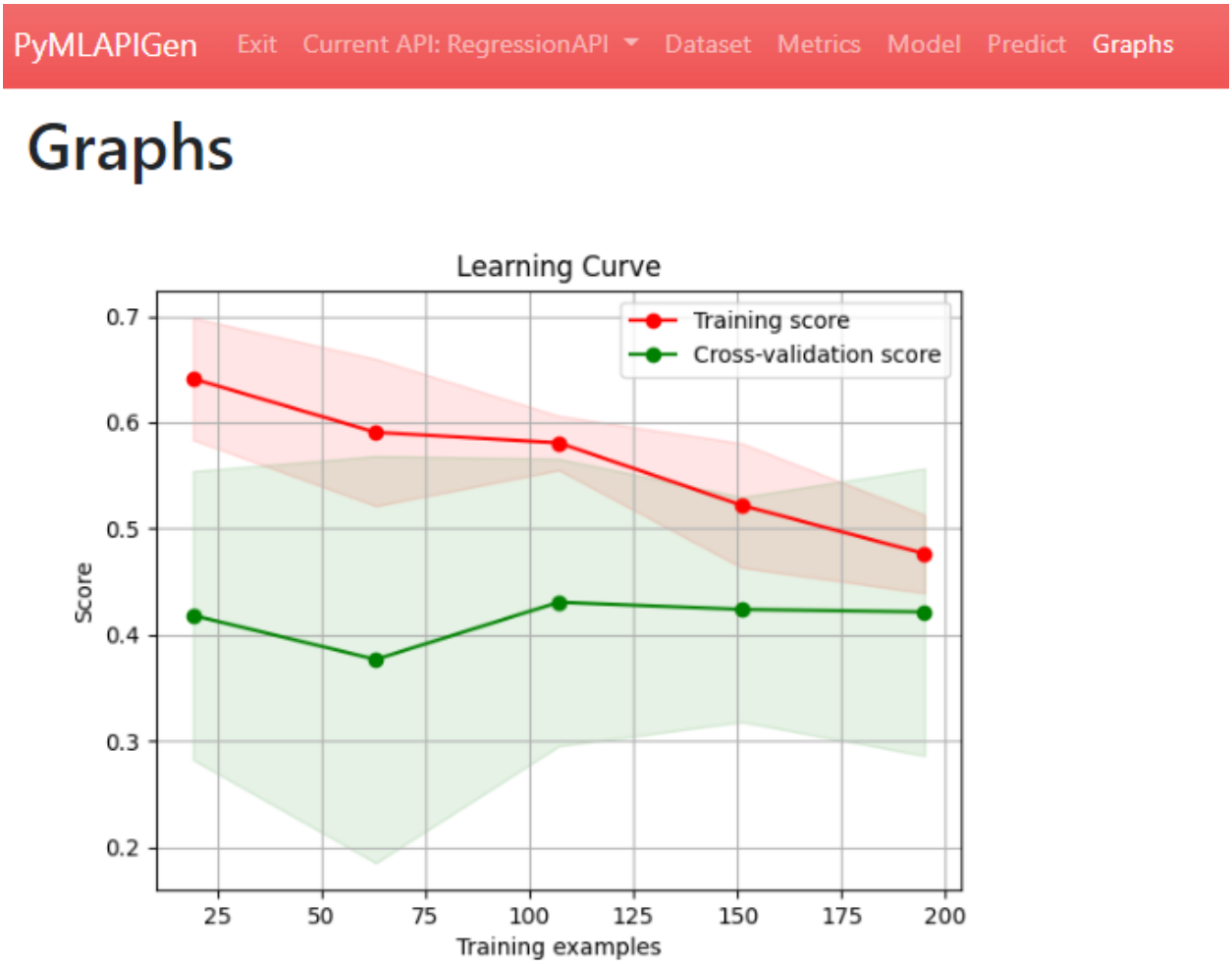
```
    "size": 4
}, {
    "total_bill": 20.45,
    "sex": "Male",
    "smoker": "No",
    "day": "Sat",
    "time": "Dinner",
    "size": 4
}, {
    "total_bill": 13.42,
    "sex": "Male",
    "smoker": "Yes",
    "day": "Fri",
    "time": "Lunch",
    "size": 2
}]
```

El resultado es el siguiente:

Prediction Result

#	total_bill	sex	smoker	day	time	size	result
1	24.08	Female	No	Thur	Lunch	4	3.9641966337558516
2	20.45	Male	No	Sat	Dinner	4	3.6125974887136274
3	13.42	Male	Yes	Fri	Lunch	2	2.4132531382683773

Gráficos



(http://localhost:5000/RegressionAPI/graphs)

En esta ventana podrás **visualizar** distintos **gráficos** en función del modelo del experimento de la API generada.

Generar API (Petición JSON)

Además de la aplicación web, es posible generar y utilizar la API a partir de **peticiones JSON** a los endpoints de la aplicación.

Petición JSON (/load)

Para generar una API equivalente al generado con la aplicación web, se debe enviar una petición **HTTP POST** al endpoint http://localhost:5000/api/load. El cuerpo de esta petición HTTP POST será este JSON.

Si todo funciona correctamente, se nos debería devolver el siguiente resultado de la operación:


```
{
    "success": "The API has been successfully generated and its now operable.",
    "endpoints": {
        "home": {
            "methods": "GET",
            "endpoint": "/api/RegressionAPIFromJSON"
        },
        "dataset": {
            "methods": "GET",
            "endpoint": "/api/RegressionAPIFromJSON/dataset"
        },
        "metrics": {
            "methods": "GET",
            "endpoint": "/api/RegressionAPIFromJSON/metrics"
        },
        "model": {
            "methods": "GET",
            "endpoint": "/api/RegressionAPIFromJSON/model"
        },
        "predict": {
            "methods": "POST",
            "endpoint": "/api/RegressionAPIFromJSON/predict"
        }
    }
}
```


Endpoints API (Peticiones JSON)

GET Dataset

Endpoint: http://localhost:5000/api/RegressionAPIFromJSON/dataset

En este endpoint podrás **consultar** el **dataset** del experimento de la API generada.

Resultado:


```
[
   {
     "day": "Sun",
     "sex": "Female",
     "size": 2,
     "smoker": "No",
     "time": "Dinner",
     "tip": 1.01,
     "total_bill": 16.99
   },
```


(continué en la próxima página)

(proviene de la página anterior)


```
  {
    "day": "Sun",
    "sex": "Male",
    "size": 3,
    "smoker": "No",
    "time": "Dinner",
    "tip": 1.66,
    "total_bill": 10.34
  },
  ...
  {
    "day": "Thur",
    "sex": "Female",
    "size": 2,
    "smoker": "No",
    "time": "Dinner",
    "tip": 3.0,
    "total_bill": 18.78
  }
]
```


GET Metrics

Endpoint: http://localhost:5000/api/RegressionAPIFromJSON/metrics

En este endpoint podrás **consultar** la **evaluación** del experimento de la API generada.

Resultado:


```
{
    "MAE": 0.7171821289867383,
    "MSE": 0.9318323215910987,
    "RMSE": 0.9653146231105684,
    "RMSLE": -0.03530119646931796,
    "R2": 0.29309667441267395
}
```


GET Model

Endpoint: http://localhost:5000/api/RegressionAPIFromJSON/model

En este endpoint podrás **consultar** los **parámetros del experimento** de la API generada.

Resultado:


```
{
    "label": "tip",
    "features": [
        "day",
        "sex",
        "size",
```


(continué en la próxima página)

(proviene de la página anterior)

```
        "smoker",
        "time",
        "total_bill"
    ],
    "problem": "Regression",
    "NanNull": "drop",
    "dropped": [],
    "algorithm": "LinearRegression",
    "algorithm_args": {},
    "dataset_size": 244,
    "training_size": 170,
    "testing_size": 74
}
```

POST Predict

Endpoint: http://localhost:5000/api/RegressionAPIFromJSON/predict

En este endpoint podrás realizar **predicciones** al experimento de la API generada.

Para ello, en el cuerpo de la petición HTTP POST se introducirá un objeto JSON con los parámetros de entrada.

Por ejemplo se va a mostrar una petición HTTP POST cuyo cuerpo es:

```
[{
     "total_bill": 24.08,
     "sex": "Female",
     "smoker": "No",
     "day": "Thur",
     "time": "Lunch",
     "size": 4
 }, {
     "total_bill": 20.45,
     "sex": "Male",
     "smoker": "No",
     "day": "Sat",
     "time": "Dinner",
     "size": 4
 }, {
     "total_bill": 13.42,
     "sex": "Male",
     "smoker": "Yes",
     "day": "Fri",
     "time": "Lunch",
     "size": 2
 }]
```

Resultado:

```
[
    3.9641966337558423,
    3.612597488713627,
```

(continué en la próxima página)

(proviene de la página anterior)

```
    2.413253138268394
]
```

2.2.3 Clustering

En este apartado se mostrará un ejemplo de uso de la aplicación para **generar una API** con un experimento de **clustering**.

En el dominio de Machine Learning, el **clustering** es un tipo de aprendizaje **no supervisado**. El objetivo es **agrupar** el conjunto de datos en una serie de **conjuntos similares** (*clústeres*).

Dataset

El dataset que se va a utilizar para este ejemplo práctico consiste en un conjunto de datos de **flores iris**.

Este dataset es **público** y tiene un formato **CSV**. Puede descargarlo pulsándo aqui.

Las columnas de este dataset son las siguientes:

sepal_length **Atributo** que indica la longitud del sépalo. (*Float*)

sepal_width **Atributo** que indica la anchura del sépalo. (*Float*)

petal_length **Atributo** que indica la longitud del petalo. (*Float*)

petal_width **Atributo** que indica la anchura del petalo. (*Float*)

species **Atributo** que indica la especie de la flor. (*String*)

En el caso ideal, los conjuntos formados coincidirán con las especies.

```
"sepal_length","sepal_width","petal_length","petal_width","species"
5.1,3.5,1.4,.2,"Setosa"
4.9,3,1.4,.2,"Setosa"
4.7,3.2,1.3,.2,"Setosa"
4.6,3.1,1.5,.2,"Setosa"
...
```

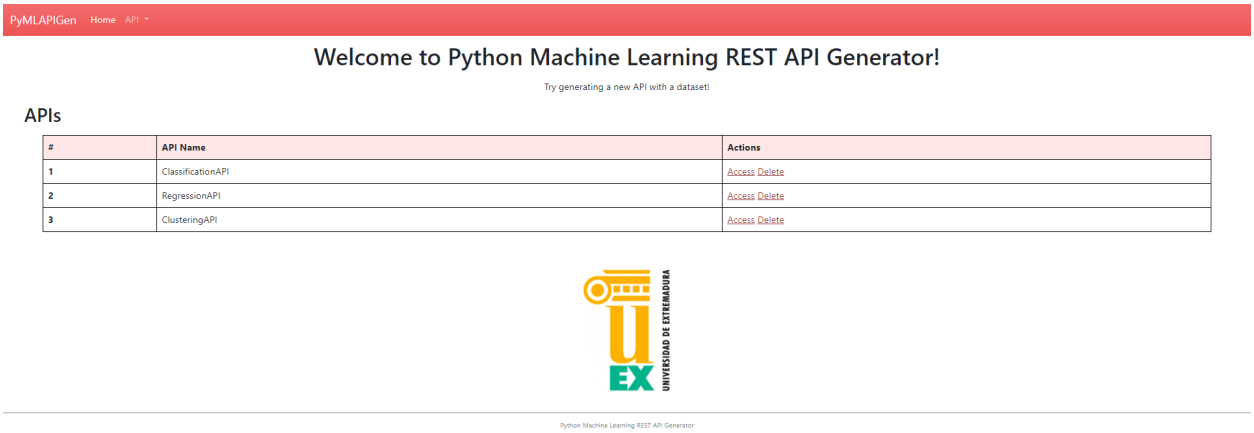
Generar API (Herramienta Web)

En primer lugar, hay que inicializar la herramienta:

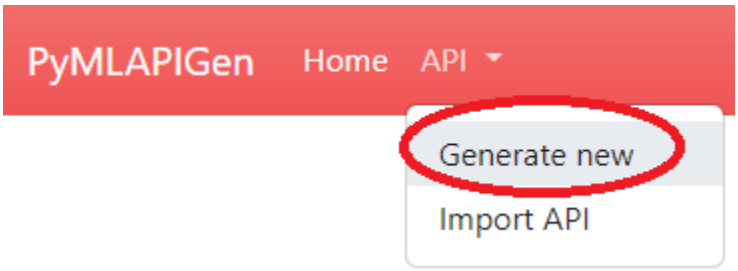
- Si se instaló como paquete usando el comando `pymlapigen`.
- Si se clonó el repositorio con el comando `python run.py`.

```
* Serving Flask app 'pymlapigen' (lazy loading)
* Environment: production
* Debug mode: off
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

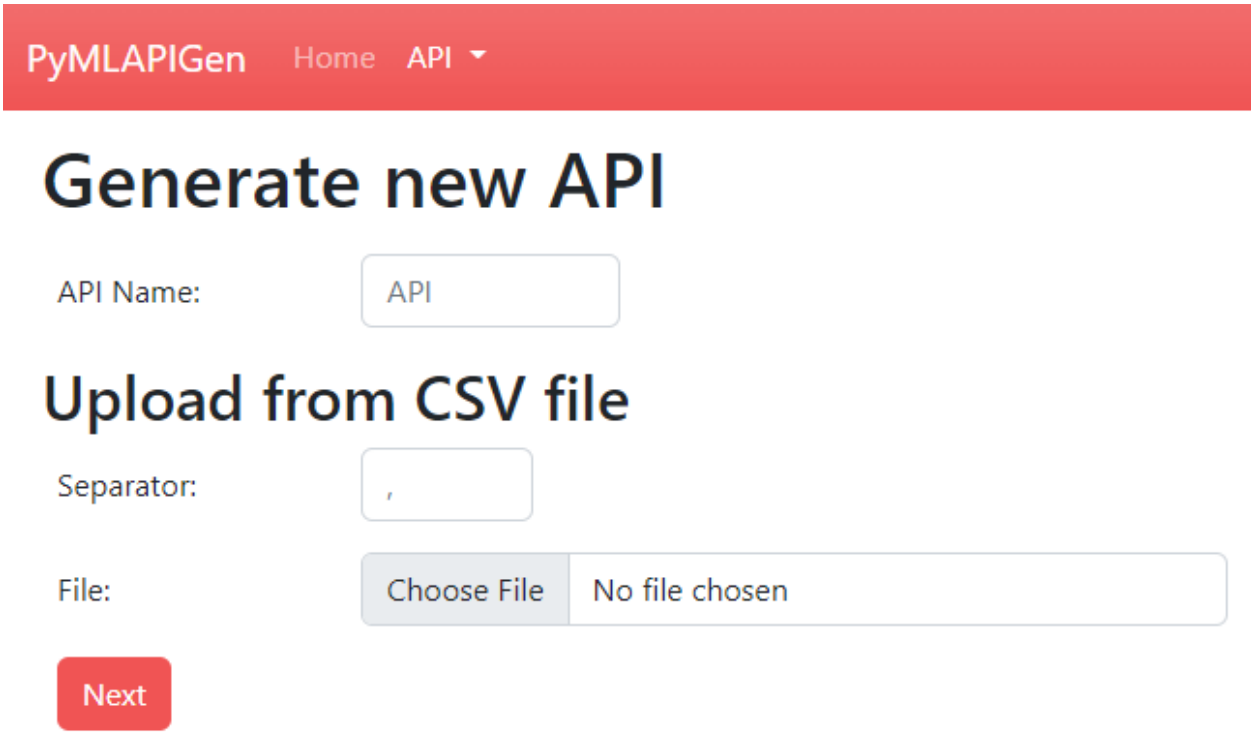
Una vez inicializada la herramienta, acceder a ella a través del URL http://localhost:5000.



Dentro de la aplicación, para comenzar la generación de la API, desplegar el menú *API* y pulsar sobre *Generate New*.



Paso 1



En este paso, se debe escoger el **nombre de la API** a generar y el **dataset** del experimento.

Para este ejemplo, introduciremos en el formulario los siguientes valores:

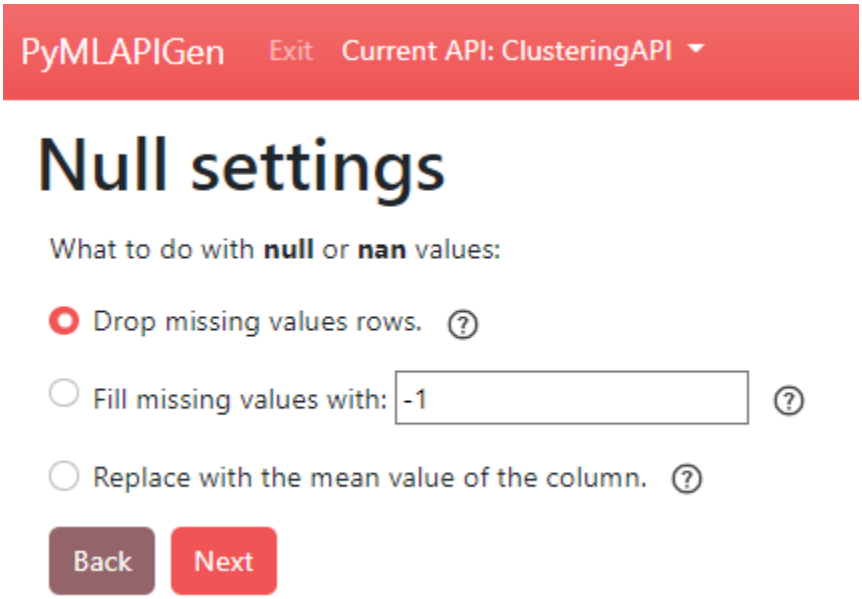
API Name *ClusteringAPI*

Separator ,

File *Fichero descargado del apartado dataset*

Una vez introducidos los parámetros, presionar el botón **Next**.

Paso 2

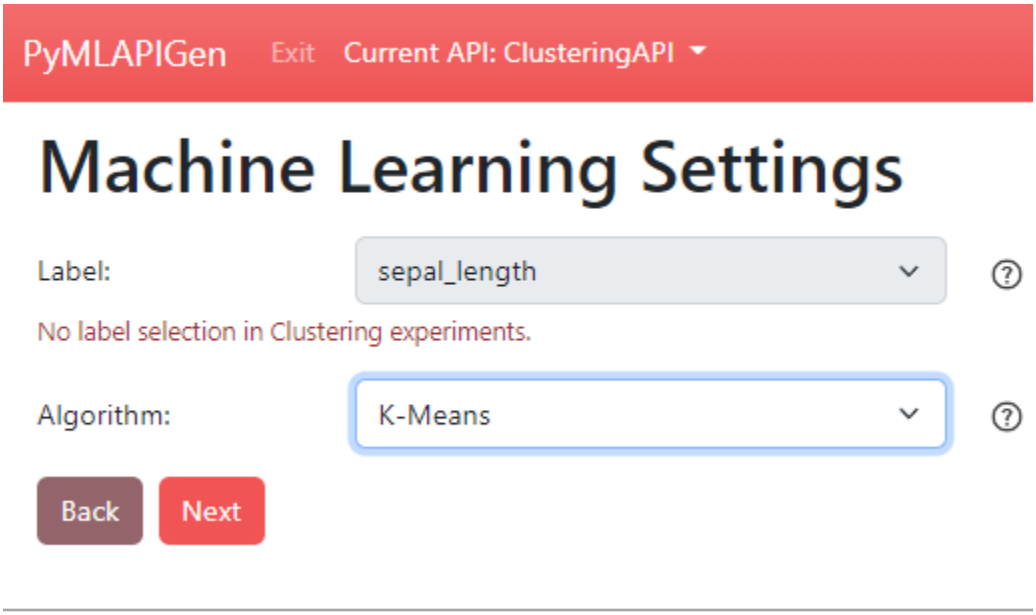


En este paso, se debe escoger **qué hacer** con los **valores que faltan** o **NaN** del dataset.

Como en el dataset de este ejemplo todos los valores **están** y son **válidos**, este paso es irrelevante.

Simplemente presionar el botón **Next**.

Paso 3



En este paso, se debe escoger el **algoritmo** de Machine Learning que llevará a cabo el modelo que se entrenará.

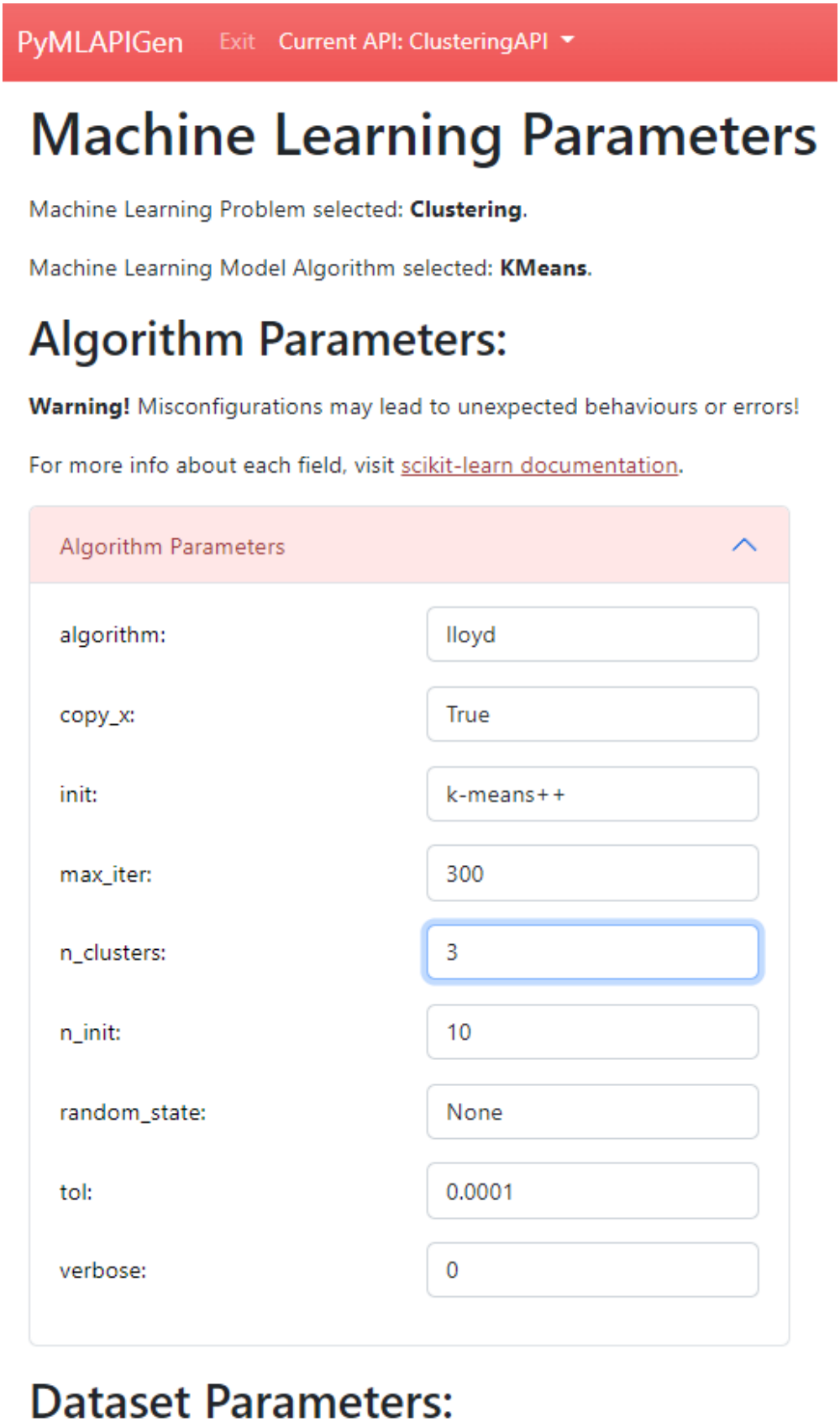
Al tratarse de un experimento de *clustering*, no hay que seleccionar ninguna **etiqueta**.

Para este ejemplo, introduciremos en el formulario los siguientes valores:

Algorithm *Clustering* > *K-Means*

Una vez introducidos los parámetros, presionar el botón **Next**.

Paso 4


PyMLAPIGen
Exit
Current API: ClusteringAPI
Machine Learning Parameters
Machine Learning Problem selected: Clustering.
Machine Learning Model Algorithm selected: KMeans.
Algorithm Parameters:
Warning! Misconfigurations may lead to unexpected behaviours or errors!
For more info about each field, visit scikit-learn documentation.
Algorithm Parameters
algorithm: lloyd
copy_x: True
init: k-means++
max_iter: 300
n_clusters: 3
n_init: 10
random_state: None
tol: 0.0001
verbose: 0
Dataset Parameters:

En este último paso, se deben seleccionar los **parámetros adicionales** del experimento.

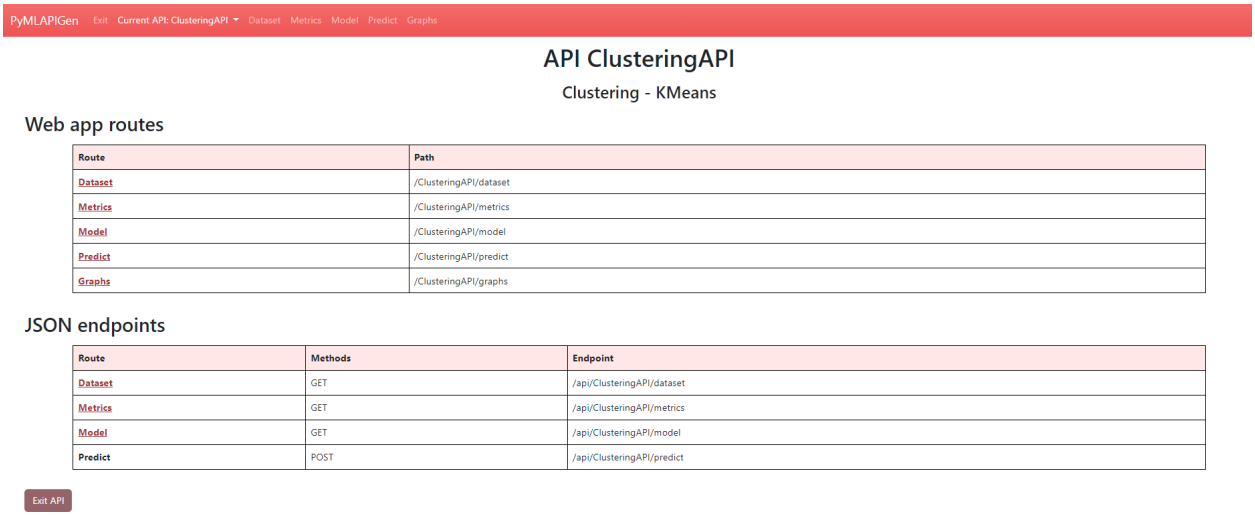
Como se esta utilizando el algoritmo *K-Means*, seleccionaremos el número de conjuntos (*clusters*) que deseamos que se formen. Se eligirá 3 ya que es el número de especies del dataset.

Para ello, desplegar **Algorithm Parameters** e introducir el valor *3* en *n_clusters*.

El resto de opciones se dejarán las **por defecto** y se presiona el botón **Finish** para comenzar la **generación de la API**.

Ventanas API (Herramienta Web)

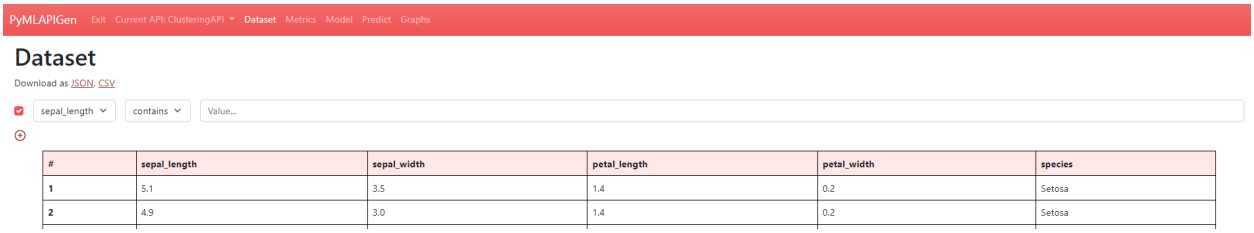
Una vez se genere la API (entrenamiento y evaluación del modelo) se nos redirigirá a la ventana **HOME** de la API (http://localhost:5000/ClusteringAPI).



En esta ventana podrás acceder a las distintas **rutas** y los diferentes **endpoints** de la API.

Además, en la **barra de navegación** podrás navegar entre las distintas ventanas de la API o salir de ella.

Dataset

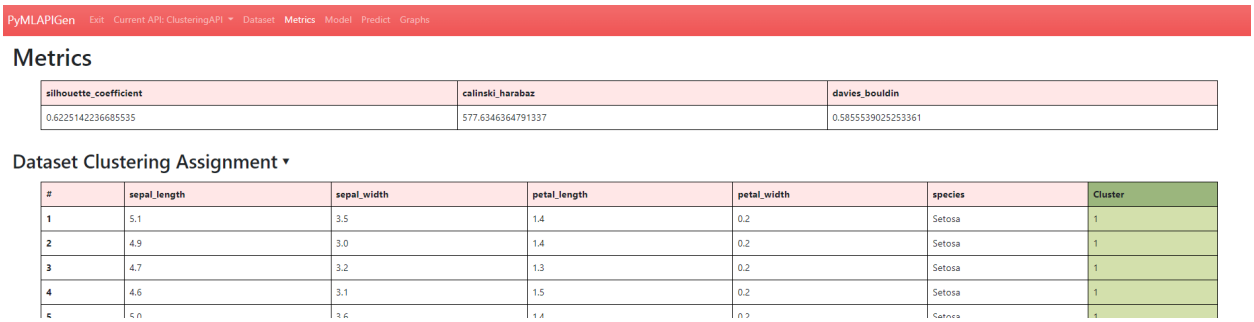


(http://localhost:5000/ClusteringAPI/dataset)

En esta ventana podrás **visualizar** el dataset del experimento de la API generada.

También puedes **ordenar** la tabla pulsando en las cabeceras y aplicar **filtros**.

Métricas

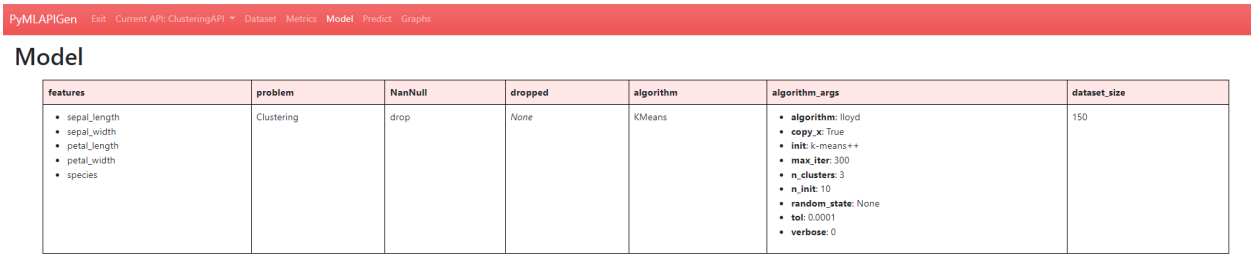


(http://localhost:5000/ClusteringAPI/metrics)

En esta ventana podrás **visualizar** las métricas del experimento de la API generada.

También puedes **desplegar** y **echar un vistazo** al resultado de la **fase de asignación de conjuntos** del experimento. En él, puedes ver el **clúster** al cual se ha **asignado** cada entrada del dataset. Al elegir 3 clústeres en la generación de la API, los tres clústeres generados son 1, 2 y 3.

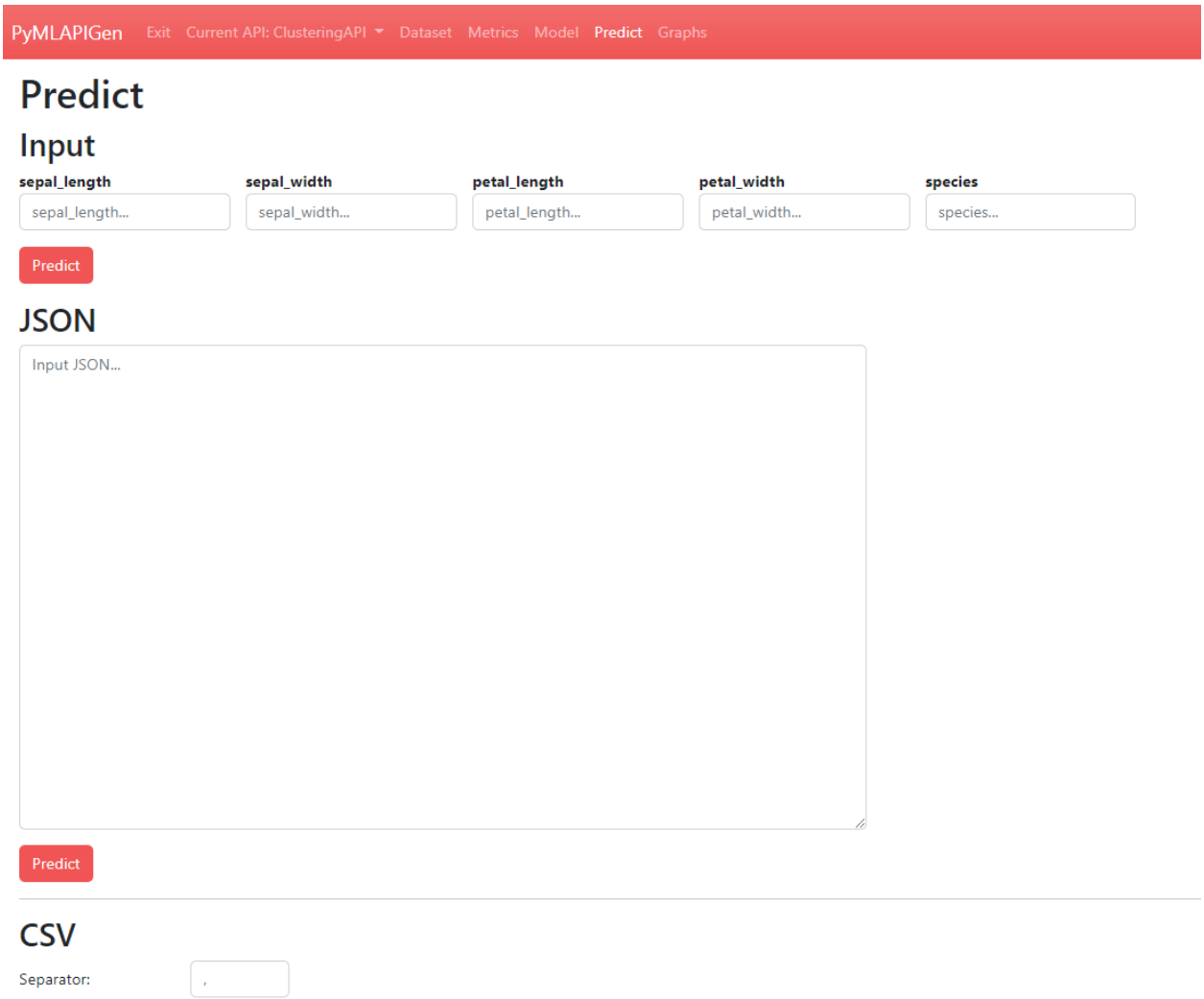
Model



(http://localhost:5000/ClusteringAPI/model)

En esta ventana podrás **visualizar** los parámetros escogidos para el experimento de la API generada.

Predecir



(http://localhost:5000/ClusteringAPI/predict)

En esta ventana podrás realizar **predicciones** utilizando el modelo entrenado del experimento de la API generada.

Puedes realizar las predicciones de diversas formas:

- Introduciendo los datos **manualmente**.
- Utilizando un objeto **JSON** como parámetro de entrada.
- A través de un fichero **CSV** de entrada.

Una vez introducidos los datos y pulsado el botón **Predict**, si los datos son correctos y no hay ningún problema, se nos mostrará abajo el resultado de la predicción.

Por ejemplo, mandar a predecir el siguiente objeto JSON:

```
[{
   "sepal_length": 4.9,
   "sepal_width": 3.0,
   "petal_length": 1.4,
```

(continué en la próxima página)

(proviene de la página anterior)

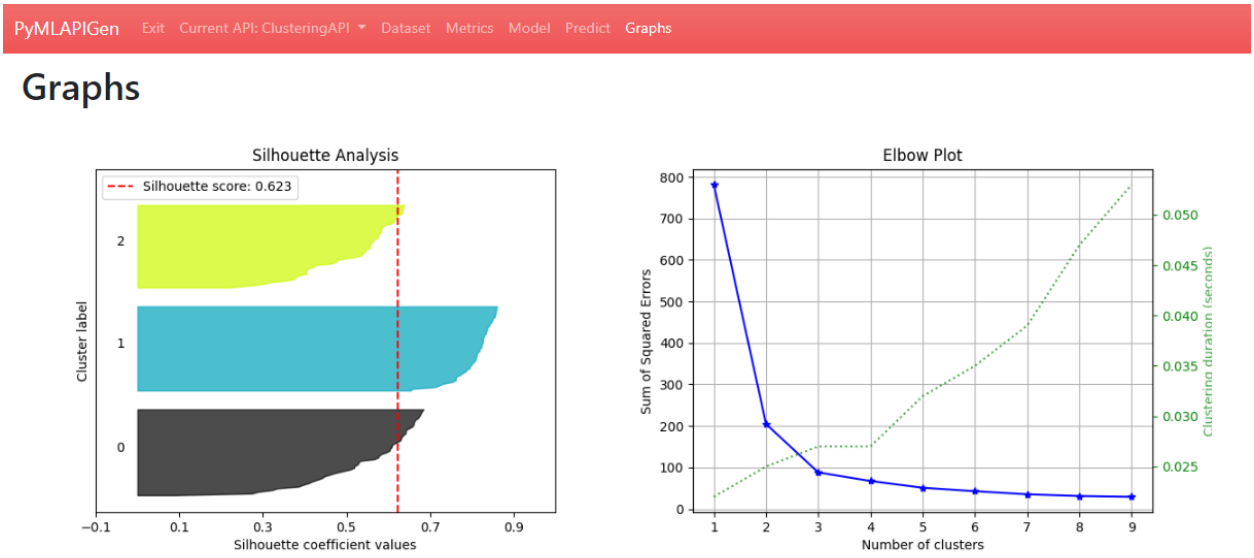
```
    "petal_width": 0.2,
    "species": "Setosa"
}, {
    "sepal_length": 6.3,
    "sepal_width": 2.7,
    "petal_length": 4.9,
    "petal_width": 1.8,
    "species": "Virginica"
}, {
    "sepal_length": 4.8,
    "sepal_width": 3.1,
    "petal_length": 1.6,
    "petal_width": 0.2,
    "species": "Setosa"
}]
```

El resultado es el siguiente:

Prediction Result

#	sepal_length	sepal_width	petal_length	petal_width	species	result
1	4.9	3.0	1.4	0.2	Setosa	1
2	6.3	2.7	4.9	1.8	Virginica	2
3	4.8	3.1	1.6	0.2	Setosa	1

Gráficos



(http://localhost:5000/ClusteringAPI/graphs)

En esta ventana podrás **visualizar** distintos **gráficos** en función del modelo del experimento de la API generada.

Generar API (Petición JSON)

Además de la aplicación web, es posible generar y utilizar la API a partir de **peticiones JSON** a los endpoints de la aplicación.

Petición JSON (/load)

Para generar una API equivalente al generado con la aplicación web, se debe enviar una petición **HTTP POST** al endpoint http://localhost:5000/api/load. El cuerpo de esta petición HTTP POST será este JSON.

Si todo funciona correctamente, se nos debería devolver el siguiente resultado de la operación:


```
{
   "success": "The API has been successfully generated and its now operable.",
   "endpoints": {
      "home": {
         "methods": "GET",
         "endpoint": "/api/ClusteringAPIFromJSON"
      },
      "dataset": {
         "methods": "GET",
         "endpoint": "/api/ClusteringAPIFromJSON/dataset"
      },
      "metrics": {
         "methods": "GET",
         "endpoint": "/api/ClusteringAPIFromJSON/metrics"
      },
      "model": {
         "methods": "GET",
         "endpoint": "/api/ClusteringAPIFromJSON/model"
      },
      "predict": {
         "methods": "POST",
         "endpoint": "/api/ClusteringAPIFromJSON/predict"
      }
   }
}
```


Endpoints API (Peticiones JSON)

GET Dataset

Endpoint: http://localhost:5000/api/ClusteringAPIFromJSON/dataset

En este endpoint podrás **consultar** el **dataset** del experimento de la API generada.

Resultado:


```
[
   {
      "petal_length": 1.4,
      "petal_width": 0.2,
```


(continué en la próxima página)

(proviene de la página anterior)

```
      "sepal_length": 5.1,
      "sepal_width": 3.5,
      "species": "Setosa"
   },
   {
      "petal_length": 1.4,
      "petal_width": 0.2,
      "sepal_length": 4.9,
      "sepal_width": 3.0,
      "species": "Setosa"
   },
   ...
   {
      "petal_length": 5.1,
      "petal_width": 1.8,
      "sepal_length": 5.9,
      "sepal_width": 3.0,
      "species": "Virginica"
   }
]
```

GET Metrics

Endpoint: http://localhost:5000/api/ClusteringAPIFromJSON/metrics

En este endpoint podrás **consultar** la **evaluación** del experimento de la API generada.

Resultado:

```
{
   "silhouette_coefficient": 0.369247326698845,
   "calinski_harabaz": 482.3198481255937,
   "davies_bouldin": 0.9080585128589639
}
```

GET Model

Endpoint: http://localhost:5000/api/ClusteringAPIFromJSON/model

En este endpoint podrás **consultar** los **parámetros del experimento** de la API generada.

Resultado:

```
{
   "features": [
         "petal_length",
         "petal_width",
         "sepal_length",
         "sepal_width",
         "species"
   ],
```

(continué en la próxima página)

(proviene de la página anterior)

```
   "problem": "Clustering",
   "NanNull": "drop",
   "dropped": [],
   "algorithm": "KMeans",
   "algorithm_args": {},
   "dataset_size": 150
}
```

POST Predict

Endpoint: http://localhost:5000/api/ClusteringAPIFromJSON/predict

En este endpoint podrás realizar **predicciones** al experimento de la API generada.

Para ello, en el cuerpo de la petición HTTP POST se introducirá un objeto JSON con los parámetros de entrada.

Por ejemplo se va a mostrar una petición HTTP POST cuyo cuerpo es:

```
[{
   "sepal_length": 4.9,
   "sepal_width": 3.0,
   "petal_length": 1.4,
   "petal_width": 0.2,
   "species": "Setosa"
}, {
   "sepal_length": 6.3,
   "sepal_width": 2.7,
   "petal_length": 4.9,
   "petal_width": 1.8,
   "species": "Virginica"
}, {
   "sepal_length": 4.8,
   "sepal_width": 3.1,
   "petal_length": 1.6,
   "petal_width": 0.2,
   "species": "Setosa"
}]
```

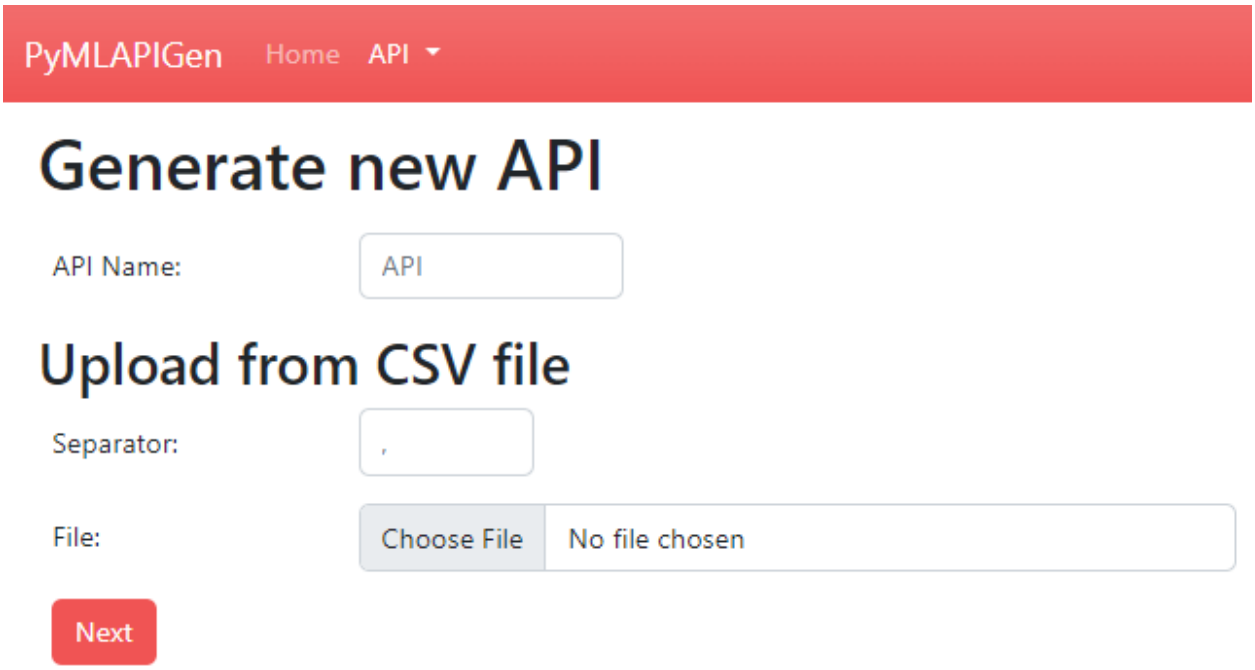
Resultado:

```
[
   1,
   2,
   1
]
```

2.3 Opciones de experimentación

En esta sección se comentará todas las **opciones** de las que se dispone a la hora de **realizar un experimento** de Machine Learning.

2.3.1 Paso 1 - Nombre y dataset CSV



(http://localhost:5000/load/0)

En este paso se debe escoger el **nombre** de la API que se va a generar y el dataset del experimento.

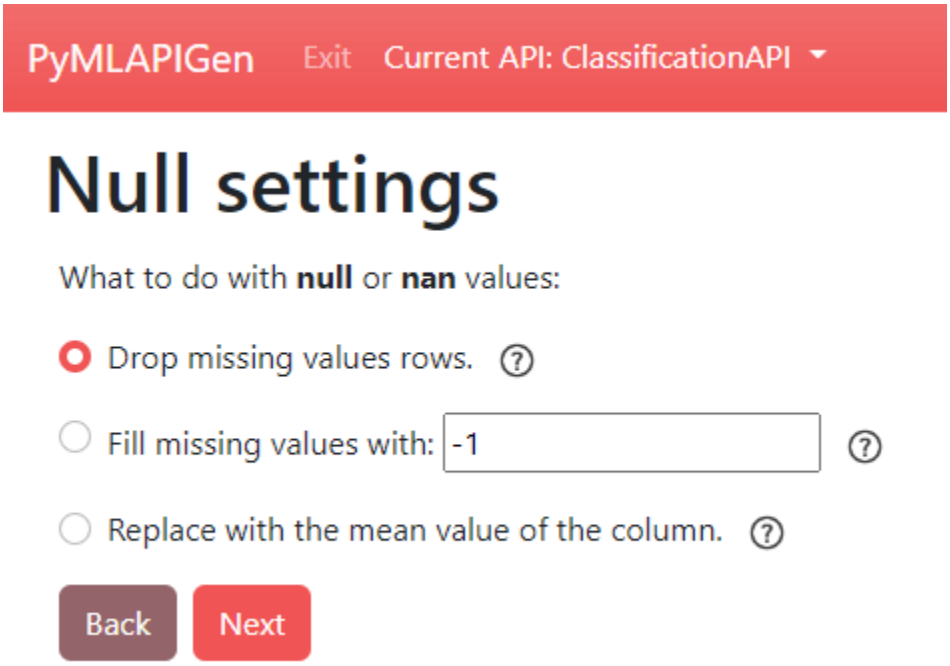
Los campos del formulario de este paso son los siguientes:

API Name **Nombre** de la API a generar. Todas las rutas de la API tendrán el prefijo */nombre/*. El nombre debe ser único y diferente a «api».

Separator **Separador** de valores del fichero CSV. Normalmente suele ser «,» aunque puede variar en función del fichero.

File **Fichero CSV** que contiene el **dataset** del experimento.

2.3.2 Paso 2 - Ajustes de valores nulos



(http://localhost:5000/NOMBREAPI/load/1)

En este paso se debe escoger el **procesado** de los valores **nulos** y **NaN**. Es decir, se decide que hacer con aquellos valores que faltan del dataset.

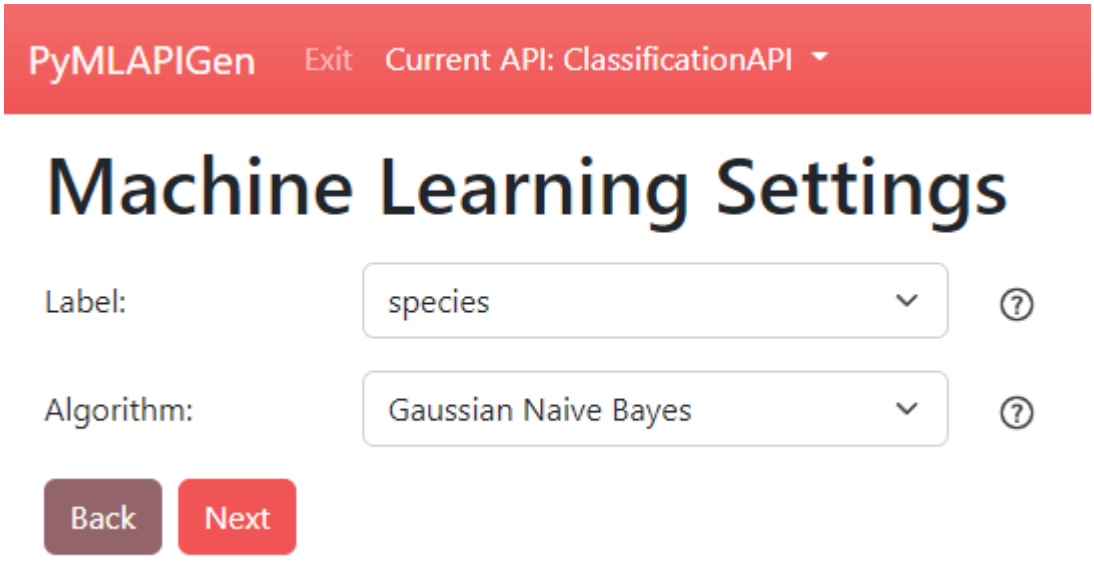
Se debe elegir entre las siguientes **opciones**:

Drop missing values rows Esta opción **desecha** del dataset subido aquellas fillas con valores nulos o NaN.

Fill missing values with <valor> Esta opción **rellena** los valores nulos o NaN con el **valor fijo** introducido.

Replace with the mean value of the column Esta opción **rellena** los valores nulos o NaN con el valor numérico **medio** de la columna o el valor discreto **más frecuente**.

2.3.3 Paso 3 - Ajustes de Machine Learning



(http://localhost:5000/NOMBREAPI/load/2)

En este paso se debe escoger la **etiqueta/columna objetivo** de los experimentos de **clasificación** y **regresión** y el **algoritmo** que ejecutará el modelo.

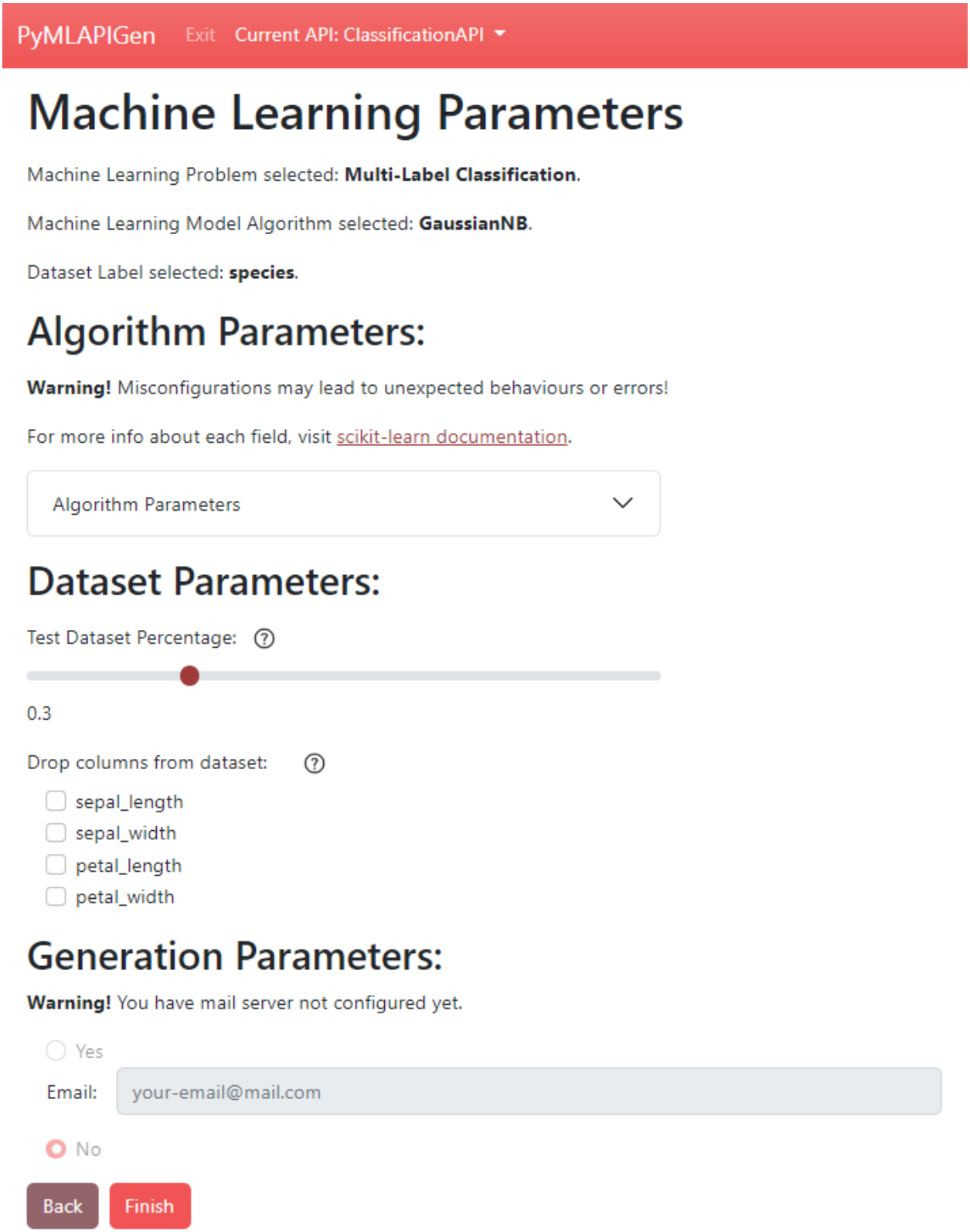
Al seleccionar el algoritmo, también se selecciona el tipo de experimento (**clasificación**, **regresión** o **clustering**).

Los campos del formulario de este paso son los siguientes:

Label **Nombre** de la **columna objetivo/etiqueta**. Disponible para **experimentos supervisados** (clasificación o regresión).

Algorithm **Algoritmo** que ejecutará **el modelo** del experimento. Al elegir un algoritmo se elige también el **tipo de experimento** (**clasificación**, **regresión** o **clustering**).

2.3.4 Paso 4 - Parámetros del experimento



(http://localhost:5000/NOMBREAPI/load/3)

En este paso se debe escoger los diferentes **parámetros adicionales** del experimento y de la API a generar

Los campos del formulario de este paso son los siguientes:

Positive label Valor de etiqueta **positiva**. Utilizado para calcular los **falsos negativos**, **falsos positivos**, **verdaderos negativos**, **verdaderos positivos** durante la fase de test. **(Únicamente disponible para experimentos de clasificación binaria)**

Algorithm Parameters **Parámetros** concretos del algoritmo. Cada algoritmo tiene sus parámetros ajustables propios. Para saber más de cada parámetro visitar la documentación de la librería Scikit-Learn.

Test Dataset Percentage **Porcentaje** del dataset que se dedicará a formar el conjunto de entrenamiento para la fase de **entrenamiento**. Por defecto a 0.3. **(Únicamente disponible para experimentos de clasificación y regresión)**

Drop columns from dataset **Atributos** del dataset que no se desean conservar. Aquellas columnas seleccionadas serán desechadas del dataset.

Email Si se ha configurado el *Servidor de correos elecrónicos (SMTP)*, en esta opción podrás seleccionar si recibir o no un **correo electrónico** una vez **finalice** la **generación de la API**.

2.3.5 Endpoint JSON POST

Es posible generar una API desde una petición **HTTP POST** al endpoint (**/api/load**).

El cuerpo de la petición POST será un **objeto JSON**. Los **atributos** de este objeto serán:

apiName **Nombre** de la API a generar. Todas las rutas de la API tendrán el prefijo */nombre/*. El nombre debe ser único y diferente a «api». **Obligatorio**.

dataset **Dataset** del experimento. Consiste en un array de JSON cuyos elementos son objetos JSON con los atributos. **Obligatorio**.

modelType **Algoritmo** del modelo experimento. En función del algoritmo se elige también el tipo de experimento (**clasificación**, **regresión** o **clustering**).. **Obligatorio**. Los algoritmos disponibles son:

- **Clasificación**
  - **GNB** - *Gaussian Naive Bayes*
  - **SVC** - *Support Vector Machine*
  - **KNN** - *K-Neighbors Classifier*
  - **DT** - *Decision Tree Classifier*
  - **RF** - *Random Forest Classifier*
- **Regresión**
  - **LR** - *Linear Regression*
  - **SVR** - *Support Vector Machine*
  - **SGDR** - *SGD Regressor*
  - **KR** - *Kernel Ridge*
  - **GBR** - *Gradient Boosting Regressor*
- **Clustering**
  - **KM** - *K-Means*

- **AP** - *Affinity Propagation*
- **MS** - *Mean Shift*
- **MKM** - *Mini-Batch K-Means*

nanNullMode **Modo de procesado** de los valores nulos. **Optativo**. Las opciones son:

- **drop** - Deshechar filas con valores nulos. **(Opción por defecto si no se especifica)**
- **fill** - Rellenar con un valor fijo. Si elige esta opción, se **debe proveer** el atributo **fillvalue**.
- **mean** - Rellenar con la media o moda.

fillvalue **Valor fijo** a rellenar. **Obligatorio si nanNullMode : «fill»**.

modelParams **Objeto JSON** con los **parámetros concretos** del algoritmo (visitar la documentación de la librería Scikit-Learn). **Optativo**.

dropColumns **Array JSON** con los **Atributos** del dataset que no se desean conservar. Aquellas columnas seleccionadas serán desechadas del dataset. **Optativo**.

testSize **Porcentaje** del dataset que se dedicará a formar el conjunto de entrenamiento para la fase de **entrenamiento**. Por defecto a 0.3. **(Únicamente disponible y optativo para experimentos de clasificación y regresión)**

Positive label Valor de etiqueta **positiva**. Utilizado para calcular los **falsos negativos**, **falsos positivos**, **verdaderos negativos**, **verdaderos positivos** durante la fase de test. **(Únicamente disponible y obligatorio para experimentos de clasificación binaria)**

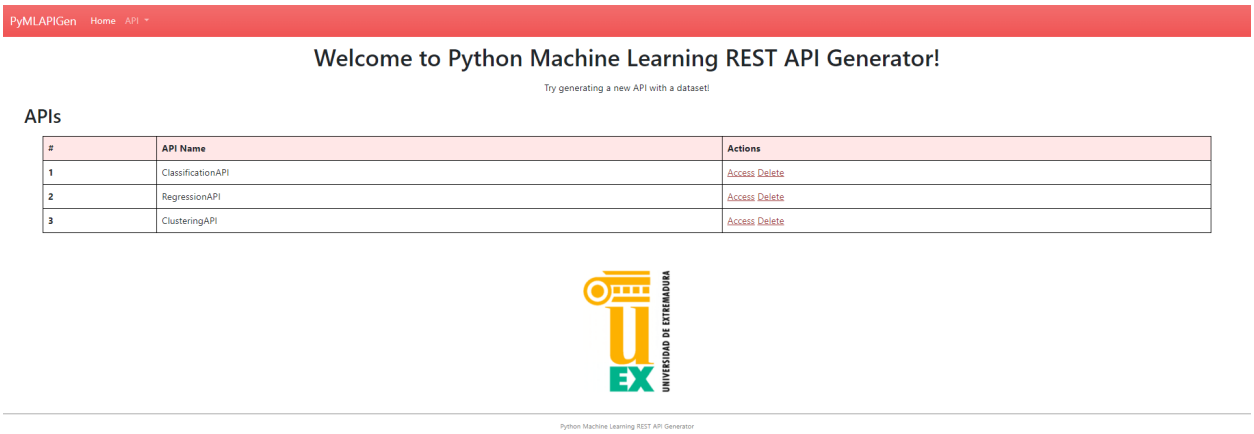
email Si se ha configurado el *Servidor de correos elecrónicos (SMTP)* y se desea recibir un correo electrónico una vez finalice la generación de la API, especifica el **email** que **recibirá el correo**. **Optativo**

Una vez se realice la petición HTTP POST, la aplicación retornará un mensaje de **éxito** (junto a los endpoints generados) o **error** en función del resultado de la operación.

2.4 Ventanas

En esta sección se comentarán las distintas **ventanas** que dispone la aplicación web.

2.4.1 Home



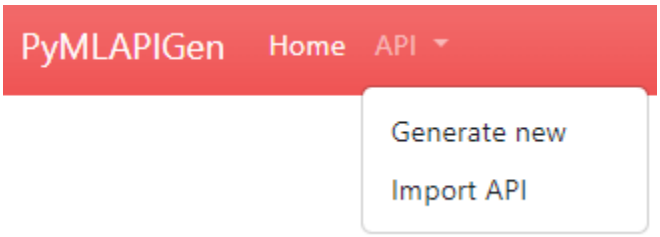
(http://localhost:5000/)

Esta es la **ventana principal** de la aplicación.

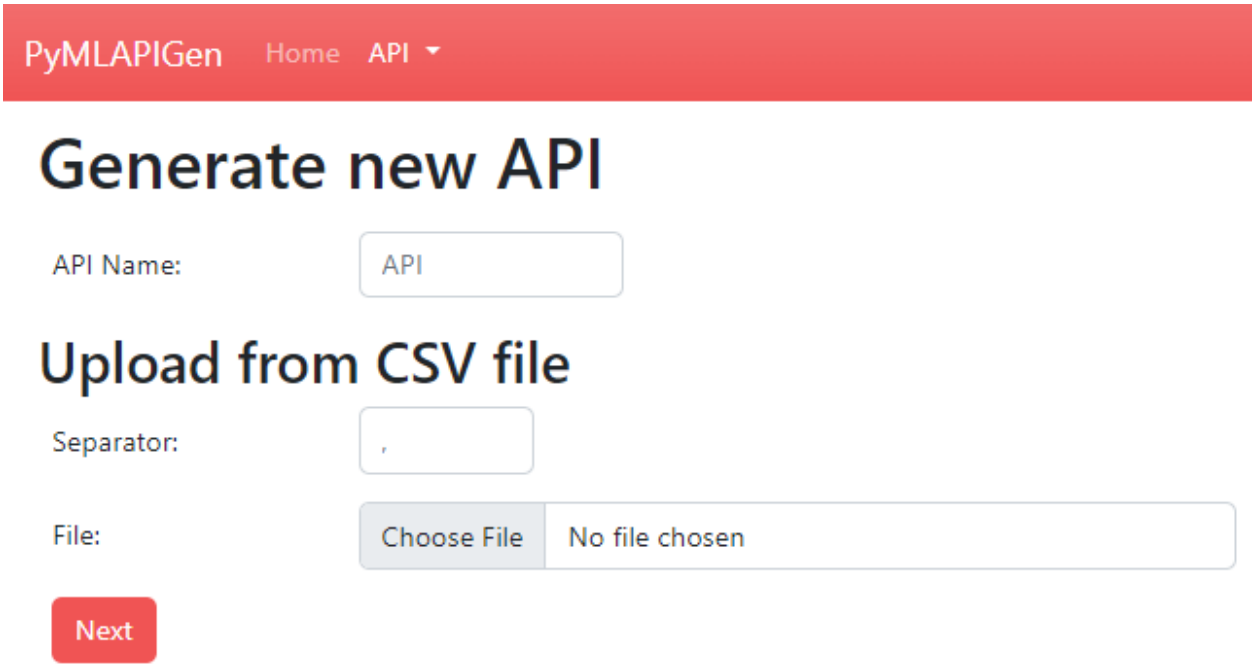
Dentro de ella podrás **consultar**, **acceder** y **eliminar** todas las APIs generadas e importadas.

Además, se dispone de una **barra de navegación** a partir de la cual se puede acceder a:

- **Generar** una *nueva API*.
- **Importar** una *API ya generada*.



2.4.2 Generación de API

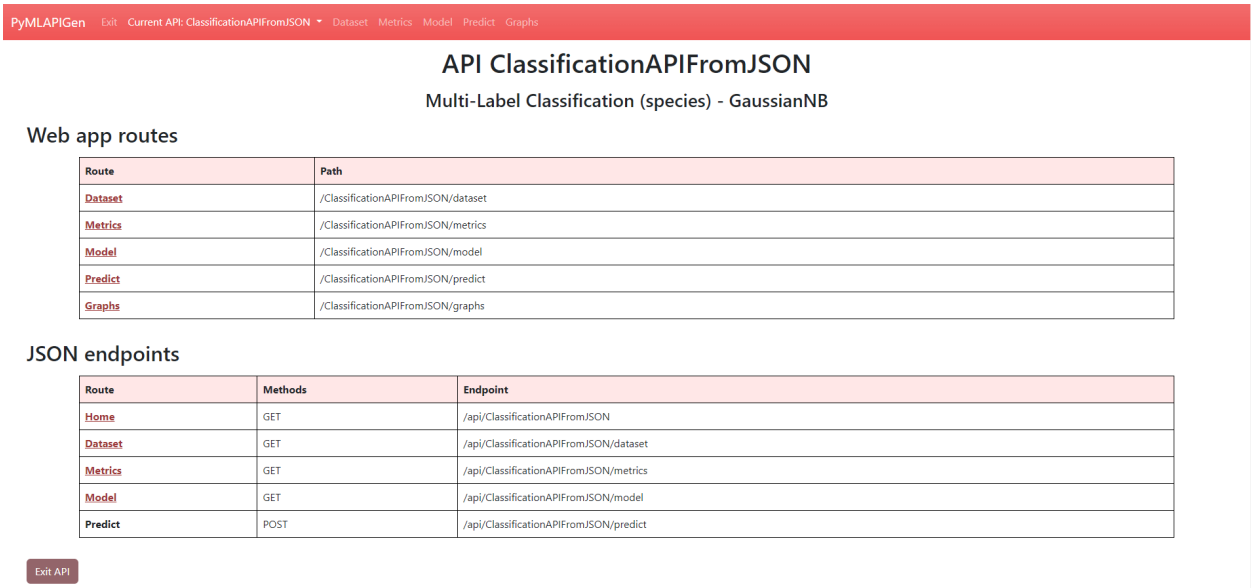


(http://localhost:5000/load/0)

En esta ventana podrás **generar una nueva API** a través de un **experimento Machine Learning**.

Para más información acerca de las opciones de generación de APIs, consulta la sección *Opciones de experimentación*.

2.4.3 API Home



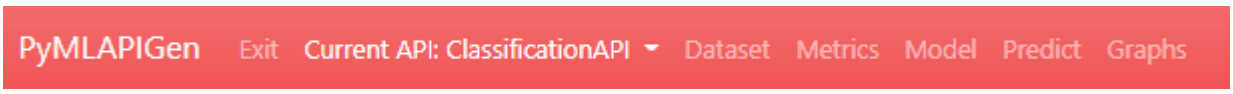
(http://localhost:5000/NOMBREAPI)

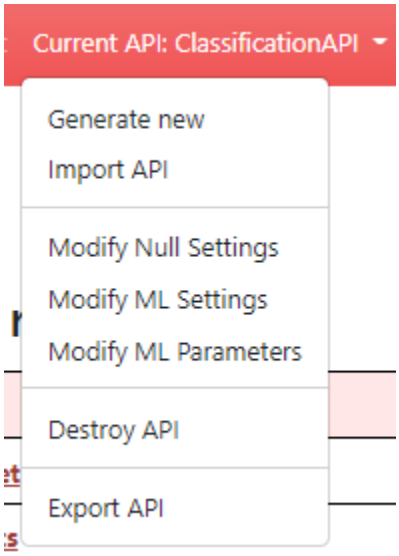
Esta es la **ventana principal** de la una API.

Dentro de ella podrás **consultar** y **acceder** a las distintas ventanas y endpoints de la **API generada**.

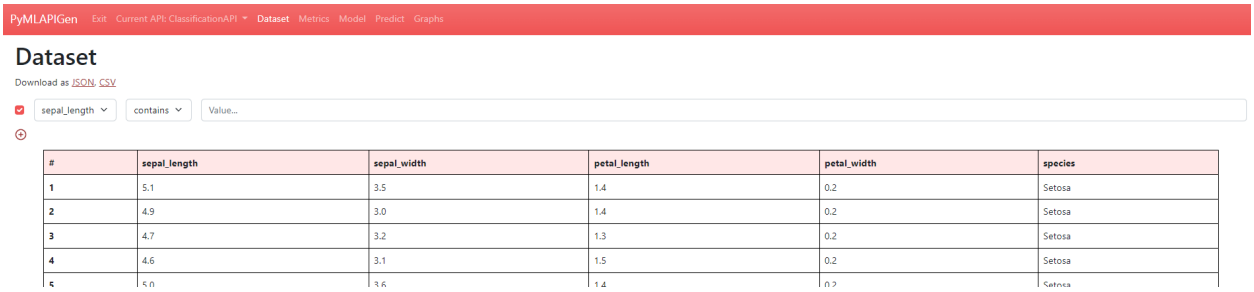
Además, se dispone de una **barra de navegación** a partir de la cual se puede acceder a:

- *Home principal <#home>* de la aplicación.
- **Menú desplegable:**
    - **Generar** una *nueva API*.
    - **Importar** una *API ya generada*.
    - **Modificar** la *API*.
    - **Eliminar** la API.
    - **Exportar** la API generada.
- *Dataset del experimento*.
- *Métricas del experimento*.
- *Modelo de la experimento*.
- *Predicciones*.
- *Gráficos de la experimento*.





2.4.4 Dataset



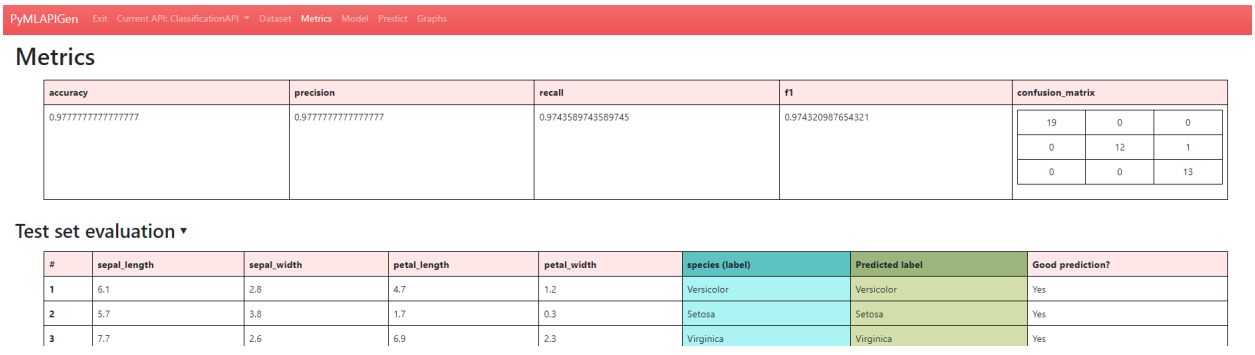
#	sepal_length	sepal_width	petal_length	petal_width	species
1	5.1	3.5	1.4	0.2	Setosa
2	4.9	3.0	1.4	0.2	Setosa
3	4.7	3.2	1.3	0.2	Setosa
4	4.6	3.1	1.5	0.2	Setosa
5	5.0	3.6	1.4	0.2	Setosa

(http://localhost:5000/NOMBREAPI/dataset)

En esta ventana podrás **visualizar** el dataset del experimento de la API generada. Además, puedes descargar el dataset como **JSON** O **CSV**.

También puedes **ordenar** la tabla pulsando en las cabeceras y aplicar **filtros**. Puedes añadir y quitar filtros pulsando los botones «+» y «-«.

2.4.5 Métricas



accuracy	precision	recall	f1	confusion_matrix
0.9777777777777777	0.9777777777777777	0.9743589743589745	0.974320987654321	19 0 0 0 12 1 0 0 13

Test set evaluation

#	sepal_length	sepal_width	petal_length	petal_width	species (label)	Predicted label	Good prediction?
1	6.1	2.8	4.7	1.2	Versicolor	Versicolor	Yes
2	5.7	3.8	1.7	0.3	Setosa	Setosa	Yes
3	7.7	2.6	6.9	2.3	Virginica	Virginica	Yes

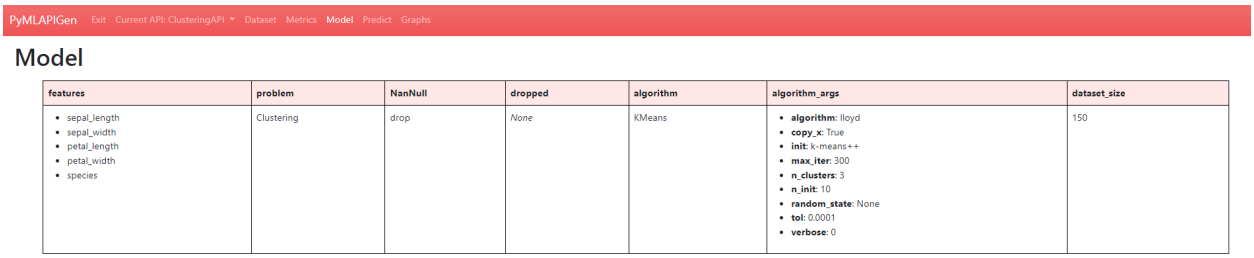
(http://localhost:5000/NOMBREAPI/metrics)

En esta ventana podrás **visualizar** las métricas del experimento de la API generada.

En experimentos de **clasificación** y **regresión**, puedes **desplegar** y **echar un vistazo** al resultado de la **fase de evaluación** del experimento. En él, puedes ver el **valor original** de la etiqueta/clase, el **valor predicho** y si es **correcto o no** (clasificación) o el **error** (regresión).

En experimentos de **clustering**, en cambio, puedes visualizar el resultado de la **fase de asignación de conjuntos** del experimento. Podrás ver el **clúster** al cual se ha **asignado** cada entrada del dataset.

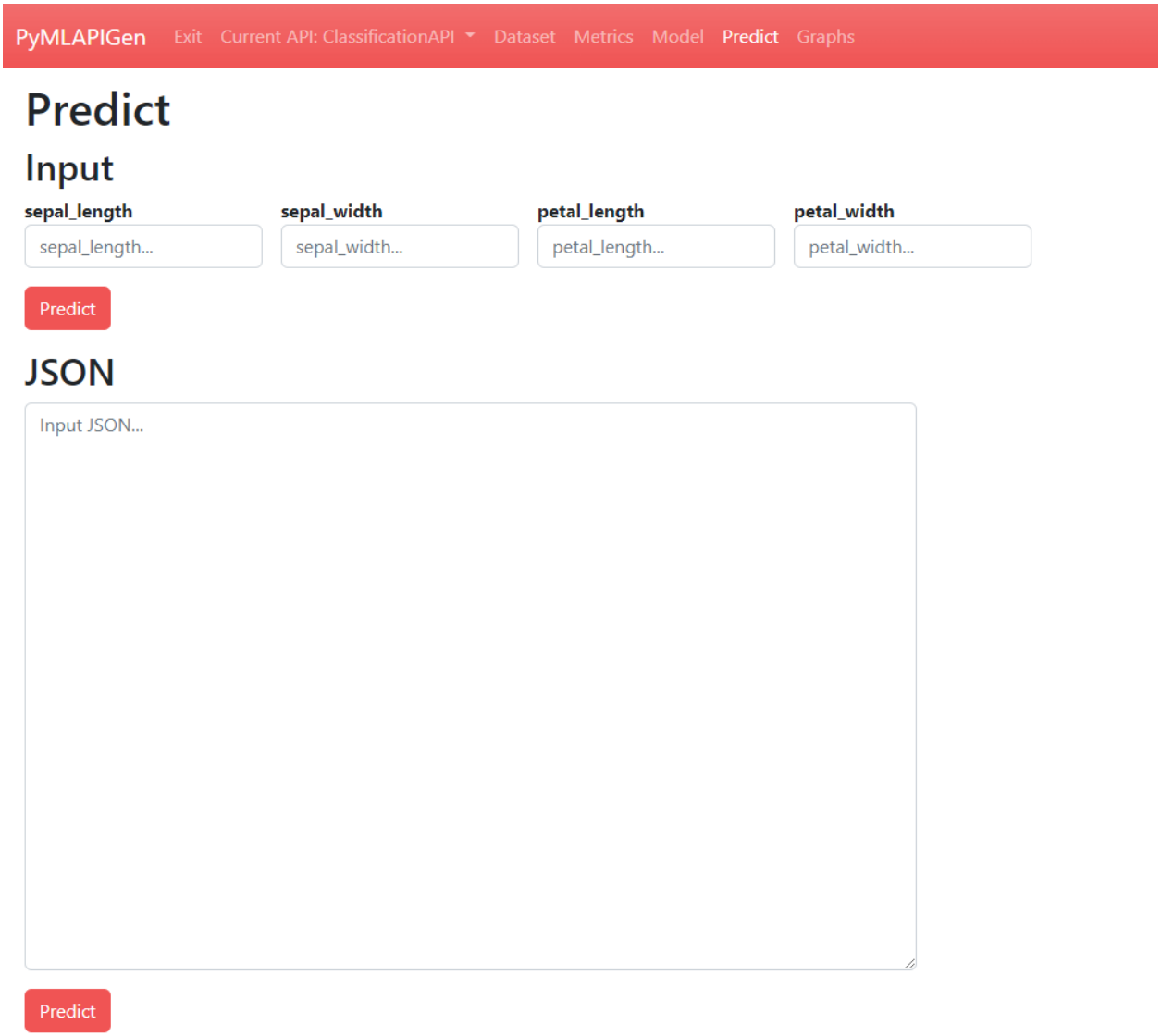
2.4.6 Modelo



(http://localhost:5000/NOMBREAPI/model)

En esta ventana podrás **visualizar** los parámetros escogidos para el experimento de la API generada.

2.4.7 Predecir



(http://localhost:5000/NOMBREAPI/predict)

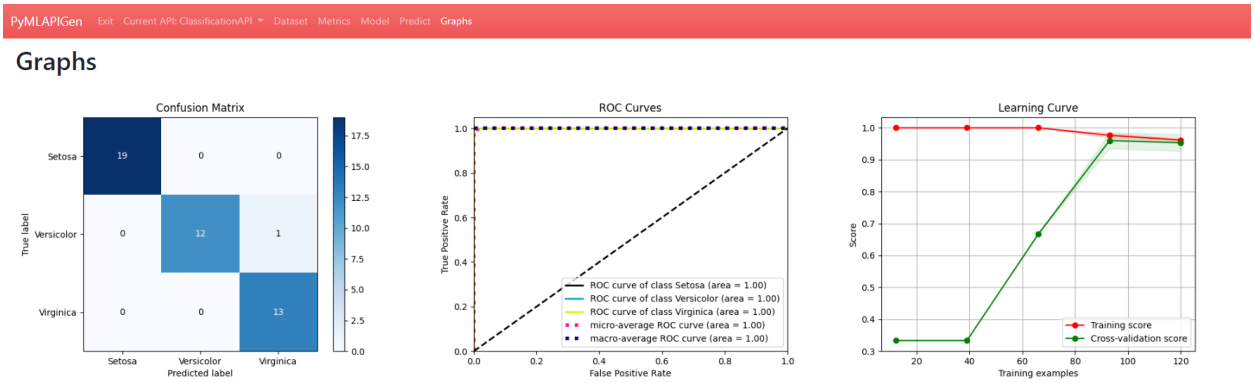
En esta ventana podrás realizar **predicciones** utilizando el modelo entrenado del experimento de la API generada.

Puedes realizar las predicciones de diversas formas:

- Introduciendo los datos **manualmente**.
- Utilizando un objeto **JSON** como parámetro de entrada.
- A través de un fichero **CSV** de entrada.

Una vez introducidos los datos y pulsado el botón **Predict**, si los datos son correctos y no hay ningún problema, se nos mostrará abajo el resultado de la predicción.

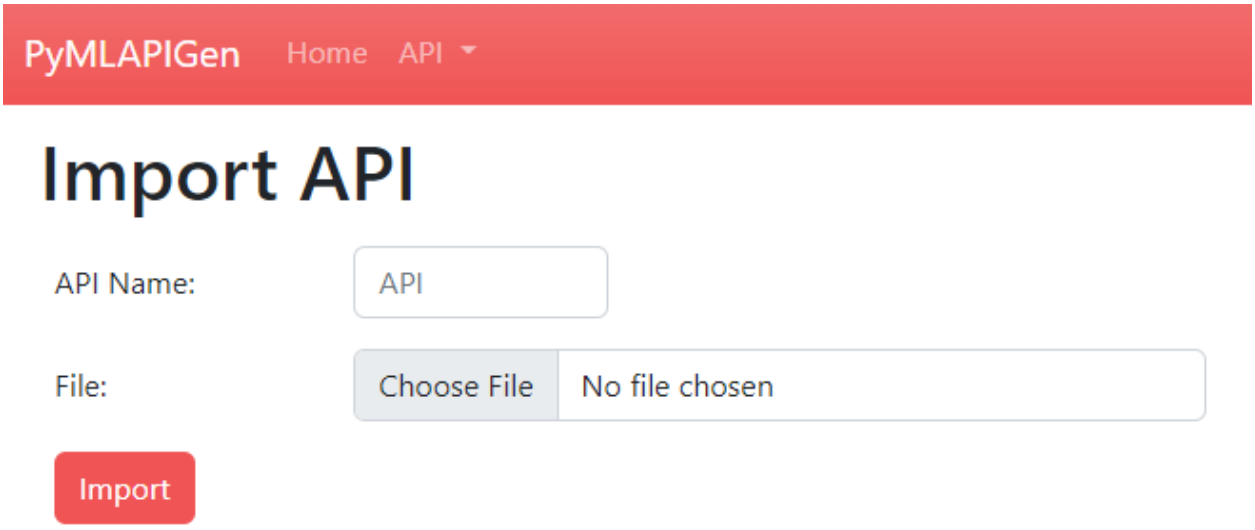
2.4.8 Gráficos



(http://localhost:5000/ClassificationAPI/graphs)

En esta ventana podrás **visualizar** distintos **gráficos** en función del modelo del experimento de la API generada.

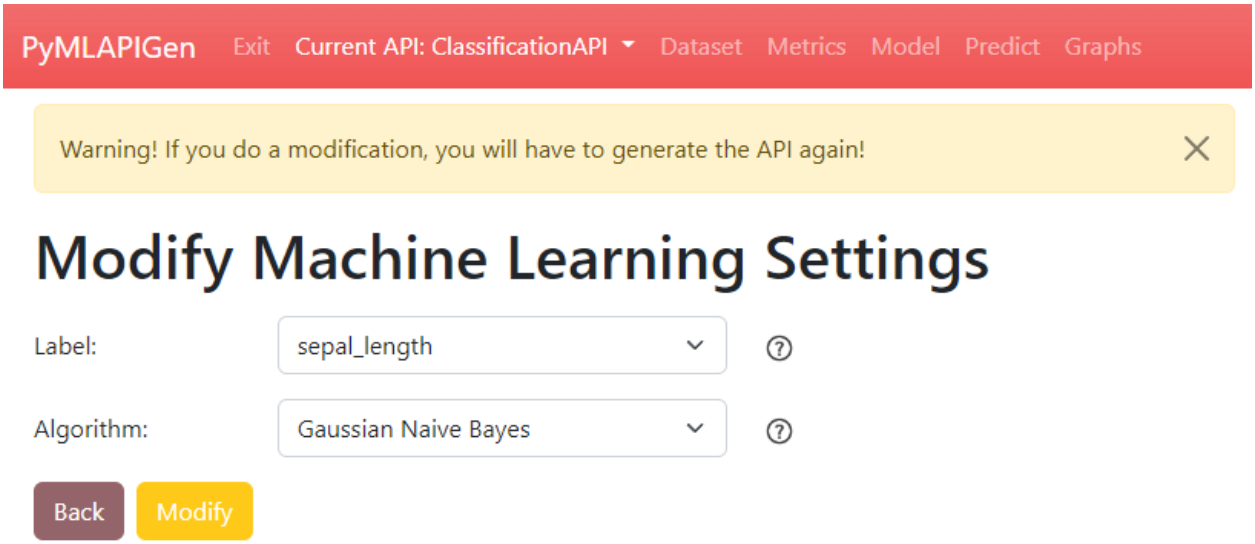
2.4.9 Importar



(http://localhost:5000/import)

En esta ventana podrás **importar una API** generada y exportada (extensión **.api**) previamente **sin necesidad de entrenar y evaluar el modelo de nuevo**.

2.4.10 Modificar API



(http://localhost:5000/load/0)

En esta ventana podrás **modificar una API generada**.

Si realizas una modificación, tendrás que realizar de vuelta los **pasos posteriores** de la generación y volver a **generar la API** (incluyendo entrenamiento y evaluación).

Para más información acerca de las opciones de generación de APIs, consulta la sección *Opciones de experimentación*.

2.5 Endpoints

En esta sección se comentarán los distintos endpoints para peticiones HTTP (GET y POST) de la aplicación.

2.5.1 Home (GET)

http://localhost:5000/api/

Este endpoint comprueba que la aplicación esté en **ejecución** y devuelve **todas las APIs** generadas y accesibles en memoria.

```
{
    "status": "Api is working",
    "generatedApis": {
        "ClassificationAPIFromJSON": "/api/ClassificationAPIFromJSON",
        "RegressionAPIFromJSON": "/api/RegressionAPIFromJSON"
        "ClusteringAPIFromJSON": "/api/ClusteringAPIFromJSON",
    }
}
```

2.5.2 Load (POST)

http://localhost:5000/api/load

Este endpoint **genera una nueva API**. Para ello, utiliza el cuerpo provisto en la **petición HTTP POST**.

Para más información acerca de esta petición, consultar la sección *Endpoint JSON POST* de *Opciones de experimentación*.

2.5.3 API Home (GET)

http://localhost:5000/api/NOMBREAPI

Este endpoint devuelve los parámetros principales de la API junto a sus endpoints.


```
{
    "apiName": "ClassificationAPIFromJSON",
    "mlProblem": "Multi-Label Classification",
    "modelAlgorithm": "GaussianNB",
    "label": "species",
    "endpoints": {
        "home": {
            "methods": "GET",
            "endpoint": "/api/ClassificationAPIFromJSON"
        },
        "dataset": {
            "methods": "GET",
            "endpoint": "/api/ClassificationAPIFromJSON/dataset"
        },
        "metrics": {
            "methods": "GET",
            "endpoint": "/api/ClassificationAPIFromJSON/metrics"
        },
        "model": {
            "methods": "GET",
            "endpoint": "/api/ClassificationAPIFromJSON/model"
        },
        "predict": {
            "methods": "POST",
            "endpoint": "/api/ClassificationAPIFromJSON/predict"
        }
    }
}
```

2.5.4 API Dataset (GET)

http://localhost:5000/api/NOMBREAPI/dataset

Este endpoint devuelve el **dataset**. Además, es posible especificar **filtros** como parámetros en la URL de la forma «atributo=valor».

Un ejemplo de filtro puede ser /api/ClassificationAPIFromJSON/dataset?species=Setosa. En este ejemplo, el filtro **?species=Setosa** implica que únicamente se devuelva aquellas filas cuya especie sea Setosa.


```
[
    {
        "petal_length": 1.4,
        "petal_width": 0.2,
        "sepal_length": 5.1,
        "sepal_width": 3.5,
        "species": "Setosa"
    },
    {
        "petal_length": 1.4,
        "petal_width": 0.2,
        "sepal_length": 4.9,
        "sepal_width": 3.0,
        "species": "Setosa"
    },
    ...
]
```


2.5.5 API Metrics (GET)

http://localhost:5000/api/NOMBREAPI/metrics

Este endpoint **retorna** las métricas del experimento de la API generada.


```
{
    "accuracy": 0.9777777777777777,
    "precision": 0.9777777777777777,
    "recall": 0.9743589743589745,
    "f1": 0.974320987654321,
    "confusion_matrix": [
        [
            19,
            0,
            0
        ],
        [
            0,
            12,
            1
        ],
        [
            0,
            0,
            13
```


(continué en la próxima página)

(proviene de la página anterior)

```
        ]
    ]
}
```

2.5.6 API Model (GET)

http://localhost:5000/api/NOMBREAPI/model

Este endpoint **retorna** los parámetros escogidos para el experimento de la API generada.

```
{
    "label": "species",
    "features": [
        "petal_length",
        "petal_width",
        "sepal_length",
        "sepal_width"
    ],
    "problem": "Classification",
    "classification": "Multi-Label",
    "labels": [
        "Setosa",
        "Versicolor",
        "Virginica"
    ],
    "NanNull": "drop",
    "dropped": [],
    "algorithm": "GaussianNB",
    "algorithm_args": {},
    "dataset_size": 150,
    "training_size": 105,
    "testing_size": 45
}
```

2.5.7 API Predict (POST)

http://localhost:5000/api/NOMBREAPI/predict

Este endpoint permite **realizar** predicciones utilizando nuevos datos. Estos nuevos deben estar incluidos en el **cuerpo de la petición POST**.

Por ejemplo, un cuerpo podría ser:

```
 [{
   "sepal_length": 4.9,
   "sepal_width": 3.0,
   "petal_length": 1.4,
   "petal_width": 0.2
}, {
   "sepal_length": 6.3,
   "sepal_width": 2.7,
```

(continué en la próxima página)

(proviene de la página anterior)

```
   "petal_length": 4.9,
   "petal_width": 1.8
}, {
   "sepal_length": 4.8,
   "sepal_width": 3.1,
   "petal_length": 1.6,
   "petal_width": 0.2
}]
```

El resultado de la petición es:

- El **valor predicho** para cada entrada en experimentos de **clasificación** y **regresión**.
- El **clúster** al que pertenece cada entrada en experimentos de **clustering**.

```
[
 "Setosa",
 "Virginica",
 "Setosa"
]
```

2.6 Opciones adicionales

En esta sección te mostraremos las opciones adicionales con las que cuenta la aplicación web.

2.6.1 IP y puerto para la aplicación

Es posible elegir la **ip** y el **puerto** en el que se va a alojar la aplicación.

Por defecto, al ejecutar la aplicación, esta se despliega en el ip **localhost** y en el puerto **5000** (http://localhost:5000/). Estos son los parámetros por defecto del **framework** Flask.

Sin embargo, si deseamos alojar la aplicación en otra dirección o puerto, a la hora de ejecutar la aplicación es posible incluir dos **parámetros de entrada opcionales**:

- La **ip** en la que se desea alojar la aplicación. (Por defecto a localhost / 127.0.0.1)
- El **puerto** en el que localizar la aplicación (Por defecto a 5000).

De esta forma, lanzar la aplicación de la siguiente forma:

- **Si se instaló a través de pip:**
    - `pymlapigen <IP> <PUERTO>`
    - **Ejemplos:**
        - `pymlapigen 0.0.0.0`
        - `pymlapigen localhost 80`
- **Si se clonó el código fuente:**
    - `python run.py <IP> <PUERTO>`
    - **Ejemplos:**
        - `python run.py 0.0.0.0`

- python run.py localhost 80

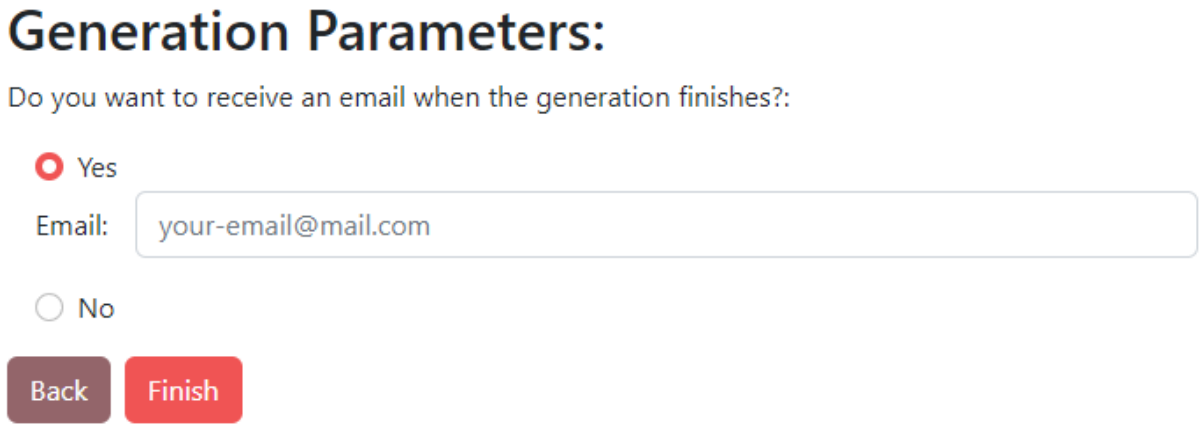
```
pymlapigen localhost 80
  * Serving Flask app 'pymlapigen' (lazy loading)
  * Environment: production
  * Debug mode: off
  * Running on http://localhost:80/ (Press CTRL+C to quit)
```

2.6.2 Servidor de correos elecrónicos (SMTP)

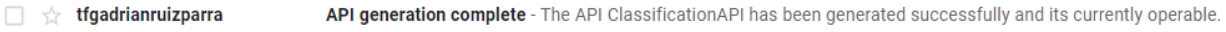
La aplicación ofrece **notificaciones** a través de **correo electrónico** para cuando las APIs completan su generación.

Esto es útil para aquellos experimentos en el que los **costes de cómputo** de generación sean **altos** (ya sea por datasets grandes o por algoritmos complicados).

De esta forma, si el usuario **marca** recibir un correo electrónico introduciendo su correo, este será enviado y recibido cuando **finalice la generación** de la API.



El email recibido tiene el siguiente aspecto:



IMPORTANTE: Para poder recibir correos es **necesario** configurar el **servidor de correos electrónicos (SMTP)**.

Para ello, acceder al fichero de configuración del código fuente *config.py*. En él, debéis **descomentar** e **introducir** los siguientes parámetros:

MAIL_SERVER Servidor de correo electrónico. (*String*)

MAIL_PORT Puerto de correo electrónico. (*Integer*)

MAIL_USERNAME Correo electrónico que enviará los correos. (*String*)

MAIL_PASSWORD Contraseña de aplicación del correo electrónico. (*String*)

MAIL_USE_TLS Utilizar TLS en los correos electrónicos. (*Boolean*)

MAIL_USE_SSL Utilizar SSL en los correos electrónicos. (*Boolean*)

Una vez introducidos estos valores y guardados, al volver a ejecutar la aplicación, la sección de correos electrónicos durante el último paso de la generación deberá aparecer **habilitado**.

2.6.3 Docker

La aplicación **es compatible** para ser ejecutada dentro de contenedores con **Docker**.

Para ello:

1. **Clonar el repositorio git** con el comando `git clone https://github.com/adruizp/PYMLAPIGEN`.

```
git clone https://github.com/adruizp/PYMLAPIGEN
  Cloning into 'PYMLAPIGEN'...
  remote: Enumerating objects: 197, done.
  remote: Counting objects: 100% (197/197), done.
  remote: Compressing objects: 100% (131/131), done.
  Receiving objects:  73% (144/197)), reused 145 (delta 61), pack-reused 0
  Receiving objects: 100% (197/197), 250.51 KiB | 1.25 MiB/s, done.
  Resolving deltas: 100% (104/104), done.
```

2. Una vez clonado, **acceder** al directorio del proyecto con el comando `cd PYMLAPIGEN`.
3. Dentro del directorio, crearemos la **imagen Docker** con el comando `docker build --tag pymlapigen-docker .`
   - Puedes consultar la lista de imagenes con `docker image ls`.

```
docker build --tag pymlapigen-docker .
    [+] Building 71.9s (15/15) FINISHED
    => [internal] load build definition from Dockerfile                                 ␣
↪                                0.8s
    => => transferring dockerfile: 215B                                                 ␣
↪                                0.0s
    => [internal] load .dockerignore                                                    ␣
↪                                0.8s
    => => transferring context: 2B                                                      ␣
↪                                0.0s
    => resolve image config for docker.io/docker/dockerfile:1                           ␣
↪                                2.0s
    => [auth] docker/dockerfile:pull token for registry-1.docker.io                     ␣
↪                                0.0s
    => CACHED docker-image://docker.io/docker/
↪dockerfile:1@sha256:443aab4ca21183e069e7d8b2dc68006594f40bddf1b15bbd83  0.0s
    => [internal] load .dockerignore                                                    ␣
↪                                0.0s
    => [internal] load build definition from Dockerfile                                 ␣
↪                                0.0s
    => [internal] load metadata for docker.io/library/python:3.10.4-slim-buster        ␣
↪                                1.8s
    => [auth] library/python:pull token for registry-1.docker.io                        ␣
↪                                0.0s
    => [1/4] FROM docker.io/library/python:3.10.4-slim-
↪buster@sha256:b59fd1a008ab77d90dca9dffeab9fdcc55475ee78ffa5de  1.7s
    => => resolve docker.io/library/python:3.10.4-slim-
↪buster@sha256:b59fd1a008ab77d90dca9dffeab9fdcc55475ee78ffa5de  0.0s
    => => sha256:b008ef62b91dcc0e4f5f9f6a45b56e4305429afbcfc07f9584b1d0e34776f9e0 3.16MB␣
↪/ 3.16MB                        1.2s
    => => sha256:b59fd1a008ab77d90dca9dffeab9fdcc55475ee78ffa5de3a0097d33936583da 1.86kB␣
↪/ 1.86kB                        0.0s
```

(continué en la próxima página)

(proviene de la página anterior)

```
    => => sha256:5386a4608a2c9339e32683254c9ddff2bccc0f826a96b22deece63b83e9c0384 1.37kB␣
↪/ 1.37kB                        0.0s
    => => sha256:e00cda196d23f412ddd977c38c47b39aafbe156d8fb13d063b7af535ace6678e 7.50kB␣
↪/ 7.50kB                        0.0s
    => => extracting␣
↪sha256:b008ef62b91dcc0e4f5f9f6a45b56e4305429afbcfc07f9584b1d0e34776f9e0                ␣
↪          0.3s
    => [internal] load build context                                                    ␣
↪                                0.3s
    => => transferring context: 3.26MB                                                  ␣
↪                                0.2s
    => [2/4] WORKDIR /pymlapigen                                                        ␣
↪                                0.2s
    => [3/4] ADD . /pymlapigen                                                          ␣
↪                                0.1s
    => [4/4] RUN pip install -r requirements.txt                                        ␣
↪                               61.2s
    => exporting to image                                                               ␣
↪                                3.5s
    => => exporting layers                                                              ␣
↪                                3.4s
    => => writing image␣
↪sha256:c42e13c735036b785a3e29e600e4d9958af4cdb64e7d8d704dd3d5a2e8dd7aac                ␣
↪       0.0s
    => => naming to docker.io/library/pymlapigen-docker                                 ␣
↪                                0.0s
```

```
docker image ls
    REPOSITORY          TAG       IMAGE ID       CREATED          SIZE
    pymlapigen-docker   latest    c42e13c73503   57 seconds ago   704MB
```

5. Con la imagen ya creada, **lanzar un contenedor** con la imagen utilizando el comando `docker run -d -p <PUERTOMAQUINAHOSPEDADORA>:5000 pymlapigen-docker` siendo **<PUERTOMAQUINAHOSPEDADORA>** el **puerto** del **equipo hospedador** en el que se desea alojar el contenedor.

    - De esta forma, si deseamos alojar el contenedor en el puerto 80, usar el comando `docker run -d -p 80:5000 pymlapigen-docker`.

```
docker run -d -p 80:5000 pymlapigen-docker
    94543cea3e5e47e9512e81d2a47b0500e8b309a60db23cce75c48eb6ee9bfb15
```

Una vez seguidos estos pasos, ya tendremos disponible un **contenedor** Docker **ejecutando la aplicación**.

CAPÍTULO 3

Código fuente

Aquí encontrarás información acerca de las diferentes funciones y clases utilizadas para construir la herramienta.

3.1 Módulo api_generator

Este módulo se encarga de realizar y gestionar un experimento de Machine Learning.

class pymlapigen.api_generator.**API_Generator**(*dataset*)

Bases: object

Manages a Machine Learning experiment

downloadCSV(*path*)

Exports the dataset into a CSV file

Parámetros **path** (*str*) – Path of the file to export

Devuelve Exported file name

Tipo del valor devuelto str

evaluateModel()

Evaluates the API's model generating metrics.

filterDataset(*args*)

Returns the dataset as JSON filtered with args

Parámetros **args** (*dict*) – Filter params

Devuelve JSON filtered dataset

Tipo del valor devuelto (dict)

getAlgorithm()

Returns the model's algorithm full name

Devuelve Model's algorithm name

Tipo del valor devuelto str

`getAlgorithmParams()`

Gets the model algorithm parameters

Devuelve Algorithm parameters

Tipo del valor devuelto dict

`getColumns()`

Return the dataset columns name

Devuelve Dataset columns name

Tipo del valor devuelto list

`getFeatures()`

Returns the dataset feature columns :returns: Dataset feature columns :rtype: list

`getInputLabel()`

Returns the input label name

Devuelve Input label name

Tipo del valor devuelto str

`getModelParams()`

Returns the API's model parameters

Devuelve API's model parameters

Tipo del valor devuelto dict

`getPossibleLabels()`

Returns the unique values of the label column. Used in Classification problems.

Devuelve Unique values of the label column.

Tipo del valor devuelto numpy.ndarray

`getPredictions()`

Returns the API's test set, the label values and predicted values

Devuelve Test set feature values. y_test (Pandas Series): Test set label values. predictions (List): Test set label predicted values.

Tipo del valor devuelto noOhe (Pandas Dataframe)

`getProblem()`

Returns the Machine Learning problem of the API

Devuelve Machine Learning Problem

Tipo del valor devuelto str

`getValues()`

Returns the rows of the dataset

Devuelve Dataset value rows

Tipo del valor devuelto numpy.ndarray

graphs()

Generates plots of the experiment and returns it as figures

Devuelve Figures plot list

Tipo del valor devuelto list[Matplotlib Pyplot Figure]

predictNewValues(*inputData*, *typeData='JSON'*, *separator=','*, *toApi=False*)

Predicts new data using the trained model

Parámetros

- **inputData** (*str, dict*) – If typeData == «Input», it consists of a dict with the input data. If typeData == «JSON», it consists of an str or dict with JSON values. If typeData ==»CSV», it contains the CSV file path
- **typeData** (*str, optional*) – Input data type. Defaults to «JSON».
- **separator** (*str, optional*) – If typeData == «CSV», it represents the CSV file data separator. Defaults to «,».
- **toApi** (*bool, optional*) – Tells whether the predicción comes from the API JSON endpoint or not. Defaults to False.

Devuelve List of predicted values Pandas Dataframe: Input Dataframe with a result column appended Pandas Dataframe: Auxiliar Dataframe that points the type of the result cells

Tipo del valor devuelto list

processNanNull(*mode='drop'*, *fillValue=- 1*)

Processes the missing values from the dataset. There are three modes: Drop mode: deletes rows with missing values. Fill mode: fills missing values with a value. Mean mode: fills missing values with the mean of the column.

Parámetros

- **mode** (*str, optional*) – Processing missing values mode. Defaults to «drop».
- **fillValue** (*int, optional*) – If «fill» mode, represents the filling value. Defaults to -1.

setAlgorithm(*mltype*, *modelType*)

Sets the Machine Learning problem and the model's algorithm

Parámetros

- **mltype** (*str*) – Machine Learning Problem.
- **modelType** (*str*) – Model's Algorithm.

setAlgorithmParams(*algorithmParams*)

Sets the model algorithm parameters

Parámetros **algorithmParams** (*dict*) – Algorithm parameters

setDropColumns(*dropColumns=[]*)

Sets the dropping columns (columns that will be deleted from the dataset) The rest of columns are designed as features.

Parámetros **dropColumns** (*list, optional*) – Dropping columns list. Defaults to [].

setInputLabel(*inputLabel*)

Inputs the label of the experiment.

Parámetros **inputLabel** (*str*) – Name of the label column

setPositiveLabel(*positiveLabel=1*)

Sets the positive label value for binary classification problems. Thanks to this positive value, it will be possible to perform the TP,FP,TN,FN calculations

Parámetros **positiveLabel** (*any, optional*) – Positive label value. Defaults to 1.

setTestSize(*testSize=0.3*)

Sets the test set size percentage

Parámetros **testSize** (*float, optional*) – Set test size percentage. Defaults to 0.3.

trainModel()

Trains the API's model.

pymlapigen.api_generator.**load_csv**(*csvFile*, *separator=','*)

Instances an API_Generator object from a CSV file.

Parámetros

- **csvFile** (*str*) – CSV file's path
- **separator** (*str, optional*) – Values separator in the CSV fil. Defaults to ",".

Devuelve API Generator.

Tipo del valor devuelto *API_Generator*

pymlapigen.api_generator.**load_json**(*inputJson*)

Instances an API_Generator object from a JSON dict.

Parámetros **inputJson** (*dict*) – JSON dict.

Devuelve API Generator.

Tipo del valor devuelto *API_Generator*

3.2 Módulo cli

Este módulo se encarga de inicializar la aplicación web desde el comando *pymlapigen*.

pymlapigen.cli.**cli**()

Runs the pymlapigen app from the console command

3.3 Módulo config

Este módulo almacena la configuración del microframework utilizado Flask.

Si quieres recibir correos electrónicos que notifiquen la finalización de la generación, debes descomentar e introducir las variables *MAIL_SERVER*, *MAIL_PORT*, *MAIL_USERNAME*, *MAIL_PASSWORD*, *MAIL_USE_TLS* y *MAIL_USE_SSL*.

pymlapigen.config.**JSON_SORT_KEYS = False**

Sorts json keys

pymlapigen.config.**UPLOAD_FOLDER = 'static/files'**

App upload files folder

3.4 Módulo routes

Este módulo contiene la lógica de las rutas de la aplicación.

pymlapigen.routes.**apiHome**(*apiName*)

API's home route

Parámetros **apiName** (*str*) – Name of the API

pymlapigen.routes.**dataset**(*apiName*)

Dataset route.

Parámetros **apiName** (*str*) – Name of the API

pymlapigen.routes.**datasetApi**(*apiName*)

JSON API Dataset route.

Parámetros **apiName** (*str*) – Name of the API

pymlapigen.routes.**defaultApiRoute**()

JSON Home route.

pymlapigen.routes.**destroy**(*apiName*)

Destroy route.

Parámetros **apiName** (*str*) – Name of the API to destroy

pymlapigen.routes.**download_CSV**(*apiName*)

Calls the download CSV function from the API and returns its filename

Parámetros **apiName** (*str*) – Name of the API

pymlapigen.routes.**export**(*apiName*)

Export route.

Parámetros **apiName** (*str*) – Name of the API

pymlapigen.routes.**get_import**()

Import GET route.

Parámetros **apiName** (*str*) – Name of the API

pymlapigen.routes.**get_load_0**()

Load GET route. Step 0.

Parámetros **apiName** (*str*) – Name of the API

pymlapigen.routes.**get_load_1**(*apiName*)

Load GET route. Step 1.

Parámetros **apiName** (*str*) – Name of the API

pymlapigen.routes.**get_load_2**(*apiName*)

Load GET route. Step 2.

Parámetros **apiName** (*str*) – Name of the API

pymlapigen.routes.**get_load_3**(*apiName*)

Load GET route. Step 3.

Parámetros **apiName** (*str*) – Name of the API

pymlapigen.routes.**graphs**(*apiName*)

Graphs route.

Parámetros **apiName** (*str*) – Name of the API

pymlapigen.routes.**home**()

Home route.

pymlapigen.routes.**homeApi**(*apiName*)

JSON API Home route.

Parámetros **apiName** (*str*) – Name of the API

pymlapigen.routes.**loadApi**()

JSON API Load POST route.

pymlapigen.routes.**metrics**(*apiName*)

Metrics route.

Parámetros **apiName** (*str*) – Name of the API

pymlapigen.routes.**metricsApi**(*apiName*)

JSON API Metrics route.

Parámetros **apiName** (*str*) – Name of the API

pymlapigen.routes.**model**(*apiName*)

Model route.

Parámetros **apiName** (*str*) – Name of the API

pymlapigen.routes.**modelApi**(*apiName*)

JSON API Model route.

Parámetros **apiName** (*str*) – Name of the API

pymlapigen.routes.**page_not_found**(*e*)

pymlapigen.routes.**post_import**()

Import POST route.

Parámetros **apiName** (*str*) – Name of the API

pymlapigen.routes.**post_load_0**()

Load POST route. Step 0.

Parámetros **apiName** (*str*) – Name of the API

pymlapigen.routes.**post_load_1**(*apiName*)

Load POST route. Step 1.

Parámetros **apiName** (*str*) – Name of the API

pymlapigen.routes.**post_load_2**(*apiName*)

Load POST route. Step 2.

Parámetros **apiName** (*str*) – Name of the API

pymlapigen.routes.**post_load_3**(*apiName*)

Load POST route. Step 3.

Parámetros **apiName** (*str*) – Name of the API

pymlapigen.routes.**predict**(*apiName*)

Predict GET route.

Parámetros **apiName** (*str*) – Name of the API

pymlapigen.routes.**predictApi**(*apiName*)

JSON API Predict POST route.

Parámetros **apiName** (*str*) – Name of the API

pymlapigen.routes.**predict_post**(*apiName*)

Predict POST route.

Parámetros **apiName** (*str*) – Name of the API

3.5 Módulo principal

Paquete PyMLAPIGen. Es el punto de entrada a la aplicación.