- model_service module
  - class model_service.PredictionRequest(*, model_folder: str, model_name: str, window_size: int, angulo: list, par: list)
    - class Config
    - protected_namespaces = ()
    - angulo : list
    - model_computed_fields : ClassVar[dict[str, ComputedFieldInfo]] = {}
    - model_config : ClassVar[ConfigDict] = {'protected_namespaces': ()}
    - model_fields : ClassVar[dict[str, FieldInfo]] = {'angulo': FieldInfo(annotation=list, required=True), 'model_folder': FieldInfo(annotation=str, required=True), 'model_name': FieldInfo(annotation=str, required=True), 'par': FieldInfo(annotation=list, required=True), 'window_size': FieldInfo(annotation=int, required=True)}
    - model_folder : str
    - model_name : str
    - par : list
    - window_size : int
  - model_service.load_model_and_scaler(model_folder, model_name, window_size)
  - model_service.predict(request: PredictionRequest)
  - model_service.read_root()
  - model_service.start_service()

model_service module

Servicio de Predicción con FastAPI

Este script implementa un servicio de predicción utilizando FastAPI. Se encarga de cargar modelos de predicción y escalar datos para realizar predicciones.

Imports: : - fastapi: Framework para construir APIs.

- pydantic: Librería para validación de datos.
- numpy: Librería para manejo de matrices y operaciones numéricas.
- joblib: Librería para cargar modelos serializados.
- os: Librería para interactuar con el sistema operativo.

Funciones: : - load_model_and_scaler: Carga el modelo de predicción y el scaler.

- predict: Realiza una predicción utilizando el modelo y los datos proporcionados.
- read_root: Ruta raíz de prueba.
- start_service: Inicia el servidor de FastAPI.

class model_service.PredictionRequest(*, model_folder: str, model_name: str, window_size: int, angulo: list, par: list)

Bases: `BaseModel`

Modelo de solicitud de predicción.

Atributos: : model_folder (str): Carpeta del modelo. model_name (str): Nombre del modelo. window_size (int): Tamaño de la ventana de datos. angulo (list): Lista de valores de ángulo. par (list): Lista de valores de par.

***class* Config**

Bases: `object`

protected_namespaces = ***()***

angulo ***: list***

model_computed_fields ***: ClassVar[dict[str, ComputedFieldInfo]] = {}***

A dictionary of computed field names and their corresponding ComputedFieldInfo objects.

model_config ***: ClassVar[ConfigDict] = {'protected_namespaces': ()}***

Configuration for the model, should be a dictionary conforming to [ConfigDict] [pydantic.config.ConfigDict].

model_fields ***: ClassVar[dict[str, FieldInfo]] = {'angulo': FieldInfo(annotation=list, required=True), 'model_folder': FieldInfo(annotation=str, required=True), 'model_name': FieldInfo(annotation=str, required=True), 'par': FieldInfo(annotation=list, required=True), 'window_size': FieldInfo(annotation=int, required=True)}***

Metadata about the fields defined on the model, mapping of field names to [FieldInfo] [pydantic.fields.FieldInfo].

This replaces Model._*fields*_ from Pydantic V1.

model_folder ***: str***

model_name ***: str***

par ***: list***

window_size ***: int***

model_service.load_model_and_scaler(model_folder, model_name, window_size)

Carga el modelo de predicción y el scaler.

Args: : model_folder (str): Carpeta del modelo. model_name (str): Nombre del modelo. window_size (int): Tamaño de la ventana de datos.

Returns: : tuple: Modelo de predicción y scaler.

Raises: : FileNotFoundError: Si el modelo o el scaler no existen.

model_service.predict(request: PredictionRequest)

Realiza una predicción utilizando el modelo y los datos proporcionados.

Args: : request (PredictionRequest): Solicitud de predicción con los datos y configuración del modelo.

Returns: : dict: Resultado de la predicción.

model_service.read_root()

Ruta raíz de prueba.

Returns: : dict: Mensaje de bienvenida.

model_service.start_service()

Inicia el servidor de FastAPI.