Polo Strategico Nazionale

Realizzazione e gestione di una nuova infrastruttura informatica al servizio della Pubblica Amministrazione denominata Polo Strategico Nazionale (“PSN”), di cui al comma 1 dell’articolo 33-septies del d.l. n. 179 del 2012

CUP: J51B21005710007

CIG: 9066973ECE

# Manuale Utente

# PaaS AI

Data: 23/06/2023

PSN_Manuale Utente PaaS AI

Ed. 1 - ver. 1.0

Polo Strategico Nazionale

QUESTA PAGINA È LASCIATA INTENZIONALMENTE BIANCA

# STATO DEL DOCUMENTO

| TITOLO DEL DOCUMENTO | | | |
|---|---|---|---|
| Manuale Utente PaaS AI | | | |
| EDIZ. | REV. | DATA | AGGIORNAMENTO |
| 1 | 1.0 | 23/06/2023 | Prima versione |
| | | | |

| | |
|---|---|
| NUMERO TOTALE PAGINE: | 45 |

| AUTORE: | |
|---|---|
| Team di lavoro PSN | Unità operative Solution Development, Technology Hub e Sicurezza |

| REVISIONE: | |
|---|---|
| Referente del Servizio | Paolo Trevisan |

| APPROVAZIONE: | |
|---|---|
| Direttore del Servizio | Antonio Garelli |

# LISTA DI DISTRIBUZIONE

INTERNA A:

- Funzione Solution Development
- Funzione Technology Hub
- Funzione Sicurezza
- Referente Servizio
- Direttore Servizio

ESTERNA A:

- Direttore dell'Esecuzione Contrattuale (DEC) PSN ing. Fabrizio Marchese

Polo Strategico Nazionale

# INDICE

# LISTA DELLE FIGURE

# LISTA DELLE TABELLE

# 1 Definizioni e Acronimi

## 1.1 Definizioni

| Definizione | Descrizione |
|---|---|
| PSN | È la nuova società che è stata costituita nell'ambito del progetto del Cloud Nazionale |
| TBC | Il tema è stato discusso ma è in attesa di conferma dalle parti coinvolte |
| TBD | Il tema non è ancora stato discusso |

Tabella 1: Glossario Definizioni

## 1.2 Acronimi

| Acronimo | Descrizione |
|---|---|
| AD | Active Directory |
| APT | Advanced Persistent Threat |
| API | Application Program Interface |
| AV | AntiVirus |
| BaaS | Backup as a Service |
| CaaS | Container as a Service |
| CLI | Command Line Interface |
| CSP | Cloud Service Provider |
| DBE | DataBase Encryption |
| DDC | Data Discovery and Classification |
| DDoS | Distributed DoS |
| DE | Data Encryption |
| DLP | Data Loss Prevention |
| DM | Data Masking |
| DMZ | DeMilitarized Zone |
| DNS | Domain Name System |
| DoS | Denial of Service |
| DWDM | Dense Wavelength Division Multiplexing |
| EDE | Endpoint Disk Encryption |
| EDR | Endpoint Detection and Response |
| FIM | File Integrity Monitoring |
| FW | FireWall |
| Gbps | Gigabits per second |
| GUI | Graphical User Interface |
| HA | High Availability |
| HSM | Hardware Security Module |

| Acronimo | Descrizione |
|---|---|
| HTTP | HyperText Transfer Protocol |
| HTTPS | HTTP Secure |
| IaaS | Infrastructure as a Service |
| IAG | Identity and Access Governance |
| I&AM | vedi IAM |
| IAM | Identity and Access Management |
| IDS | Intrusion Detection System |
| IP | Internet Protocol |
| IPS | Intrusion Prevention System |
| iSCSI | Internet SCSI |
| ISO | International Organization for Standardization |
| KMS | Key Management System |
| L2 | Layer 2 (della pila ISO/OSI) |
| L3 | Layer 3 (della pila ISO/OSI) |
| L4 | Layer 4 (della pila ISO/OSI) |
| LAG | Link Aggregation Group |
| LAN | Local Area Network |
| LM | Log Management |
| LOM | Lights Out Management |
| MAC | Media Access Control |
| MC-LAG | Multi Chassis LAG |
| MDM | Mobile Device Management |
| MFA | Multi Factor Authentication |
| MPLS | MultiProtocol Label Switching |
| NAC | Network Access Control |
| NGFW | Next Generation FW |
| NL-SAS | Near Line SAS |
| NPB | Network Packet Broker |
| NTP | Network Time Protocol |
| OOB | Out of band |
| OSI | Open Systems Interconnection |
| PaaS | Platform as a Service |
| PA | Pubblica Amministrazione |
| PAM | Privileged Access Management |
| PdL | Postazione di Lavoro |
| PSN | Polo Strategico Nazionale |
| rpm | Rotation per minute |
| SaaS | Software as a Service |
| SAN | Storage Area Network |
| SAS | Serial Attached SCSI |
| SCSI | Small Computer System Interface |
| SEG | Security Email Gateway |
| SFP | Small Form-factor Pluggable |
| SFP+ | Enhanced SFP |
| SIEM | Security Information and Event Management |
| SNMP | Simple Network Management Protocol |
| SOAR | Security Orchestration, Automation and Response |

| Acronimo | Descrizione |
|---|---|
| SOC | Security Operation Center |
| SQL | Structured Query Language |
| SR | Short Reach |
| SWG | Secure Web Gateway |
| TB | TeraByte |
| TBC | To Be Confirmed |
| TBD | To Be Defined |
| TI | Threat Intelligence and Infosharing |
| ToR | Top of Rack |
| VBR | Veeam Backup & Replication |
| VDOM | Virtual DOMain (Contesto Virtuale) |
| VLAN | Virtual LAN |
| VM | Vulnerability Management |
| VPN | Virtual Private Network |
| WAF | Web Application Firewall |
| WAN | Wide Area Network |
| XSS | Cross-Site Scripting |

Tabella 2: Glossario Acronimi

# 2 Executive Summary

## 2.1 Scopo del documento

Il documento ha lo scopo di fornire una guida all'utente finale delle funzionalità rilasciate nei servizi PaaS AI, ovvero AI Platform, Semantic Knowledge Search, Text Analytics e Audio Analytics.

# 3 PaaS AI – AI Platform

In questa sezione verranno visualizzati i dettagli di utilizzo degli strumenti inclusi nell'AI Platform del PSN.
Gli strumenti che verranno descritti sono:

- Jupyter Notebook
- Mlflow
- MinIO
- BentoML

L'utilizzo degli strumenti verrà descritto nell'ambito di un esperimento, che inizierà tramite lo sviluppo del modello di AI tramite Jupyter Notebook, l'avvio di un esperimento che verrà monitorato su Mlflow, l'accesso a MinIO per visionare gli artefatti generati dall'addestramento ed infine il serving del modello tramite BentoML. Per visualizzare gli step descritti in maniera approfondita, si può fare riferimento alla Figura 1.

## 3.1 Jupyter Notebook

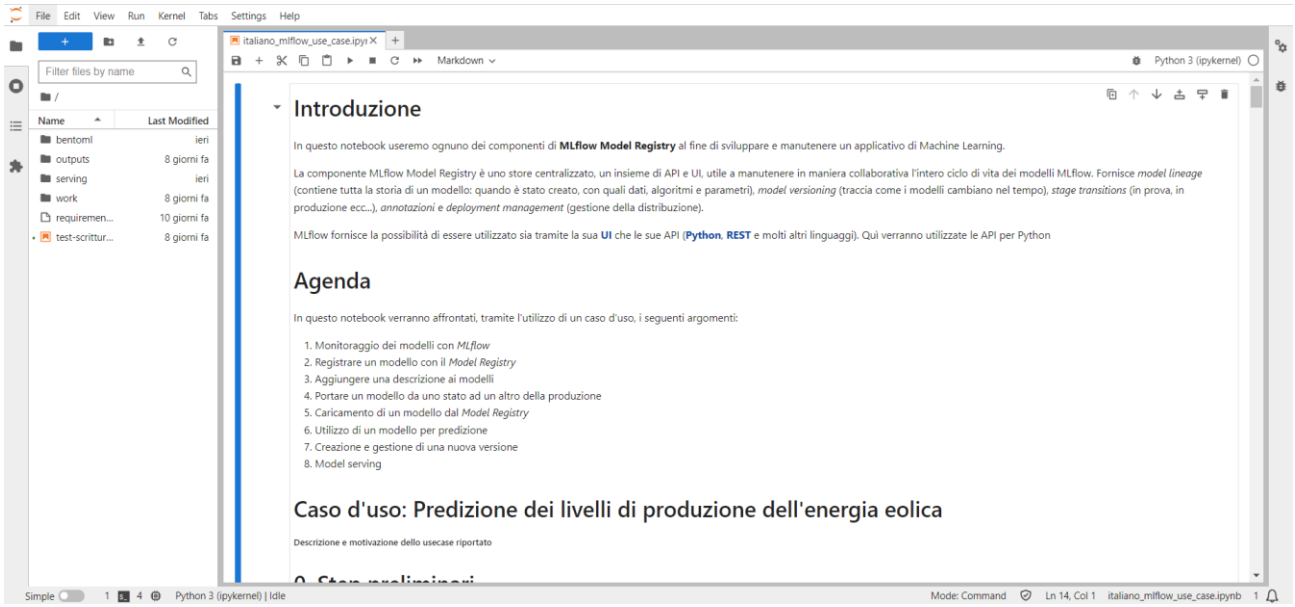



Figura 1 – Dashboard Principale di Jupyter Lab

In Figura 1 è raffigurata la dashboard principale di accesso a Jupyter Lab.
La schermata è composta da un pannello in alto, una parte centrale in cui è possibile visionare e fare operazioni sui Notebook (e.g. run, modifiche del codice) ed infine una barra laterale rappresentata in Figura 2.

Dalla “navbar” laterale è possibile visualizzare i file disponibili tramite la sezione “File Browser”, visualizzare l’indice del notebook tramite “Table of Contents”, visualizzare il kernel attivo tramite “Running Terminal and Kernels” ed infine si possono visualizzare i pacchetti installati nel Kernel tramite “Extension Manager”.

Jupyter Notebook consente di effettuare i passi principali dello sviluppo di un modello di AI utilizzando Mlflow per il monitoraggio e BentoML per il serving:

- Caricamento del dataset
- Definizione del modello di AI utilizzando librerie preinstallate (e.g. Pytorch, Tensorflow, Keras)
- Monitoraggio con Mlflow (Figura 3)
- Registrazione del modello con il Model Registry
- Aggiungere descrizione ai modelli
- Portare un modello da uno stato ad un altro della produzione
- Caricamento di un modello da un Model Registry
- Utilizzo di un modello per predizione
- Creazione e gestione di una nuova versione
- Model Serving

I dettagli tecnici sono ampiamente dettagliati nel Notebook nell’Allegato 1.

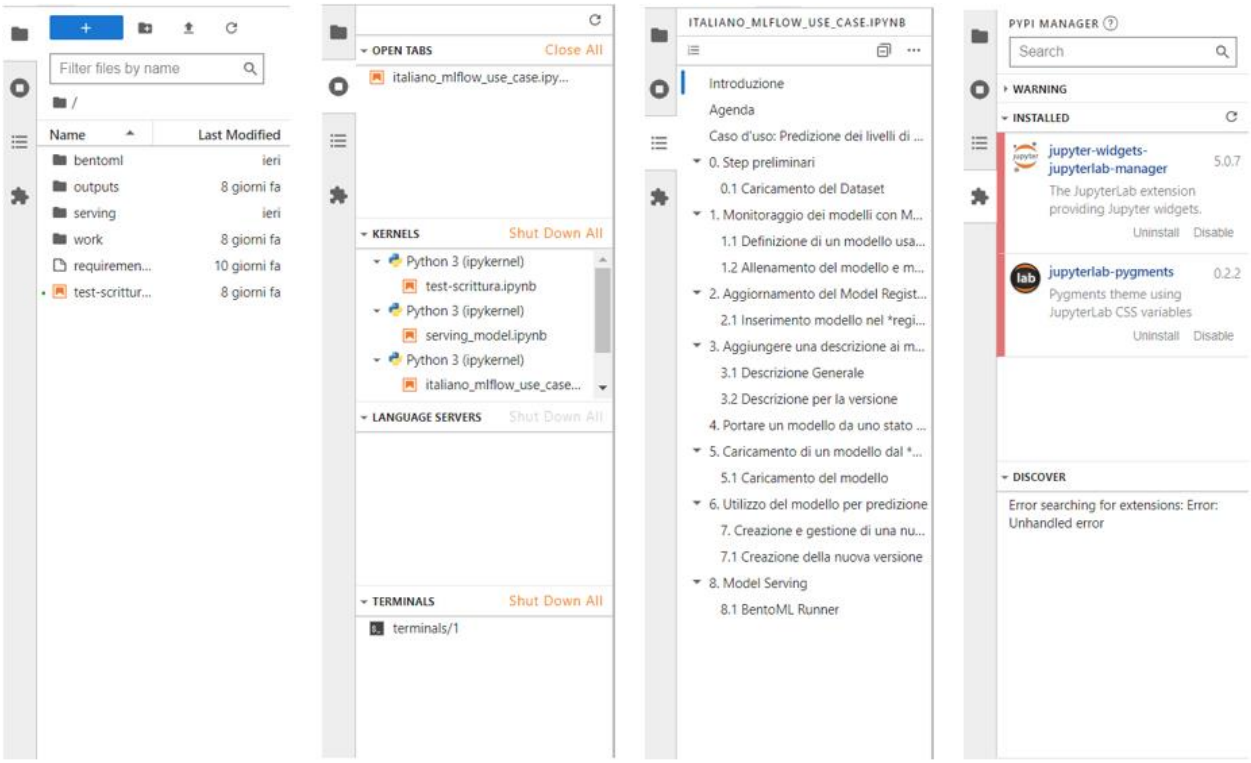



Figura 2 – Dettaglio della Navbar laterale e delle sue quattro schermate disponibili: File Browser, Running Terminals and Kernels, Table of Contents, Extension Manager.

Per i dettagli tecnici sullo sviluppo di questi passi si faccia riferimento al Notebook allegato:

Esempio_Di_Training_
Con_Jupyter_Noteboo

Allegato 1 – Esempio di Training con Jupyter Notebook. Il linguaggio utilizzato è Python.

```
## 1.2 Allenamento del modello e monitoraggio con MLFlow

Con la seguente cella gli iper-parametri, indicatori delle performance, codice sorgente ed artefatti verranno tracciati usando MLFlow.

import mlflow
import mlflow.keras
import mlflow.tensorflow

X_train, y_train = get_training_data()
X_train = X_train.to_numpy()
y_train = y_train.to_numpy()

with mlflow.start_run(run_name='training-tensorflow-power-forecasting-model-0'):
    # Automatically capture the model's parameters, metrics, artifacts, and source code with the `autolog()` function
    mlflow.keras.autolog()

    train_keras_model(X_train, y_train)
    run_id = mlflow.active_run().info.run_id
```

Figura 3 – Esempio di codice in Python che permetto il monitoraggio del modello con MLflow

## 3.2 Mlflow

L'utente potrà accedere alla piattaforma accedendo al link tramite browser qualora fosse collegato alla rete aziendale o rete VPN opportuna. In Figura 4 è raffigurata la dashboard di atterraggio di MLflow. La schermata ha due tab principali: "Experiments" e "Models".

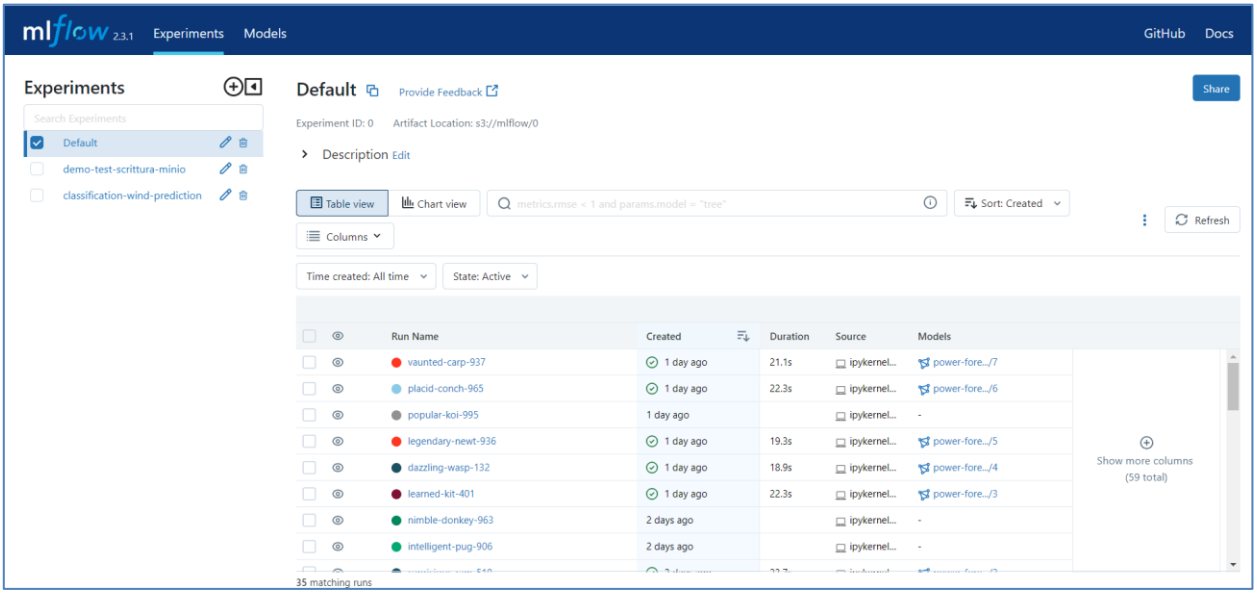



Figura 4 – Dashboard principale di MLflow: tab "*Experiments*"

### 3.2.1 Experiments

Nel tab *"Experiments"* a sinistra sono elencati gli esperimenti eseguiti, selezionandone uno è possibile visualizzare le *"Run"* eseguite. Cliccando su un *"Run"* si possono visualizzare i dettagli (Figura 5). In alto si possono visualizzare i metadati riguardo il *run* dell'esperimento, come ad esempio: run ID, data di esecuzione, user.
In basso sono disponibili dei menù ad apertura che consentono di visualizzare i seguenti dettagli:

- Description
- Parameters
- Metrics
- Tags
- Artifacts

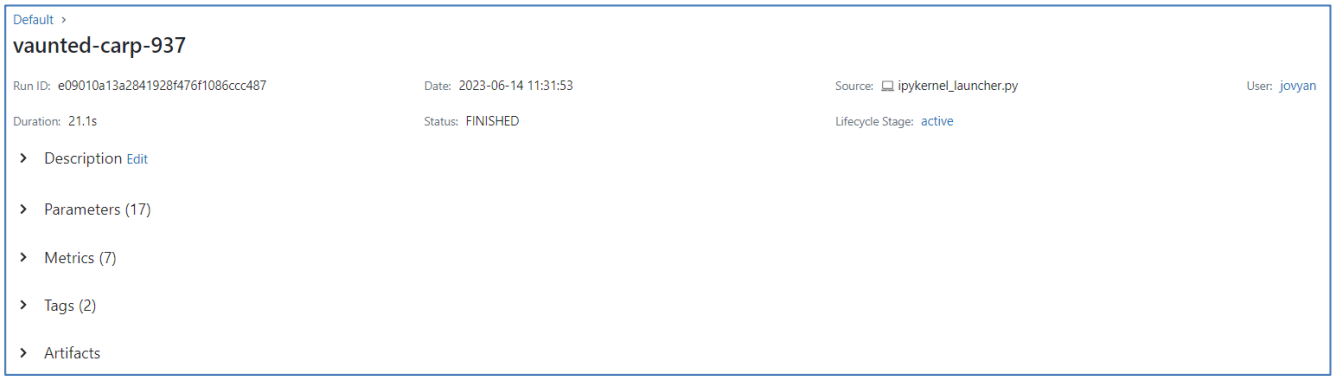



Figura 5 – Dettagli di visualizzazione di una esecuzione (*run)* di un esperimento

In Figura 6 è raffigurato il menù ad espansione degli artefatti del *run* di un esperimento. Qui si potranno visualizzare gli artefatti salvati una volta eseguito l'esperimento. Inoltre, in alto è possibile visualizzare il percorso di salvataggio dei file all'interno del MinIO (in questo caso s3://mlflow/0/e09010a13a2841928f476f1086ccc487/artifacts/model)

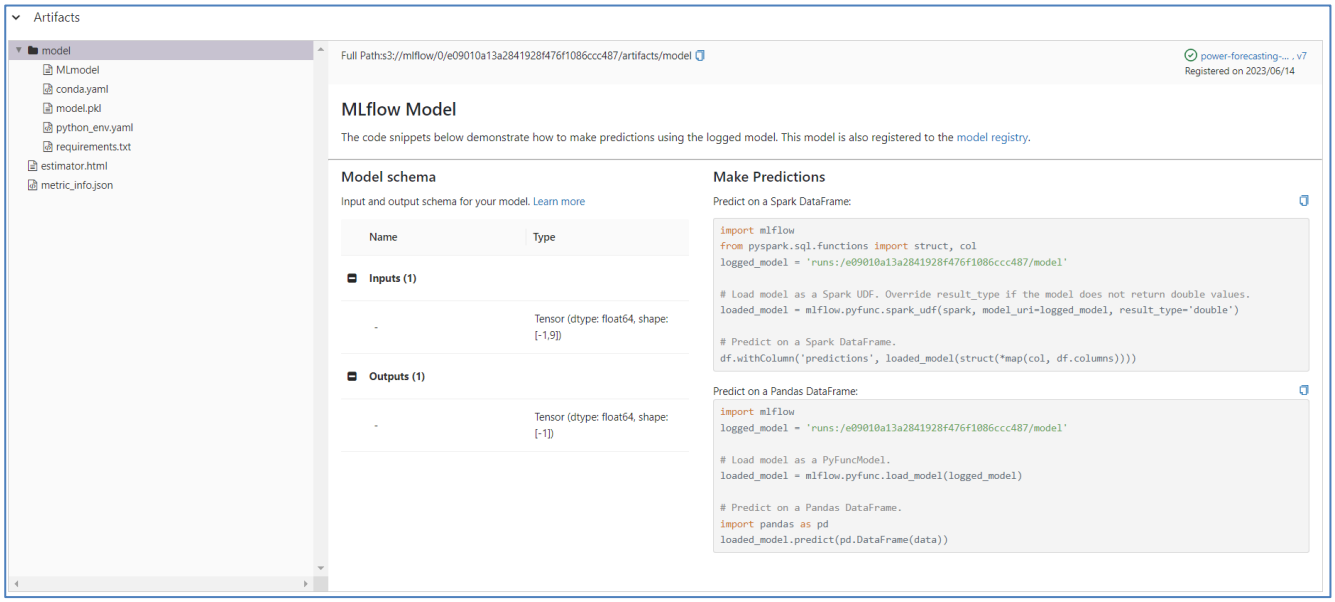



Figura 6 – Dettaglio del menù ad espansione *Artifacts*

## 3.2.2 Models

In Figura 7 si può visualizzare la schermata del tab Models in cui è possibile visionare i modelli registrati.

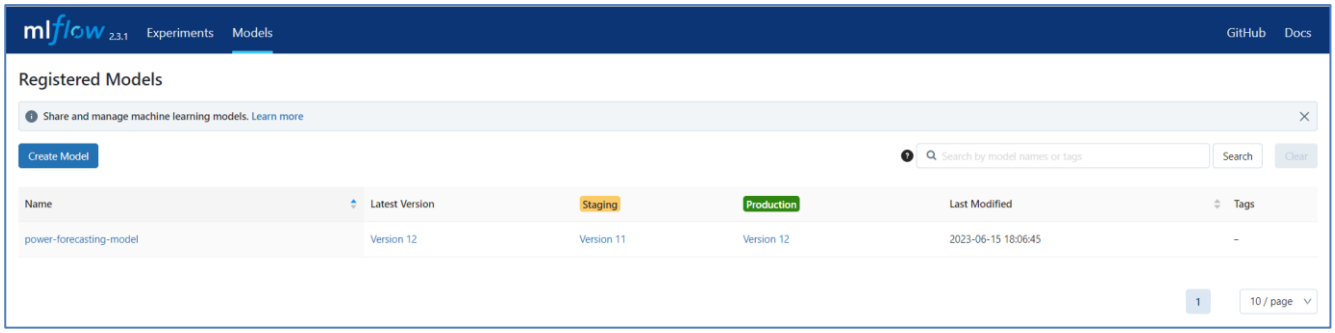



Figura 7 – Mlflow: tab "*models*"

Nella schermata principale si può visualizzare l'ultima versione del modello, la versione in *staging*, la versione in produzione, la data dell'ultima modifica e il tag associato.
Cliccando sul modello di interesse si possono visualizzare ulteriori dettagli, presenti in Figura 8.

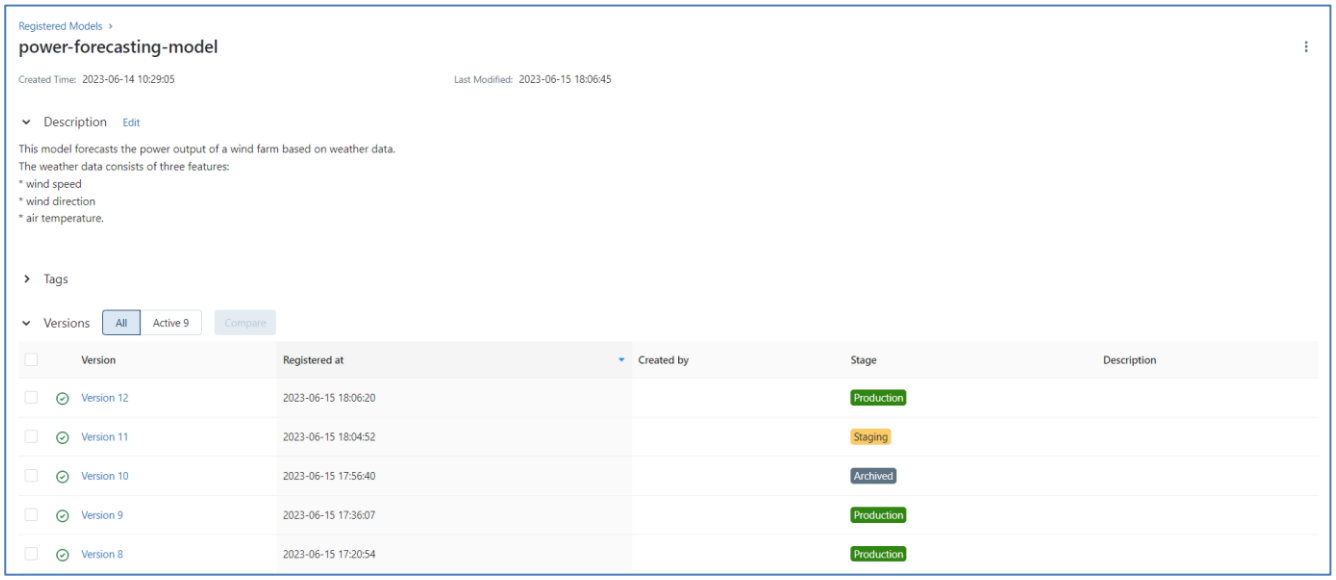



Figura 8 – Modello registrato

## 3.3 MinIO

L'accesso a MinIO si effettua tramite link opportuno e per visualizzare i file all'interno dell'*object storage* occorre effettuare il Login (Figura 9).

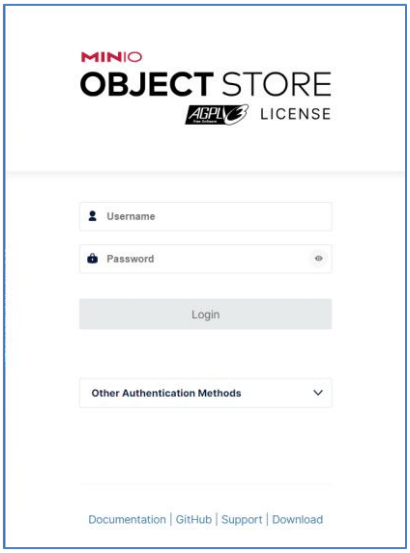



Figura 9 – Login di accesso a MinIO

Nella dashboard principale sono disponibili vari bucket (Figura 10). All'interno del bucket sono disponibili varie cartelle, una per ogni *"experiment"*. Si noti, infatti, che le cartelle presenti in Figura 11 sono 3, e si riferiscono agli *experiment* in *mlflow* presenti in Figura 4.

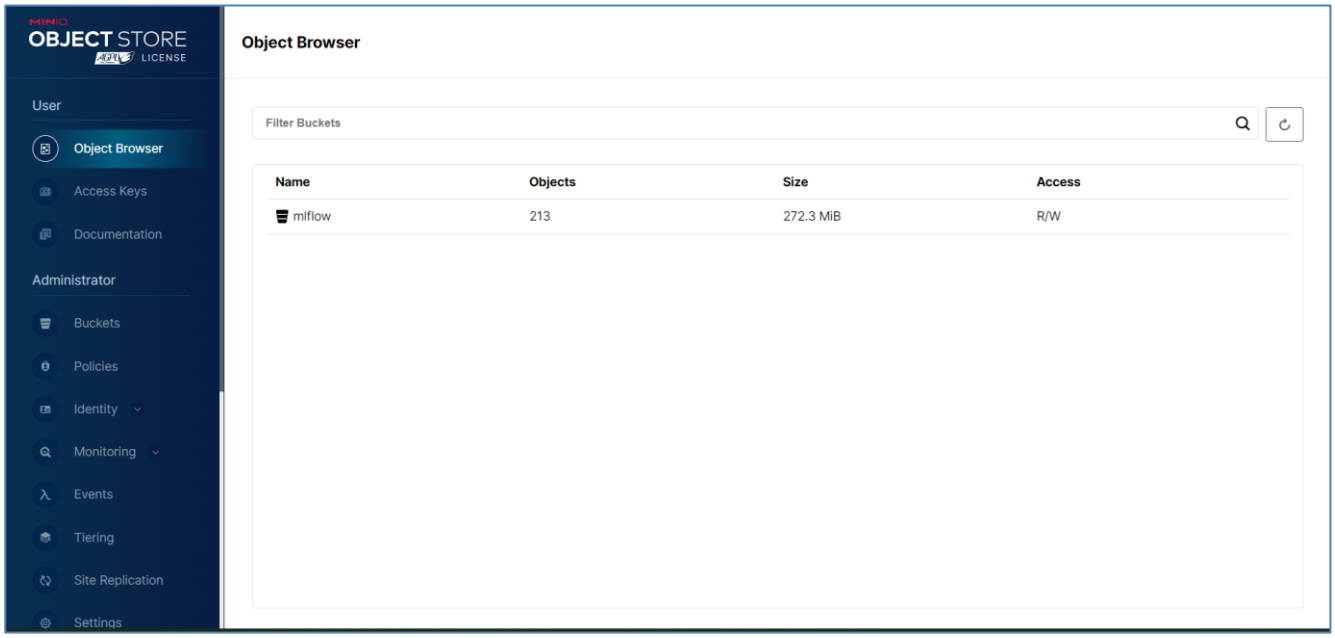



Figura 10 – Dashboard principale di MinIO

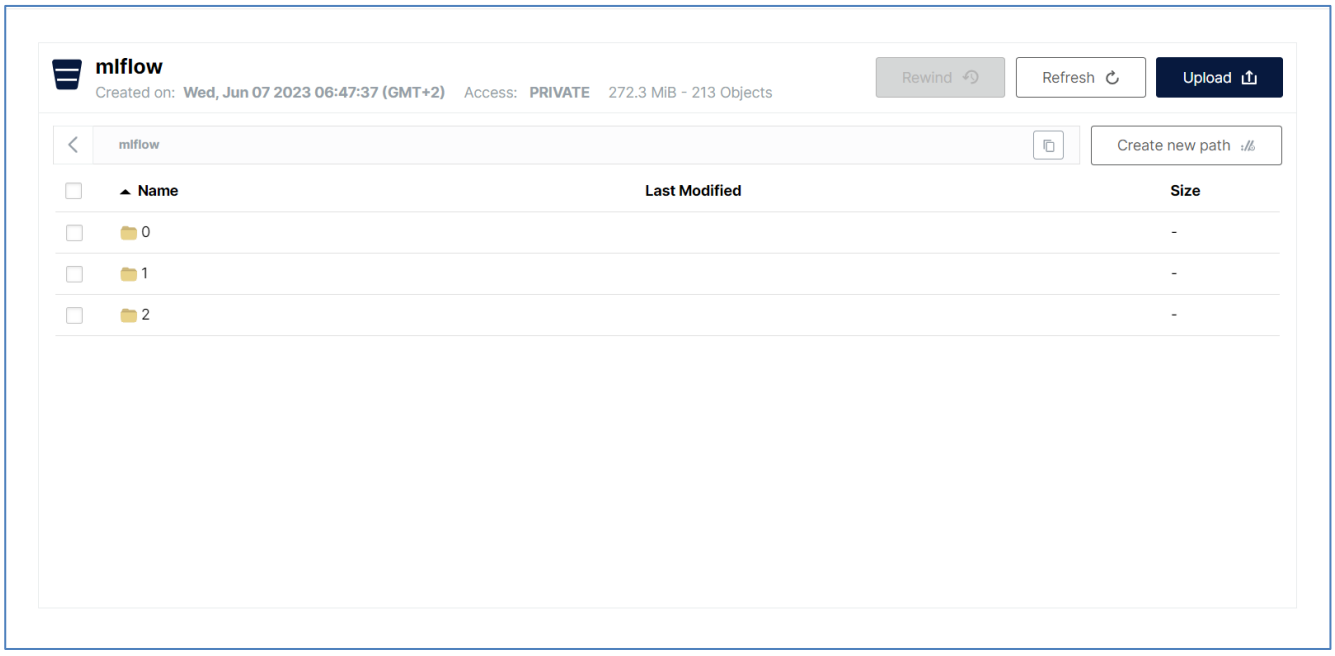



Figura 11 – Visualizzazione del bucket *mlflow: sono presenti 3 folder all'interno*. Ogni cartella è relativa ad un *esperimento*

All'interno di ogni folder all'interno del bucket, sono presenti gli artefatti generati ad ogni *experiment (*Figura 12*)*. Gli artefatti generati sono visualizzabili sono presenti in Figura 13.

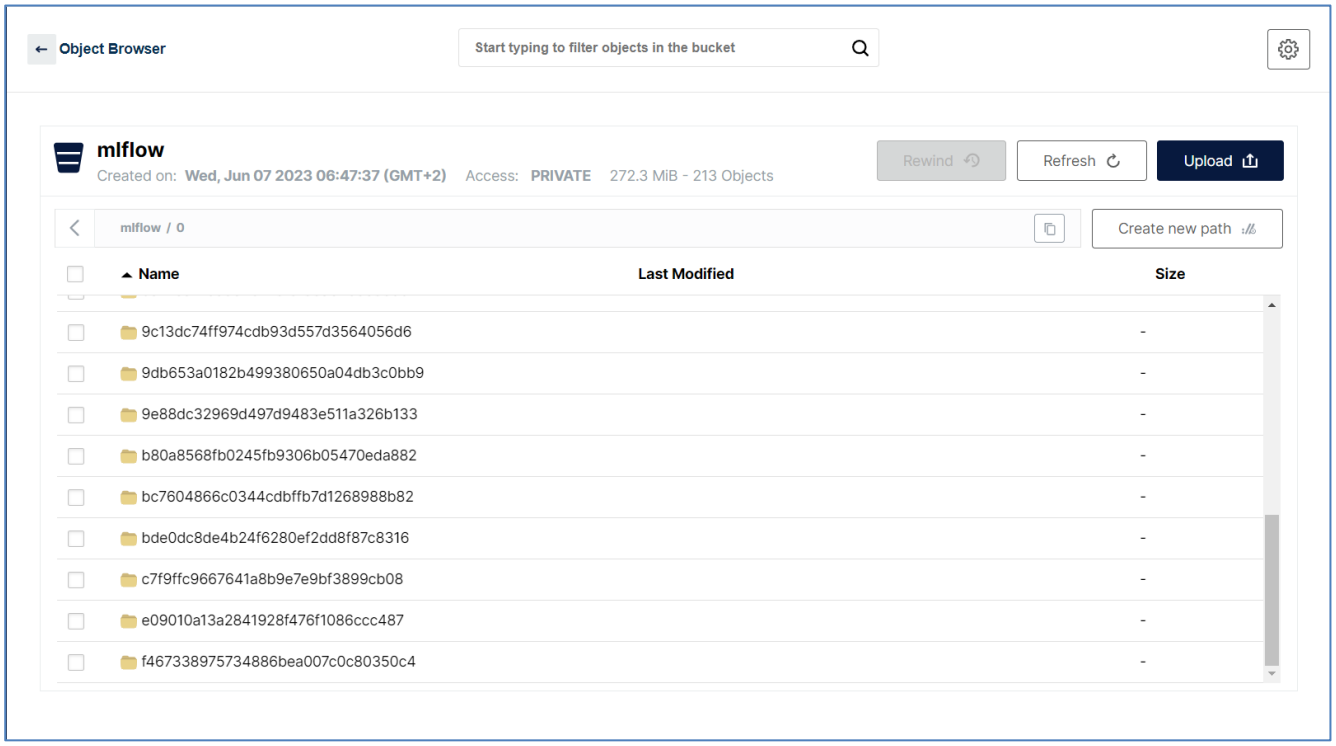



Figura 12 – Visualizzazione dei run effettuati all'interno dell'experiment "*2*"

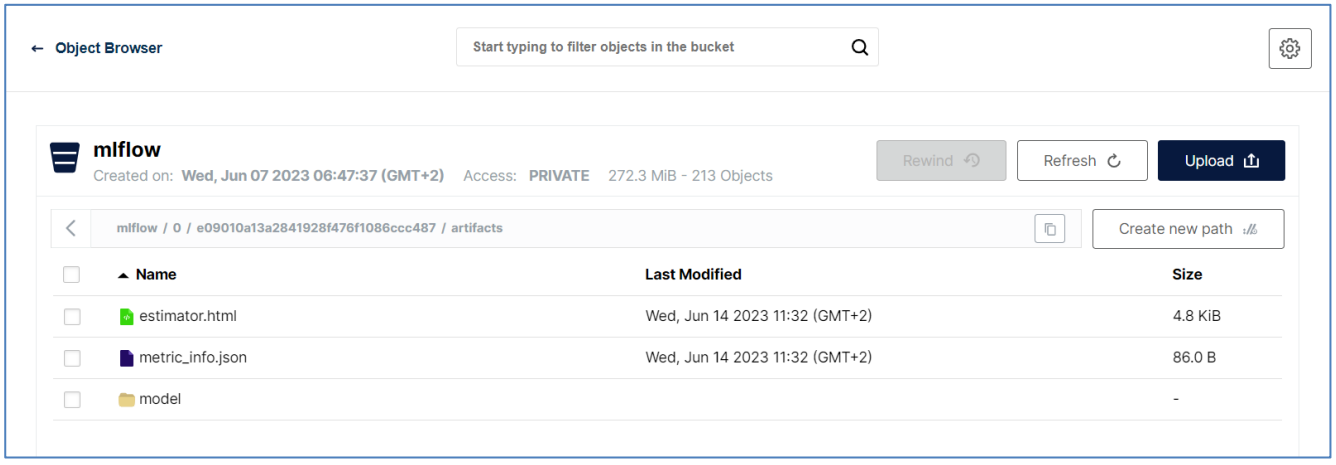



Figura 13 – Visualizzazione degli artefatti all'interno di uno specifico "*run*"

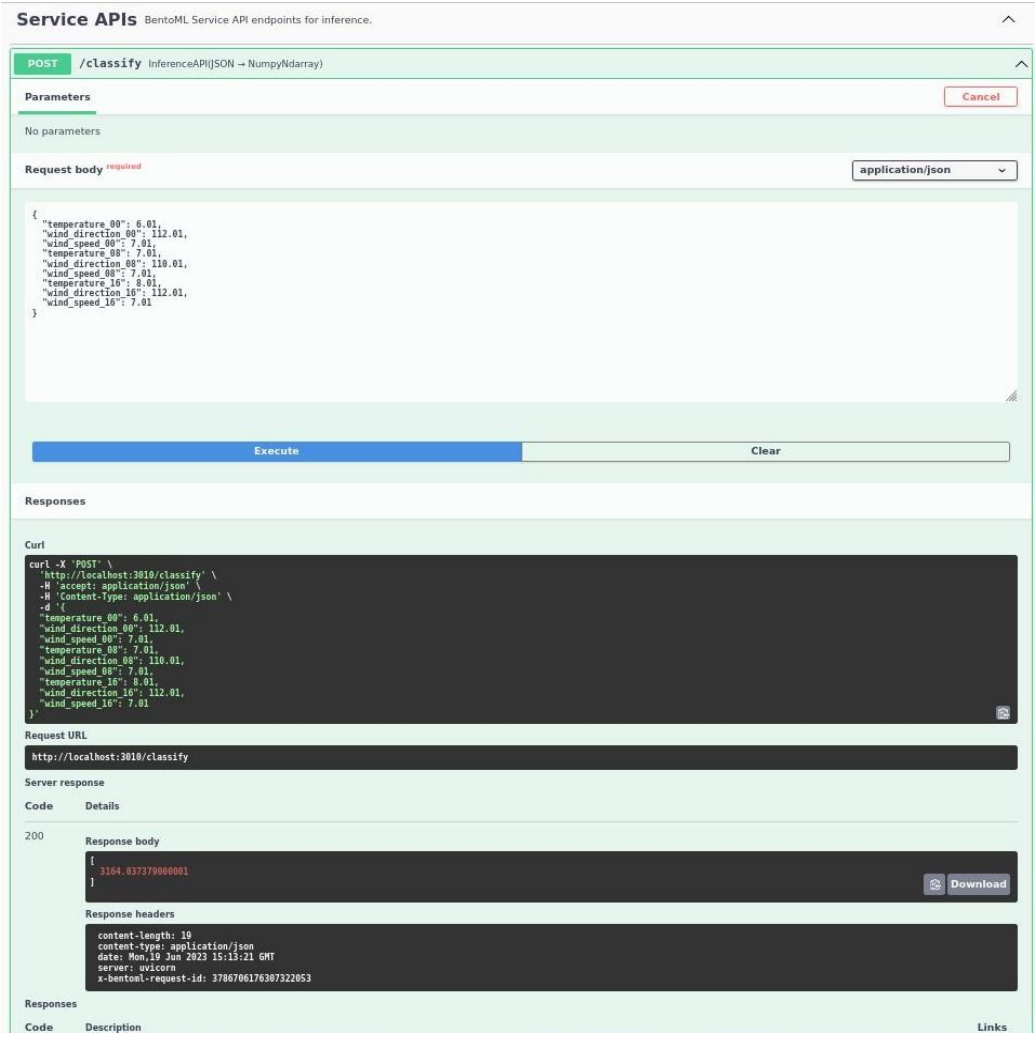



Figura 14 – Esempio di inferenza tramite chiamata REST API

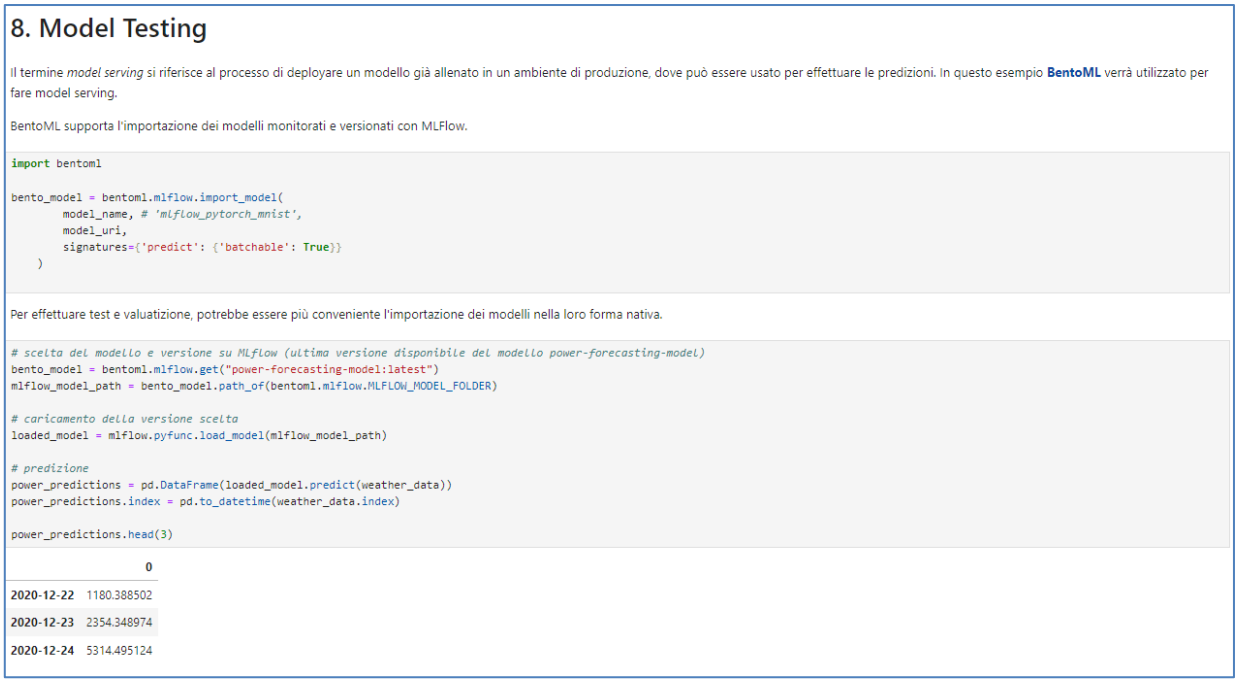
## 3.4 BentoML

BentoML è una libreria potente e flessibile che svolge un ruolo chiave nel processo di testing e serving dei modelli di machine learning. Una volta che un modello è stato addestrato ed è pronto per il deploy in un ambiente di produzione, BentoML fornisce gli strumenti necessari per testare il modello ed effettuare inferenze attraverso delle Rest API.

### 3.4.1 Testing del modello importato da MLflow

Una delle principali caratteristiche di BentoML è la sua integrazione con MLFlow. In particolare, è possibile importare dei modelli monitorati e versionati con MLFlow.
Quando si utilizza BentoML, il primo passo consiste nel testare il modello importato da MLFlow. Questo consente di verificare la corretta funzionalità del modello prima di procedere con il

serving. BentoML semplifica il processo di test, fornendo strumenti intuitivi e funzionalità per eseguire facilmente predizioni sul modello importato.



## 8. Model Testing

Il termine *model serving* si riferisce al processo di deployare un modello già allenato in un ambiente di produzione, dove può essere usato per effettuare le predizioni. In questo esempio **BentoML** verrà utilizzato per fare model serving.

BentoML supporta l'importazione dei modelli monitorati e versionati con MLFlow.

```
import bentoml

bento_model = bentoml.mlflow.import_model(
        model_name, # 'mlflow_pytorch_mnist',
        model_uri,
        signatures={'predict': {'batchable': True}}
    )
```

Per effettuare test e valuatizione, potrebbe essere più conveniente l'importazione dei modelli nella loro forma nativa.

```
# scelta del modello e versione su MLflow (ultima versione disponibile del modello power-forecasting-model)
bento_model = bentoml.mlflow.get("power-forecasting-model:latest")
mlflow_model_path = bento_model.path_of(bentoml.mlflow.MLFLOW_MODEL_FOLDER)

# caricamento della versione scelta
loaded_model = mlflow.pyfunc.load_model(mlflow_model_path)

# predizione
power_predictions = pd.DataFrame(loaded_model.predict(weather_data))
power_predictions.index = pd.to_datetime(weather_data.index)

power_predictions.head(3)
```

| | 0 |
|---|---|
| **2020-12-22** | 1180.388502 |
| **2020-12-23** | 2354.348974 |
| **2020-12-24** | 5314.495124 |

Figura 15 – Esempio di Model Testing in ambiente Jupyter Notebook

### *3.4.2 Build delle API su BentoML*

Successivamente, BentoML offre la possibilità di creare API per il modello importato. Queste API consentono di effettuare richieste per ottenere predizioni utilizzando il modello. BentoML semplifica il processo di creazione delle API, consentendo agli utenti di definire in modo chiaro e strutturato le funzionalità e gli input necessari per le predizioni.

### *3.4.3 Generazione dell'immagine docker del modello importato su BentoML*

Per implementare il modello in un ambiente di produzione, BentoML offre la generazione di un'immagine Docker del modello. Questa immagine Docker contiene tutte le dipendenze e le risorse necessarie per eseguire il modello in modo isolato e scalabile. L'immagine Docker può quindi essere distribuita su piattaforme di cloud computing o in ambienti on-premise.
Nella piattaforma saranno messi a disposizione un file Python o uno Shell Script in grado di creare un'immagine docker del modello tramite BentoML (Figura 16 e Figura 17).

```python
import argparse
import subprocess
import shlex
import mlflow
import bentoml
import os


if __name__ == '__main__':
    parser = argparse.ArgumentParser()
    parser.add_argument('--model_name', type=str, default = 'power_forecasting_model', help='MLflow model name')
    parser.add_argument('--model-tag', type=str, default = '1.0.0',help='MLflow model version')
    parser.add_argument('--bento-ml-path', type=str, default = './bentofile.yaml', help='Path bentoml file')
    opt = parser.parse_args()
    subprocess.run(["chmod", "+x", "create_docker.sh"])
    subprocess.call(shlex.split(f"./create_docker.sh %s %s %s"%(opt.model_name, opt.model_tag, opt.bento_ml_path))
```

Figura 16 – Esempio di codice in Python che permette la generazione dell'immagine Docker del modello

```bash
#!/bin/bash
MODEL_NAME=$1
MODEL_TAG=$2
BENTOFILE=$3


bentoml build -f $BENTOFILE --version $MODEL_TAG --verbose


bentoml list | grep $MODEL_NAME:$MODEL_TAG


bentoml containerize $MODEL_NAME:$MODEL_TAG


docker images | grep $MODEL_NAME:$MODEL_TAG
```

Figura 17 – Esempio di codice Bash che permette la generazione dell'immagine Docker del modello

### 3.4.4 Serving del modello e testing attraverso Swagger API

Infine, BentoML fornisce un'interfaccia Swagger API che facilita il testing e l'interazione con il modello tramite richieste API. L'interfaccia Swagger API offre una documentazione chiara delle API del modello, consentendo agli sviluppatori e agli utenti di comprendere facilmente come interagire con il modello per ottenere predizioni accurate e attendibili.

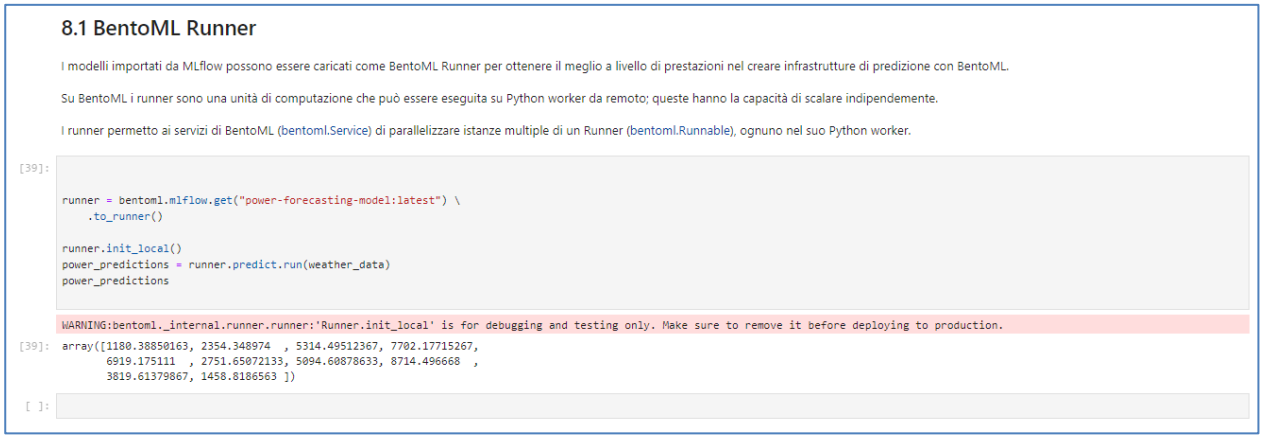
### 3.4.5 BentoML Runner

In BentoML, Runner è responsabile dell'esecuzione di calcoli su worker Python remoti e ha la capacità di scalare in modo indipendente.

Il Runner consente a "*bentoml.Service*" di parallelizzare diverse istanze di una classe "*bentoml.Runnable*", ognuna eseguita su un worker Python separato. Quando viene avviato un BentoServer, un gruppo di processi worker runner viene creato e le chiamate al metodo run effettuate dal codice di bentoml.Service vengono distribuite tra questi worker runner.

Il Runner supporta anche il batching adattivo. Per una classe "*bentoml.Runnable*" configurata per il batching, diverse chiamate al metodo run provenienti da altri processi possono essere raggruppate dinamicamente in un'unica esecuzione batch in tempo reale. Questo è

particolarmente vantaggioso per carichi di lavoro intensi come l'inferenza dei modelli e contribuisce a ottenere prestazioni migliori attraverso tecniche come la vettorizzazione o il multithreading.



### 8.1 BentoML Runner

I modelli importati da MLflow possono essere caricati come BentoML Runner per ottenere il meglio a livello di prestazioni nel creare infrastrutture di predizione con BentoML.

Su BentoML i runner sono una unità di computazione che può essere eseguita su Python worker da remoto; queste hanno la capacità di scalare indipendemente.

I runner permetto ai servizi di BentoML (bentoml.Service) di parallelizzare istanze multiple di un Runner (bentoml.Runnable), ognuno nel suo Python worker.

[39]:
```
runner = bentoml.mlflow.get("power-forecasting-model:latest") \
    .to_runner()

runner.init_local()
power_predictions = runner.predict.run(weather_data)
power_predictions
```

```
WARNING:bentoml._internal.runner.runner:'Runner.init_local' is for debugging and testing only. Make sure to remove it before deploying to production.
```

[39]:
```
array([1180.38850163, 2354.348974  , 5314.49512367, 7702.17715267,
       6919.175111  , 2751.65072133, 5094.60878633, 8714.496668  ,
       3819.61379867, 1458.8186563 ])
```

[ ]:

Figura 18 – Esempio in ambiente Jupyter Notebook dell'utilizzo di BentoML Runner

# 4 PaaS AI – Semantic Knowledge Search

## 4.1 Dashboard Principale

La dashboard principale è presente in Figura 19.

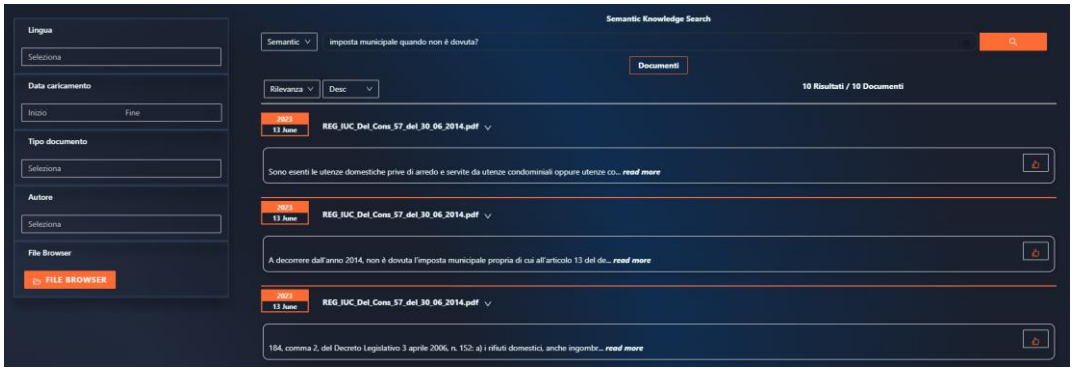



Figura 19 - Dashboard Principale

L'utente avrà a disposizione la barra di ricerca sulla destra, nella quale potrà inserire la query in linguaggio naturale. L'icona della lente di ingrandimento permette di avviare la ricerca.
Sulla sinistra sono presenti i filtri di ricerca.
I filtri disponibili sono:

- Lingua
- Data caricamento
- Tipo documento
- Autore

Inoltre, c'è anche la possibilità di caricare file tramite File Browser.

## 4.2 Ricerca semantica e ricerca sintattica

Prima di effettuare una ricerca, l'utente ha la possibilità di scegliere se effettuare una ricerca semantica o sintattica (per parola chiave). È disponibile sulla sinistra un menù a tendina che ne permette la scelta (Figura 20).

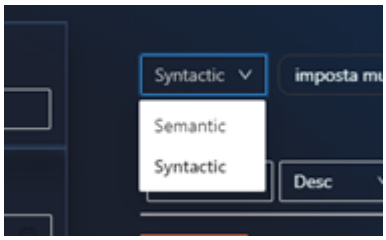



Figura 20 – *Scelta tra Semantic e Syntactic Search*

La ricerca semantica è specializzata sul contenuto dei documenti ricercati, mentre la ricerca sintattica è una ricerca per parola chiave. È consigliato l'utilizzo di entrambe le ricerche qualora non si trovino subito le informazioni necessarie.
Per avviare la ricerca occorre scrivere la query nella "search bar" e cliccare sulla lente d'ingrandimento.
In Figura 21 è presente un esempio di query in linguaggio naturale.

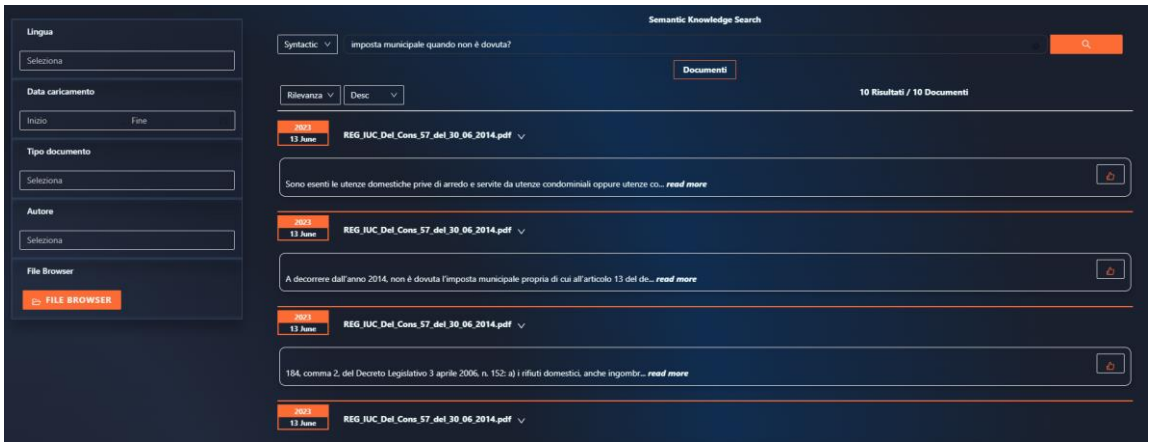



Figura 21 – *Esempio di ricerca semantica*

## *4.3 Filtri di ricerca*

Sulla sinistra sono presenti dei componenti per inserire dei filtri all'interno della ricerca.
I filtri da inserire sono selezionabili una volta che si clicca sull'apposito riquadro all'interno del componente del filtro (Figura 22).

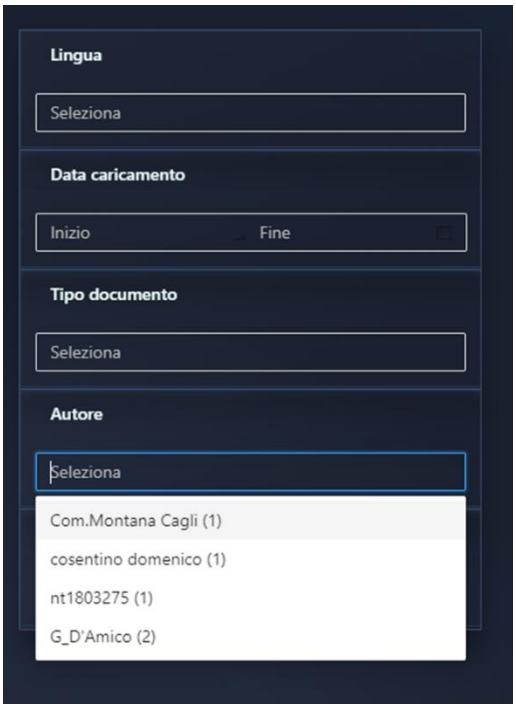



Figura 22 – Filtri disponibili

## 4.4 Visualizzazione risultati

I risultati della ricerca vengono presentati in un elenco sotto la barra di ricerca (Figura 21). A sinistra di ogni risultato viene mostrata la data di chiusura del documento e accanto ad essa il suo oggetto. Sotto l'oggetto è presente l'inizio del testo del contenuto e infine è presente un pulsante con un pollice in alto che consente all'utente di fornire un "feedback" positivo sui documenti più rilevanti per la ricerca (Figura 23 e Figura 24).
È inoltre possibile modificare l'ordine dei risultati in base alla rilevanza, alla data o al feedback fornito dagli utenti, in ordine ascendente o discendente (Figura 25).
Cliccando su un Ticket è possibile visualizzare i dettagli del risultato trovato (Figura 27).

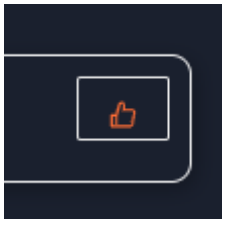


Figura 23 – Pulsante di feedback positivo sul risultato della query di ricerca

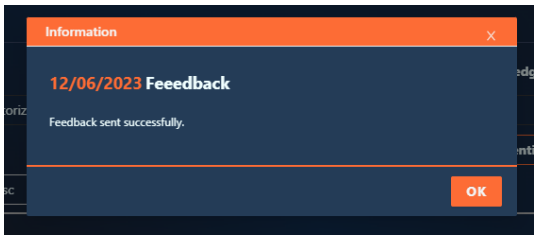



Figura 24 – Processo di Feedback. Questa schermata si apre una volta che viene correttamente fornito il feedback positivo sul risultato

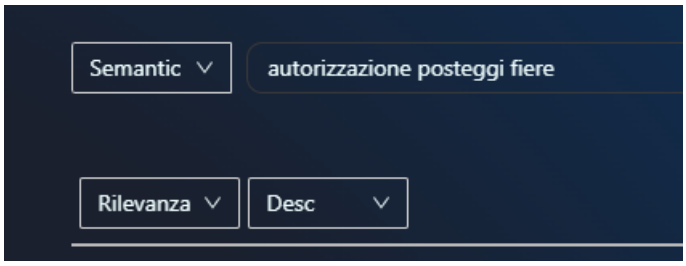



Figura 25 - Scelta dell'ordine dei risultati in base alla rilevanza, al feedback degli utenti e alla data, in ordine ascendente o discendente

Inoltre, cliccando sulla freccia in Figura 26 è possibile visualizzare i metadati dei risultati.


Figura 26 – Visualizzazione metadati dei risultati

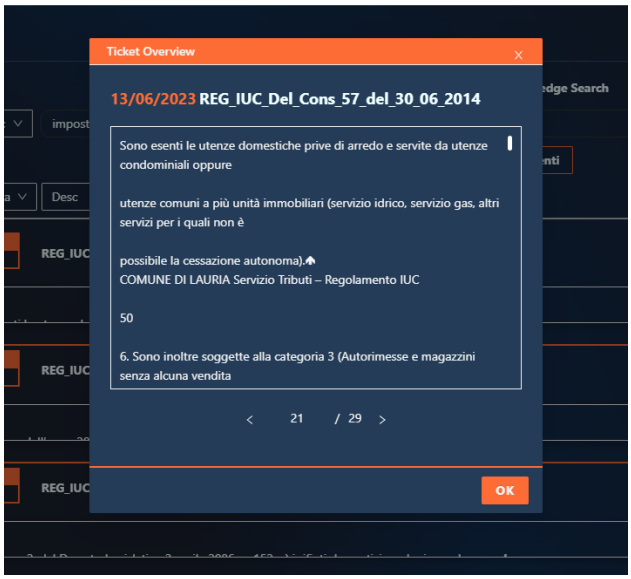



Figura 27 – Ticket overview

## 4.5 Upload di file per aggiungerli alla piattaforma

Gli utenti amministratori hanno la possibilità di caricare un file di tipo docx o pdf tramite l'apposito usando la funzionalità di upload del file. La funzionalità di upload permette di indicizzare il testo a partire da un file applicando tecniche di pre-processing presenti nel campo del Natural Language Model o NLP.
Il bottone per aggiungere un file si trova nella dashboard a sinistra (Figura 28).

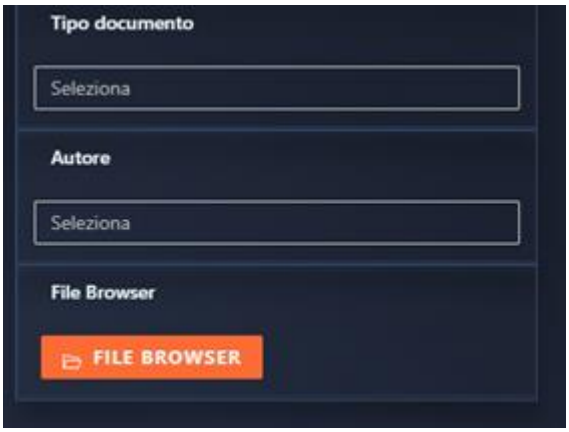



Figura 28 - File Browser per l'upload di un file

Il sistema permetterà di caricare un file presente sul proprio pc (Figura 29). Avviato il caricamento, è presente una schermata di "Loading" fino a che il processo è terminato (Figura 30 e Figura 31).
Infine è possibile visualizzare la lista dei file indicizzati nel DB semantico (Figura 32).

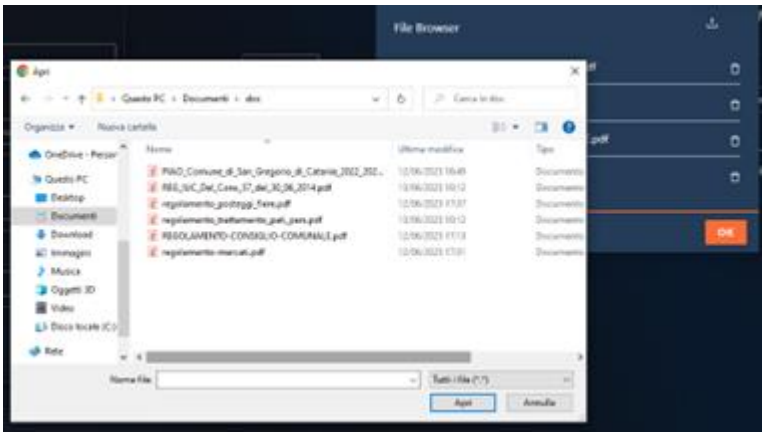



Figura 29 – Scelta di un file locale nel computer

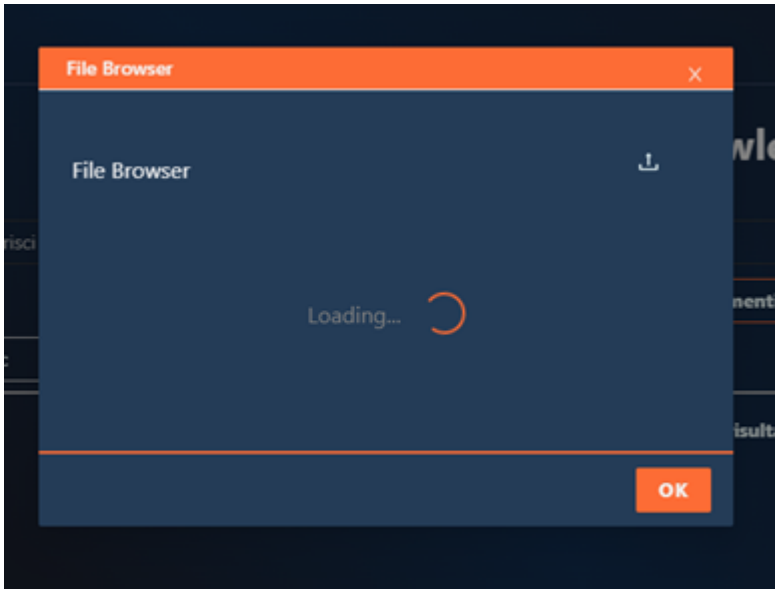



Figura 30 – Schermata di caricamento di un file all'interno dell'applicativo

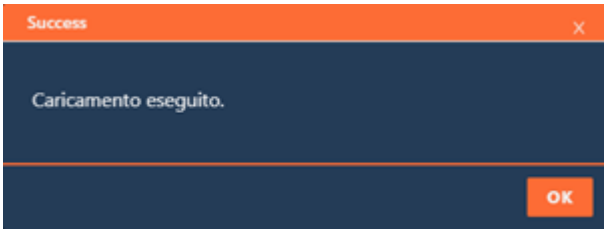



Figura 31 – Fine del processo di upload, file indicizzato


Figura 32 – Lista dei file caricati e indicizzati nel BD Semantico

# 5 PaaS AI – Text Analytics

## 5.1 Key Phrases Extraction

Il servizio di Key Phrases Exraction offre una chiamata API di tipo POST per estrarre le parole chiavi da un testo.

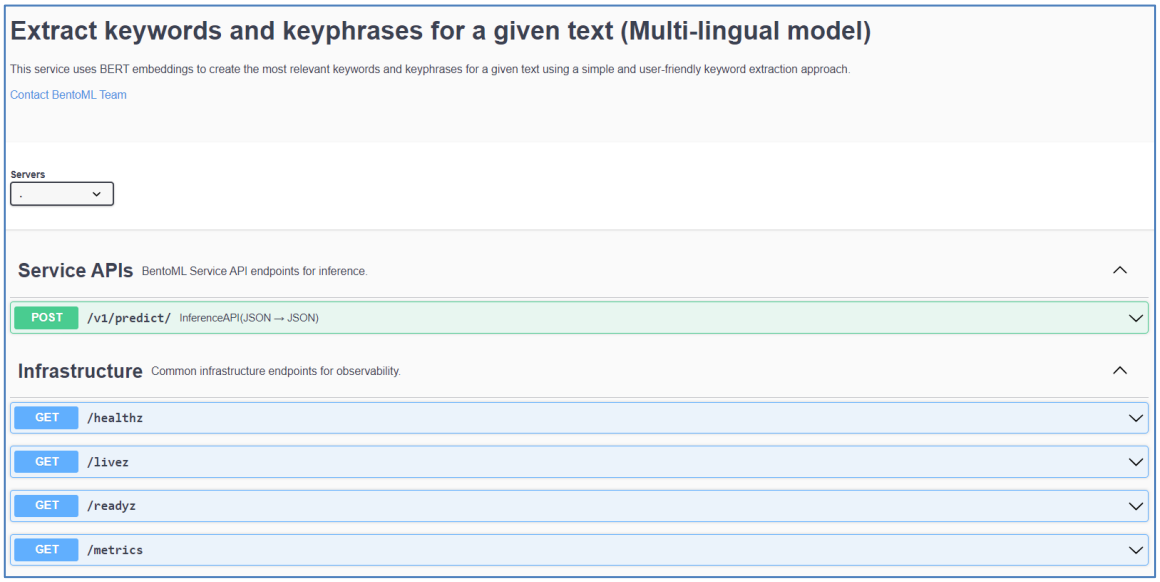



Figura 33 – Swagger UI del servizio di Key Phrases Extraction

La chiamata di inferenza sulla lingua è "/v1/predict" (Figura 33). L'utente deve cliccare sulla freccia per aprire la finestra, inserire nel body la richiesta ed infine cliccare su "Execute" (Figura 34).

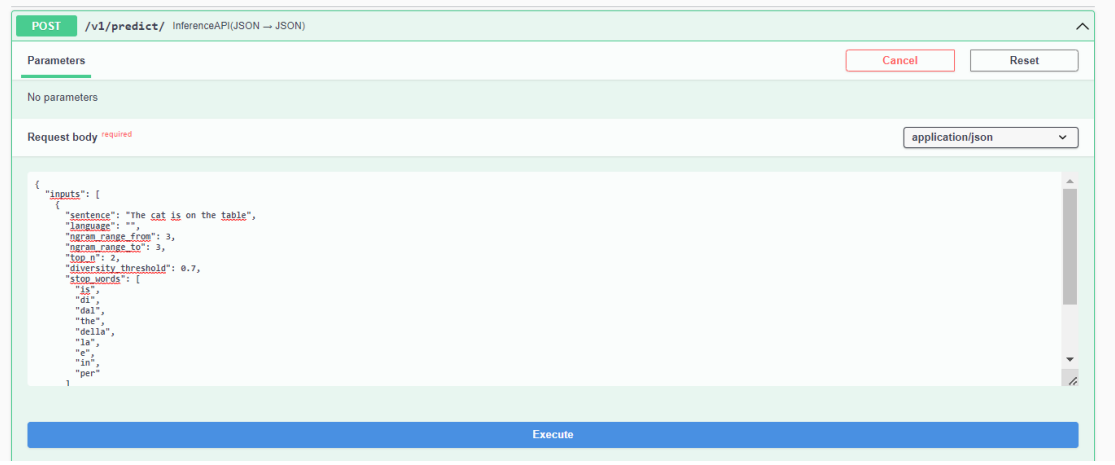



Figura 34 - Swagger UI: dettaglio della chiamata di inferenza

Una volta effettuata la chiamata, sarà disponibile la risposta dell'inferenza (Figura 35).


Figura 35 – Risposta di inferenza del servizio di Key Phrases Extraction

## 5.2 Language Detection

Il servizio di Language Detection offre una chiamata API di tipo POST per effettuare inferenza sulla lingua (Figura 36).

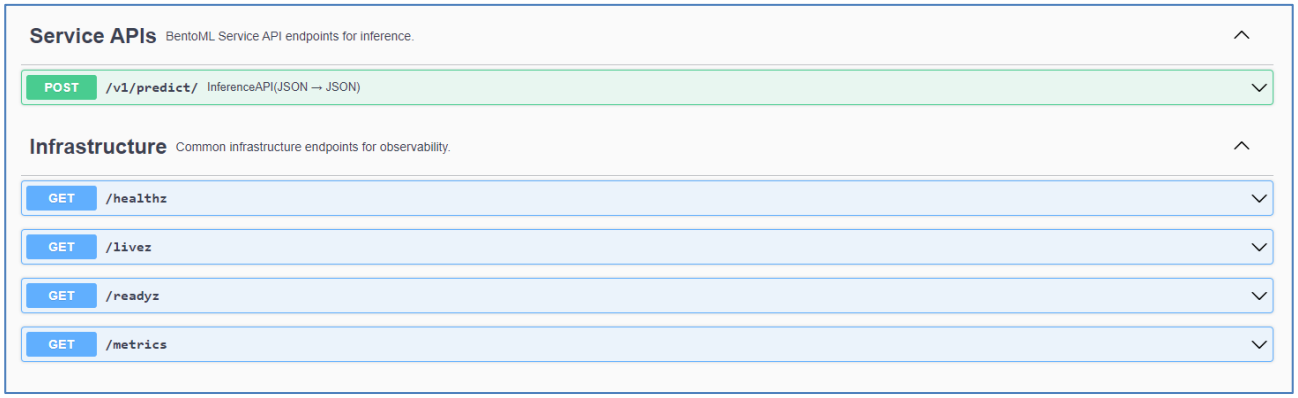



Figura 36 – Swagger UI del servizio di Language Detection

La chiamata di inferenza sulla lingua è "/v1/predict". L'utente deve cliccare sulla freccia per aprire la finestra, inserire nel body la richiesta ed infine cliccare su "Execute" (Figura 37).

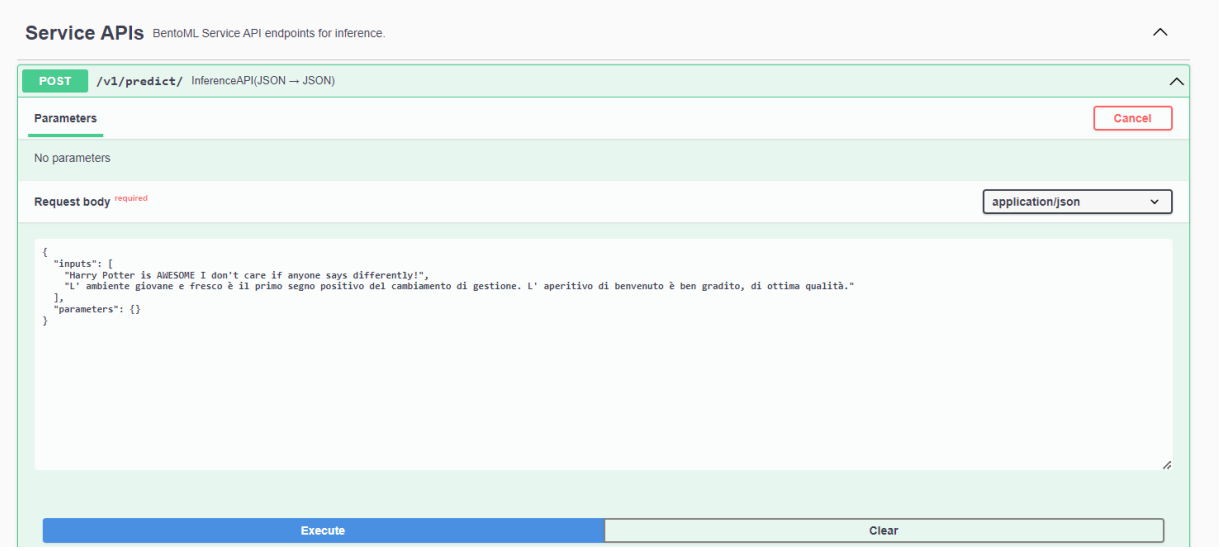



Figura 37 – Swagger UI: dettaglio della chiamata di inferenza "Language Detection"

Una volta effettuata la chiamata, sarà disponibile la risposta dell'inferenza (Figura 38).


Figura 38 – Risposta di inferenza del servizio di Language Detection

## 5.3 Sentiment Analysis

Il servizio di Sentiment Analysis offre una chiamata API di tipo POST per effettuare inferenza sulla lingua (Figura 39).

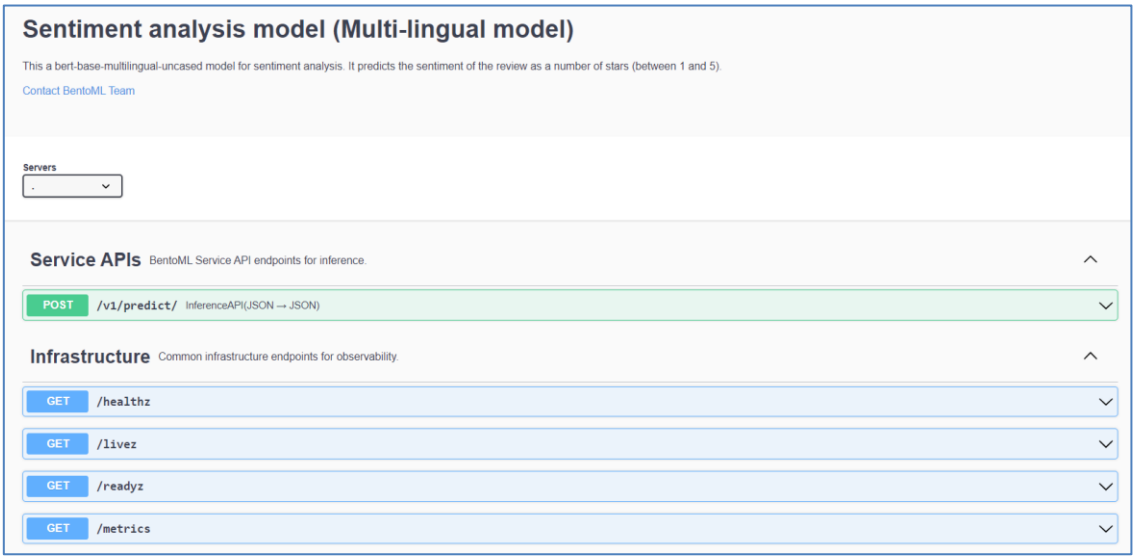



Figura 39 – Swagger UI del servizio Sentiment Analysis

La chiamata di inferenza sulla lingua è "/v1/predict". L'utente deve cliccare sulla freccia per aprire la finestra, inserire nel body la richiesta ed infine cliccare su "Execute" (Figura 40).

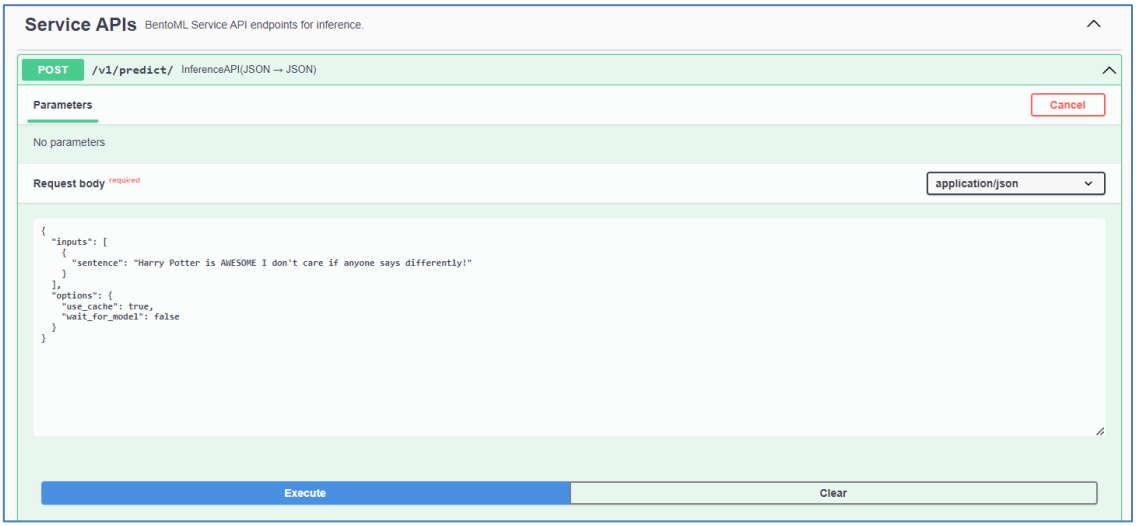



Figura 40 - Swagger UI: dettaglio della chiamata di inferenza di Sentiment Analysis

Una volta effettuata la chiamata, sarà disponibile la risposta dell'inferenza (Figura 41).

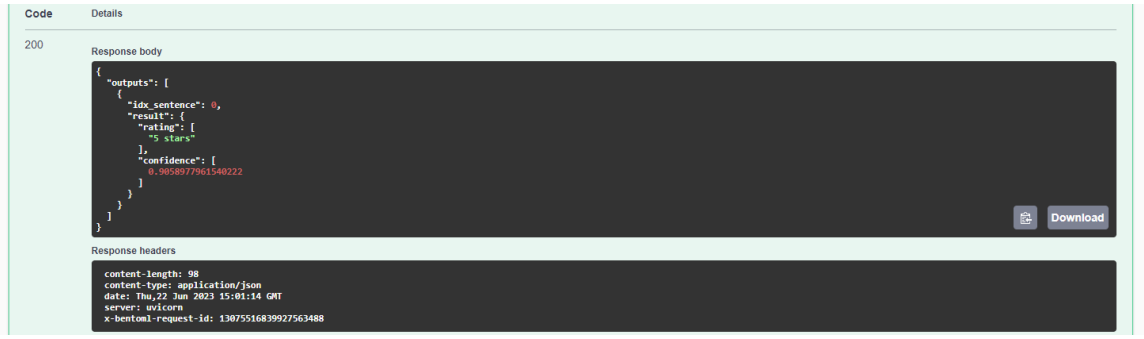



Figura 41 – Risposta di inferenza del servizio di Sentiment Analysis

## 5.4 Named Entity Recognition

Il servizio di Named Entity Recognition offre una chiamata API di tipo POST (Figura 42).

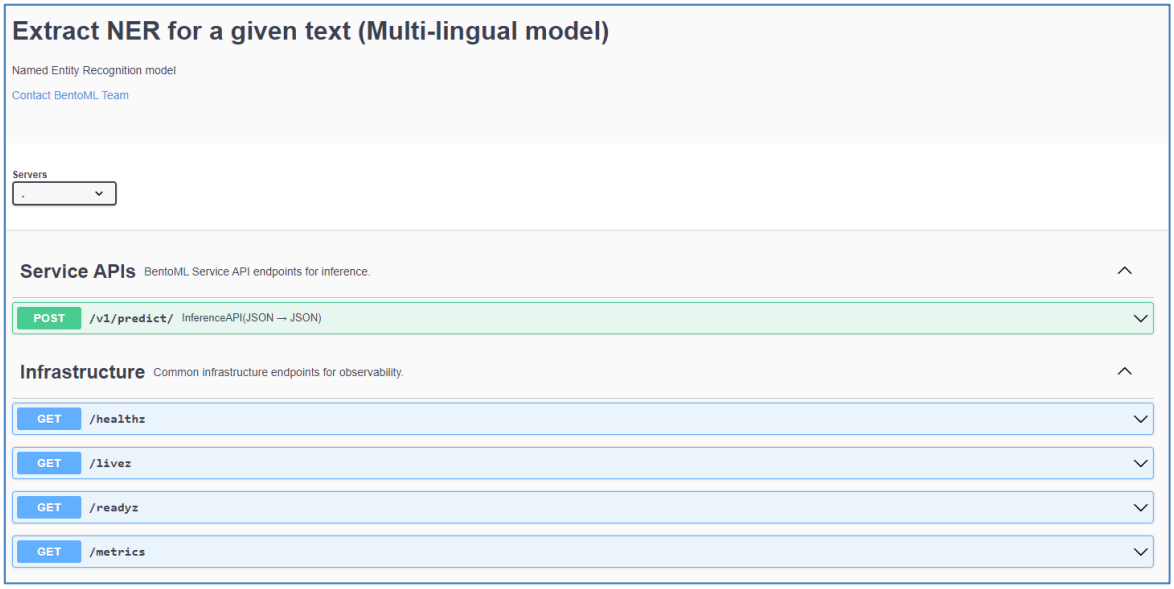



Figura 42 - Swagger UI del servizio di Named Entity Recognition

La chiamata di inferenza sulla lingua è "/v1/predict". L'utente deve cliccare sulla freccia per aprire la finestra, inserire nel body la richiesta ed infine cliccare su "Execute" (Figura 43).

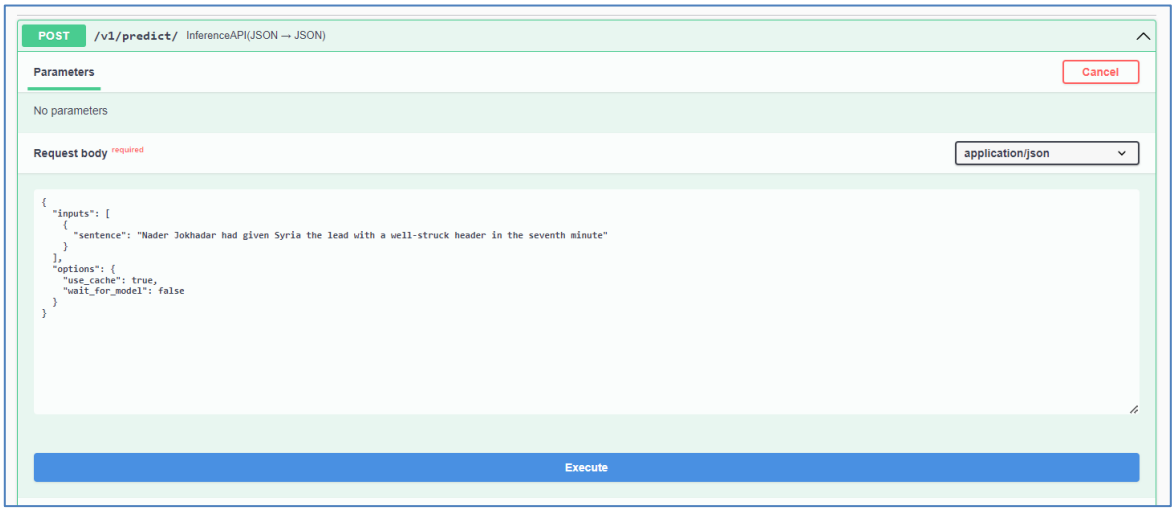



Figura 43 – Swagger UI: dettaglio della chiamata di inferenza di Sentiment Analysis

Una volta effettuata la chiamata, sarà disponibile la risposta dell'inferenza (Figura 44).

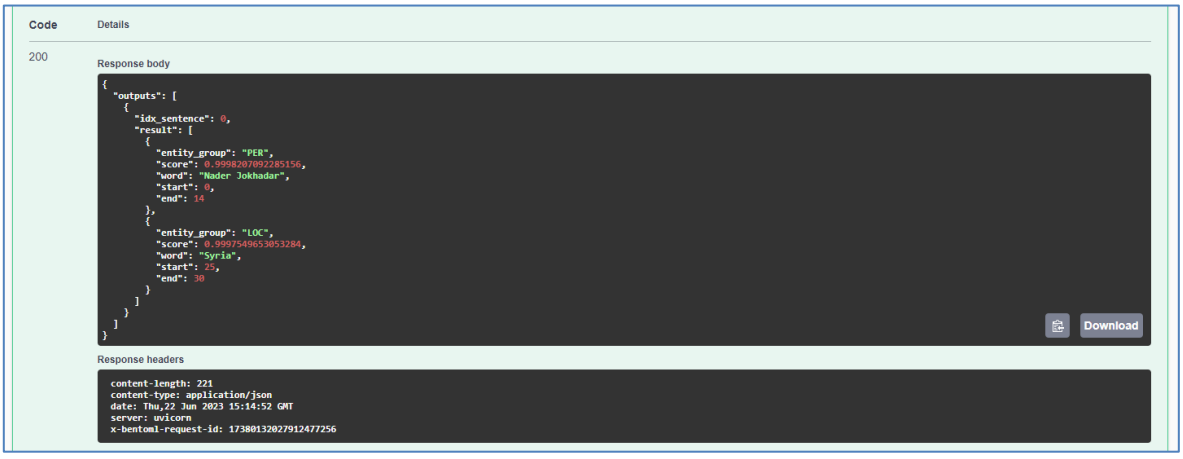



Figura 44 - Risposta di inferenza del servizio di Sentiment Analysis

# 6 PaaS AI – Audio Analytics

## 6.1 *Environment Classification*

Il servizio di Environment Classification offre due chiamate API che sono raffigurate in Figura 45:
Una chiamata "/api" di tipo "GET" che permette di ottenere lo stato del servizio (Figura 46).
Una chiamata "/api/audio_environment_classification/inference" di tipo "POST" con cui è possibile inviare un file al servizio ed effettuare inferenza tramite il modello di AI (Figura 47).

Chiamata "/api":

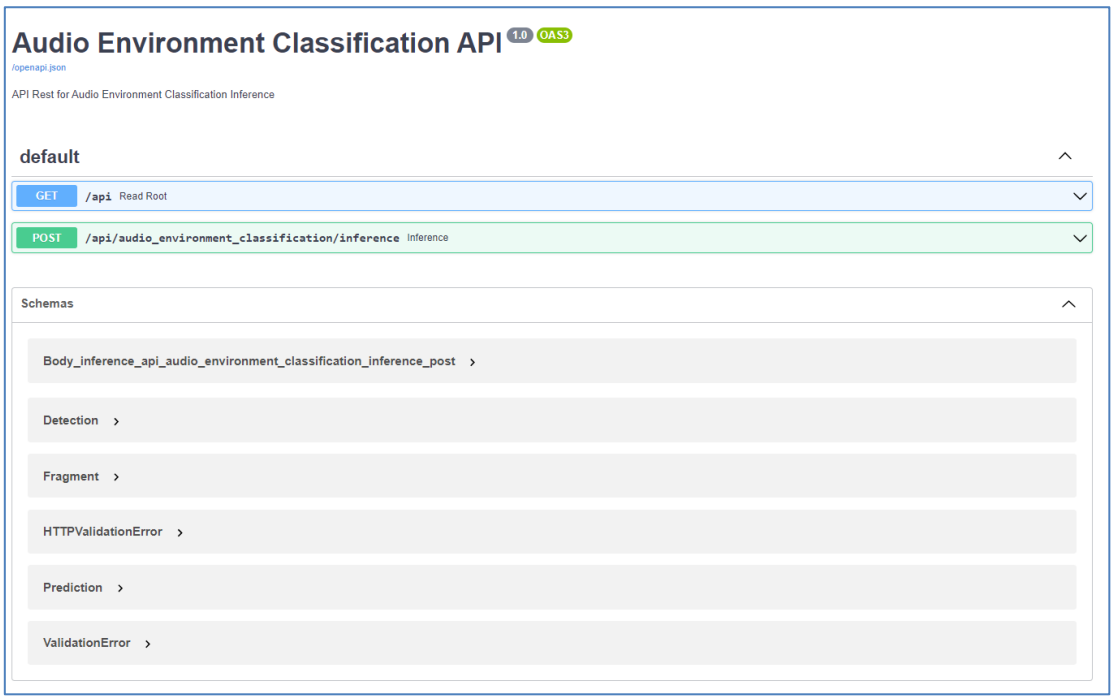



Figura 45 – UI di Swagger API per il servizio Audio Environment Classification API

La chiamata "*/api*" (Figura 46) fornisce informazioni sullo stati del servizio.

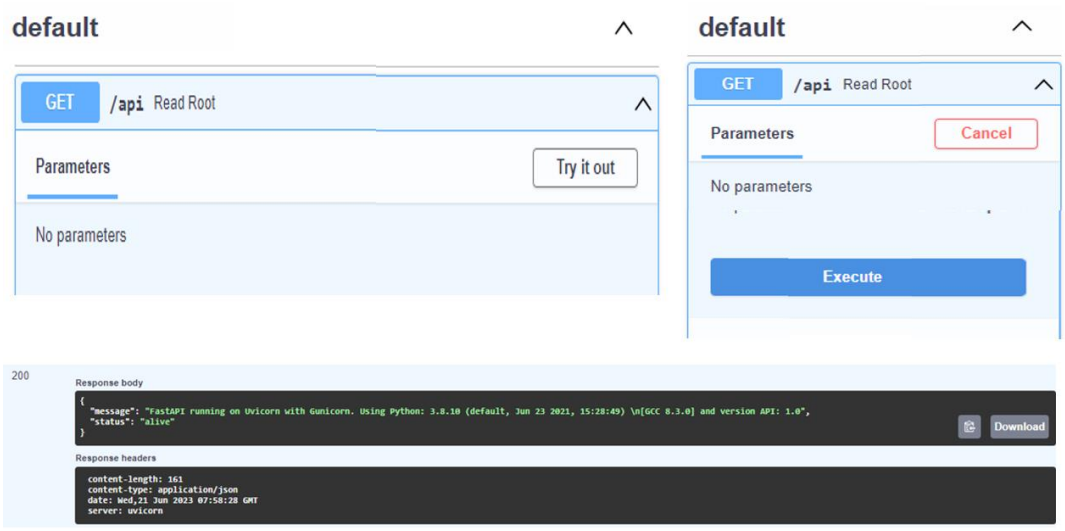



Figura 46 – Dettaglio di come eseguire una chiamata API. Esempio eseguito sulla chiamata GET "/api". In questo caso il sistema risponde con dettagli sull'architettura del servizio e sul suo status.

Chiamata *"/api/audio_environment_classification/inference"*:
La chiamata *"/api/audio_environment_classification/inference"* è di tipo post. Una volta cliccato su "*Try it Out*", analogamente a quanto descritto in Figura 45, è possibile inserire un file audio per cui si vuole effettuare inferenza. Nell'esempio mostrato in Figura 47 c'è l'esempio della risposta.


Figura 47 – Chiamata *POST* per effettuare inferenza.

## 6.2 Anomaly Detection

Training:
Il primo servizio di Audio Anomaly Detection è un servizio di Training del modello. Le chiamate disponibili sono raffigurate in Figura 48.

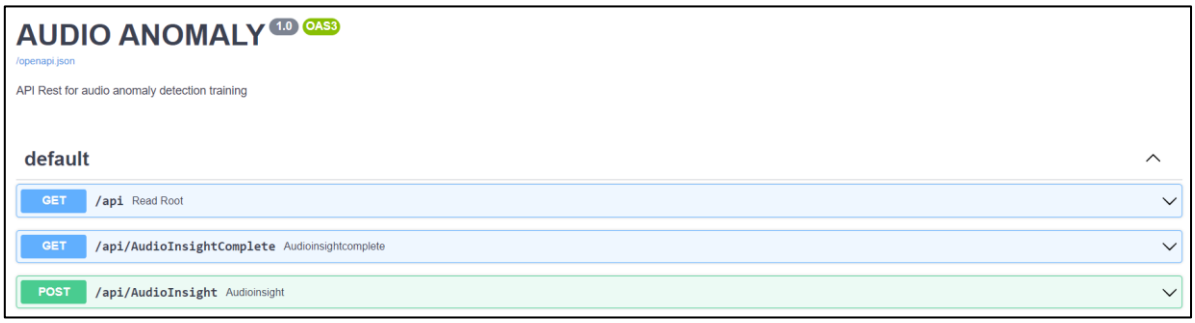



Figura 48 – Chiamate API per il servizio di Audio Anomaly Detection Training

La prima chiamata "*/api*" è analoga alla chiamata descritta in Figura 46.
Per effettuare il traning vero e proprio sul modello, occorre effettuare la POST "*/api/AudioInsight*".
Per visualizzare lo stato del training occorre effettuare la GET "*/api/AudioInsightComplete*".
I parametri da inserire nella POST "*/api/AudioInsight*" sono rappresentati in Figura 49 e sono un ID di cui occorre tenere traccia e i file su cui effettuare l'addestramento del modello. Cliccando poi su "*Execute*" è possibile avviare il training. Il Json in risposta avrà il parametro: "TRAINING_ID" (Figura 50).
Il parametro "TRAINING_ID" è il parametro da passare alla chiamata "*api/AudioInsightComplete*" per otteenere lo status del training (Figura 51). L'esempio di risposta di questa chiamata è presente in Figura 52.

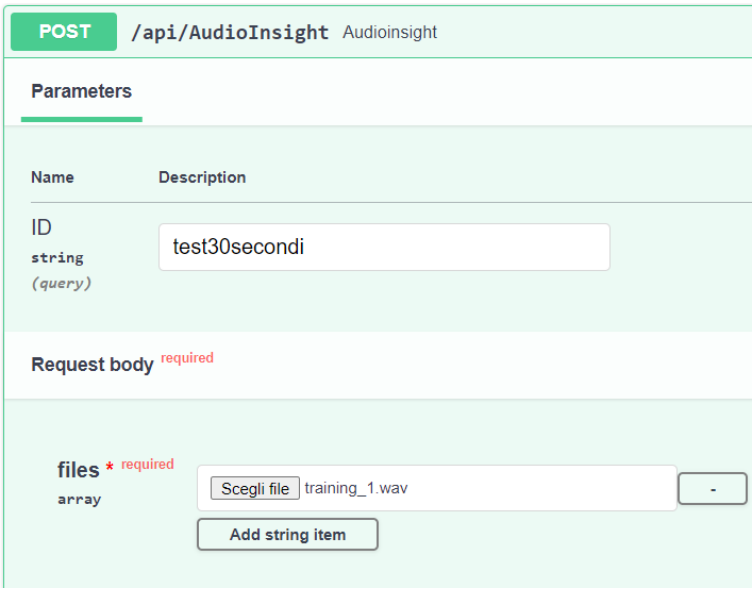



Figura 49 – Parametri da inserire nella chiamata "*api/AudioInsight*"


Figura 50 – Json di risposta della chiamata "*api/AudioInsight*". *Si prenda nota del parametro "TRAINING_ID".*

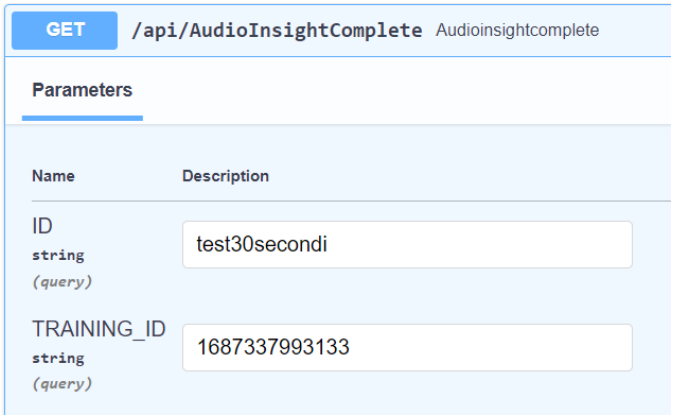



Figura 51 – Parametri da inserire nella chiamata "*api/AudioInsightComplete*"

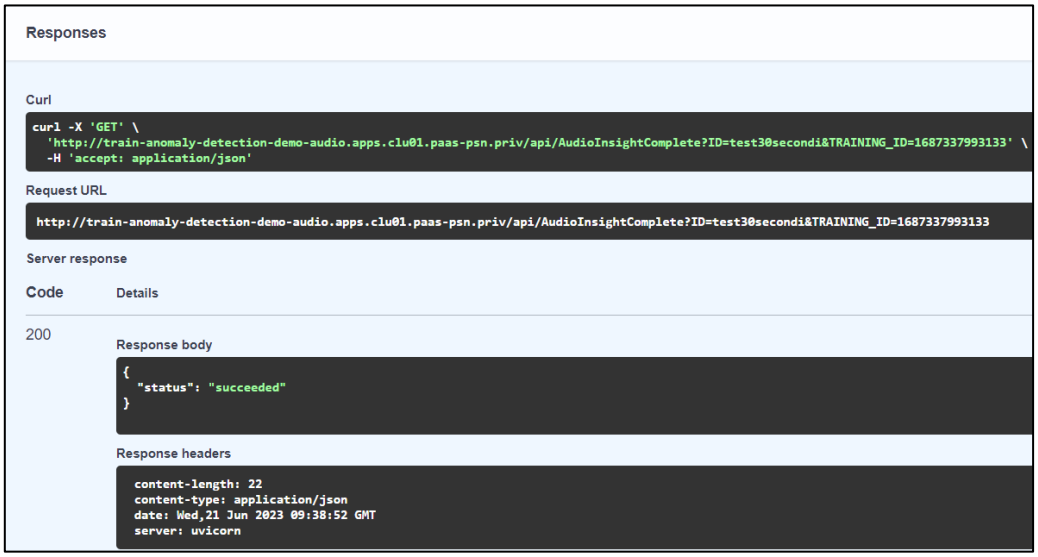



Figura 52 – Esempio di risposta alla chiamata "*Api/AudioInsightComplete*".

Inferencing:

L'altro servizio di Audio Anomaly Detection è un servizio di Inferencing del modello. Le chiamate disponibili sono raffigurate in Figura 53.
Le chiamate disponibili sono:

- *"/api":* questa chiamata è analoga alla chiamata descritta in Figura 46 e restituisce lo status ed i dettagli del servizio.
- *"/api/Anomaly"*

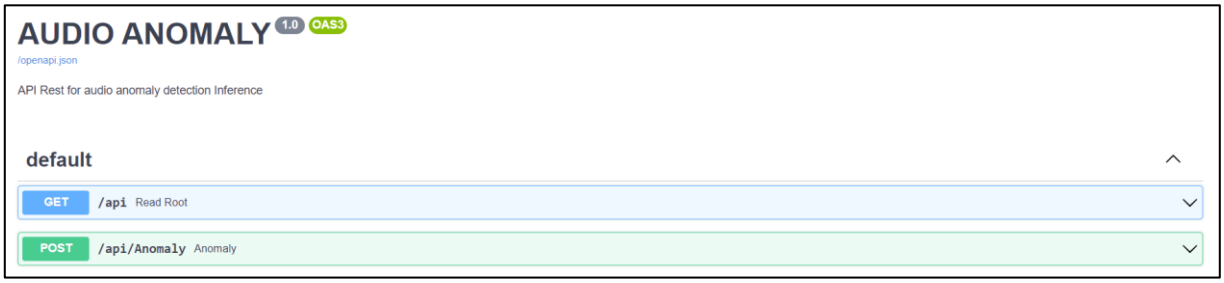



Figura 53 – Chiamate API per Audio Anomaly Detection Inference

Per eseguire la chiamata "*/api/Anomaly*" occorre inserire dei parametri, come in Figura 54. L'esempio di risposta della chiamata è rappresentato in Figura 55.

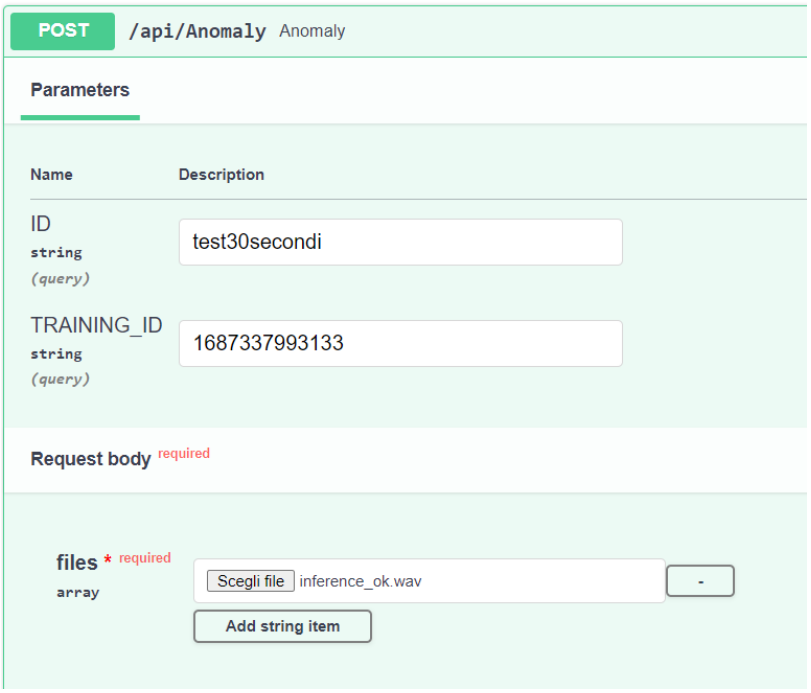



Figura 54 – Parametri da inserire per eseguire la chiamata "*/api/Anomaly*"

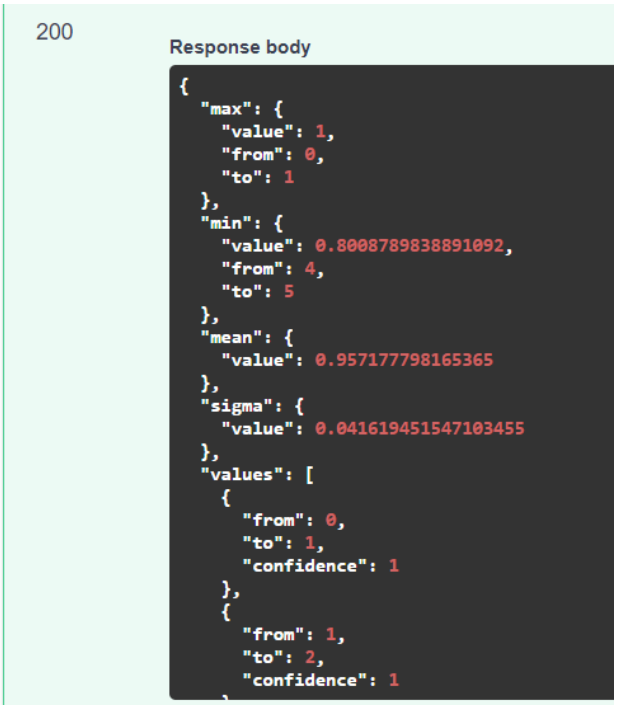



Figura 55 – Esempio di JSON di risposta della chiamata di infer

## 6.3 *Speaker Identification*

Training:
Le chiamate del servizio disponibili per il traning sono (Figura 56):

- "*/api*": questa chiamata è analoga alla chiamata descritta in Figura 46 e restituisce lo status ed i dettagli del servizio.
- *"/api/speaker_identification/training"*
- *"/api/speaker_identification/training/status":* chiamata che restituisce lo status del training.

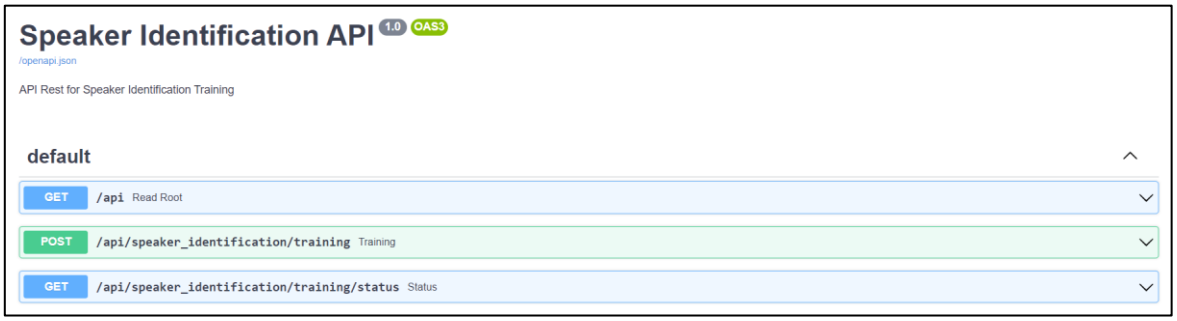



Figura 56 - Chiamate API per Speaker Identification Training

Per eseguire il training del modello occorre eseguire la chiamata "*/api/speaker_identification/training*" e inserire i parametri come specificato in Figura 57. Nella Figura 57 vengono passati i parametri "*subject*" ed il file audio di riferimento. La "*response body*" restituisce un "TRAINING ID".


Figura 57 – Parametri da inserire per eseguire la chiamata "*/api/speaker_identification/training*"

Inferencing:
Le chiamate del servizio disponibili per il traning sono (Figura 56):

- "*/api*": questa chiamata è analoga alla chiamata descritta in Figura 46 e restituisce lo status ed i dettagli del servizio.
- *"/api/speaker_identification/inference"*

La chiamata *"/api/speaker_identification/inference"* permette di eseguire inferenza (Figura 59). È possibile passare un "*training ID*" (si è passato il "*Training ID*" ottenuto dalla chiamata eseguita in Figura 57).
La risposta della chiamata è presente in Figura 60.

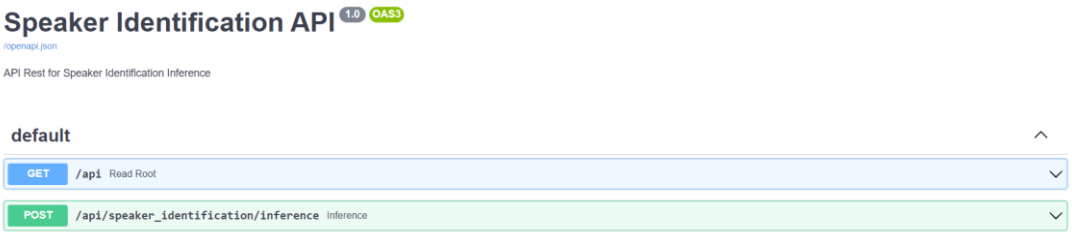



Figura 58 – Chiamate API per Speaker Identification Inferencing

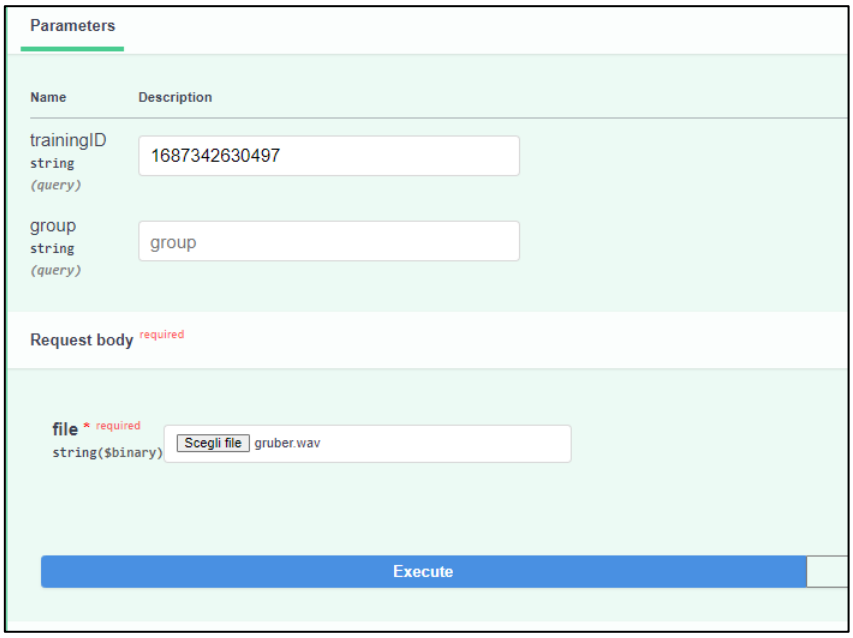



Figura 59 – Parametri da passare nella chiamata /api/speaker_identification/inference


Figura 60 – *Response Body* della chiamata /api/speaker_identification/inference