

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 Al

Data: 23/06/2023

PSN_Manuale Utente PaaS AI

Ed. 1 - ver. 1.0



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



INDICE

1		Definizioni e Acronimi	9
	1.1	DEFINIZIONI	9
	1.2	ACRONIMI	9
2		Executive Summary	12
	2.1	Scopo del documento	12
3		PaaS AI – AI Platform	13
	3.1	Jupyter Notebook	13
	3.2	MLFLOW	15
		3.2.1 Experiments 3.2.2 Models	16 17
	3.3	MinIO	18
	3.4	BentoML	21
		3.4.1 Testing del modello importato da MLflow	21
		3.4.2 Build delle API su BentoML	22
		3.4.4 Serving del modello e testing attraverso Swagaer API	22
		3.4.5 BentoML Runner	23
4		PaaS AI – Semantic Knowledge Search	25
	4.1	Dashboard Principale	25
	4.2	RICERCA SEMANTICA E RICERCA SINTATTICA	25
	4.3	FILTRI DI RICERCA	26
	4.4	VISUALIZZAZIONE RISULTATI	27
	4.5	Upload di file per aggiungerli alla piattaforma	29
5		PaaS AI – Text Analytics	32
	5.1	Key Phrases Extraction	32
	5.2	Language Detection	33
	5.3	SENTIMENT ANALYSIS	34



	5.4	Named Entity Recognition	
6	F	PaaS AI – Audio Analytics	
	6.1	Environment Classification	
	6.2	Anomaly Detection	
	6.3	SPEAKER IDENTIFICATION	



LISTA DELLE FIGURE

Figura 2 – Dettaglio della Navbar laterale e delle sue quattro schermate disponibili: File Browser, Running Terminals and Kernels, Table of Contents, Extension Manager. 14 Figura 3 – Esempio di codice in Python che permetto il monitoraggio del modello con MLflow 15 Figura 4 – Dashboard principale di MLflow: tab "Experiments". 15 Figura 5 – Dettagli di visualizzazione di una esecuzione (run) di un esperimento. 16 Figura 6 – Dettaglio del menù ad espansione Artifacts. 17 Figura 7 – Mlflow: tab "models" 17 Figura 9 – Login di accesso a MinIO 18 Figura 10 – Dashboard principale di MinIO 19 Figura 11 – Visualizzazione del bucket mlflow: sono presenti 3 folder all'interno. Ogni cartella è relativa ad un esperimento. 19 Figura 12 – Visualizzazione degli artefatti all'interno dell'experiment "2" 20 Figura 13 – Visualizzazione degli artefatti all'interno di uno specifico "run" 20 Figura 14 – Esempio di inferenza tramite chiamata REST API 21 Figura 15 – Esempio di Model Testing in ambiente Jupyter Notebook 22 Figura 16 – Esempio di codice in Python che permette la generazione dell'immagine Docker del modello 23
Running Terminals and Kernels, Table of Contents, Extension Manager.14Figura 3 – Esempio di codice in Python che permetto il monitoraggio del modello con MLflow 15Figura 4 – Dashboard principale di MLflow: tab "Experiments"15Figura 5 – Dettagli di visualizzazione di una esecuzione (run) di un esperimento.16Figura 6 – Dettaglio del menù ad espansione Artifacts17Figura 7 – Mlflow: tab "models"17Figura 9 – Login di accesso a MinIO18Figura 10 – Dashboard principale di MinIO19Figura 11 – Visualizzazione del bucket mlflow: sono presenti 3 folder all'interno. Ogni cartella èrelativa ad un esperimento19Figura 12 – Visualizzazione dei run effettuati all'interno dell'experiment "2"20Figura 13 – Visualizzazione deji artefatti all'interno di uno specifico "run"20Figura 14 – Esempio di inferenza tramite chiamata REST API21Figura 16 – Esempio di codice in Python che permette la generazione dell'immagine Docker del23
Figura 3 – Esempio di codice in Python che permetto il monitoraggio del modello con MLflow 15 Figura 4 – Dashboard principale di MLflow: tab "Experiments"
Figura 4 – Dashboard principale di MLflow: tab "Experiments"15Figura 5 – Dettagli di visualizzazione di una esecuzione (run) di un esperimento.16Figura 6 – Dettaglio del menù ad espansione Artifacts.17Figura 7 – Mlflow: tab "models"17Figura 8 – Modello registrato18Figura 9 – Login di accesso a MinIO18Figura 10 – Dashboard principale di MinIO19Figura 11 – Visualizzazione del bucket mlflow: sono presenti 3 folder all'interno. Ogni cartella èrelativa ad un esperimento19Figura 12 – Visualizzazione dei run effettuati all'interno dell'experiment "2"20Figura 13 – Visualizzazione degli artefatti all'interno di uno specifico "run"20Figura 14 – Esempio di inferenza tramite chiamata REST API21Figura 15 – Esempio di codice in Python che permette la generazione dell'immagine Docker del23
Figura 5 – Dettagli di visualizzazione di una esecuzione (run) di un esperimento.16Figura 6 – Dettaglio del menù ad espansione Artifacts17Figura 7 – Mlflow: tab "models"17Figura 8 – Modello registrato18Figura 9 – Login di accesso a MinIO18Figura 10 – Dashboard principale di MinIO19Figura 11 – Visualizzazione del bucket mlflow: sono presenti 3 folder all'interno. Ogni cartella èrelativa ad un esperimento19Figura 12 – Visualizzazione dei run effettuati all'interno dell'experiment "2"20Figura 13 – Visualizzazione degli artefatti all'interno di uno specifico "run"20Figura 14 – Esempio di inferenza tramite chiamata REST API21Figura 15 – Esempio di codice in Python che permette la generazione dell'immagine Docker del23
Figura 6 – Dettaglio del menù ad espansione Artifacts17Figura 7 – Mlflow: tab "models"17Figura 8 – Modello registrato18Figura 9 – Login di accesso a MinIO18Figura 10 – Dashboard principale di MinIO19Figura 11 – Visualizzazione del bucket mlflow: sono presenti 3 folder all'interno. Ogni cartella èrelativa ad un esperimento19Figura 12 – Visualizzazione dei run effettuati all'interno dell'experiment "2"20Figura 13 – Visualizzazione degli artefatti all'interno di uno specifico "run"20Figura 14 – Esempio di inferenza tramite chiamata REST API21Figura 15 – Esempio di Model Testing in ambiente Jupyter Notebook22Figura 16 – Esempio di codice in Python che permette la generazione dell'immagine Docker del23
Figura 7 – Mlflow: tab "models"17Figura 8 – Modello registrato18Figura 9 – Login di accesso a MinIO18Figura 10 – Dashboard principale di MinIO19Figura 11 – Visualizzazione del bucket mlflow: sono presenti 3 folder all'interno. Ogni cartella èrelativa ad un esperimento19Figura 12 – Visualizzazione dei run effettuati all'interno dell'experiment "2"20Figura 13 – Visualizzazione degli artefatti all'interno di uno specifico "run"20Figura 14 – Esempio di inferenza tramite chiamata REST API21Figura 15 – Esempio di Model Testing in ambiente Jupyter Notebook22Figura 16 – Esempio di codice in Python che permette la generazione dell'immagine Docker del23
Figura 8 – Modello registrato18Figura 9 – Login di accesso a MinIO18Figura 10 – Dashboard principale di MinIO19Figura 11 – Visualizzazione del bucket mlflow: sono presenti 3 folder all'interno. Ogni cartella èrelativa ad un esperimento19Figura 12 – Visualizzazione dei run effettuati all'interno dell'experiment "2"20Figura 13 – Visualizzazione degli artefatti all'interno di uno specifico "run"20Figura 14 – Esempio di inferenza tramite chiamata REST API21Figura 15 – Esempio di Model Testing in ambiente Jupyter Notebook22Figura 16 – Esempio di codice in Python che permette la generazione dell'immagine Docker del23
Figura 9 – Login di accesso a MinIO18Figura 10 – Dashboard principale di MinIO19Figura 11 – Visualizzazione del bucket mlflow: sono presenti 3 folder all'interno. Ogni cartella èrelativa ad un esperimento19Figura 12 – Visualizzazione dei run effettuati all'interno dell'experiment "2"20Figura 13 – Visualizzazione degli artefatti all'interno di uno specifico "run"20Figura 14 – Esempio di inferenza tramite chiamata REST API21Figura 15 – Esempio di Model Testing in ambiente Jupyter Notebook22Figura 16 – Esempio di codice in Python che permette la generazione dell'immagine Docker del23
Figura 10 – Dashboard principale di MinIO19Figura 11 – Visualizzazione del bucket mlflow: sono presenti 3 folder all'interno. Ogni cartella èrelativa ad un esperimento19Figura 12 – Visualizzazione dei run effettuati all'interno dell'experiment "2"20Figura 13 – Visualizzazione degli artefatti all'interno di uno specifico "run"20Figura 14 – Esempio di inferenza tramite chiamata REST API21Figura 15 – Esempio di Model Testing in ambiente Jupyter Notebook22Figura 16 – Esempio di codice in Python che permette la generazione dell'immagine Docker del23
Figura 11 – Visualizzazione del bucket mlflow: sono presenti 3 folder all'interno. Ogni cartella èrelativa ad un esperimento19Figura 12 – Visualizzazione dei run effettuati all'interno dell'experiment "2"20Figura 13 – Visualizzazione degli artefatti all'interno di uno specifico "run"20Figura 14 – Esempio di inferenza tramite chiamata REST API21Figura 15 – Esempio di Model Testing in ambiente Jupyter Notebook22Figura 16 – Esempio di codice in Python che permette la generazione dell'immagine Docker del23
relativa ad un <i>esperimento</i>
Figura 12 – Visualizzazione dei run effettuati all'interno dell'experiment "2"
Figura 13 – Visualizzazione degli artefatti all'interno di uno specifico " <i>run</i> "
Figura 14 – Esempio di inferenza tramite chiamata REST API
Figura 15 – Esempio di Model Testing in ambiente Jupyter Notebook
Figura 16 – Esempio di codice in Python che permette la generazione dell'immagine Docker del modello
modello
20
Figura 17 – Esempio di codice Bash che permette la generazione dell'immagine Docker del
modello 23
Figura 18 – Esempio in ambiente Jupyter Notebook dell'utilizzo di BentoML Runner 24
Figure 19 - Dashboard Principale 25
Figura 20 – Scelta tra Semantic e Svntactic Search 26
Figura 21 – Esempio di ricerca semantica
Figura 22 – Filtri disponibili
Figure 22 – Pulsante di feedback positivo sul risultato della query di ricerca 27
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 deali utenti e alla
data in ordine ascendente o discendente 28
Figura 26 – Visualizzazione metadati dei risultati 28
Figure 27 – Ticket overview 29
Figura 28 - File Browser per l'upload di un file
Figura 29 – Scelta di un file locale nel computer 30
Figura 30 – Schermata di caricamento di un file all'interno dell'applicativo 30
Figura 31 – Fine del processo di upload file indicizzato 30
Figura 32 – Lista dei file caricati e indicizzati nel BD Semantico
Figura 33 – Swagger UI del servizio di Key Phrases Extraction
Figura 34 - Swaager UI: dettaalio della chiamata di inferenza
Figura 35 – Risposta di inferenza del servizio di Kev Phrases Extraction 33
Figura 36 – Swagaer UI del servizio di Language Detection
Figura 37 – Swagger UI: dettaglio della chiamata di inferenza "Language Detection"



Figura 38 – Risposta di inferenza del servizio di Language Detection	34
Figura 39 – Swagger UI del servizio Sentiment Analysis	35
Figura 40 - Swagger UI: dettaglio della chiamata di inferenza di Sentiment Analysis	36
Figura 41 – Risposta di inferenza del servizio di Sentiment Analysis	36
Figura 42 - Swagger UI del servizio di Named Entity Recognition	37
Figura 43 – Swagger UI: dettaglio della chiamata di inferenza di Sentiment Analysis	37
Figura 43 – Swagger UI: dettaglio della chiamata di inferenza di Sentiment Analysis	37
Figura 45 – UI di Swagger API per il servizio Audio Environment Classification API	ta GET
Figura 46 – Dettaglio di come eseguire una chiamata API. Esempio eseguito sulla chiama	status.
"/api". In questo caso il sistema risponde con dettagli sull'architettura del servizio e sul suo	30
Figura 47 – Chiamata <i>POST</i> per effettuare inferenza. Figura 48 – Chiamate API per il servizio di Audio Anomaly Detection Training Figura 49 – Parametri da inserire nella chiamata " <i>api/AudioInsight</i> " Figura 50 – Json di risposta della chiamata " <i>api/AudioInsight</i> ". <i>Si prenda nota del para</i> " <i>TRAINING_ID</i> ". Figura 51 – Parametri da inserire nella chiamata " <i>api/AudioInsightComplete</i> " Figura 52 – Esempio di risposta alla chiamata " <i>Api/AudioInsightComplete</i> " Figura 53 – Chiamate API per Audio Anomaly Detection Inference Figura 54 – Parametri da inserire per eseguire la chiamata " <i>Api/Anomaly</i> " Figura 55 – Esempio di JSON di risposta della chiamata di infer Figura 57 – Parametri da inserire per eseguire la chiamata di si fer Figura 57 – Parametri da inserire per eseguire la chiamata " <i>Api/Speaker_identification/tr</i> .	
Figura 58 – Chiamate API per Speaker Identification Inferencing	45
Figura 59 – Parametri da passare nella chiamata /api/speaker_identification/inference	45
Figura 60 – <i>Response Body</i> della chiamata /api/speaker_identification/inference	45

LISTA DELLE TABELLE

Tabella 1: Glossario Definizioni	9
Tabella 2: Glossario Acronimi	11

Definizioni e Acronimi

1.1 Definizioni

Definizione	Descrizione
PSN	È la nuova società che è stata costituita nell'ambito del progetto del Cloud Nazionale
ТВС	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
laaS	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
ТВ	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'Al 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 Al 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 Al utilizzando Mlflow per il monitoraggio e BentoML per il serving:

- Caricamento del dataset
- Definizione del modello di Al 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:





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



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



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".

mlflow 23.1 Experiments Mod	leis			GitHub Docs
Experiments Search Experiments Default demo-test-scrittura-minio	Default Provide Feedback C Experiment ID: 0 Artifact Location: s3://milflow/0 > > Description Edit > >			Share
classification-wind-prediction 🧷 🔒	Time created: All time Chart view Q metrics.rmse < 1 and p Columns Colum		O Fig. Sort: Created v	: C Refresh
	Run Name	Created = Duration Source	Models	
	vaunted-carp-937	⊘ 1 day ago 21.1s □ ipykernel	S power-fore/7	A
	Image:	⊘ 1 day ago 22.3s □ ipykernel	S power-fore/6	
	oppular-koi-995	1 day ago 📃 ipykernel		
	Ill legendary-newt-936	⊘ 1 day ago 19.3s □ ipykernel	S power-fore/5	+
	azzling-wasp-132	⊘ 1 day ago 18.9s □ ipykernel	S power-fore/4	Show more columns (59 total)
	Iearned-kit-401	⊘ 1 day ago 22.3si ipykernel	S power-fore/3	(
	Imple-donkey-963	2 days ago 📃 ipykernel		
	 intelligent-pug-906 	2 days ago 📃 ipykernel		
	35 matching runs	O 2 da 22 7. □ induand	art	-

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

Default → vaunted-carp-937			
Run ID: e09010a13a2841928f476f1086ccc487	Date: 2023-06-14 11:31:53	Source: 🛄 ipykernel_launcher.py	User: jovyan
Duration: 21.1s	Status: FINISHED	Lifecycle Stage: active	
> Description Edit			
> Parameters (17)			
> Metrics (7)			
> Tags (2)			
> 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)



model MLmodel	Full Path:s3://mlflow/0/e09	010a13a2841928f476f1086ccc487/artifacts/mode	el 🕽 📀 power-forecas Registered on 202	ing , v7 3/06/14
값 conda.yaml 값 model.pkl 政 python_env.yaml 과 requirements.txt	MLflow Model The code snippets below	demonstrate how to make predictions using	the logged model. This model is also registered to the model registry.	
metric_info.json	Model schema		Make Predictions	_
	Input and output schema f	or your model. Learn more	Predict on a Spark DataFrame:	0
	Name	Туре	<pre>import mlflow from pyspark.sal.functions import struct. col</pre>	
	Inputs (1)		<pre>logged_model = 'runs:/e09010a13a2841928F476f1086ccc487/model' # Load model as a Spark UDF. Override result_type if the model does not return double values.</pre>	
		Tensor (dtype: float64, shape: [-1,9])	<pre>loaded_model = mlflow.pyfunc.spark_udf(spark, model_uri=logged_model, result_type='double') # Predict on a Spark DataFrame.</pre>	
	Outputs (1)		<pre>df.withColumn('predictions', loaded_model(struct(*map(col, df.columns)))) Predict on a Pandas DataFrame</pre>	a
		Tensor (dtype: float64, shape: [-1])	<pre>import mlflow logged_model = 'runs:/e00010a13a2841928f476f1086ccc487/model'</pre>	
			<pre># Load model as a PyFuncHodel. loaded_model = mlflow.pyfunc.load_model(logged_model)</pre>	
			<pre># Predict on a Pandas DataFrame. import pandas as pd loaded model.predict(od.DataFrame(data))</pre>	

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.

mlflow 2.3.1 Experiments Models					GitHub Docs
Registered Models					
Share and manage machine learning models. Learn more					×
Create Model				Q Search by model names or tags	Search Clear
Name	Latest Version	Staging	Production	Last Modified	≑ Tags
power-forecasting-model	Version 12	Version 11	Version 12	2023-06-15 18:06:45	-
					1 10 / page V

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.



Peoistered Models >				
power-forecasting-model				:
Created Time: 2023-06-14 10:29:05	Last Modified: 2023-06-15 18:06:45			
✓ Description Edit				
This model forecasts the power output of a wind fam The weather data consists of three features: * wind speed * wind direction * air temperature.	m based on weather data.			
Versions All Active 9 Compare				
Version	Registered at	Created by	Stage	Description
Version 12	2023-06-15 18:06:20		Production	
Version 11	2023-06-15 18:04:52		Staging	
Version 10	2023-06-15 17:56:40		Archived	
Version 9	2023-06-15 17:36:07		Production	
Version 8	2023-06-15 17:20:54		Production	





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).

OBJECT ST	ORE
2 Username	
Password Login	0
Other Authentication Methods	~
Documentation GitHub Suppor	rt Download

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.



MINK	2					
OB	JECT STORE	Object Browser				
User		Filter Buckete				0
	Object Browser	File Buckets				u 0
		Name	Objects	Size	Access	
		The milliow	213	272.3 MiB	R/W	
Adm	inistrator					
۲						
٢	Settings					



<	mlflow		Create new path ://
	▲ Name	Last Modified	Size
	0		-
	i 1		-
	2		-

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.



Object	t Browser	Start typing to filter objects in the bucket	Q	
	mifiow Created on: Wed, Jun 07 2023 06:47:37 (GMT+2)	Access: PRIVATE 272.3 MiB - 213 Objects	Rewind	ය න Refresh උ Upload 🖞
<	miflow / 0			Create new path ://
	▲ Name	Last Modified		Size
	늘 9c13dc74ff974cdb93d557d3564056d6			
	9db653a0182b499380650a04db3c0bb9			-
	늘 9e88dc32969d497d9483e511a326b133			-
	b80a8568fb0245fb9306b05470eda882			
	bc7604866c0344cdbffb7d1268988b82			
	bde0dc8de4b24f6280ef2dd8f87c8316			-
	늘 c7f9ffc9667641a8b9e7e9bf3899cb08			-
	e09010a13a2841928f476f1086ccc487			-
	늘 f467338975734886bea007c0c80350c4			

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

Imifiow Rewind Refresh Refresh Refresh Refresh Refresh Refresh Created on: Wed, Jun 07 2023 06:47:37 (GMT+2) Access: PRIVATE 272.3 MiB - 213 Objects Refresh Created mifliow / 0 / e09010a13a2841928f476f1086ccc487 / artifacts Imifiow Created Imifiow Created Created Imifiow Imifiow Created Imifiow Imifiow Created Imifiow Imifiow	Şç		Q	objects in the bucket	Start typing to filte	r	ect Browser
Miffiow / 0 / e09010a13a2841928f476f1086ccc487 / artifacts Creat Name Last Modified a estimator.html Wed, Jun 14 2023 11:32 (GMT+2) Med, Jun 14 2023 11:32 (GMT+2) b metric_info.json Wed, Jun 14 2023 11:32 (GMT+2) Med, Jun 14 2023 11:32 (GMT+2)	Upload 🟦	d 🕤 Refresh 🖒	Re	272.3 MiB - 213 Objects	2) Access: PRIVATE	n: Wed, Jun 07 2023 06:47:37 (GMT	miflow Created or
Name Last Modified estimator.html Wed, Jun 14 2023 11:32 (GMT+2) metric_info.json Wed, Jun 14 2023 11:32 (GMT+2)	ate new path ://	Cre			7 / artifacts	/ 0 / e09010a13a2841928f476f1086ccc4	mlflow
Image: Sestimator.html Wed, Jun 14 2023 11:32 (GMT+2) Image: Sestimator.html Wed, Jun 14 2023 11:32 (GMT+2)	Size			Last Modified		ne	🔺 Nam
Wed, Jun 14 2023 11:32 (GMT+2)	4.8 KiB		(GMT+2)	Wed, Jun 14 2023 11:		imator.html	😽 esti
	86.0 B		(GMT+2)	Wed, Jun 14 2023 11:		tric_info.json	net 📄
model	-					del	i moc

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



POST	/classify InferenceAPI(JSON → NumpyNdarray)	
Paramete	rs	Canc
No parame	iters	
Request b	pody required	application/json
{ "temper "wind d "winds; "temper "wind d "wind_s; } }	ature_00": 6.01. irection 00": 112.01. peed_00": 7.05. irection 00": 7.01. irection 00": 7.01. irection 10": 112.01. peed_10": 7.01	
	Execute	Clear
Response	15	
<pre>curl -X 'P 'http:// -H'acce -H'Cont -d'{ "tempera "wind_di "wind_sp "tempera "wind_di "wind_sp }'</pre>	V057'\ \ Vocalhort:3010/classify' \ \ pt:sp01ication/json'\ \ strre:40":6.01	
http://lo	lL calhost:3010/classify	
Server resp	ponse	
Code	Details	
200	Response body	
	3164.03737906001	Downia
	Response headers	
	context-length: 19 context-lengt: uplication/json date: Won_10 Jun 2023 15:13:21 GMT server: wicorm x-bentom!-request-id: 3780706176307322053	
Responses	server: wicorn x-bentonl-request-id: 3706706176307322053	

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 Polo Strategico Nazionale

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
<pre>bento_model = bentoml.mlflow.import_model(model_name, # 'mlflow_pytorch_mnist', model_uri, signatures=('predict': ('batchable': True)))</pre>
Per effettuare test e valuatizione, potrebbe essere più conveniente l'importazione dei modelli nella loro forma nativa.
<pre># scelta del modello e versione su MLFLow (ultima versione disponibile del modello power-forecasting-model) bento_model = bentonl.mlFLow.get("power-forecasting-model:latest") mlFLow.model_path = bento_model.path_of(bentoml.mlFLow.MLDEL_FOLDER) # caricamento della versione scelta Londer ando = _mfore unione scelta</pre>
<pre>status model = miles.pruit.iom_model_mating_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)</pre>
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

creare un'immagine docker del modello tramite BentoML (Figura 16 e Figura 17).

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



import argparse
import subprocess
import shlex
import mlflow
import bentoml
import os
<pre>ifname == 'main':</pre>
parser = argparse.ArgumentParser()
<pre>parser.add_argument('model-name', type=str, default = 'power-forecasting-model', help='MLflow model name')</pre>
<pre>parser.add_argument('model-tag', type=str, default = '1.0.0', help='MLflow model version')</pre>
<pre>parser.add_argument('bento-ml-path', type=str, default = './bentofile.yaml', help='Path bentoml file')</pre>
opt = parser.parse_args()
<pre>subprocess.run(["chmod", "+x", "create_docker.sh"])</pre>
<pre>subprocess.call(shlex.split(f"./create_docker.sh %s %s %s %s (opt.model_name, opt.model_tag, opt.bento_ml_path))</pre>

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

<pre>#!/bin/bash MODEL_NAME=\$1 MODEL_TAG=\$2 BENTOFILE=\$3</pre>
bentoml build -f <mark>\$BENTOFILE</mark> version <mark>\$MODEL_TAG</mark> verbose
bentoml list grep
bentoml containerize <pre>\$MODEL_NAME:\$MODEL_TAG</pre>
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

Polo Strategico Nazionale

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]:	<pre>runner = bentoml.mlflow.get("power-forecasting-model:latest") \ .to_runner() runner.init_local() power_predictions = runner.predict.run(weather_data) power_predictions</pre>
	WARNING:bentomlinternal.runner.runner.init_local' is for debugging and testing only. Make sure to remove it before deploying to production.
[39]:	array([1180.38659163, 2354.348974 , 5314.49512367, 7702.17715267, 6919.175111 , 2751.65072135, 504.68078633, 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.

Ineres		Semantic Knowledge Search	
	Semantic V imposta municipale quando non é dovuta?		Q
Seleziona		Documenti	
Data caricamento	Rilevanza v Desc v	10 Rise	itati / 10 Documenti
Inizio Fine	2023 REG_IUC_Del_Cons_57_del_30_06_2014.pdf v		
Tipo documento			
Seleziona	Sono esenti le utenze domestiche prive di arredo e servite da utenze condomi	niali oppure utenze co read more	
Autore	1		
Seleziona	REG_JUC_Del_Cons_57_del_30_06_2014.pdf V		
File Browser	A decorrere dall'anno 2014, non è dovuta l'imposta municipale propria di cui a	ll'articolo 13 del de read more	
	13 June REG_IUC_Del_Cons_\$7_del_30_06_2014.pdf v		
	184, comma 2, del Decreto Legislativo 3 aprile 2006, n. 152 a) i rifiuti domesti	i, anche ingombr reed more	

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.

Herein	Semantic Knowledge Search	
	Syntactic V imposta municipale quando non è dovuta?	<u> </u>
Seleziona	Documenti	
Data caricamento	Rievanza V Desc V 10 Risultati / 10 Documen	ni
Inizio Fine	7023 13 Juna REG JUC Del Cons. 57. del 30. 06. 2014.pdf ∨	
Tipo documento		
Seleziona	Sono esenti le utenze domestiche prive di arredo e servite da utenze condominiali oppure utenze co read more	
Autore		
Seleziona	13 June REG_IUC. Del, Cons. 57, del 30_06_2014.pdf ∨	
File Browser	A decorrere dall'anno 2014, non è dovata l'imposta municipale propria di cui all'articolo 13 del de read more	
	2021 REG JUC, Del, Cons, 57, del 30, 06, 2014.pdf 13 June REG JUC, Del, Cons, 57, del 30, 06, 2014.pdf	
	184. comma 2, del Decreto Legislativo 3 aprile 2006, n. 152: a) i rifiuti domestici, anche ingombr read more	رت
	13 Anne REG, IUC, Del, Come, 57, del 30, 06, 2014 pdf 🤍	

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).



Lingua		
Seleziona		
Data caricamento		
Inizio	Fine	
Tipo documento		
Seleziona		
Autore		
Seleziona		
Seleziona Com.Montana Cagli (1)		
Seleziona Com.Montana Cagli (1) cosentino domenico (1)		
Seleziona Com.Montana Cagli (1) cosentino domenico (1) nt1803275 (1)		

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

[Semantic ∨	autorizzazione posteggi fiere
_		
_	Rilevanza ∨	Desc V

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.

tilevanza 🗸 Desc 🗸	10 Risultati / 10 Documenti	
2023 regolamento_posteggi_fiere.pdf >		
Metadati		
id documento: regolamento_posteggi_fiere		
Lingua: it		
Nome file: regolamento_posteggi_fiere.pdf		
Autore: G_D'Amico		
Data importazione: 12/06/2023 18:08		

Figura 26 – Visualizzazione metadati dei risultati



	Ticket Overview X	
. V impost	13/06/2023 REG_IUC_Del_Cons_57_del_30_06_2014	edge Search
	Sono esenti le utenze domestiche prive di arredo e servite da utenze l condominiali oppure	nti
a ∨ Desc	utenze comuni a più unità immobiliari (servizio idrico, servizio gas, altri servizi per i quali non è	
REG_IUC	possibile la cessazione autonoma).♠ COMUNE DI LAURIA Servizio Tributi – Regolamento IUC	
	50	
REG_IUC	6. Sono inoltre soggette alla categoria 3 (Autorimesse e magazzini senza alcuna vendita	
<u> </u>	< 21 / 29 >	
REG_IUC	ок	

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).

Seleziona		
Autore		
Seleziona		
File Browser		
- FILE BRO	WSER	

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

Success	×
Caricamento eseguito.	
	ок

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.

Extract keywords and keyphrases for a given text (Multi-lingual model)	
This service uses BERT embeddings to create the most relevant keywords and keyphrases for a given text using a simple and user-friendly keyword extraction approach. Contact BentoML Team	
Servers	
Service APIs BentoML Service API endpoints for inference.	^
POST /v1/predict/ InferenceAPI(JSON → JSON)	\sim
Infrastructure Common infrastructure endpoints for observability.	^
GET /healthz	\sim
GET /livez	\sim
GET /readyz	\sim
CET /metrics	\sim

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).

POST /v1/predict/ InferenceAPI(JSON → JSON)	^
Parameters	Cancel Reset
No parameters	
Request body required	application/json V
<pre>("inputs": [</pre>	•
Execute	





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).

Service APIs BentoML Service API endpoints for inference.	^
POST /v1/predict/ InferenceAPI(JSON → JSON)	\checkmark
Infrastructure Common infrastructure endpoints for observability.	^
GET /healthz	\checkmark
GET /livez	\checkmark
GET /readyz	\checkmark
GET /metrics	V

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).



Service APIs BentoML Service API endpoints for inference.		^
POST /v1/predict/ InferenceAPI(JSON → JSON)		^
Parameters		Cancel
No parameters		
Request body required		application/json V
"inputs": ["Harry Potter is AWESOME I don't care if anyone says differently!", "L'amblente glovane e fresco è il primo segno positivo del cambiamento di gestione. L'aperitivo], parameters": {} }	di benvenuto è ben gradito, di ottima qualità."	
Execute	Clear	

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

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

Code	Details
200	Response body
	<pre>{ *outputs*: [*idx_sentence": 0, *sent": "Marry Potter is AMESOME I don't care if anyone says differently!", *language": en; *core: 0.999997812735563 } }</pre>
	<pre>"sent": "L' ambiente giovane e fresco è il primo segno positivo del cambiamento di gestione. L' aperitivo di benvenuto è ben gradito, di ottima qualità.", "language": "it", "score": 0.9999973815910759] }</pre> Download
	Response headers
	content-length: 367 content-type: application/json dat: We(2J Jun 2023 14:32:04 GWT server: uvicorm x-berGomI-request-id: 6938098912335727155

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).



Sentiment analysis model (Multi-lingual model)	
This a bert-base-multilingual-uncased model for sentiment analysis. It predicts the sentiment of the review as a number of stars (between 1 and 5).	
Contact BentoML Team	
servers	
Service APIs BentoML Service API endpoints for inference.	^
POST /v1/predict/ InferenceAPI(JSON → JSON)	\sim
Infrastructure Common infrastructure endpoints for observability.	^
GET /healthz	\sim
GET /livez	\sim
GET /readyz	\sim
GET /metrics	\sim

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).

Service APIs BentoML Service API endpoints for inference.	^
POST /v1/predict/ InferenceAPI(JSON \rightarrow JSON)	^
Parameters	Cancel
No parameters	
Request body required	application/json v
<pre>{ "input": [</pre>	8
Execute	Clear

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).



Code	Details	
200	Response body	
	<pre>("outputs": [</pre>	
) }	₿ Dow
	Response headers	
	content-length: 38 content-y-pe: application/json date: hwi,22 Jan 2023 5:00:14 GMT server: workner:	

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).

Extract NER for a given text (Multi-lingual model)	
Named Entity Recognition model	
Contact BentoML Team	
Servers	
Service APIs BentoML Service API endpoints for inference.	^
POST /v1/predict/ InferenceAPI(JSON JSON)	\sim
Infrastructure Common infrastructure endpoints for observability.	^
GET /healthz	\checkmark
GET /livez	\sim
GET /readyz	\checkmark
GET /metrics	\sim

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).



Parameters			L	Cance
No parameters				
Request body required			application/json	
<pre>' "sentence": "Nader Jokhadar had given Syria the le } / options": { "use_cache": true, "wait_for_model": false } </pre>	ad with a well-struck header in the seventh	minute"		

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).

Code	Details
200	Response body
	<pre>{ * outputs": [</pre>
	*start: 25, } } Download
	Response headers
	context-length: 221 context-type: application/joan date: Tbu/27 Jun 2023 15:14:52 GWT server: uwiconn x-bentoml-request-id: 17380132027912477256

Figura 44 - Risposta di inferenza del servizio di Sentiment Analysis



6 PaaS Al – 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":

Audio Environment Classification API 🚥 🚥	
Vermary swi PPI Rest for Audio Environment Classification Inference	
default	^
GET /api Read Root	\sim
POST /api/audio_environment_classification/inference Inference	~
Schemas	^
Body_inference_api_audio_environment_classification_inference_post >	
Detection >	
Fragment >	
HTTPValidationError	
Prediction >	
ValidationError >	

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

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



efault	∧ default	^
GET /api Read Root	GET /api Read Ro	oot 🔨
Parameters	Try it out No parameters	Cancel
No parameters	Execute	
Response body { *message*: "Fast&PI running on Uvicorn with Gunicorn. U *tatus": "alive" } }	ising Python: 3.8.10 (default, Jun 23 2021, 15:28:49) \n[GCC 8.3.0] and version APT: 1.0°,	B Download
Response headers content-length: 161 content-type: application/json date: Hed,21 un 2023 07:55:28 GMT		

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.

API Rest for audio anomaly detection training	
default	^
GET /api Read Root	\sim
GET /api/AudioInsightComplete AudioinsightComplete	\sim
POST /api/AudioInsight Audioinsight	\checkmark

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.

POST	/api/AudioInsight Audioinsight
Parameters	s
Name	Description
ID string (query)	test30secondi
Request bo	ody ^{required}
files * ' array	Scegli file training_1.wav Add string item

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



Curl curl -X 'http:	- 9057' \ ///Trajo unamaly_detection_deem_puljo_pogs_clp81.pass_pom_priv/pp1/AudioTesightYDPtest98second1' \	
-H 'ac -H 'Co -F 'fi	cospt: application/joen '\ ontri-Type: mil/serv/form-data'\ Lles@ftraining_l.uwy:typeaudia/uw'	8
Request U	lRL train-anomaly-detection-demo-mudio.apps.cluθl.paas-psn.priv/api/AudioInsight7ID=test30secondi	
Server res	sponse	
Code	Details	
202	Response body	
	("TRAINING_ID": "1687337993133")	Download
	Response headers	
	access-central-alla-credentials: true access-central-alla-credentials: access-central-alla-credentials: centext-type application/sen detex Wed, 21 Jun 2023 96:59:50 GOT server: wickern	

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

GET /a	pi/AudioInsightComplete Audioinsightcomplete
Parameters	
Name	Description
ID string	test30secondi
(query) TRAINING_II	D 1687337993133
(query)	

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

Respons	es
Curl	
curl -X 'http:, -H 'ace	'GET' \ //train-anomaly-detection-demo-audio.apps.clu01.paas-psn.priv/api/AudioInsightComplete?ID=test30secondi&TRAINING_ID=1687337993133' \ cept: application/json'
Request U	RL
http://t	train-anomaly-detection-demo-audio.apps.clu01.paas-psn.priv/api/AudioInsightComplete?ID=test30secondi&TRAINING_ID=1687337993133
Server res	ponse
Code	Details
200	Response body
	{ "status": "succeeded" }
	Response headers
	content-length: 22 content-type: application/json date: Wed,21 Jun 2023 09:38:52 GMT server: uvicorn

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"

API Rest for audio anomaly detection Inference	
default	^
GET /api Read Root	\sim
POST /api/Anomaly Anomaly	\sim

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.

Name	Description	
ID string (query)	test30secondi	
TRAINING_ID string (query)	1687337993133	
Request body ^r	equired	

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



200	Response body
200	<pre>Response body { "max": { "value": 1, "from": 0, "to": 1 }, "min": { "value": 0.8008789838891092, "from": 4, "to": 5 }, "mean": { "value": 0.957177798165365 }, "sigma": { "value": 0.041619451547103455 }, "values": [{ "from": 0, "to": 1, "confidence": 1 } }</pre>
	}, {
	"from": 1, "to": 2, "confidence": 1

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.

Speaker Identification API 100 CASS	
API Rest for Speaker Identification Training	
default	^
GET /api Read Root	\checkmark
POST /api/speaker_identification/training Training	\checkmark
GET /api/speaker_identification/training/status Status	\checkmark

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"*.

POST /ap:	i/speaker_identification/training Training
This is a POST n	nethod that runs the training of speaker identification.
Parameters	
 subject. The group: Grou file: Audio Fi 	speaker name p name of speaker. If group is empty, it will generate by default lie
Returns	
The Response th	at has as message:
<pre>{ "start": "training }</pre>	True, z_id": training id
Parameters	
Parameters	Description
Parameters Name subject * require	Description
Parameters Name subject * require string (auery)	Description d Gruber
Parameters Name subject * require string (query)	Description d Gruber
Parameters Name subject * require string (query) group string	Description d Gruber group
Name subject * require string (query) group string (query)	Description d Gruber group
Name Subject * require string (query) group string (query) Request body re	Description d Gruber group
Name subject * require string (query) group string (query) Request body re	Description d Gruber group
Parameters Name subject * require string (query) group string (query) Request body file * required	Description d Gruber group

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.



Speaker Identification API Image: Speaker Identification Inference API Rest for Speaker Identification Inference Image: Speaker Identification Inference default Image: Speaker Identification Inference Image: Speaker Identification Inference Image: Speaker Identification Inference POST /api/speaker_identification/inference



Parameters	
Name	Description
trainingID string (query)	1687342630497
group string (query)	group
Request body ^{required}	
file * required string(\$binary) Scegli file gruber.wav	
Execute	

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



