CAPSTONE PROJECT

POWER SYSTEM FAULT DETECTION AND CLASSIFICATION

Presented By:
Aksa Jiji Thomas - Fr. Conceicao Rodrigues Institute of Technology, Vashi - Information Technology

edunet
foundation

OUTLINE

- **Problem Statement**
- **Proposed System/Solution**
- **System Development Approach** (Technology Used)
- **Algorithm & Deployment**
- **Result (Output Image)**
- **Conclusion**
- **Future Scope**
- **References**

PROBLEM STATEMENT

Design a machine learning model to detect and classify different types of faults in a power distribution system. Using electrical measurement data (e.g., voltage and current phasors), the model should be able to distinguish between normal operating conditions and various fault conditions (such as line-to-ground, line-to-line, or three-phase faults). The objective is to enable rapid and accurate fault identification, which is crucial for maintaining power grid stability and reliability.

PROPOSED SOLUTION

- This project proposes a machine learning-based fault detection and classification system that utilizes electrical parameters such as voltage, current, power load, weather conditions, and component health to predict the type of fault in a power distribution network. The Random Forest Classifier was identified as the best-performing model with the highest accuracy among the tested algorithms.
- **Components of the solution:**

- Data Collection: Kaggle dataset with fault-related electrical parameters
- Data Preprocessing: Cleaning, normalization, and feature engineering
- Model Training: AutoAI-generated pipelines using IBM Watsonx.ai Studio
- Best Model: Random Forest Classifier with 0.409 optimized accuracy
- Deployment: Hosted and tested using IBM Cloud Lite deployment services

SYSTEM APPROACH

The "System Approach" section outlines the overall strategy and methodology for developing and implementing the rental bike prediction system.

- **System requirements**
- IBM Watsonx.ai Studio
- IBM Cloud Lite (Deployment)
- Cloud Object Storage
- Watsonx.ai Runtime
- Multiple languages supported (auto-generated code by IBM for scoring endpoint): Python, Java, JavaScript, cURL, Scala
- REST API (for real-time model scoring)

- **Library required to build the model**
- These are the internal libraries IBM AutoAI uses to build and score models:
- Scikit-learn: Core modeling (Random Forest, Logistic Regression, etc.)
- Pandas and NumPy: Data preprocessing and handling
- Requests: REST API interaction during deployment and scoring

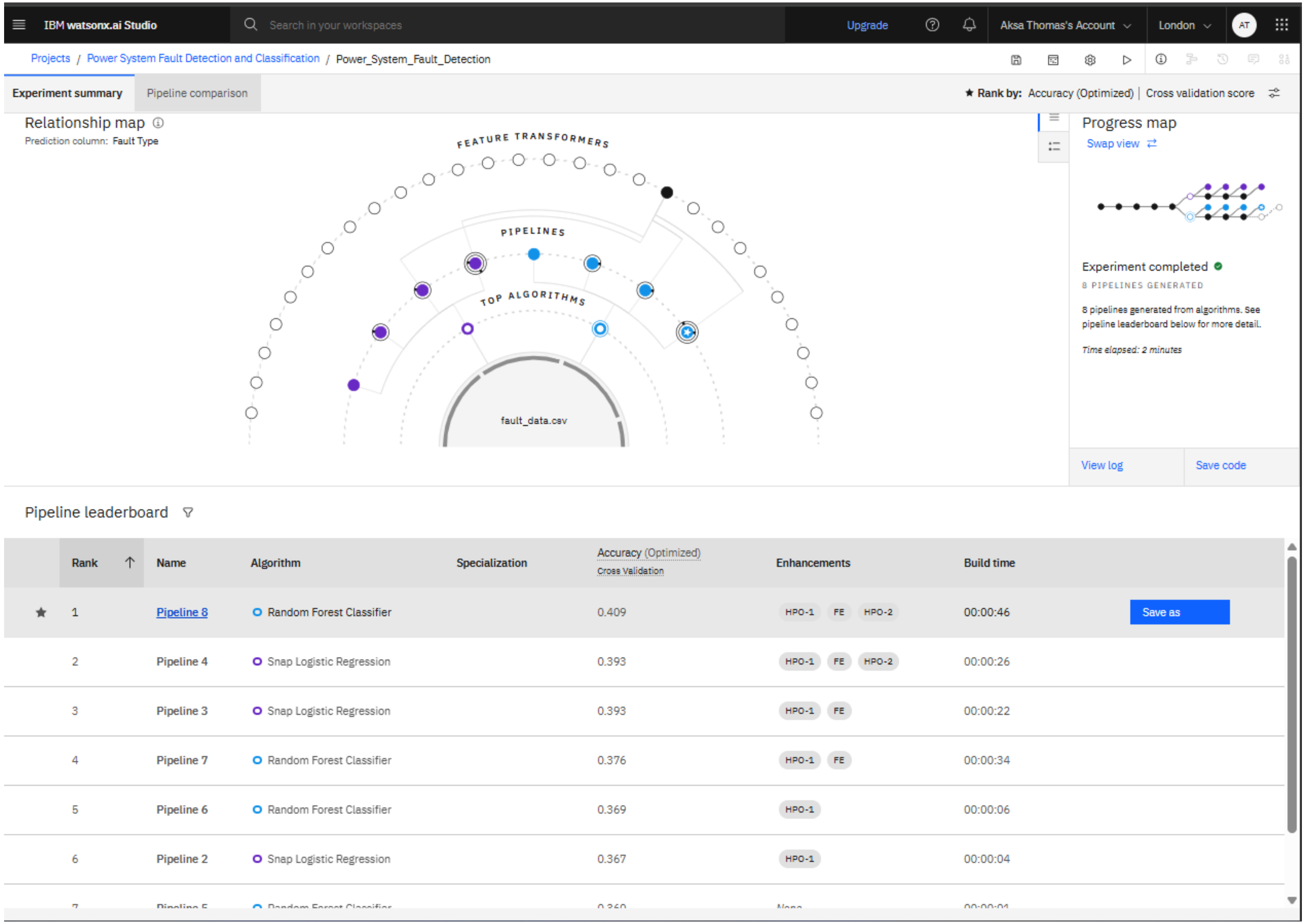
ALGORITHM & DEPLOYMENT

- Random Forest Classifier
- AutoAI tested 8 different model pipelines.
- Pipeline 8 using Random Forest achieved the best performance.
- Snap Logistic Regression was the closest competitor but underperformed on recall/accuracy.
- AutoAI enhancements like Feature Engineering (FE), Hyperparameter Optimization (HPO-1, HPO-2) were applied to improve results.
- **Training Process:** The training process was fully managed by IBM Watsonx's AutoAI, which handled data preprocessing, feature engineering, algorithm selection, and hyperparameter optimization automatically. Each pipeline was trained using internal cross-validation to reduce overfitting and improve the model's ability to generalize to unseen data. Pipeline 8, which used the Random Forest algorithm, emerged as the top performer.
- **Prediction Process:** Once the best pipeline was deployed, predictions were made by submitting data via manual input or API. When new feature values are submitted, the model processes them and returns a predicted fault category along with confidence scores for all other possible classes.

ALGORITHM & DEPLOYMENT

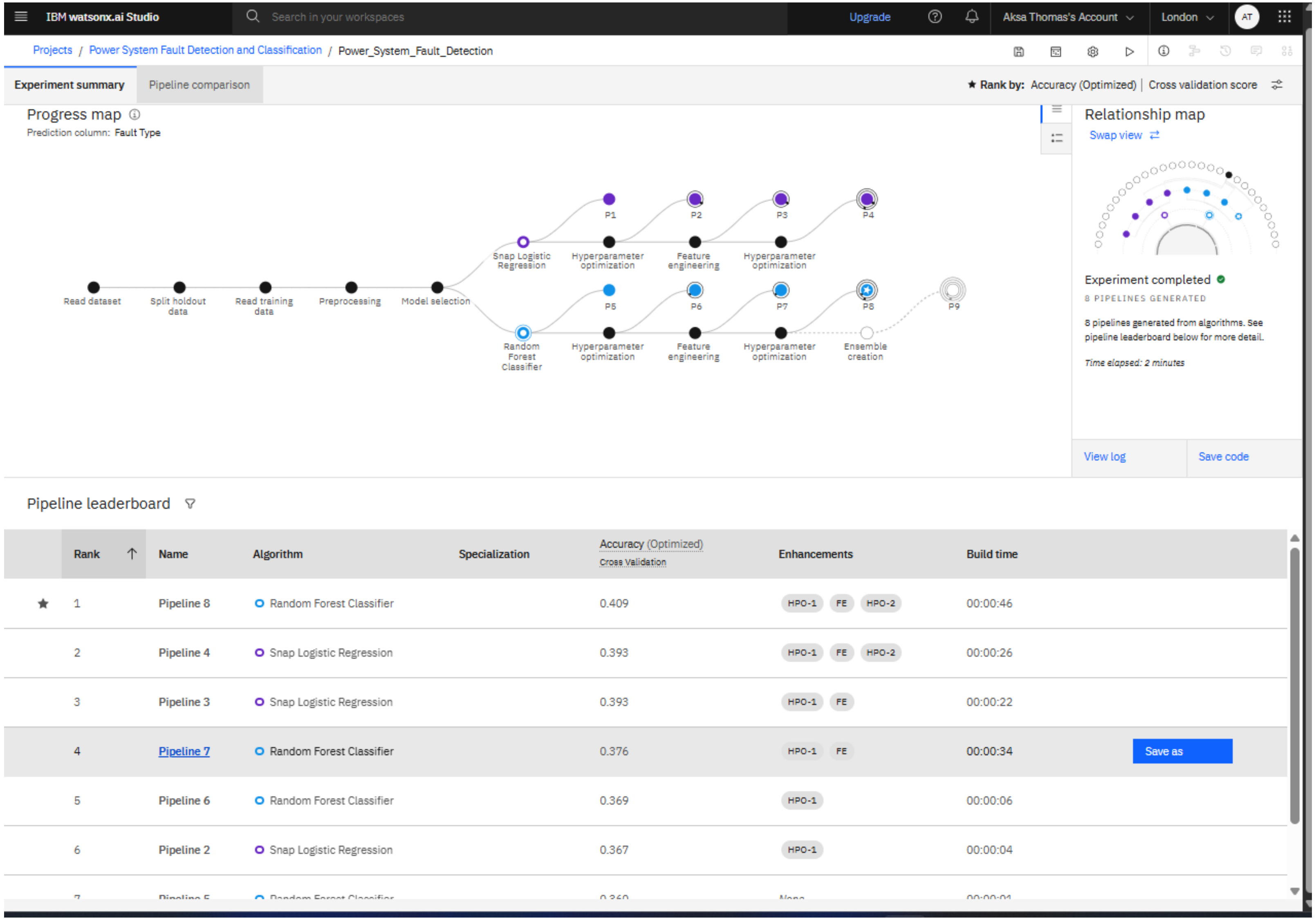
- **Input Features Considered:**
  - Electrical: Voltage, Current, Power Load
  - Environmental: Temperature, Wind Speed, Weather Condition
  - Operational: Component Health, Maintenance Status
  - Fault Metadata: Duration, Downtime, Fault Location
- **Deployment Details:**
  - The best model (Pipeline 8) was deployed using Watsonx Deployment Space.
  - A REST API endpoint was automatically generated.
  - IBM Watsonx auto-generated code snippets in multiple languages, including Python, Java, JavaScript, cURL, and Scala, to call the API endpoint.
  - For testing purposes, 10 sample rows were selected from the original Kaggle dataset and manually entered into the Watsonx prediction UI, instead of using a CSV file upload.
  - Output includes the predicted fault class and a probability/confidence score for each possible fault type.

RESULT



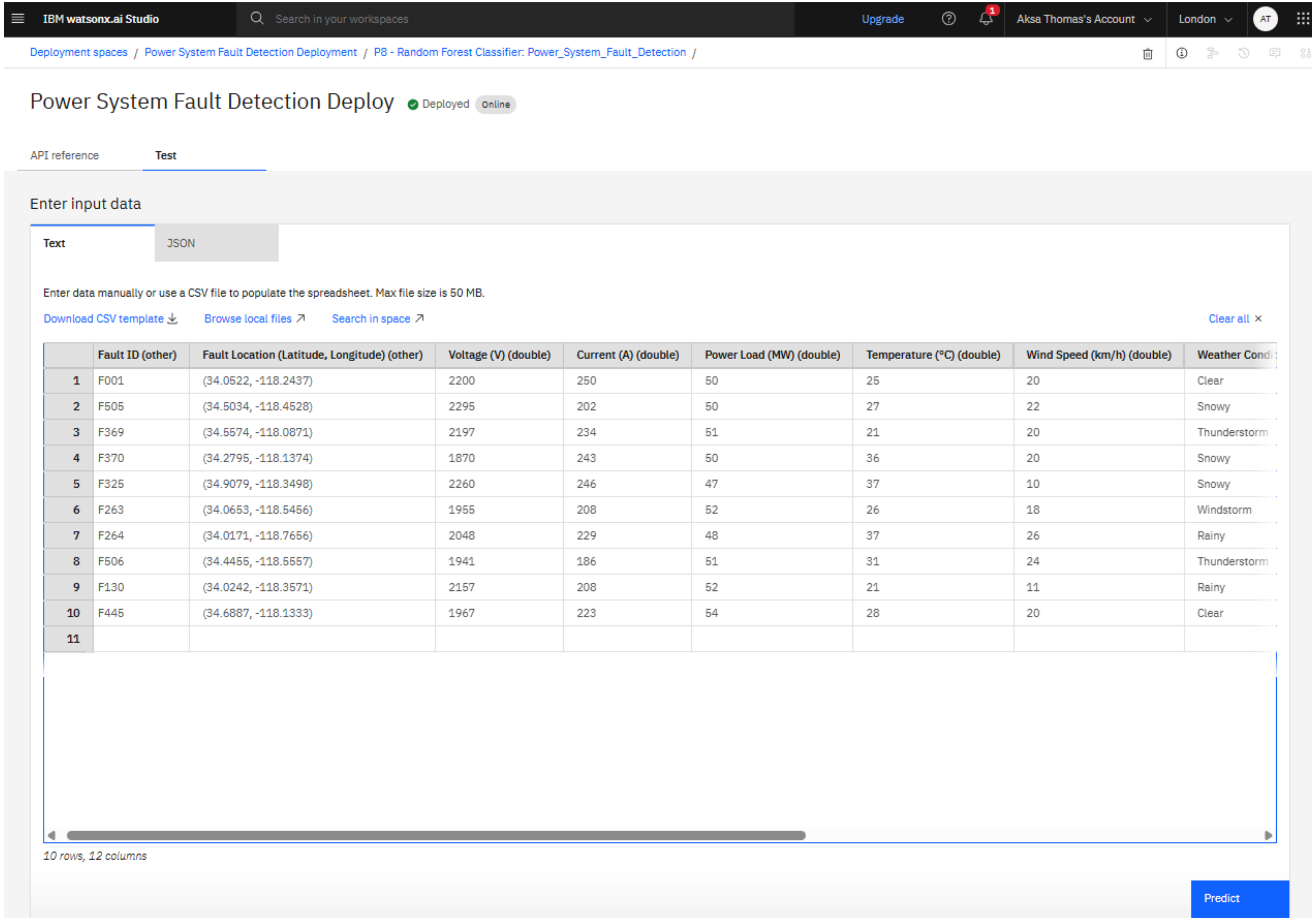
edunet foundation

RESULT



edunet foundation

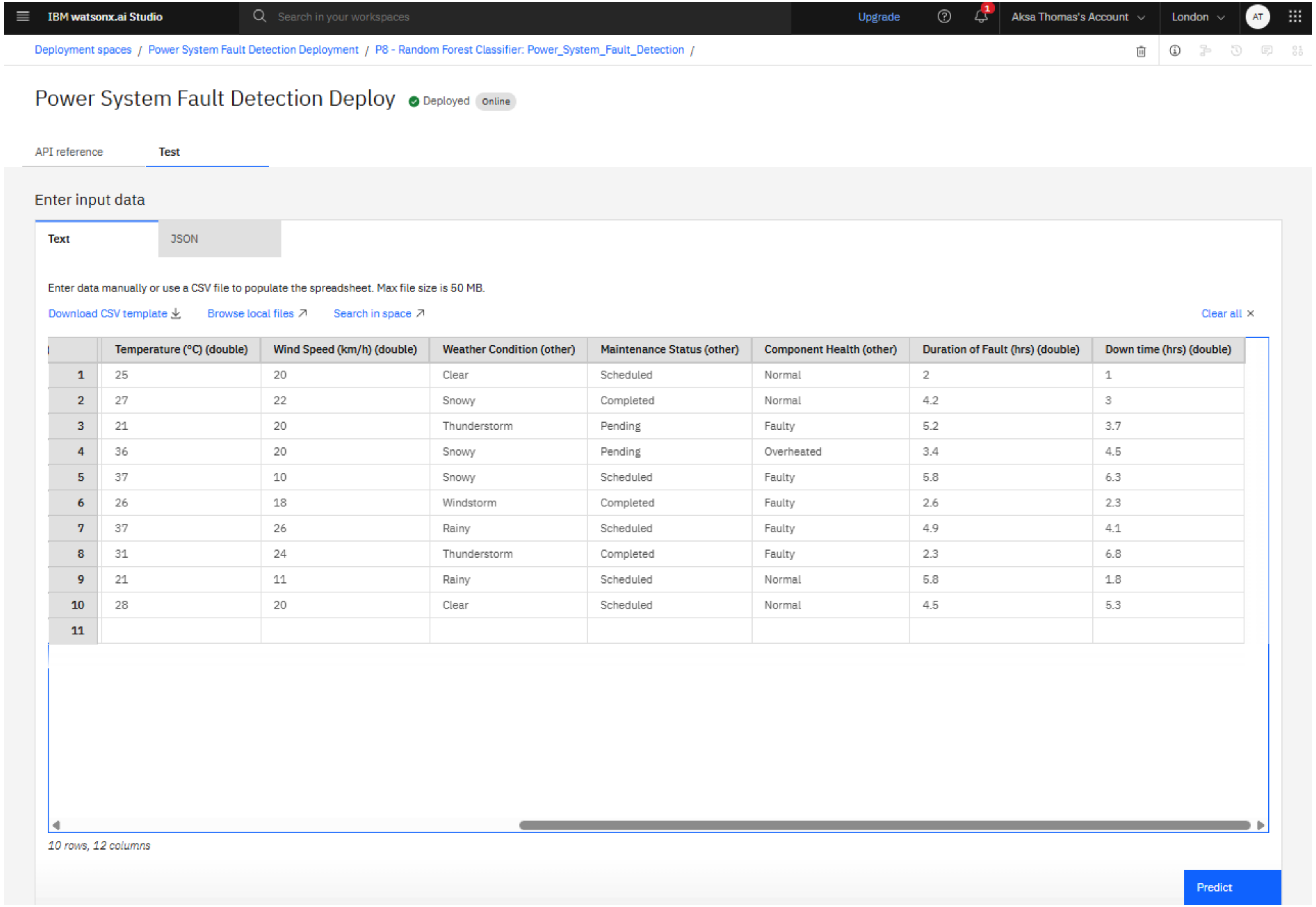
RESULT



	Fault ID (other)	Fault Location (Latitude, Longitude) (other)	Voltage (V) (double)	Current (A) (double)	Power Load (MW) (double)	Temperature (°C) (double)	Wind Speed (km/h) (double)	Weather Cond
1	F001	(34.0522, -118.2437)	2200	250	50	25	20	Clear
2	F505	(34.5034, -118.4528)	2295	202	50	27	22	Snowy
3	F369	(34.5574, -118.0871)	2197	234	51	21	20	Thunderstorm
4	F370	(34.2795, -118.1374)	1870	243	50	36	20	Snowy
5	F325	(34.9079, -118.3498)	2260	246	47	37	10	Snowy
6	F263	(34.0653, -118.5456)	1955	208	52	26	18	Windstorm
7	F264	(34.0171, -118.7656)	2048	229	48	37	26	Rainy
8	F506	(34.4455, -118.5557)	1941	186	51	31	24	Thunderstorm
9	F130	(34.0242, -118.3571)	2157	208	52	21	11	Rainy
10	F445	(34.6887, -118.1333)	1967	223	54	28	20	Clear
11								

edunet foundation

RESULT



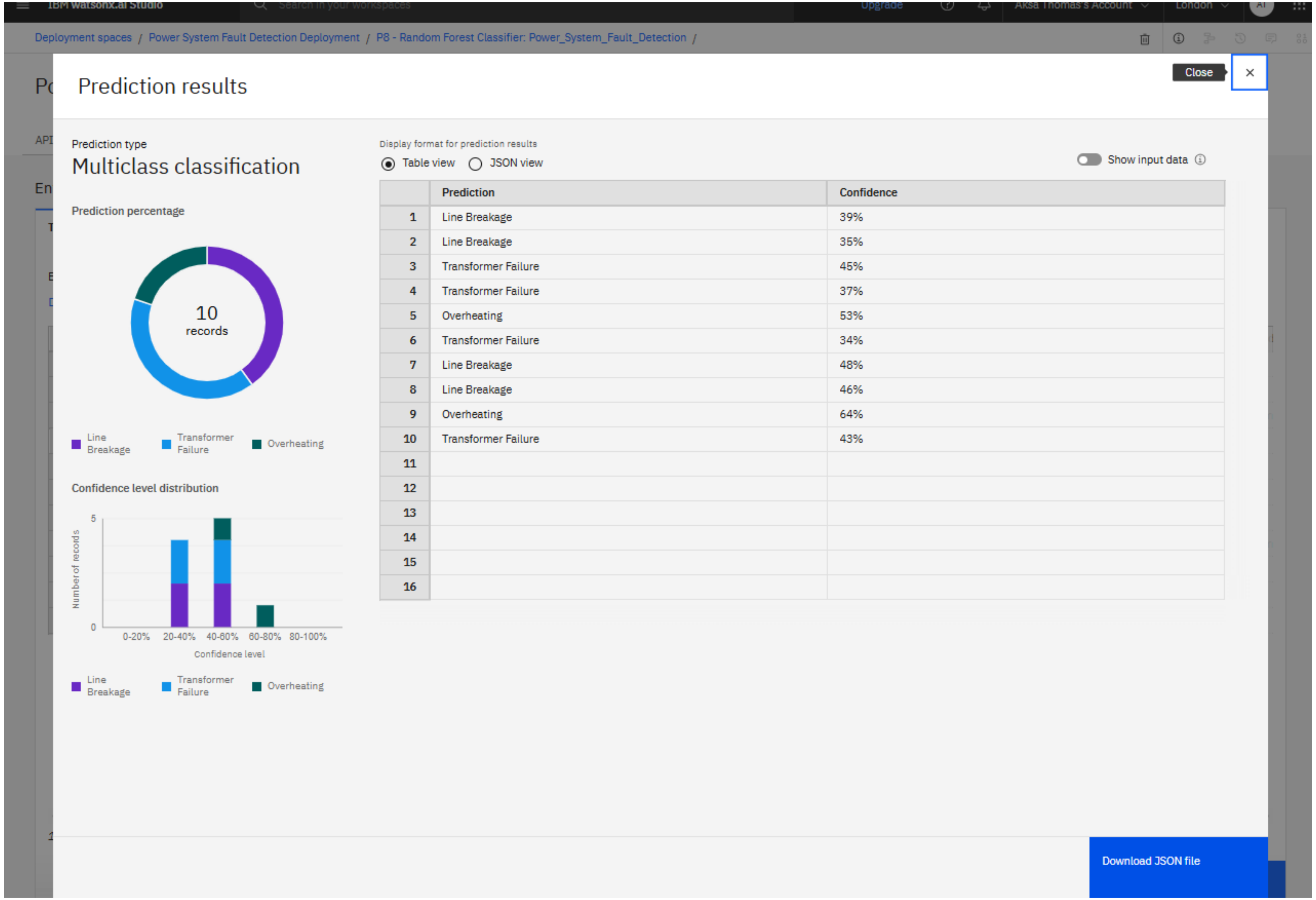
	Temperature (°C) (double)	Wind Speed (km/h) (double)	Weather Condition (other)	Maintenance Status (other)	Component Health (other)	Duration of Fault (hrs) (double)	Down time (hrs) (double)
1	25	20	Clear	Scheduled	Normal	2	1
2	27	22	Snowy	Completed	Normal	4.2	3
3	21	20	Thunderstorm	Pending	Faulty	5.2	3.7
4	36	20	Snowy	Pending	Overheated	3.4	4.5
5	37	10	Snowy	Scheduled	Faulty	5.8	6.3
6	26	18	Windstorm	Completed	Faulty	2.6	2.3
7	37	26	Rainy	Scheduled	Faulty	4.9	4.1
8	31	24	Thunderstorm	Completed	Faulty	2.3	6.8
9	21	11	Rainy	Scheduled	Normal	5.8	1.8
10	28	20	Clear	Scheduled	Normal	4.5	5.3
11							

10 rows, 12 columns

Predict

edunet foundation

RESULT



edunet foundation

CONCLUSION

- The proposed system successfully demonstrates a practical and automated approach to detect and classify faults in power systems using machine learning. The use of IBM Watsonx's greatly simplified model creation, optimization, and deployment by eliminating the need for manual tuning. While the current system used a limited number of test entries, it still validated the core predictive logic. The project lays a strong foundation for future scaling in real-time, data-rich environments and showcases the effectiveness of cloud-based AI tools in modernizing power infrastructure.

FUTURE SCOPE

- Improve model accuracy with a larger and more diverse dataset
- Use real-time IoT-based data feeds from smart meters/sensors
- Extend fault detection to include 3-phase faults, transient faults, and cable insulation failures
- Build a web dashboard that uses the REST API to display real-time fault classification results
- Integrate SMS or email-based alerts for immediate field action

REFERENCES

- Kaggle dataset link – https://www.kaggle.com/datasets/ziya07/power-systemfaults-dataset
- IBM Watsonx Documentation

edunet foundation

IBM CERTIFICATIONS

In recognition of the commitment to achieve professional excellence

Getting Started with Artificial Intelligence

IBM SkillsBuild

IBM

Aksa Jiji

Has successfully satisfied the requirements for:

Getting Started with Artificial Intelligence

Issued on: Jul 18, 2025

Issued by: IBM SkillsBuild

Verify: https://www.credly.com/badges/c6f19fbb-a576-4d3e-bdd5-fa27241b77f3

IBM

IBM CERTIFICATIONS

Journey to Cloud: Envisioning Your Solution

IBM SkillsBuild

IBM

In recognition of the commitment to achieve professional excellence

Aksa Jiji

Has successfully satisfied the requirements for:

Journey to Cloud: Envisioning Your Solution

Issued on: Jul 20, 2025

Issued by: IBM SkillsBuild

Verify: https://www.credly.com/badges/45ad7bae-3842-42c1-8a23-a39eb056ee2b

IBM

IBM CERTIFICATIONS

IBM **SkillsBuild** Completion Certificate

This certificate is presented to

Aksa Jiji

for the completion of

Lab: Retrieval Augmented Generation with LangChain

(ALM-COURSE_3824998)

According to the Adobe Learning Manager system of record

Completion date: 23 Jul 2025 (GMT)

Learning hours: 20 mins

edunet foundation

THANK YOU