UNIVERSIDAD SAN FRANCISCO DE QUITO USFQ

Colegio de Posgrados

Supervised Learning for Failure Detection in an Engine Generation Unit Leveraging Data Infrastructure, Feature Engineering and Monitoring through a Machine Learning Dashboard

Proyecto de Titulación

Jonathan Vinicio Espín Martin

Israel Pineda, Ph.D.

Director de Trabajo de Titulación

Trabajo de titulación de posgrado presentado como requisito para la obtención del título de Magíster en Ciencia de Datos

Quito, 02 de diciembre 2024

UNIVERSIDAD SAN FRANCISCO DE QUITO USFQ
COLEGIO DE POSGRADOS

HOJA DE APROBACIÓN DE TRABAJO DE TITULACIÓN

Supervised Learning for Failure Detection in an Engine Generation Unit Leveraging Data Infrastructure, Feature Engineering and Monitoring through a Machine Learning Dashboard

Jonathan Vinicio Espín Martin

Nombre del Director del Programa:	Felipe Grijalva
Título académico:	Ph.D. en Ingeniería Eléctrica
Director del programa de:	Maestría en Ciencia de Datos

Nombre del Decano del Colegio Académico:	Eduardo Alba
Título académico:	Doctor en Ciencias Matemáticas
Decano del Colegio:	Ciencias e Ingenierías

Nombre del Decano del Colegio de Posgrados:	Dario Niebieskikwiat
Título académico:	Doctor en Física

Quito, diciembre 2024

© DERECHOS DE AUTOR

Nombre del estudiante: Jonathan Vinicio Espín Martin

Código de estudiante: 00339306

C.I.: 1803933660

Lugar y fecha: Quito, 02 de diciembre de 2024

ACLARACIÓN PARA PUBLICACIÓN

Nota: El presente trabajo, en su totalidad o cualquiera de sus partes, no debe ser considerado como una publicación, incluso a pesar de estar disponible sin restricciones a través de un repositorio institucional. Esta declaración se alinea con las prácticas y recomendaciones presentadas por el Committee on Publication Ethics COPE descritas por Barbour et al. (2017) Discussion document on best practice for issues around theses publishing, disponible en http://bit.ly/COPETheses.

UNPUBLISHED DOCUMENT

Note: The following graduation project is available through Universidad San Francisco de Quito USFQ institutional repository. Nonetheless, this project – in whole or in part – should not be considered a publication. This statement follows the recommendations presented by the Committee on Publication Ethics COPE described by Barbour et al. (2017) Discussion document on best practice for issues around theses publishing available on http://bit.ly/COPETheses.

DEDICATORIA

A Dios y mi familia

AGRADECIMIENTOS

A Dios. A mi familia por su apoyo. A la Universidad San Francisco de Quito mi alma mater. A mi tutor de titulación y cada profesor.

RESUMEN

Detectar fallos en maquinaria industrial con alta precisión y exactitud es crucial para extender los ciclos de vida operativos y garantizar la fiabilidad del sistema. Esta investigación tiene como objetivo desarrollar e implementar el uso de modelos de Machine Learning supervisado con resultados probabilístico para predecir fallos en un generador eléctrico industrial de gran capacidad. El generador es un componente crítico en un campo remoto de extracción de petróleo y gas en Ecuador, que alimenta los subsistemas con la energía principal necesaria para garantizar la continuidad de la seguridad industrial y la producción. Este estudio implementará la detección de fallos basada en aprendizaje supervisado para clasificación binaria, utilizando un conjunto de datos de 8 millones de muestras con datos en tiempo real e históricos. Se extraen aproximadamente 90 características a partir de variables del proceso: temperatura, presión, potencia, entre otras métricas operativas.

La recopilación y almacenamiento de datos se llevan a cabo mediante el sistema SCADA Ignition de Inductive Automation. Ignition interactúa fácilmente entre los protocolos del controlador del generador y los servidores de bases de datos basados en SQL. Por lo tanto, el software actúa como capa intermedia e integración en la recopilación, estructuración y procesamiento de datos para análisis predictivos. En esta investigación se proponen tres algoritmos de ML: Random Forest (RF) como método base, XGBoost como un modelo más robusto y escalable, y Redes Neuronales Artificiales (ANN) multicapa de perceptrones (MLP) como representante de enfoques avanzados de aprendizaje profundo. Se aplica ajuste de hiperparámetros para optimizar el rendimiento del modelo y lograr las mejores predicciones posibles de la probabilidad de fallos.

Los modelos de Machine Learning entrenados se implementan posteriormente a través de Ignition, que ejecuta scripts de Python para realizar predicciones tanto en tiempo real como históricas. Esta integración permite que se analice constantemente los datos entrantes y proporcione resultados probabilísticos que pueden visualizarse mediante una interfaz de ML hecho con Ignition. El dashboard o interfaz en intuitiva en cuanto a las probabilidades de fallo previstas y la importancia de las características para facilitar la toma de decisiones operativas. Con modelos de ML explicables y supervisados, este sistema proporcionará información procesable con la que los usuarios podrán predecir y prevenir fallos potenciales. Además, mejora la fiabilidad del motor generador eléctrico y aumenta su eficiencia operativa, demostrando así el valor de integrar ML en los sistemas de monitoreo y control industrial.

Palabras clave: Aprendizaje de Máquina, Predictor de Fallas, Motor Genedor Eléctrico, Aprendiza Supervisado, Random Forest, XGBoost, ANN, SCADA-Ignition, Dashboard

ABSTRACT

Detecting faults in industrial machinery with high precision and accuracy is crucial to extend operational life cycles and ensure system reliability. This research aims to develop and implement the use of supervised Machine Learning models with probabilistic results to predict failures in a large capacity industrial electrical generator. The generator is a critical component in a remote oil and gas extraction field in Ecuador, feeding the subsystems with the main power necessary to ensure the continuity of industrial safety and production. This study will implement supervised learning-based fault detection for binary classification, using a dataset of 8 million samples with real-time and historical data. Approximately 90 characteristics are extracted from process variables: temperature, pressure, power, among other operational metrics.

Data collection and storage are carried out using Inductive Automation's SCADA Ignition system. Ignition easily interfaces between generator driver protocols and SQL-based database servers. Therefore, the software acts as an intermediate layer and integration in collecting, structuring and processing data for predictive analytics. In this research, three ML algorithms are proposed: Random Forest (RF) as a base method, XGBoost as a more robust and scalable model, and Artificial Neural Networks (ANN) multilayer perceptron (MLP) as a representative of advanced deep learning approaches. Hyperparameter tuning is applied to optimize model performance and achieve the best possible failure probability predictions.

The trained Machine Learning models are then deployed through Ignition, which runs Python scripts to make both real-time and historical predictions. This integration allows incoming data to be constantly analyzed and provides probabilistic results that can be visualized using an ML interface made with Ignition. The dashboard or interface is intuitive in terms of the expected failure probabilities and the importance of the characteristics to facilitate operational decision making. With explainable and supervised ML models, this system will provide actionable insights with which users can predict and prevent potential failures. Furthermore, it improves the reliability of the electric motor generator and increases its operational efficiency, thus demonstrating the value of integrating ML into industrial monitoring and control systems.

Key words: Machine Learning, Failure Prediction, Engine Electric Generator, Supervised Learning, Random Forest, XGBoost, ANN, SCADA-Ignition, Machine Learning Dashboard

TABLE OF CONTENTS

TABLES INDEX

FIGURES INDEX

Supervised Learning for Failure Detection in an Engine Generation Unit Leveraging Data Infrastructure, Feature Engineering and Monitoring through a Machine Learning Dashboard

Jonathan Espín Martin, *Member, IEEE Universidad San Francisco de Quito*

***Abstract*—An accurate and high precision predictor for industrial machine failure detection is crucial to enhancing its operational lifecycle. This research shows the results of various Machine Learning (ML) models in classifying a target feature, upon observations and evaluation metrics that better predict failure events. A bore industrial electrical engine generator and its failures over time are the focus of analysis for developing and implementing a probabilistic ML predictor. Gathered process variable information, as features, alongside operational data over time as observations, represents the source of the tabular dataset. These features correspond to various monitoring variables such as temperatures, pressure, power, and similar metrics, with about 90 characteristics analyzed. A set of 8 million samples is used to train and test a supervised machine learning model for binary classification, capable of predicting failure probability events with real-time or the latest historical values. This data was collected from a specific engine generator used as the main electrical power unit supply for a remote Oil & Gas extraction field in Ecuador. Data storage and extraction are hosted and managed by a Supervisory Control and Data Acquisition (SCADA) software server-agnostic system, Ignition by Inductive Automation, which handles Structured Query Language (SQL) clauses for data acquisition from database servers. The runtime operability of this engine generator is critical, as it provides energy to the subsystems that maintain industrial safety and production operations. This leads to the need for probabilistic predictions that deliver valuable information to operational control and monitoring of the generator itself. Furthermore, the deployment of a user-friendly dashboard is useful as a visualization tool for analysis and decision-making, offering insights into the probability of failure events and feature importance. The ML classification algorithms used in this research include Random Forest (RF) as a standard method, Extreme Gradient Boosting (XGBoost) as a more robust strategy, and Artificial Neural Network (ANN) as a more sophisticated approach. Once the best model hyperparameters are obtained via tuning, Ignition is used as a task orchestration system to gather process variables either from current process values or from the historical database source, and to make predictions. In both cases, Ignition acts as middleware between the driver and protocol gathering information on the production field and the database, using its Open Database Connectivity (ODBC) capabilities. Additionally, it loads trained ML models into memory, executes Python scripts, and delivers predictions based on the latest data points. This system also hosts the ML dashboard, deploying the output of the predictors as explainable visual insights through graphics that help users understand input changes in a clear and interpretable manner.**

***Index Terms*—Machine Learning, Failure Prediction, Engine Electric Generator, Supervised Learning, Random Forest, XGBoost, ANN, SCADA-Ignition, Machine Learning Dashboard.**

I. Introduction

THIS research analyzes how Machine Learning (ML), as a prediction technique, is applicable to any task and process. The Oil & Gas Ecuadorian industry is not the exception, data generation from this industry processes and production areas, enables to obtain data information to perform analytics capable to deliver valuable insights to enhance productivity for efficiency expectations and to mitigate possible downtime events with predictions. Industrial Electrical Generators fuel-based engines are an important component in the hydrocarbons sector due to its capacity to provide electrical energy to various sub-systems for petroleum extraction processes. The operational time of these systems is meant to be continuous and reliable since they are usually located in remote locations that may not be connected to the national electrical systems. This means that these electrical power generator systems are the main power source for production, personnel sustainability and safety. The engine-generator system may be sliced into two functional sections, the combustion engine and electrical generator; mechanical wear is a key factor to inspect when the goal is to keep these operations maximized overtime without harming the lifespan of the engine or generator. Furthermore, various trackable variables like temperature, pressure, oil, water, fuel, and vibrations at the engine generator or even system is complementary electronics may lead into a failure event when abnormal behavior happens. Intrinsically, any variable as independent variable or as a conjunction of values that follows explainable regressions, are utilized for predictions of a system failure [1]. In ML terminology, these are the correlations of a labeled target and data context itself within a supervised learning scope, aimed at obtaining probabilistic prediction results for binary classification.

Future events are inherently impossible to predict with

absolute precision or accuracy. A machine learning model provides quantified predictions suggesting the likelihood of a failure occurring. The advantages of suggesting provisional maintenance towards productivity and mitigating long-term downtimes are not only important from the productivity point of view, but also important because it collects valuable information prior, while or after a failure event. This information is in fact the most valuable information overall since it contains values from features that are important for understanding failure/non-failure behavior events [2].

The strategy for this machine failure prediction aims to use Supervised Learning techniques that leverage labeled dataset usage for training, testing and even validating. This enables the model to recognize patterns by establishing the relations between the inputs and outputs as well to predict outcomes. Hyperparameters tuning is important since it keeps the best possible model attributes by testing multiples ML strategies like balancing, regularization, Principal Component Analysis (PCA), learning rates and more [3], [4].

With about 90 features, the analysis of the correlation between variables its importance, may provide hints of data characterization that suggests more relevance of some features against others. Feature importance engineering is performed as part of this research to obtain valuable information toward features observation, aiming to the utilization of this information to users to emphasize attention to specific features that are more relevant when failure probabilistic prediction is the goal.

For classification, Random Forest algorithm is used as a standard method due to its capacity to handle high dimensional datasets and excels in complex decisions boundaries. The idea behind this algorithm is based on building decision trees in training phase and it uses majority voting to determine the class label correspondence. Another good reason to use this algorithm is because it supports nonlinearity and offers a way to extract features importance [5].

Extreme Gradient Boosting, as a robust strategy, is highly efficient for binary classification tasks and it is suitable because it has some characteristics, ideal for binary classification. It builds decision trees where further depth tree is a corrected version of the prior tree. It also supports imbalanced data, which is common for binary classification, by setting a parameter in the model definition. Regularization L1 and L2 and depth control are also supported and it is important to avoid overfitting. It also offers a way to extract features importance, which is important to accomplish the purposes of this research [6].

Artificial Neural Network is a more general approach, it excels on capturing non-lineal relationships in the dataset by using some known classification functions like ReLU or Sigmoid. This algorithm offers the capacity of hierarchical feature learning automatically, letting it capture important patterns without necessarily doing feature engineering manually. It also provides the possibility to perform transfer learning and its scalability due to its capacity to be executed on (Graphical Processing Unit) GPU [7].

The strategy to evaluate the model performanace is by using F2 score. This is a variation of F1 score, but it emphasizes recall more than precision [8]. This is particularly useful in a binary classification analysis for imbalanced target features. Thus, sensitivity has higher weight on F2, which means it penalizes the model less for false positives but more for false negatives [9]. With this asseveration, it is established that the scoring strategy will not tolerate false negatives, in other words, it is not acceptable to categorize an event as non-failure when its probability to be a failure is high.

Before starting, a series of steps must be carried out to get a pathway aimed at obtaining probabilistic prediction models for a particular engine generator machine failure event. Firstly, data acquisition strategy, which is based on SQL, was defined and used as the mechanism to obtain and analyze the dataset. Variables collection is based on a series of sensors and signals on field, connected to the engine-generator machine, publishing data values into the advanced SCADA Ignition system. Regarding data historization, this software performs batch insertion to a structured partitioned tables upon a system hosting SQL engine. In addition, it provides Python scripting functions to sinuously extract historical data information, solving historical calls by executing SQL clauses against the self-managed table's structure database. Afterwards, data preprocessing was necessary to build this structured file holding all the key information for this analysis into a comma-separated value (csv) file. For instance, the dataset was split into separate *.csv* files due to the size of observation and features counts for portability and operability purposes. The range of information for this analysis contains a historical range of 26 months overall that contains around 11 gigabytes of raw plain information. Afterwards, an Exploratory Data Analysis (EDA) provided insights to enable the best set of information. Cleaning and transforming data is necessary to obtain a labeled dataset. This means that failure detections on a time basis are not part of data information metrics itself, and they handled manually on real scenarios. Thus, it is necessary to integrate these failure events into the historical dataset and the Ignition platform offers a scripting mechanism to push data in place specifying the value and the timestamp of occurrence. Keeping in might that this system manages the database through pre-configured queries, it is suitable to generate and obtain the labeled dataset as expected, further details are described below. Ones desired data is in place, as mentioned before, a scripting function command executed historical variables information was specified in desired in a specific range and in a controlled time span.

For this implementation, ten seconds sample time was used against all variables associated with the monitoring of the engine generator, separated on daily groups across two years of information. As data is important, so is the framework to perform the model training, using an in-place folder structure holding input and output file storing

the execution plans as a Machine Learning repository. As follows, high specs hardware was required to perform data processing and training in manageable times. As referential mentions, a hardware system capable of delivering at least 150GB of RAM and 30GB of GPU is recommended for dataset operations and training steps. Those values are mentioned after a series of shared experiences when dealing with the size of the mentioned dataset into the Python 3 runtimes using Pandas, NumPy and Scikit-learn frameworks, just to mention a few. Subsequently, it is crucial to properly perform programing strategies aimed at avoiding unexpected exceptions while executions and model characterization on phases of loading, cleaning, training, evaluation and feature importance observations. After completing all the phases to obtain a probabilistic prediction model, those models were saved as serializable files. As the system that will run the models is apart from the environment where training was taking place. Finally, Ignition as a SCADA system capable of performing Python calls, performs prediction orchestration taking process data variables from raw driver protocols, historize data changes into a database and pull back the information into those calls where the probabilistic prediction response is expected. Hence, this research shares a proposal of how a live predictions dashboard should expose prediction and insights upon data changes using trained models.

II. Related Work

Recent researches in supervised learning for failure detection in engine generation units have been influenced by studies focusing on data infrastructure, feature engineering, and monitoring through machine learning dashboards. For example, Vago et al. [10] conducted an industrial case study predicting machine failures from multivariate time series, underlining the impact of reading and prediction window sizes on model performance. Their findings indicate that deep learning methods are especially good in classifying data with a variety of time-dependent patterns before failures. In the same manner, the work of Amaya-Sanchez et al. [11] discusses the application of machine learning models for fault diagnosis in power generators. Their study emphasis the importance of feature engineering and robust data infrastructure. Also, it presents an overview of the performance of different supervised learning techniques for the considered operating conditions. Moreover, practical applications, such as the truck failure detection project by Wang [12], show how machine learning algorithms combined with real-time monitoring through dashboards work, therefore proving the practical utility of supervised learning in failure detection. Other major contributions include sensor-based failure detection systems presented by Meenatchi et al. [2], which demonstrate an innovative monitoring framework applied for predictive maintenance. Also, in the review of generative AI models for fault detection [1], the advantages of generative AI in traditional failure prediction systems has been discussed.

All these collectively point to the importance of data infrastructure, feature engineering, and monitoring tools in the development of effective supervised learning models for failure detection in engine generation units. However, none of this related work shows significant progress in the dashboard visualization platform to perform runtime predictions. This is where Ignition excels as the right tool to execute the models, gather data, store it in a database, and display insights using a dashboard with the latest technology.

III. PROBLEM STATEMENT

In this section, issues related to the use of electric generator engines as energy provision systems for the oil and extraction sector are addressed, as well as how their high availability is necessary to maximize production. The early detection of potential failures will enable immediate actions to prevent major damage. Monitoring various variables and standard Piping and Instrumentation Diagram (P&ID) schematics, along with human inspection, are not sufficient to predict failures, which is why a supervised machine learning model is used to create failure prediction models with low tolerance for false negatives. Additionally, the sensitivity of data extraction and processing methods is discussed. Furthermore, it describes how Ignition is the appropriate tool to orchestrate and synchronize information to utilize trained models integrated within the same production platform of the oil extraction company in Ecuador.

1) Ignition SCADA Overview and Machine Learning Integration: Ignition, by Inductive Automation, is a software that excels in many tasks for the latest industrial operations demands. Likewise, the utilization of Machine Learning algorithms is not the exception since this platform can manage model executions via Jython commands. The core engine of this software runs over Java Virtual Environment (JVM), this brings significant advantages like cross-platform compatibility and a large set of development toolkits over Java and Python libraries and languages classes. Ignition can seamlessly connect to a wide range of devices, using self-embedded protocols, and enabling the creation of unlimited tags, and provides this information across the entire platform scope. A tag is referenced as an object within the system that its main role is to hold data point values in real-time or historical mode, it also contains a set of configurations such engineering units, datatype, historization configurations, alarming, Jython based scripting, just to mention a few. The platform's versatility extends the customization of the solution upon applications design, connectivity and integration. Ignition acts as a central controller hub, integrating plant-floor equipment data, NoSQL or SQL databases and bridging the gap between production OT and IT layers. Using languages such as SQL clauses and Jython, and protocols such: Open Platform Communications Unified Architecture (OPC-UA), Modbus TCP/RTU, Allen-Bradley Ethernet/IP, Siemens S7, and Message Queuing Telemetry Transport (MQTT).

Regarding SQL databases, Java Database Connectivity API (JDBC) drivers are required to accomplish historization and transactional operations, supported by a wide range of well-known databases. Custom historization can be configured against each tag or set of tags like deadband, times between samples and high-level configuration like how often should table partitions time span and the age of data to keep. This system offers a wide range of custom scripting functions implemented with Jython. These functions are easy to use and are well documented. The framework where this function gets executed is in a Jython 2.7 interpreter. Within this set of functions, there is a dedicated package for data historization and extraction. The goal of these functions is to provide an easy way to request historized data. This is desired because internal operations perform well structured queries against the tables structure while we only need to specify the time frames to request historical data. There are a lot of possible functions to work with that Ignition supported natively.
However, this is not enough when it is desired to perform Machine Learning tasks, and it is required to rely on another programming approach. The main reason why it is required to relay in the capabilities of external Python calls is because Ignition, on its current version supports Jython version 2.7, which represents a challenge when dealing with libraries compatibility and Machine Learning implementations that are mostly developed and supported on dependencies of version 3 of CPython, known as Python. Nevertheless, it is totally possible and easy to invoke side *.py* codes written in CPython from the Ignition framework via subprocess or API calls [13]. These calls can be executions of predictions requests to a serialized Machine Learning model file, further details are described in the following section. Moreover, Ignition has a web-based visualization module that supports the foundation of web technologies used to set structure, style, and interactivity to modern websites. This web-based visualization module supports modern frontend implementations in HTML5, CSS3, JavaScript and a graphical designer tool based on React components that allows the development of views as desired and effortless deployment. Ideal for dashboards development, it allows instant web access to a Machine Leaning Dashboard application across various industrial clients or sessions systems from web browsers, desktops, mobile devices and more. With these features, relevant information that runs in a backend, like model predictions and feature importance selection, shows using modern frontend techniques.

2) Electrical Generator Bore Engine and Failure Events and SCADA: To provide electrical energy to an industrial process of Gas & Oil extraction field in a reliable way and in low-maintenance times. The Wärtsilä 16V32 is a suitable solution, it is a recurrently performing main engine that belongs to the family called Wärtsilä 32, known for its high-power density, low fuel consumption and long proven record. This engine machine supports multiple fuel types, meaning it performs well in a wide range of applications with various fuels. However, its primary focus is on the main uses such as Heavy Fuel Oil (HFO), Marine Diesel Oil (MDO), and liquid biofuels. All of this makes the engine more flexible to several types of energy sources in various productive sectors. For remote locations like offshore/marine platforms or extraction fields in the Ecuadorian Amazon Rainforest, this engine is ideal and important in that, it provides electrical energy supply to extraction platforms, drilling rigs, and security and mission critical systems. This is the reason why it is imperative to avoid failure events by making predictions to maximize the runtime hours and productivity. Wärtsilä 16V32 is a very great energy source for this operational type, considering its performance under high loaded operations, and where it powers mechanical tools, auxiliar sub-systems, and an electrical generator.[14]

Figure 1. Wärtsilä 16V32 Bore Engine

Despite its strength and capabilities, there are several factors that could lead to unplanned downtime. A common example of how failure can occur is that low-quality fuel may grow on or block the fuel system due to corrosion, whereas a disturbance in the cooling system may lead to overheating causing damage to engine components such as pistons or cylinder heads. The turbocharger, pistons, and crankshafts are particularly important components, any failure in these can spoil the working of the engine and it will require a major repair within downtime of weeks. The manufacturer has made the engine in such a way that it can be neglected for maintenance periods up to 24,000 hours and hence condition-based maintenance, which is essentially repairs before the problem might appear, it in turn will minimize the risk of sudden halts. Nevertheless, human inspection by visual and sonorous methods, is not capable of providing an advanced diagnostic of a failure event that is likely to occur. That is the reason why a probabilistic failure predictor model, designed and implemented with machine learning strategies, is more useful and advanced for these predictions. Furthermore, displaying features and failure events on a dashboard interface enhances the operability of the electrical energy generator system. According to the specifications of the engine, Wärtsilä 16V32 withstands tests designed to work, without failure-indicative data for the given period of years. This reliability makes the first step in the validation process more feasible, where the failure occurrences are measured by the criteria of engine downtime, which is used for continuous monitoring. For the high-level operational activities, the key metrics related to the engine insulation of fuel efficiency, power generation, thermal conditions, and some other performance-related key indicators are aggregated into a control system.

Ignition, the SCADA system uses communication protocols

to enable multiple devices connection capabilities, including the sensor unit devices of the Wärtsilä 16V32 engine. The collected data information can be distributed across the system in any desired manner and purpose such historization or operations of any flavor. The relevance of the system is that it can provide diagnosis of downtimes. In fact, this system is capable to show when the failure event had happened, as it is configured based on the alarms control logics, but this is not a predictive operation. Thus, the integration of Machine Learning models working side by side with field device data and SCADA systems is possible, this means that enhanced tasks can be developed powered with this integration. This promotes faultless motor management and discharge of the maximum up time. For instance, a regular visualization (top view) of a real production system visualization of the engine is shown fellow. Implemented in Ignition, this visualization shows some of the data variables and perhaps shows a high-level monitoring for process monitoring but limited on regard with any failure prediction insight.

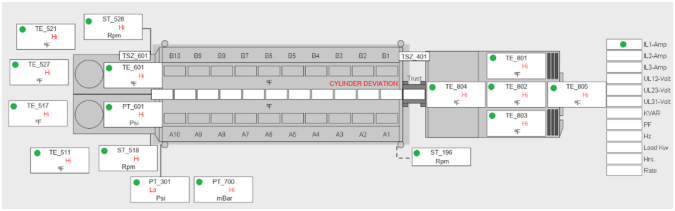

Figure 2. PI&D Visualization Standard Scada of the Machine (top view). Green dots shows some of the monitoring variables

The previous visualization lacks explainability for the user, as it does not display any information on feature importance or the probability of a failure event. In fact, it only intends to display all available data in one sight, which can be hard to manage and visually overloaded. For this research, some logical grouping of this engine generator process variables or features are shown as follows. Further results show the importance of these features agnostic to the corresponding model.

Table I
Important variable grouping of features.

Category	Variables
Control	Cooling water freq. conv. control value
Current	Generator phase current L1, L2, L3
Flow	Fuel oil consumption from flow rate, Fuel oil inlet flow, Fuel oil outlet flow
Frequency	Generator frequency
Position	Actuator position %
Power	Generator reactive power, Maximum allowed power, Generator active power, Generator apparent power, Generator power factor
Pressure	Receiver press. control, setpoint, sp. max. limit, HT-water pressure jacket inlet, CA pressure engine inlet, etc.
Speed	Engine speed, Turbo A speed, Turbo B speed
States	Engine stopped
Temperature	Main bearing temp., HT-water temp. jacket inlet, Exhaust gas temp., LT-water temp. CAC inlet, etc.
Voltage	Generator main voltage U12, U23, U31

3) Orchestration data flows, models and dashboard using Ignition: When it is required to perform orchestration data flows and deliver production data to a prediction model object in real-time while on production, a series of steps are carried on achieve this. The purpose of making this information available to the Machine Learning model is to enable the generation of failure prediction results as a probabilistic output as the productions happen. To start with, data source information on production operations is delivered to a Machine Learning model input. To accomplish this, it is important to select a tool that is capable of casting process data as utilizable information. Ignition, with its embedded driver's compatibilities and a series of industrial protocols, is capable to accomplish this. By connecting devices protocols to the system, the utilizable information lies on tags objects. These objects hold data values as the production happens. For instance, if a driver connected to a Transmission Control Protocol (TCP) device transmitting temperature in an analog signal, Ignition connects to this device and set the value input in a tag object and see the value changes live as its changes. So far, data values are available in the SCADA system. With its historization capabilities described before, this information can be stored as it changes. Therefore, data information records are stored and available as a source of consultation. Additionally, Ignition offers and wide variety of operations like scripting under timers and cronjobs that lead to gather process values, these are important to accomplish predictions as it enables execution pipelines. On its own, Ignition does not offer predictor models, expected from a Machine Learning system standard. However, it has Jython libraries that are capable of interacting with the operative system seamlessly. In this manner, data streams can be forwarded to a specialized model on memory and obtain predictions. This information can be easily collected back and whatever is required with it, like historization, reports or informative dashboards. For instance, the data flow is completed, and the remaining step involves deploying a dashboard that helps the end user to see information about the predictions and feature importance at a glance.

IV. Methodology

This section discusses the handling of the dataset and the structuring process required to make it available as a training object. It describes the requirements for obtaining data from a private oil extraction company. The dataset's characteristics, such as time intervals and sampling, are detailed, along with a general description of the training environment. Subsequently, the three failure prediction algorithms for engines Random Forest, XGBoost, and ANN-MLP are discussed. In general, all procedures have followed a strategic order of execution, including data cleaning and organization accompanied by normalization. Stratified training with cross-validation of some parameters is then applied, depending on the model, taking into account performance metrics and regularization [15].
Once this is completed, using the best parameters and the highest score, a results evaluation is performed through

learning evolution analysis or by reviewing confusion matrices. Next, the models are exported as serializable files, and, in the case of using Compute Unified Device Architecture (CUDA) - Graphics Processing Unit (GPU), instructions are given on saving these files with device index 0. Once this is done, the models are loaded from the serialized files to test the model's correct functionality in memory. From there, feature importance is extracted, and depending on the model type, one strategy or another can be applied.
Finally, it is discussed how this can be achieved with Ignition, and a distributed architecture is proposed, starting from field signals, the database for historical data collection, and the visualization of the model. Furthermore, the most important aspect of the system continuous predictions is highlighted, which sends information based on requests from the system management platform, Ignition.

The performance score metric utilized for this research is F2 that emphasis on recall [8] and is given by:

$$F_\beta = (1+\beta^2)\cdot\frac{P\cdot R}{\beta^2\cdot P+R} \quad \Bigg| \quad \begin{array}{l} P = \text{Precision} = \dfrac{TP}{TP+FP} \\ R = \text{Recall} = \dfrac{TP}{TP+FN} \\ \beta \in \mathbb{R},\ \beta > 0 \end{array} \tag{1}$$

Note: TP (True Positive), FP (False Positive), FN (False Negative)

1) Dataset Gathering and Training Environment: A private Oil & Gas company, with operations in the Ecuadorian Amazonic Forest, has been leveraging petroleum extraction for decades. This process demands high electrical energy and it is self-provided by a Wärtsilä 16V32 Bore engine generator. Ignition is the system that monitors and controls the operations of this machine. During machine runtime, a wide set of information has been produced, compounded by failures events that led to electrical backouts in the past. Ignition has been the system that manages information storage in a production MySQL database. To avoid overloading the production systems by crashing the servers and to minimize the risk of data loss and downtimes, the path for data extraction was stablished to gather information straight from the database. This has two mayor benefits, avoids the dependency of the SCADA system and obtains the raw data stored into the database. To recall, Ignition is capable to query the historical data against the database by its embedded functions which its simple and straight forward to do. However, this method has two mayor problems in these circumstances, data interpolation and the risk of overloading the SCADA server due to the time span of information desired of 28 months between 2019-05 and 2021-11. Therefore, the implementation of an advanced query is required to gather data information straight from the database as stored. To achieve the objectives of this investigation, a set of tasks were taken in the acquisition of the dataset phase:

- Privacy agreement and approval plan.
- Development of an advanced query, agnostic to the table structure given by the Ignition historization engine, developed and tested in a sandbox environment for data extraction.
- Query testing in production, failover plan, kill-switch, and approval pipeline.
- Query execution in controlled timeframes and database health monitoring; the query saves a *.csv* file with raw table information.
- Physical data extraction and transportation via hard drive. Extraction via internet tunneling was not permitted by the company.

A set of plain files *.csv* format is the result of the previous acquisition phase. Failure events were recorded manually and kept separately from the database in *.xlsx* files. Therefore, programming operations were required to merge this data information with the information previously collected. Merging this information is critical since it is the target feature value utilized for supervised learning. In Addition, merging failure events on a timestamp basis needs to happen under the Ignition context because data migration implements in a development environment. This means that, once data has been successfully merged into plain files by simple Python scripts, it needs to be pushed back into a development database under the historical table structure managed by Ignition. This historization context is capable to return a symmetrical tabular dataset, this is especially important because raw data operations are sampled in different and individual rated as defined by the operation. For instance, the samples taken from the engine generated frequency are higher than a level tank. Overall, the steps taken can be described as:

- Merging manual failure events with production historical values in plain files; 1 indicates a failure, 0 indicates no failure.
- Preparation of a development environment with Ignition and a MySQL database.
- Insert information into the database using the Ignition context via its scripting dedicated to manual history insertions. This function is called and available in the Ignition Designer framework as `system.tag.storeTagHistory()`.
- Extraction of the desired symmetrical dataset with all the required features, including the target feature, in the desired time sample using `system.tag.queryTagHistory()`.
- Loading the dataset files in the training environment and preparing them for data processing.
- Implement a supervised machine learning execution plan to obtain a predictor model with feature importance and insights.

These steps were taken to obtain the dataset to further operations such training and results analysis. Daily interval files at 10 second sample rate produced 8.1 million observations spread in 945 plain files and exactly with 89 features

to train. A High-Performance Computing (HPC) unit is required to handle the dataset size and procedures like Exploratory Data Analysis (EDA), training with hyperparameters tunning and feature selections information for each model in the scope of this research. A CPU with 250 GB of RAM and four 32GB RAM GPU per unit make up the training environment, which is capable to perform tasks within manageable times. As reference, with Randomized Search Cross-Validation operation on each model with about 8 to 15 hyperparameters distribution tuning, the training time took around 4 to 6 days, using measured capacity of the environment to not overload the kernel. Further details about training methodology are described hereafter.

2) Random Forest Classifier, Review & Training Script: In Machine Learning applications that search for atypical events, such as machine failures within high dimensional datasets, Random Forest (RF) provides benefits in terms of regularization and robustness. Regularization is inherently achieved using randomized feature selection for each decision tree generated. This minimizes correlations between trees and reduces overfitting, this is a critical advantage when handling complex datasets with large observations and feature counts [5], [16]. By doing this, the model effectively manages its complexity, this makes it suitable for detecting failure patterns that are challenging to predict [17], [18]. The criterion for RF helps to split quality, gini for binary classification tasks was used for this research. This setting lets the model evaluate the purity of each split, enhancing its ability to differentiate between failure/no-failure states by favoring splits that maximize class binary separation [19]. Alternatively, entropy criterion that is gain-wise may be used in scenarios that require enhanced precision for split decisions [20]. Another regularization technique for RF is its use of a validation technique that estimates the predictive performance of the model based on samples dropped during tree construction, offering an unbiased, this minimizes overfitting [21], [22]. With bootstrapping included, helps to sample using replacement, this introduces additional randomness and reduces sensitivity to noise.

Additionally, bagging known as Bootstrap Aggregation, lets the RF to gain model stability by creating independent decision trees based on randomly sampled subsets of data. Through majority voting by classification or averaging by regression, bagging aggregates the outputs of individual trees to achieve the final prediction, as result it reduces variance and improves predictive accuracy [23]. Regarding diversity, each tree building is based on distinct data samples, which helps in avoiding overfitting and ensures that the model property generalizes [24]. Furthermore, RF is known for handling missing values and providing feature importance insights, which are beneficial for features hierarchical observance [25].

This model is capable of handling categorical and continuous data types of underscores, therefore is applicable to diverse utilizations, in this case, for anomaly detection of a failure in a generalized prediction task [26]. These algorithm characteristics make it adaptable due to its validations and flexibility techniques with split criteria that make it an adaptable and effective choice of a failure event and generate valuable operability operation.

The methodology of the script execution for this Machine Learning algorithm is by executing a Python (3.10.12) code against the HPC in a Jupyter notebook file. It follows a workflow that the main goal is to obtain a serializable model for the failure prediction of this engine-generator. The first task is to read the dataset that is spread in several *.csv* files and load the dataset in memory. Doing this lets for perform a standard Exploratory Data Analysis (EDA) by doing data cleaning, filling in missing values and clipping negative values for consistency. Also, verification of missing or infinite values as well as removing features with zero variance to avoid redundancy dimensions. Onward, no further operations over the dataset are performed because it is desired to conserve data natural changes for accurate catch of information behavior. The dataset is divided into a training set and a test set through stratified sampling. This is important because it helps that train and testing set to have a balanced target feature value on each. A pipeline is created as an execution plan, starting with SMOTE for handling class imbalance by oversampling the minority class, then SelectKBest to select the top 20 most informative features, MinMaxScaler for feature normalization that is key for this type of dataset where the target is binary. PCA for dimensionality reduction of the components. Finally at the pipeline, the utilization Random Forest Classifier class for modeling. Them, it performs hyperparameter tuning with RandomizedSearchCV using stratified K-fold cross-validation. The performance metric is F2 score to maximize recall and minimize false negatives tolerance. After cross-validation cycles with best hyperparameters tunning such:

- SMOTE sampling strategy with values near 0.8 to reduce bias by creating synthetic values for the minority class.
- Number of neighbors for SMOTE sampling can remain low; synthetic data are created around this number, not exceeding 10.
- Establish the depth of the Random Forest (RF) tree to avoid overfitting. For this research, explainability is achieved with a depth not greater than 10.
- The estimator will enhance the RF model's performance and precision. It is used to manage training times, with values not exceeding 100 established for this research.
- Principal Component Analysis (PCA) for information control measured by the variance of the principal components. Both "None" and 0.95 are used to determine if reducing components is beneficial.

After model evaluation, the model runs the test set to get metrics such as the F2 score, confusion matrix, and classification report for performance assessment, then best parameters are assigned in the pipeline and into the model

so it can be saved as a serializable *.pkl* or *.pth* file. Also, loading friendly feature names from a reference *.csv* file to map feature names so that results can be more interpretable when displaying. The newly stored model is dump back and perform feature importance analysis. This is executed to understand which variables have the most influence for failure event predictions. Also, a decision tree from the Random Forest is visualized by Graphviz library to illustrate the model's decision-making process.

3) Extreme Gradient Boosting (XGBoost), Review & Training Script: XGBoost is an advanced Machine Learning algorithm and highly efficient, it was developed to be robust in performance for classification and regression tasks. Unlike standard Random Forest, XGBoost utilizes the gradient boosting an ensemble technique that is corrected by gradient descents from previous errors sequences [6]. XGBoost is especially good at handling non-linear and complex relationships, applicable to the nature of the dataset of this research, and is better in accuracy compared to standard methods, especially where high-dimensional is present in the dataset.

XGBoost has various hyperparameters available to enhance regularization and control model complexity to avoid overfitting. Regularization parameters are lambda, which is L2 regularization, and alpha, that is L1 regularization, resulting in a penalty on the weights of features. This makes the model more conservative which benefits generalization [27]. The learning rate, which controls how much each new tree contributes to the ensemble, balances again accuracy versus the risk of overfitting. Smaller values of learning rates slow down learning but increase model stability. Speaking of performance, this model has a parameter for the decision-tree method that allows to use the GPU acceleration taking advantage of devices with CUDA for faster training times. The histogram-based tree method is optimized for execution on a GPU, hence is part of the strategy of improved training [28]. Another characteristic of XGBoost is capable of handling imbalanced data by using the `scale_pos_weight` parameter, which weighs negative and positive classes to make the performance more balanced for skewed target's dataset distributions. In addition, to control the portion of data by features it gets subsampled for training set of each tree by using `colsample_bytree`. These add regularization because it reduces the dependence on any specific sample or set of features [29].

For the training script phase, it follows a workflow to predict the failure of an engine-generator, using the approach of using GPU acceleration with CUDA devices enhancement. To start, the dataset is loaded memory via various files reads in an appended data-frame for cleaning using first filling in missing values and clipping negative data to remove unreal noisy data. Perform Quick EDS for check of missing and infinite values and remove features with zero variance to eliminate redundancy or weight-less component. Then, split the dataset into training and testing sets utilizing stratified sampling to keep class distribution. Next, it's required to initialize a Dask CUDA Cluster to distribute the computations load across all GPUs while training. For these steps, it was configured for specific CUDA devices available. This helps to use dynamic allocation and therefore allows the model to move across different CUDA devices during processing. This is crucial for optimizing resource utilization in shared hardware environments. Finally, all gets pushed to device to CUDA 0 before saving the model, because most devices are very likely to have at least one CUDA enabled GPU, this means it was compatible when loaded is a separated hardware with different system specifications. As follows, a pipeline is created including SMOTE for class balancing by oversampling the minority class, MinMaxScaler for feature normalization, this is important because as mentioned before, it improves the convergence and control dominant features. PCA for reducing the dimensions, and an XGBoost Classifier optimized for GPU computation. Hyperparameter tuning such:

- The number of neighbors for SMOTE sampling can remain low; synthetic data is created nearby these numbers, not exceeding 10 in this research.
- PCA analysis for information control measured by the variance of the principal components. "None" and 0.95 was used to determine if reducing components is beneficial.
- Establish the depth of the tree to avoid overfitting. For this research, explainability is achieved with a depth not greater than 10.
- The estimator will enhance the Random Forest (RF) model's performance and precision. It is used to manage training times, with values not exceeding 100 established for this research.
- Learning rates of 0.007 or 0.01 balance the trade-off between the speed of convergence and the impacts on the accuracy of the model. A smaller learning rate slows down learning but can lead to a more accurate model.
- Alpha (L1 regularization), set to 0 for no regularization and 0.1 to apply a penalty to overfitting behaviors.
- Lambda (L2 regularization), set to 1 and 1.1, with a base regulation of 1 and 1.1 as a stronger penalty to overfitting.

As follows, using RandomizedSearchCV with repeated stratified K-fold cross-validation for the F2 score metric that as mentioned before, emphasizes the recall and reduces tolerance to false negatives. An important observation is that the imbalance target data is managed by the XGBoost class library itself, that is the reason why a sampling strategy is not required for this approach. After getting the best parameters and inspecting the evaluation of the model using metrics such the evaluation of the score, confusion matrix and classification report understand the model performance. The best model parameters sets to the pipeline and exported for later use as a *.pkl* file. However, before saving its crucial no move the model to CUDA 0 because most of external hardware to load the model

may at least the one. This avoided leading errors. Then, reading friendly feature names from a reference file helped to talk about feature importance. Followed by visualization plotting a decision tree from the XGBoost model helps to understand the classifications paths. Finally, influential feature observation analysis and plot that the model object has an attribute generated in the training phase under the hood.

4) Artificial Neural Network Multi Layer Perceptron (ANN - MLP), Review & Training Script: Artificial Neural Networks (ANN) fully connected with Multi-Layer Perceptron (MLP) is the foundation of the ANNs variations. This type of network is the most sophisticated for binary classification problems. An MLP consists of multiple layers of nodes or neurons, including an input layer, one or more hidden layers, and an output layer. The network learns to model the relationship between the input data and target labels through a process called backpropagation, which optimizes the weights in the network's connections to minimize errors in prediction. MLPs are sufficient for binary classification, as they can model complex nonlinear relationships between input features and the binary target simply by iteratively adjusting the weights to make them adaptable to various types of classification problems [30].

The reason being that MLPs are considered sophisticated models is mainly because they can approximate any continuous function at any level of accuracy with enough hidden neurons and training data. This allows MLP to learn complex patterns from data sources, making it much more powerful in comparison with simpler models like logistic regression or decision trees. In addition, MLPs allows the usage of functions like ReLU or sigmoid, which introduce non-linearity in the model to learn and model generalization compared to linear models. This is particularly important when there is a highly non-linear relationship between inputs and outputs, as often happens with failure events of engine-generator from this analysis [31].

Using PyTorch for Python, the MLP can be implemented easily by inheriting the class torch.nn.Module. PyTorch's Sequential model allows users to stack layers of neurons. Each layer is a fully connected layer, with activation functions applied between the layers. PyTorch is built on a framework that allows efficient training and computation hardware by enabling CUDA GPUs. Training MLPs is hence much faster, especially for big datasets. This capability is very important to allow the training of deep networks, efficiently using the computational power hardware [32]. The flexibility and simplicity of the implementation of MLP in PyTorch make it a suitable tool to deal with binary classification, allowing to experiment and deploy machine learning models swiftly [33].

For the scripting methodology phase some steps are followed. First, it configures the computation environment, listing the availability of CUDA-enabled GPUs and setting the appropriate device while printing out the device type along with the number of CPU cores for processing. After that, several *.csv* files are loaded and appended in a dataframe. Straight forward simple EDA process is done by filling in missing values with zeros and clips negative values. Thereafter, features and the target variable of machine failure are separated and drops the rows containing missing or non-finite values, this avoids problems in the training phase. After that, stratified splitting of the data into training, validation, and test sets to maintain class distribution in all three sets, is performed. Then, features removal by having zero variance, this helps to reduce redundancy and remove undesired contributions. Next, scaling the rest of the features using MinMaxScaler to normalize these between zero and one, useful when target feature its binary as well. To handle class imbalance, SMOTE algorithm is used upon training data to oversample the minority class. Then, it instantiates an ANN model in PyTorch using class inheritance, with tunable numbers of hidden layers and dropout rates for regularization, wrapped into a scikit-learn-compatible classifier. With that, it is possible to execute stratified hyperparameter optimization by RandomizedSearchCV. Some hyperparameters tuning are:

- Hidden layers with dimensions not larger than 5 and not more than 40 neurons per layer. Some architectures like balanced and bottleneck were tested.
- Dropout rates not exceeding 0.2, used as a regularization technique to randomly drop neurons during training.
- Learning rates not exceeding 5×10^{-5}, used to control the gradient steps. Smaller values result in significantly longer training times.
- No more than 100 epochs for model convergence, combined with early stopping. This ensures stabilization of performance scores and loss values.
- Batch size not exceeding 64, determining the number of samples processed before updating model parameters. This is beneficial for memory constraints and introduces stochasticity to escape local minima.
- Patience not exceeding 35, which is lower than the total number of epochs. This parameter is used to prevent overfitting by waiting for validation loss improvement before moving to the next epoch.
- Accumulation steps not exceeding 8, which accumulate gradients of this size before model updates. This effectively increases the batch size, providing better stability while overcoming hardware constraints.

On the training step, an early stops training phase has been added to avoid overfitting using a validation loss. It subsequently calls for the evaluation of the best model, trained on the test set for emphasis on recall with F2 score. Before saving the model, it is advisable to move the model to CUDA device 0 to increase compatibility across different hardware setups. When training is completed, and the evaluation scores look acceptable then the model is saved as a serializable file, this allows make the model portable. As further steps, plotting the metrics of training and validation along epochs for visualization of performance

metric. This is followed by the substitution of feature names with descriptive labels for easy interpretation names. Finally, a feature importance analysis is performed by using Integrated Gradients that attributes the output of the model to its features ingress by integrating the gradients along a path from based on baseline input to an actual input. A plot helps to see the importance of the features.

5) Machine Learning Prediction using Ignition Orchestration: As mentioned earlier, Ignition provides the main dataflow control, as it has capabilities that allow the deployment of the explanatory Machine Learning dashboard for machine failure predictions. For a better understanding of the platform, it's important to mention that this software offers comprehensive digital transformation tools as a solution for industrial operations, allowing unlimited connectivity, data collection, and application design capabilities. It also empowers users to streamline processes, increase productivity with compatibility and flexibility [34].

Scripting in Jython is a key characteristic that allows to easily accomplish and leverage a predictor model from real-time or historical data. These data values are being generated and stored by the system in a configurable and automated way. With a server-centric model and cross-platform compatibility, various architectures configurations can be deployed to fit specific requirements. The following architecture proposal fulfills the scope of the research and can be applied for a standalone architecture on-premises capable of SCADA operation with an active and integrated Machine Learning phase. The reference is based on spread-out server services with a dedicated server for each operational role.

Any compatible database engine, ML Server Machine and the Ignition hub orchestration system; this server's disposition may have variations if required. For example, the database server might be separated but the Ignition system and the Machine Learning unit might be on the same server. In another proposal, all the services can coexist on the same server as separated services, though this is a possible configuration, it's recommended only for development and testing environmental purposes. For deployment in the production environment, it is better to have these services separated into isolated by hardware or virtualization.

Having mentioned all the capabilities of integration, flexibility and compatibility of this software, it's crucial to understand how this system is capable of handling data orchestration. When data values are collected and connected using industrial driver protocols, it can be published into the internal OPC-UA server. These data values are available via tags subscripting, allowing to build a Unified Name Space (UNS) upon data structures and data points that are available in the system's gateway scope. These tag objects not only contain the operation value on their own, but they also provide quality and timestamp as part of a qualified value. This means that the data information is not only the value of any type, but

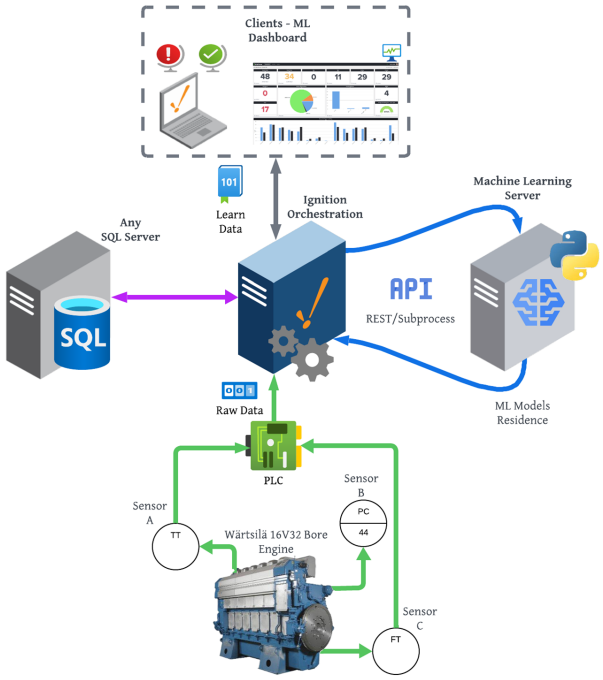

Figure 3. Data Flow Orchestration for Trained Model Utilization.

it also has the metadata related to any value changes. More information and configurations are available as properties, an important property is the enable of the historization option and the source place to store. The source place is known as the history provider that it's attached to a database connection. Under the hood, the Ignition platform does the hard job that prepares the corresponding queries against the customizable structured tables. This means that data values, qualities and timestamps are available to be customizable as the developer needs. It only requires specifying which variables are required in a specific range of time and sample rate. This information is essentially variables in a script runtime and can be managed in any possible way. Ignition has more features, where combined developments can bring in customizable data flows as complex as required. In this case, the executions of scripts under a clock frequency is used. The intention of these timers' scripts is to pull in historical information as required and expose it to a side subprocess call where historical data sends as JavaScript Object Notation (JSON) files to the Machine Learning Phase. This side script takes information from the input and performs executions that load the model in memory and the probabilities of failures are obtained. This information is pulled back to the platform and allows that the prediction made by the model can be exposed and evaluated immediately. The execution rate of the timer script are configured as decided cycles, it's recommended to allow the current process to complete before making a new call with the Fixed Delay. With each cycle, predictions serve as updates for a Machine Learning dashboard with probabilistic failure values and feature importance analysis.

V. Results

The analysis of results is closely tied to the origin of the data for the specific use case. There is a certain degree of generalization, but it is not expected that these models will perform optimally for similar study cases. The strategy for obtaining results was based on retaining the best parameters for the highest performance metric.

Please visit the *Git Repository* to access the source code and results of the implementation [35]. The source dataset is not shared due privacy policy agreement.

1) Random Forest Performance and Feature Importance: The model obtained from Random Forest algorithm training on failure detection in the engine-electric generator showed satisfactory results, especially when the concern is not to miss any event of a failure. For the model training, data inconsistencies in terms of missing or infinite values were handled to ensure clean and robust training. Stratification into training and testing sets shows a balanced representative in non-failure and failure events with better results. This is because failure events from actual data is naturally imbalanced. This balance preparation in data has ensured that the model learns from the identification of failure patterns with no tolerance of missing any failure from target [36]. The pipeline includes important steps, including SMOTE for oversampling the minority class for the model that requires enough data of the failure to learn. Then there are feature sections with SelectKBest for the best 20 features and MinMaxScaler for data normalization. Although PCA was used for dimensionality reduction, the model score turns out better without it, therefore retaining all selected features enhance performance metric. The RandomForestClassifier best parameters were given by `max_depth` equal to 8 and `n_estimators` equal to 70 obtained by RandomizedSearchCV with stratified cross-validation to balance precision and recall.
This model was then evaluated on a confusion matrix showing 1,417,214 true negatives (TN), 91,005 false positives (FP), 239 false negatives (FN), and 124,502 true positives (TP). This is further supported by a value of the F2 score of 0.9450, showing that this model is very good at minimizing false negatives. Thus, its reliability to capture failures is satisfactory. The recall for failures was 1.00, with no events of failure being missed. However, the precision of the failure class was 0.58, this means several false positives. This might result in occasional false alarms while in operations, this is the reason why probabilistic outputs are shown in the ML dashboard operation. The decision tree path is wide and hard to visualize. By feature importance observation and low impurity values, a summarized tree is shown below.

The feature importance analysis showed that the main predictors of engine-generator events show interesting results. The most important for this model is the "Actuator position % feature", whose importance scores were 0.355 that is the most important in predicting failures. This feature is related to the control of the voltage delivery and frequency while the load changes. This is like the

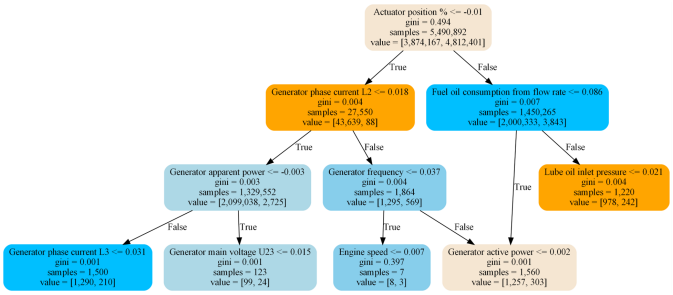

Figure 4. Random Tree Decision Tree Summarized by impurity.

acceleration system of any bore engine. This shows that the load is too large for the engine, or the control of this actuator is poor causing failure events. Other important features to observe from insights given by this model are "generator phase current L2" with 0.139, "generator apparent power" with 0.119 and "fuel oil consumption from flow rate" with importance score of 0.106. These form the basis of pointing out control mechanisms that monitor and provide for power stability and fuel flow, which act as predominant indicators toward the early detection of potential issues of an engine.
However, this has a direct relationship with the shutdown phase of an engine, meaning that, in a way, it is somewhat expected. On the other hand, there are other characteristics that are not highly representative but are still worth observing among the most important ones, those are. "Fuel oil consumption from flow rate", "Lube oil pressure A and B" and even the "hot water pressure on jacket inlet". The image below shows the 20 most important features from the model output meaning that those are worth monitoring in that order in the ML Dashboard.

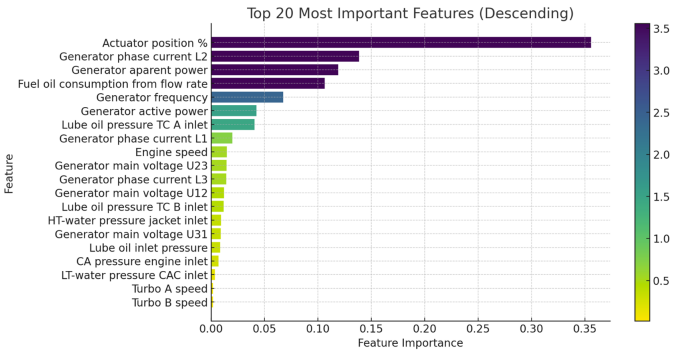

Figure 5. Feature Importance Results by Random Forest.

Potential improvements are focus on the parameters optimization using alternative oversampling methods to better handle the imbalance. This might help to improve precision by reducing false positives and avoid the “crying wolf” effect. Additional feature engineering and fine-tuning parameters with a larger stratified fold value and adjustment of some parameters given by the class.

2) XGBoost Performance and Feature Importance: The XGBoost model was trained using a pipeline, to handle

imbalance SMOTE was used for balancing the classes and show satisfactory results. Using `k_neighbors` equal to 9 and `sampling_strategy` equal to 0.8. Then MinMaxScaler for feature data normalization, followed by PCA to reduce the level of dimensionality. Again, using dimension compression given by principal component analysis did not make a positive contribution to the model, better performance metric score was obtained without this algorithm.

By using RandomizedSearchCV for hyperparameter tuning applied to XGBoost model was done through a grid search over a range of parameters. The results of this implementation from which the most adequate parameters values obtained are `max_depth` equal to 8, `n_estimators` equal to 150, `learning_rate` equal to 0.01, alpha equal 0, and lambda equal 1. Some observations about these results are than the values of this estimator and max deep are the highest offered as a selection which means those values could be larger but leading to overfitting. Regarding the learning rate, it was observed that reducing this value did not show a significant improvement in the score and increased the training time significantly, therefore the value provided seems to be adequate. Regarding regularization, the model reacted by avoiding penalties and the slight increase in these parameters was rejected. The best hyperparameters were chosen, taking into consideration not only better performance but the least amount of false negatives tolerance to be considered in this failure detection task. Stratified cross-validation ensured classes to be balanced while training and testing with better results.

The confusion matrix provides a summary of the performance of classification. These values are 1,493,314 true negative (TN), 14,905 false positive (FP), 37 false negative (FN), and 124,704 true positive (TP). The F2 score was 0.9908, showing performance of this model was in finding just a few errors and essentially making sure not to miss any potential failures. Precision on the positive class was 0.89, while the recall was 1.00, showing that there were no missed failures but still some false alarms, like RF. Also, the decision path of a summarized tree is shown below based on the selection of the less impure values of features. This shows a cascade effect that describes a failure event learned by the model.

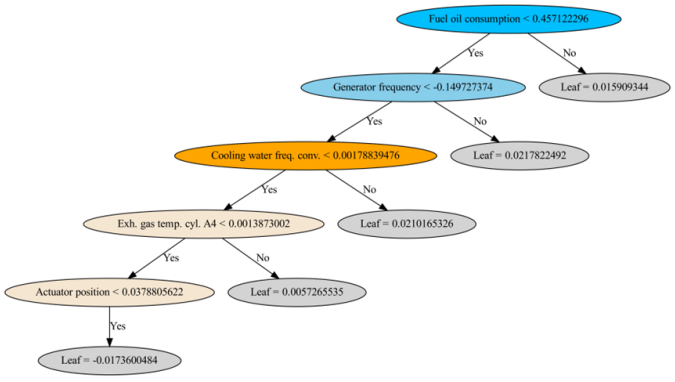

Figure 6. XGBoost Decision Tree Summarized by impurity.

Feature importance analysis from the trained XGBoost model, used as top contributing features investigation, shows consistency as the ones obtained in RF. The most important feature was "Actuator position %" with an important score of 0.544, followed by "generator apparent power" of 0.112, "exhaust gas temperature for cylinder B1" of 0.050 and "fuel oil consumption from flow rate" with 0.042. These are ranked high because those are highly influential in predicting possible engine failures. These features have a tight relation with control mechanisms, power metrics, and temperature indicators. This gives information on which parameters are highly informative regarding the health of the engine, thus guiding monitoring and preventive actions upon these observations. Further analysis indicates that some features are expected, but others seem to be more interesting like temperatures. This suggests that temperatures observation is important against a failure event [37].

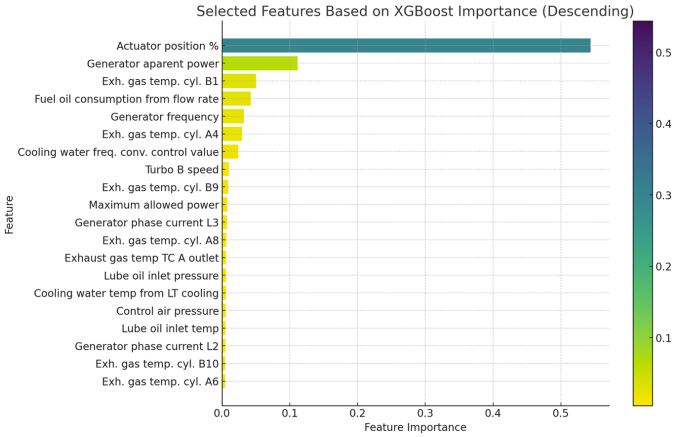

Figure 7. Feature Importance Results by XGBoost.

Some improvements are focus on oversampling and avoiding overfitting. This could provide fewer false positives and get a performance score more realistic and generalizable. Moreover, more feature engineering could be impremented to make the model learn even more complex relationships. Fine-tuning over `max_depth` and `n_estimators` would further allow get a more generalization for this model.

3) ANN MLP Performance and Feature Importance: The proposed ANN model for detection includes an input layer with 86 features, three hidden layers with each unit composed of 15 units, and one neuron in the output layer. ReLU has been applied as an activation function in the hidden layers, while dropout is used at a rate of 0.2 to avoid network overfitting. The model is trained by using the Adam optimizer, learning rate 5×10^{-5} for 100 epochs. Early stopping is set to a patience of 35, which allows it to stop training when performance on the validation set has stabilized.

The next step was to do hyperparameter tuning with RandomizedSearchCV. The best parameters were patience 35, epochs 100, learning rate 5×10^{-5} as now improvement was seen with any lower value, the number of hidden layers was 3, units in each layer 15, dropout 0.2, batch size 64, and

accumulation steps equaled 8. Train on GPU CUDA must be performed previous monitoring system status and use an available device; this is being used for faster computation. The F2 score plot shows training stabilization around 0.98, while the validation F2 score increased gradually to 0.95, hence strong model performance on training data and a good generalization for validation data.

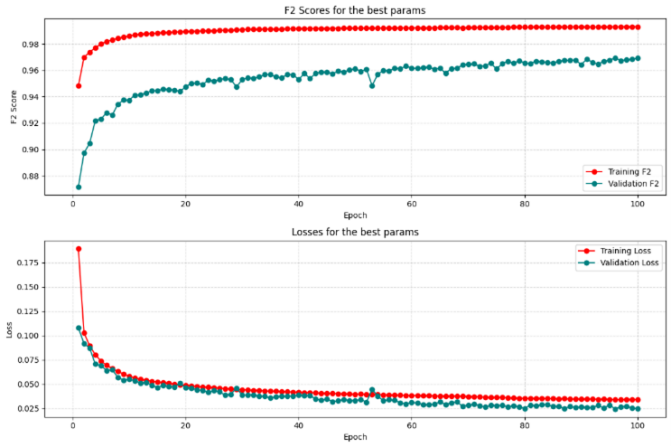

Figure 8. Performance Score and Loss over Epoch by ANN.

The loss plots are an indicative of a decrease in both training and validation losses quite stably over epochs. The initially high loss decreases rapidly in the first few epochs and then starts to converge. There is not much difference between the losses of training and validation, which is a good indication that the model is avoiding. This demonstrates that dropout and early stopping as regularization techniques took effect and hence this a sophisticated model with high performance score.

The feature importance scores extracted using Integrated-Gradients draw attention to the important inputs considered by the model for predictions. In particular, "generator reactive power" has the highest importance, close to 260, followed by "generator phase current L1" and "fuel oil inlet flow," both close to 250. Other features are "fuel oil consumption from flow rate" and "generator apparent power," both close to 230. These highly important features provide critical insight into what the model is focusing on to identify possible failures. Moreover, when giving a more detailed observation to features importance beyond the direct relation of an engine stopping, the following observations are found. Features like “Main bearing 6 & 8 temperatures” are shown, other like “Gen. ND- bearing temperature”. Ones again, domain professional observation suggests that temperatures are an important variable to observe that might suggest a failure event.

Some improvements could be aimed at using larger or additional hidden layers to model to catch more complex relationships. A learning rate scheduler can dynamically adjust the learning rate through training, hence enhancing convergence while training. Further regularization can be carried out by combining L2 regularization and dropout may reduce risks of overfitting. Ensemble methods, by

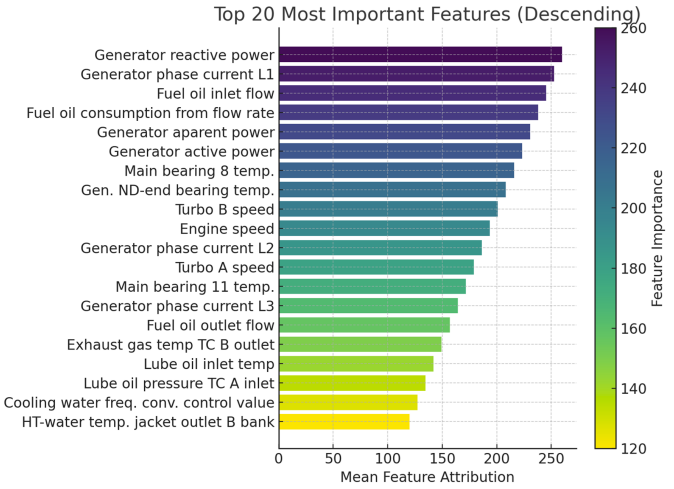

Figure 9. Feature Importance Results by ANN.

stacking this ANN with other classifiers, may combine diverse strengths and improve robustness model. This would also involve additional analysis variables by feature importance and enable to increase interpretability and predictive capabilities for critical failure indicators.

4) Results Overview: In general, it is observed that the performance metrics results are adequate. It can be seen that, in all cases, each algorithm achieves the expected results for predicting failures in the generator engine. Some aspects can be refined, and a deeper evaluation of the models' overfitting can be conducted. However, the applied regularization provides a certain degree of confidence that the results obtained, within the scope and context of this research, are the desired ones.

Table II
Results Summary Evaluation Report and Feature Importances

Algorithm	Evaluation Report	F2 Score	Top 3 Feature Importances
Random Forest	False Negatives: 239	0.945	Actuator position %, Generator phase current L2, Generator apparent power
XGBoost	False Negatives: 37	0.9908	Actuator position %, Generator apparent power, Exhaust gas temp. cyl. B1
ANN MLP	Training Loss: 0.037	0.98	Generator reactive power, Generator phase current L1, Fuel oil inlet flow

The observation of the most important features shows consistency and some similarity among the models' results. By removing the direct relationship between failure and certain features, others become more noteworthy, such as:

- Fuel oil consumption flow rate
- Lube oil pressure TC A inlet
- Exhaust gas temperature cylinder A4
- Cooling water frequency converter

- Main bearing 8 temperature
- Generator ND-end bearing temperature

By observing these features, which are not directly related to the occurrence of an engine shutdown, they are still important to consider, although their importance, according to the model, is lower than that of directly related features. The machine learning dashboard provides a visual aid by running the models based on the real-time field variable data that continuously flows in. This information is processed, and probabilistic predictions are displayed. This allows certain observations to be made. That simultaneous increase in exhaust gas temperature and bearing temperatures could suggest excessive loading or problems in the cooling system. Also, a decrease in lubricating oil pressure along with an increase in main bearing temperature could indicate a failure in the lubrication system. These observations are based on a deep analysis of feature observations and domain expertise.

5) Machine Learning Dashboard using Ignition: With trained models available as serialized files, a Machine Learning Dashboard hosted by Ignition displays crucial information about failure events. The workflow operation to obtain probabilistic predictions occurs as soon as new data comes to the system in real-time and predictions executes. On a configurable time-rate or crontab operation, in this case 2 second unlimited loop, it uploads these features data information from the Ignition system to the Machine Learning backend and from back to Ignition. The Ignition system performs this execution using a subprocess task script library or API calls with bottle or flask libraries that manages and triggers a script command for prediction results. For this implementation, the subprocess was used. With that information in place, some of the visualizations read and show these values on the main dashboard for failure predictions. Some other visualizations, such the history explorer with sorted feature importance, also takes information from the Machine Learning layer to deploy visualizations. In addition, the attributes explorer for advanced users, lets access further information of the training phase and the models obtained as attributes summary. Finally, a live input/output predictor, that lets to visualize data input and analyses probabilistic predictions outputs live.
To recall, the information returned by the Machine Layer system is displayed as follows. The main dashboard is meant to show the probability of failure events given the current data. Those predictions are shown by the three models trained before: the Random Forest, XGBoost and ANN. On the chart below, there's also a dynamic feature importance chart that helps to monitor the values as it changes overtime. This lets to see the feature importance and the prediction probability in a glance of what's happening with the operation in term of event failure prediction results.

The history explorer has a menu available to extract and see all the historization data that it's been stored as it changes in a database using the historization engine of

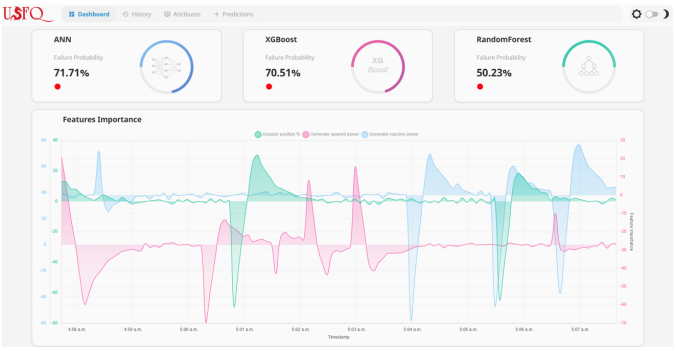

Figure 10. Main Machine Learning Dashboard about Failure Prediction of Available Models and Feature Importance History.

Ignition. The table on the right gives crucial information about real-time value changes and the current value of operation for that variable.

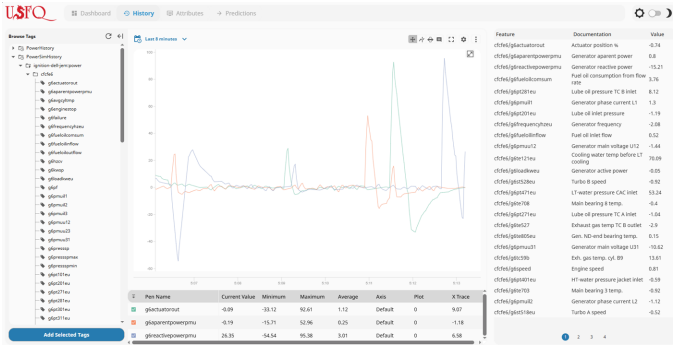

Figure 11. History Explorer and Entire Feature Importance.

The attributes analyzer enables exploration and retrieval of important information of the trained model, interesting values to visualize are for instance, best score, best parameters and further model details. This information is shown in a JSON format that is easy to read and may be the foundation base of further training if desired. This information has been gathered using Python attributes extractor of the object, in this case a Machine Learning Model.

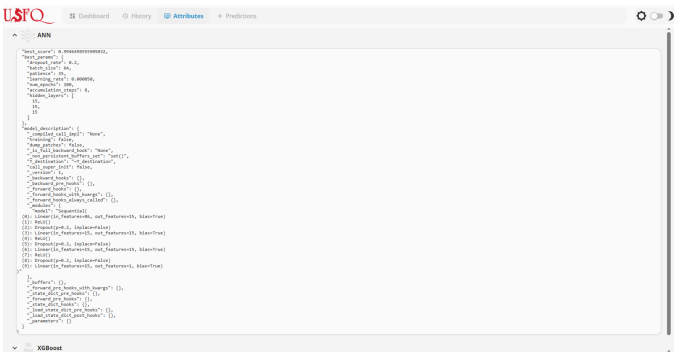

Figure 12. Model Attributes Visualizer on Training.

As follows, the following visualization of its data describes while doing the predictions. It shows data-values on real-time of all the features used for the training as it changes

as output, shown in JSON format. This “Predict Data” it’s the information that would be delivered to the model and a little timer on screen shows the time running until a probabilistic response it received.

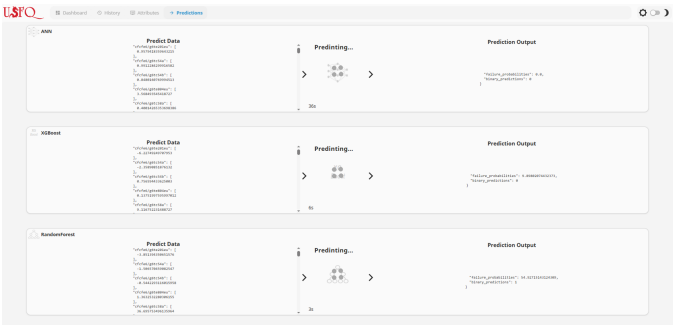

Figure 13. Live Predictions from Actual Data In and Predictions Out.

All these visualizations let the user navigate and explore the state of the predictions plan while in production. These applications support unlined visualization sessions without overheating the predictor system, since it is only executed by the core system called the gateway. This gateway executes the probabilistic operations and publishes the same calculations to the session as a single source of truth. Additionally, motor generator data can occur once in each time frame, ensuring that this is the same live data for all users connected to the application.

VI. Conclusions

The comprehensive framework through which this research uses Supervised Machine Learning to detect and predict failures for an engine-generator unit will integrate machine learning methodologies, along with data infrastructure, feature engineering, and operational monitoring through a machine learning dashboard. The insights provided by this research through data science along with industrial operations techniques provided technical and practical solutions for the efficient operation and Machine Learning foundations for this industry.

Data acquisition journey, which was facilitated by the Ignition SCADA platform, shows that this phase is critical for this research goal. Data extraction over 8 million data points across 89 features of 26 months of operational history, it’s easily supported with Ignition. The failure events were labeled with care by hand and merged with the historical data, enabling to obtain a labeled tabular dataset on a separated development environment apart from production framework.

This research underlines the importance of feature engineering and processing in Machine Learning Pipelines. Some techniques to handle imbalance were employed to remove bias due to the nature of the target feature. This is necessary due to the nature of data as it shows a high non-failure occurrence against failure events. Therefore, some considerations have to be implemented in the dataset so that the models could learn effectively from failure and non-failure events. Further tunning techniques were tested and integrated like Principal component analysis that, for this dataset type and problem statement, did not offer a contribution. Some other algorithms like the Selection K-Best were beneficial for models such as Random Forest. This enabled the identification of those variables that are most crucial for the engine generator’s performance. These processes not only serve to enhance model performance but also provide actionable insight into which the operational parameters that are most correlated with failures.

Three machine learning models were used comparatively as binary classification algorithms Random Forest, XGBoost, and ANN. Regarding Random Forest, as standard and easy to interpret, model, obtained strong recall scores to avoid missing any failure events. Since it gives a ranking for feature importance. There was adequate insight into which variables were critical in the determination of failures. About, XGBoost as a robust method, its gradient boosting technique gave it the most accurate delivery with minimal false positives and without substantially losing recall. However, having a high score, may lead to the “Crying Wolf” effect as false failure predictions alarms may involve unnecessary warning while on operations. However, since the goal is to be not tolerant with false negative events, this might be manageable. Regarding Artificial Neural Networks, a sophisticated method, the ability to capture nonlinear and complex interactions in the data enabled it to model complex feature interactions. Performing feature importance analysis through Integrated Gradients allowed more insight into the underlying pattern that exists within the data and the failure events[38]. The F2 score harmonic mean metric emphasizes recall over precision, this is a more accurate performance metric due to the nature of the problem statement. Hence, the Random Forest gave an F2 score of 0.945, while XGBoost performed best overall in terms of precision with an F2 score of 0.9908, thus assuring it to deliver reliable predictions with very low levels of false negatives. ANN, being flexible and robust in architecture, strikes a good balance in performance on recall and precision of view for interaction of features and dynamics of failure F2 score of 0.98. An interesting observation about ANN is that a symmetric layer’s architecture shows better performance scores.

This research important contribution is the development of a Machine Learning Dashboard hosted on Ignition SCADA platform. It serves as a link between analytics and predictions layer and industrial operations, converting model predictions outputs into accessible and actionable insights for the production stage on decision making and observance of features. The integration of real-time monitoring dashboard with probabilistic predictions and a feature importance visualization within this dashboard enhances the users to make informed decisions, early actions and even prioritize maintenance activities. This research emphasizes how a well-structured pipeline of data, advanced analytics, can work out failure predictions for these

Oil & Gas operations, highly dependent on the availability of an engine generator operability. The application of supervised learning models on historical and real-time data shows the relation between data science and operational management. Furthermore, the deployment of the Machine Learning Dashboard ensures proper communication of these insights between technical analysis and operational decision-making.

The applied methodology shows commitment to rigor and innovation in the study, there is not a clear proposal on the literature of how to accomplish a Machine Learning system fully integrated with Ignition. From the complexity of data acquisition process to the strategic selection of adequate Machine Learning models, all these elements in the study were driven toward providing maximum reliability and applicability of the findings. Advanced methods like hyperparameter tuning and cross-validation ensure that the performance for all models was optimized for generalization applied to this engine-generator. Moreover, interpretability thought feature importance is relevant for this research. For instance, the feature importance analyses show in general that direct relations between a stopped engine and parameters that inheritably describe a stopped motor, are highly related. Been this a positive sight of model learning is not a considerable contribution. This means that intermediate nondirected relations with a stopped motor are more interesting to observe as temperatures. Another contribution of this research is to provide a foundation aimed to better industrial operations powered by Machine Learning insights. This means that the probabilistic predictions from the models may inform maintenance schedules, reducing downtime and optimizing resource allocation. Perhaps, the most adequate step would be a further model sharpness for building and testing ensembles of alternative algorithms. Also, the incorporation of transfer learning integration with predictive models would enhance predictive performance for real-time operations and monitoring.

This therefore shows what kind of change supervised learning can bring about in industrial applications via advanced SCADA systems such Ignition. With the full utilization of data infrastructure, feature engineering, and deep machine learning models, the problem of failure detection might be solved with this solid foundation approximations. This deployment unsured that translation of data into decision making practices was effectively connected to the analytics phase operations and back. This research moves the frontier in predictive failure events and raises the expected results regarding machine learning applications in industrial processes as Oil & Gas in Ecuador.

References

[1] A. Shaheen *et al.* (2024) Machine failure prediction using joint reserve intelligence with feature selection technique. Taylor & Francis Online. [Accessed: Mar. 3, 2024]. [Online]. Available: https://www.tandfonline.com/doi/epdf/10.1080/1206212X.2023.2260619?needAccess=true

[2] M. V. T., S. Gnanambal *et al.*, "Comparative study and analysis of classification algorithms through machine learning," *International Journal of Computer Engineering and Applications*, vol. 9, no. 1, pp. 247–252, 2018.

[3] Z. Jaadi. (2024) A step-by-step explanation of principal component analysis (pca). Builtin. [Accessed: Feb. 29, 2024]. [Online]. Available: https://builtin.com/data-science/step-step-explanation-principal-component-analysis

[4] H. Tatsat, S. Puri, and B. Lookabaugh. (2020) Machine learning and data science blueprints for finance. O'Reilly Media, Inc. [Accessed: Mar. 1, 2024]. [Online]. Available: https://www.oreilly.com/library/view/machine-learning-and/9781492073048/ch04.html

[5] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001. [Online]. Available: https://link.springer.com/article/10.1023/A:1010933404324

[6] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 785–794. [Online]. Available: https://dl.acm.org/doi/10.1145/2939672.2939785

[7] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[8] I. Cingillioglu, "Detecting ai-generated essays: the chatgpt challenge," *International Journal of Information and Learning Technology*, vol. 40, no. 3, pp. 259–268, 2023. [Online]. Available: https://doi.org/10.1108/IJILT-03-2023-0043

[9] J. Davis and M. Goadrich, "The relationship between precision-recall and roc curves," in *Proceedings of the 23rd International Conference on Machine Learning (ICML)*, 2006, pp. 233–240.

[10] N. O. P. Vago, F. Forbicini, and P. Fraternali, "Predicting machine failures from multivariate time series: an industrial case study," *arXiv*, Feb. 2024, [Accessed: Dec. 1, 2024]. [Online]. Available: https://arxiv.org/abs/2402.17804

[11] Q. Amaya-Sanchez, M. J. de M. Argumedo, A. A. Aguilar-Lasserre, O. A. R. Martinez, and G. Arroyo-Figueroa, "Fault diagnosis in power generators: A comparative analysis of machine learning models," *Big Data and Cognitive Computing*, vol. 8, no. 11, p. 145, Nov. 2024, [Accessed: Dec. 1, 2024]. [Online]. Available: https://www.mdpi.com/2504-2289/8/11/145

[12] Z. Wang. (2016) Truck aps failure classification using machine learning - ida 2016. GitHub. [Accessed: Dec. 1, 2024]. [Online]. Available: https://github.com/zhendong3wang/kaggle-truck-failure-detection

[13] K. McClusky, "Python in ignition," Inductive Automation Support, 2023, [Accessed: Oct. 11, 2024]. [Online]. Available: https://support.inductiveautomation.com/hc/en-us/articles/360056397252-Python-In-Ignition

[14] W. Corporation. (2024) Wärtsilä 16v32 bore engine. [Accessed: Feb. 28, 2024]. [Online]. Available: https://www.maritimeinformed.com/w-rtsil-16v32-prime-mover-technical-details.html

[15] Z. Zhang, Y. Lei, X. Mao, M. Yan, and L. Xu, "A study of the effectiveness of deep learning in locating real faults," *Information and Software Technology*, vol. 137, p. 2021, 2021. [Online]. Available: https://yanmeng.github.io/papers/IST202.pdf

[16] A. Liaw and M. Wiener, "Classification and regression by randomforest," *R News*, vol. 2, no. 3, pp. 18–22, 2002. [Online]. Available: https://www.researchgate.net/publication/228451484_Classification_and_Regression_by_randomForest

[17] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* Springer Science & Business Media, 2009. [Online]. Available: https://link.springer.com/book/10.1007/978-0-387-84858-7

[18] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011. [Online]. Available: https://jmlr.org/papers/v12/pedregosa11a.html

[19] C. Chen, A. Liaw, and L. Breiman, "Using random forest to learn imbalanced data," University of California, Berkeley, Tech. Rep., 2004. [Online]. Available: https://statistics.berkeley.edu/sites/default/files/tech-reports/666.pdf

[20] T. K. Ho, "Random decision forests," in *Proceedings of the 3rd International Conference on Document Analysis and Recognition*, vol. 1. IEEE, 1995, pp. 278–282. [Online]. Available: https://ieeexplore.ieee.org/document/598994

[21] A. Vidhya. (2024) Understanding random forest algorithm with examples. [Online]. Available: https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest/

[22] I. R. König *et al.*, "On safari to random jungle: A fast implementation of random forests for high-dimensional data," *Bioinformatics*, vol. 26, no. 14, pp. 1752–1758, 2010. [Online]. Available: https://academic.oup.com/bioinformatics/article-abstract/26/14/1752/177075

[23] G. Biau and E. Scornet, "A random forest guided tour," *TEST*, vol. 25, no. 2, pp. 197–227, 2016. [Online]. Available: https://link.springer.com/article/10.1007/s11749-016-0481-7

[24] J. C. Quiroz, M. R. Mehrjou, M. Izadi *et al.*, "Fault detection of broken rotor bar in ls-pmsm using random forests," *Measurement*, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0263224117307066

[25] L. Auret and C. Aldrich, "Unsupervised process fault detection with random forests," *Industrial & Engineering Chemistry Research*, vol. 49, no. 21, pp. 10 359–10 369, 2010. [Online]. Available: https://pubs.acs.org/doi/abs/10.1021/ie901975c

[26] N. Amruthnath and T. Gupta. (2019) Factor analysis in fault diagnostics using random forest. arXiv preprint. [Online]. Available: https://arxiv.org/abs/1904.13366

[27] T. Chen and C. Guestrin. (2016) Xgboost documentation. [Online]. Available: https://xgboost.readthedocs.io/en/stable/parameter.html

[28] X. Li, L. Wang, and Z. Li, "Exploring the performance of xgboost for large scale data analysis," in *IEEE International Conference on Big Data*, 2018, pp. 2002–2008. [Online]. Available: https://ieeexplore.ieee.org/document/8622255

[29] J. Brownlee. (2016) How to configure xgboost for imbalanced classification. Machine Learning Mastery. [Online]. Available: https://machinelearningmastery.com/xgboost-for-imbalanced-classification

[30] S. A. Shafiq, M. Izhar, R. Sawhney *et al.* (2024) Investigating the influence of ages on the preparation and validation performance of mlp. ResearchSquare. [Online]. Available: https://www.researchsquare.com/article/rs-3848073/latest.pdf

[31] K. D. Toennies. (2024) Multi-layer perceptron for image classification. [Online]. Available: https://link.springer.com/chapter/10.1007/978-981-99-7882-3_7

[32] T. Yu, W. Guo, J. C. Li *et al.* (2022) Mctensor: A high-precision deep learning library with multi-component floating-point. arXiv preprint. [Online]. Available: https://arxiv.org/pdf/2207.08867

[33] S. Sharma, V. Narayan, R. Sawhney *et al.* (2023) A comparative assessment of artificial intelligence models for early prediction of chronic kidney disease. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2772662223000097

[34] I. University. (2024) Ignition scada training courses online. Inductive Automation. [Accessed: Mar. 3, 2024]. [Online]. Available: https://www.inductiveuniversity.com/

[35] J. E. Martin, "Supervised learning for failure detection in engine generators," 2024, [Accessed: Jul. 21, 2024]. [Online]. Available: https://github.com/Jonathan-Espin-Martin/DataScience-Master-Degree-Project

[36] S. E. R. (2024) Understand random forest algorithms with examples (updated 2024). Analytics Vidhya. [Accessed: Mar. 13, 2024]. [Online]. Available: https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest/

[37] M. Filho. (2024) How to get feature importance in xgboost in python. Forecastegy. [Accessed: Mar. 15, 2024]. [Online]. Available: https://forecastegy.com/posts/xgboost-feature-importance-python

[38] S. Ray. (2024) Top 10 machine learning algorithms to use in 2024. Analytics Vidhya. [Accessed: Mar. 2, 2024]. [Online]. Available: https://www.analyticsvidhya.com/blog/2017/09/common-machine-learning-algorithms/