MultiCraft

International Journal of Engineering, Science and Technology
Vol. 14, No. 1, 2022, pp. 28-51

INTERNATIONAL JOURNAL OF ENGINEERING, SCIENCE AND TECHNOLOGY

www.ijest1-ng.com
www.ajol.info/index.php/ijest

Design of software-oriented technician for vehicle's fault system prediction using AdaBoost and random forest classifiers

M. Kiruba Thomas [1], S. Sumathi [2]*

[1,2] Department of Electrical and Electronics Engineering, PSG College of Technology, Coimbatore , INDIA*
**Corresponding Author, Email: ssi.eee@psgtech.ac.in, Tel.: +99947 59330*
ORCID iDs: http:/orcid.org/0000-0001-5165-0212 (Sumathi)

Abstract

Detecting and isolating faults on heavy duty vehicles is very important because it helps maintain high vehicle performance, low emissions, fuel economy, high vehicle safety and ensures repair and service efficiency. These factors are important because they help reduce the overall life cycle cost of a vehicle. The aim of this paper is to deliver a Web application model which aids the professional technician or vehicle user with basic automobile knowledge to access the working condition of the vehicles and detect the fault subsystem in the vehicles. The scope of this system is to visualize the data acquired from vehicle, diagnosis the fault component using trained fault model obtained from improvised Machine Learning (ML) classifiers and generate a report. The visualization page is built with plotly python package and prepared with selected parameter from On-board Diagnosis (OBD) tool data. The Histogram data is pre-processed with techniques such as null value Imputation techniques, Standardization and Balancing methods in order to increase the quality of training and it is trained with Classifiers. Finally, Classifier is tested and the Performance Metrics such as Accuracy, Precision, Re-call and F1 measure which are calculated from the Confusion Matrix. The proposed methodology for fault model prediction uses supervised algorithms such as Random Forest (RF), Ensemble Algorithm like AdaBoost Algorithm which offer reasonable Accuracy and Recall. The Python package joblib is used to save the model weights and reduce the computational time. Google Colabs is used as the python environment as it offers versatile features and PyCharm is utilised for the development of Web application. Hence, the Web application, outcome of this proposed work can, not only serve as the perfect companion to minimize the cost of time and money involved in unnecessary checks done for fault system detection but also aids to quickly detect and isolate the faulty system to avoid the propagation of errors that can lead to more dangerous cases.

Keywords: Machine Learning (ML), Random Forest (RF), Ensemble Algorithms like AdaBoost Algorithm, Visualization, Fault Prediction

DOI: http://dx.doi.org/10.4314/ijest.v14i1.4

Cite this article as:
Thomas M.K., Sumathi S. 2022. Design of software-oriented technician for vehicle's fault system prediction using AdaBoost and random forest classifiers. *International Journal of Engineering, Science and Technology*, Vol. 14, No. 1, pp. 28-51. doi: 10.4314/ijest.v14i1.4

Received: December 1, 2021; Accepted: May 8, 2022; Final acceptance in revised form: May 19, 2022

1. Introduction

With the continuous advancements in automobile and electronic industry, the structure of vehicles becomes more complex and increase in degree of automations required. The autonomous vehicles are also being developed to reduce the human intervention which leads to the utilization of predetermined number of ECUs (Electronic Control Unit) for the optimized control of different subsystems like Engine, transmission, drive wheel etc. The demand for the control system and ECUs depends on the application and autonomous levels (Xie et al., 2019). For example, an autonomous excavator needs much control and communication networks like CAN bus (Controlled Area Network) for efficiently transferring data packets between nodes of the subsystem.

A typical vehicle consists of 90 ECUs, each designed and integrated to perform a selected function. Due to the complex character of interconnected ECUs and the large amount of data available within each ECU, manually checking all available vehicle data is complex. (Svärd et al., 2014). The Engine ECU controls the fuel injection and valve timing of the inlet and exhaust ports by evaluating the crankshaft position and drive-wheel teeth patterns which links crankshaft and camshaft. Some advanced automobile manufacturers like Volvo, Volkswagen design ECU not only for monitoring and control but also for storing the parameter to check fault events and diagnostics in the ECU's NVM (Non-volatile Memory). The figure 1 shows the block diagram of Engine ECU and their functions.

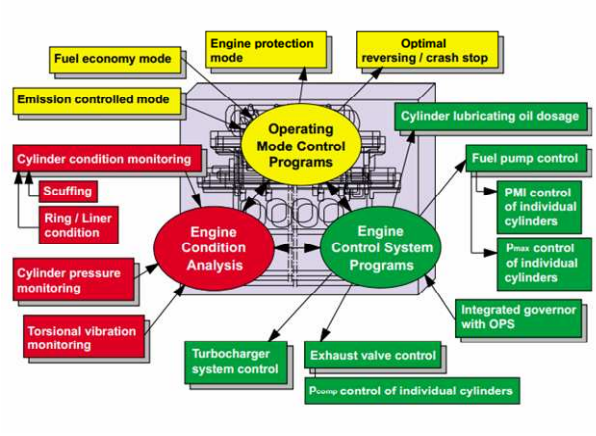

Fig 1. Engine Control Unit Functions

The goal of this paper is to deliver a User Interface model which aids the professional technician or Vehicle user with basic automobile knowledge to access the working condition of the vehicles and detect the fault subsystem in the Vehicles (Sankavaram et al., 2015). The scope of this system is to visualize the data acquired from vehicle and diagnosis the fault component using trained fault model obtained from Improvised ML classifiers.

Our aim is to attenuate the prices dealt with:

- Unwanted checks done by a mechanic.
- Missing a fault, which may lead to a failure in the future.
- Frequent replacement of good parts of vehicles
- High technology tools required to check vehicle condition by substituting it with ML based web application.

Vehicles have the potential to change the urban traffic landscape. The complexity of automotive systems is increasing at a rapid rate to improve vehicle performance. There is also an increasing reliance on electronic control units (ECUs) to monitor the condition of vehicle components. (Gomes and Wolf, 2021). This is because operating problems associated with degraded components, faulty sensors and inefficient controls affect the efficiency, safety and reliability of the vehicle, thereby leading to high costs. Warranty fees can be high. Therefore, it is important to quickly detect and isolate the faulty system to avoid the propagation of errors that can lead to more dangerous cases (Sankavaram et al., 2010). It is a tedious process to detect the fault component manually without the help of an experienced technician, it costs more time and improper repair service may also result in more adverse scenarios.

Detecting and isolating faults on cars is very important because it helps maintain high vehicle performance, low emissions, fuel economy, high vehicle safety and ensures repair and service efficiency (Xie et al., 2018). These factors are important because they help reduce the overall life cycle cost of a vehicle (Quanqi et al., 2011). Hence the data must be acquired from the concerned vehicle periodically and stored in a remote database, which are analyzed later to monitor the operational condition and predict the fault component in the system reducing the cost and time which arises from fault component replacement and repair.

Section 2 explores the literature survey made on fault detection techniques and Supervised Machine Learning Classifiers. Section 3, which describes the Implementation of the system proposed. Section 4 details the results and discussion. The performance analysis and implementation process is elaborated in section 5. Conclusion and References are presented in sections 6 and 7.

2. Literature review

In Obodoeze et al. (2018), the author tends to propose a vehicle fault detection system is a computerized application with advanced online fault reporting functionality. This system works by computer software that captures and displays the faults to the user and the user can also send the report back to the mechanic via email and SMS for immediate feedback. The proposed system reduces human labor and works rigorously in trying to detect defects. It also helps to reduce vehicle maintenance and troubleshooting costs by reducing time wasted fixing test failures.

ID	part	description	Location	Click to Add
	4 PERFORMANCE BATTERY	CAPTURES BATTERY FUNCTION	BOUNETT ENGI	
	5 BATTERYBOX	DETECT BATTERY EFFICIENCY	BOUNTETT	
	6 DISTRIBUTOR	POWER CELL	CARSAC	
	7 IGNITION BOX	START UP	ENGINE	
	8 IGNITION COIL	CONNECT BATTERY AND ENGINE	STRIP CAN	
	9 ENGINE LEVEL SENSOR	MEASUREMENT OF WORKLOAD	PART1	
	10 FUEL SENSOR	DETERMINES THE VOLUME OF FUEL	PART3	
	11 CO2SENSOR	DETERMINES WATER LEVEL	PRS	
	12 STARTER DRIVE	IGNIT	part4	
	13 GLOW PLUG	LOW 1	part	
	14 STARTER MOTOR	LOW2	part engine	
	15 OIL SENSOR	LOW3	structure 3	
	16 ADJUSTING MECHANISM	LOW4	desire1	
	17 BRAKE PAD	LOW4	gent1	
	18 BRAKE LINNING	LOW6	weight1	
	19 GASOLINE ENGINE	LW7	base5	
	20 AIR DUCT	BEST1	cool2	
	21 FAN BELT	VIEWER2	coop4	

Fig 2. Database Storage Model

The fault detection and isolation (FDI) system is used to detect the occurrence of fault events. This system has been designed using Microsoft Visual Studio.net as the front end and thus Microsoft Access Database Structured Query Language (SQL) as the backend to store error string information that can may come from the system. The database model created is depicted in Figure 2. The author designed the software to have 5 forms of interaction with the user in every possible way to show the errors detected by FDI (Sankavaram e al., 2015). The computer system can understand the digital signal coming from the ADC (Analogue to Digital Converter) and control the signal and translate it into a high-level language. All incident activities related to the vehicle are normally tracked and even generated using a log form for observation.

Due to the safety-critical nature of hybrid/electric vehicles, data is collected from test drives, road tests, customer logs, etc. and analyzed in detail to find potential errors and failures. However, due to the huge amount of data available on these vehicles, it is not possible for humans to manually check each data point (Nair and Koustubh, 2017). This paper proposes modern techniques based on machine learning technology to process hybrid/electric vehicle data and detect anomalies in vehicle behavior. Explore some powerful machine learning algorithms, fine-tuned to fit the use case of hybrid/electric vehicle data analytics for faster and more accurate anomaly detection.

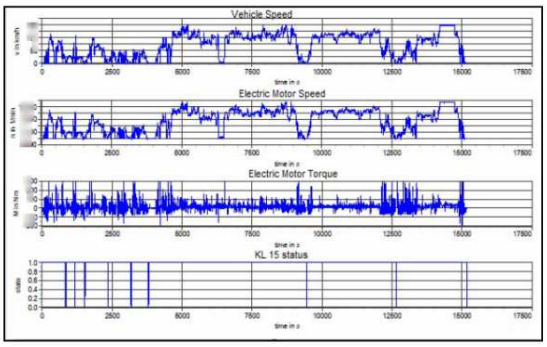

Fig 3 Data visualization of test parameters

Using Machine learning mechanisms, the difference between normal and abnormal behaviours can be learnt from the available vehicle data and then, when a new actual vehicle test data is fed to the system, the system can use the results learned in the past to confidently detect errors in the vehicle data and about abnormal behavior. common to vehicle data, the system can alert experts to further look into certain areas of data where a suspicious anomaly may be hidden.The figure 3 depicts some test signals recordings from vehicle ECU. The Single layer Support vector data descriptor (SVDD) technique for hybrid electric vehicle data analysis using RBF kernel is discussed in this article and additional techniques for more robust and fast anomaly detection have been described in more detail. It is observed that SVDD generalizes to decision boundaries better and faster with kmeans clustering (Pedregosa et al., 2011). Multi-glass error detection was used by implementing a rule-based classifier in addition to conventional SVDD. According to the paper, further improvements in this technique can be achieved by using the power of artificial neural networks.

In Quanqi et al. (2011), the author extrapolates the general design of a bus data acquisition and fault diagnosis system based on OBD (OnBoard Diagnostics), focusing on its lower computer systems such as ARM microcontrollers and design principles computer software system higher. This system is based on widely used CAN bus technology, to extract information about vehicle condition or fault. The CAN bus adopts the SAE (Society of Automotive Engineers) J1939 protocol standard. The CAN bus data acquisition and analysis system follows the SAEJ1939 protocol, providing about 400 errors.

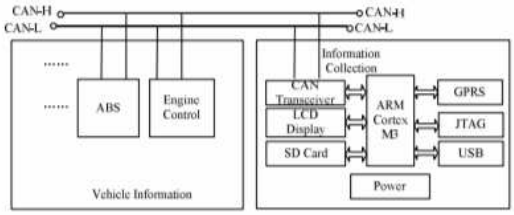

Fig. 4. The structure of the fault diagnosis and monitoring system

The System hardware includes integrated processor control circuit, power supply circuit, CAN bus circuit, LCD display(Liquid-Crystal Display) circuits, GPRS (General Packet Radio Service) circuit, JTAG (Joint Test Action Group) circuit, and USB (Universal Serial Bus) circuit.The structure of the fault diagnosis and monitoring system is shown in Figure 4. The system is connected to the vehicle via the OBD interface and receives and analyzes parameters such as engine oil level, pressure, and water temperature. cooling, etc. The LCD display shows the fault tree analysis to the pilots through the fault tree principle. The fault tree model that describes the purpose, function, and relationship of a structure is a type of qualitative and causal model.

The information can be sent to the higher computer system via USB or to the management center via GPRS, to realize remote vehicle monitoring and fault diagnosis. In software design, the goal is to realize the functionality of the system through the program that corresponds to the hardware system. The program is designed according to the integrated development platform IAR EW (Embedded Workbench) providing rich library reading functionality for the system. Configuration and initialization of the system clock and peripherals are handled by the system software. In Costa and Nascimento (2016), machine learning solutions to the IDA 2016 Industrial Challenge is presented to predict imminent failure of a specific component in a vehicle's Air Pressure System. This problem is a binary classification type of problem where each instance is classified as positive or negative (positive denotes failure in the APS component and negative denotes failure is not APS related. The data from heavy Scania trucks is used as the dataset for the experiment. The aim of the challenge is to build an efficient model that reduces the cost of misclassification.

The paper evaluates various state of the art ML classification models like Support Vector Machine (SVM), Logistic Regression (LR), k-Nearest Neighbors (k-NN) and Random Forest (RF). Their work compares the performance of the algorithms while dealing with problems like imbalanced dataset and non-trivial amounts of missing data (Gosain and Sardana, 2017). Figure 5 shows the flow of the process of identifying the best prediction model.

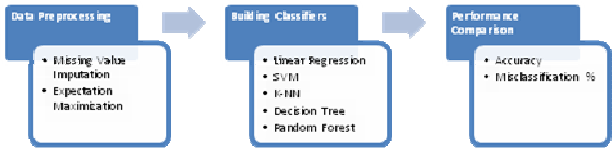

Fig 5 Work Flow Diagram

As there are very small number of instances belonging to positive class (1000 instances) when compared to negative instances (59000) the models may not learn much from positive instances resulting in misclassification of positive instances. In SVM and LR, class-specific weights are assigned to deal with imbalanced data problem. In k-NN and RF, the actual threshold for classifying an instance as positive or negative is 50%. This threshold is increased so that it classifies an instance as negative only if it is very confident. Mean Imputation and Soft Impute methods are used to impute the missing values. K-fold cross validation is used to simulate how well the models would perform in the test dataset. The experimental result showed that the Random Forest classifier was cost-wise 92.56% better than the other models thus concluded as the best model.

3. Implementation of the proposed system

The flow and overview of the proposed methodology is described briefly with the help of block diagrams in this section and also it helps the readers to evaluate the reliability and validity of this proposed work. The main objective of this proposed work is to create a software tool where the parameters obtained from ECU memory can be analysed to detect the working condition of the vehicles and diagnose the fault component if any. The proposed system involves three stages and they are data visualization manual observation, fault model creation and report the fault detection for efficient repair and services. Since the software visualizes the data to be analysed, the user need not depend on the service technician to identify the fault component to be repaired or replaced.

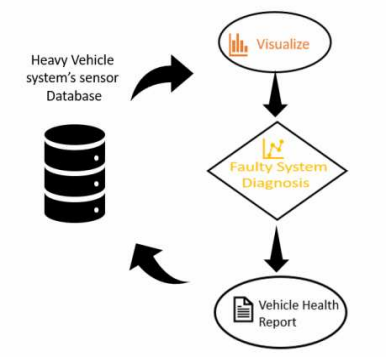

Fig 6. Proposed system overview

The GUI terminal is created for easy user interface using flask and data visualization is performed using python bokeh library. Error diagnosis methods can be classified into the following three approaches: data-driven, model-driven, and knowledge-driven (Sankavaram et al., 2010). Model-based approach is used for determining the fault component and it is performed using python-based libraries like sklearn, pandas etc. The figure 6 shows the overview of the proposed system. The sensor data from vehicles are collected using proprietary communication adapter for specific control units and stored in the form of database files like xml and csv scripts. This data is then fetched from the database and is used for performing various tasks like visualizing the sensor data, diagnosing the Air pressure System for faults and generating a report about the condition of the vehicle.

The proposed model comprises of three main components as discussed earlier. The Data visualization is achieved for better understanding the health status of the vehicles and Fault ML based Model creation is to detect the fault component and isolate it. Since the original dataset with fault classification for different subsystem in a vehicle is considered as proprietary information and generation of such data values requires valuable tools and experience, the Air pressure system (APS) fault dataset is chosen from UCI repository (Dua et al., 2019). This is a binary classification problem where the positive class tells us that the error is due to a specific component of the APS, while the negative class tells us that the error has nothing to do with that component.

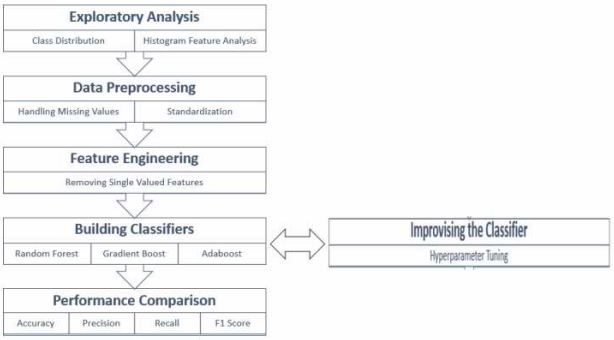

Fig 7 Flowchart of the Proposed Fault Detection System

The proposed fault detection model flow is depicted in the figure 7. Initially, exploratory data analysis is done to analyze and investigate the data in order to discover patterns in it. And then processes such as data preprocessing and feature engineering are followed to remove imbalances and anomaly values so that the dataset is made ready to be given as an input to the models for training. Following this, the machine learning models Random Forest and Adaboost are built and trained with the training data. Steps are taken to improve the performance of the fault detection models and performance comparison is done with metrics like Accuracy, Precision, Recall and F1 Score. Finally, the best model is deployed in the proposed software tool. Python is popular for its packages and support, for Math computations, metric calculation, classifier library, graph plotting. All jobs need specific packages or libraries for implementation, which are given in this section.

A. Data Visualization

1. MatplotLib

Matplotlib is a very useful plotting library for Python and other high-level languages and its NumPy numerical math extension. It provides an object-oriented API for integrating batches into applications using general-purpose GUI toolkits such as Tkinter, wxPython, Qt, or GTK+. There is also a "pylab" procedural interface supported on state machines (such as OpenGL), which is designed to closely resemble that of MATLAB, although its use is discouraged.

2. Seaborn

Seaborn is a matplotlib-based Python data visualization library that is tightly integrated with pandas' data structures in Python. Visualization is a key component of Seaborn supporting the discovery and understanding of digital data. It provides a high-level interface for drawing attractive and informative statistical graphs. It can be used to create almost any statistical chart. Seaborn is a dataset driven graphing function that can be used on both chunks and arrays of data.

3. Plotly

The Plotly Python Library is an interactive open-source library. It can be a very useful tool to visualize data and understand data in a simple and easy way. The plotted graphics object is a very easy to use high-level interface for plotting. It can draw different types of charts and graphs like scatter chart, line chart, bar chart, box chart, histogram, pie chart, etc. It is a web-based data visualization tool that provides many useful ready-to-use charts.The library works alright in interactive web applications.

B. Data Processing

1. Pandas

The Pandas library is the usual API for managing facts. Data is all statistics around, statistics which are to be bear in mind of interest. Data may be numeric or textual. For the obligations of classifying numerical facts. Pandas offers gear to address small to huge function bodies, the principle one being a facts frame.

2. Numpy

NumPy brings the computing power of languages like C and Fortran to Python, which is much easier to learn and use. With this power comes simplicity: an answer in NumPy is often clear and elegant.

3. Scipy

SciPy (pronounced "sigh pie") is an ecosystem of Python-based open-source math, science, and engineering software. This is a scientific calculation library using NumPy below. SciPy stands for Scientific Python. It provides more useful functions for optimization, statistics, and signal processing. Like NumPy, SciPy is open source, so you can use it freely. SciPy was developed by Travis Olliphant, the creator of NumPy.

C. Data Modelling

1. Sklearn

Scikit-learn (formerly scikits. learn, also known as sklearn) is a free software machine learning library for the Python programming language Pedregosa et al., 2011). There are various classification, regression, and clustering algorithms such as support vector machines, random forests, gradient boosting, , numerical and scientific Python-LibrariesNumPy and SciPy. Designed.

2. Imblearn

Imbalancedlearn (imported as imblearn) is an MIT-licensed open-source library based on scikit-learn (imported as sklearn) that provides tools for handling classifications in imbalanced classes.Imblearn library is specifically designed to affect imbalanced datasets. It provides various methods like undersampling, oversampling, and SMOTE to handle and removing the imbalance from the dataset (Ramentol et al., 2012). This library consists of varied ensemble methods like bagging classifiers, random forest and boosting classifiers which will be wont to train models for imbalanced data sets with very efficient accuracy.

3. XGBoost

XGBoost is an open source library that provides a high performance implementation of gradient support decision trees. The combination of the underlying C ++ code base and the underlying Python interface provides a very powerful yet easy-to-implement package. XGBoost is an implementation of the gradient support decision tree designed for speed and performance that dominates competitive machine learning.

4. LightGBM

LightGBM (LGBM) is an open-source gradient boosting library. It can be used to train models with tabular data with incredible speed and accuracy. This performance comes from how LightGBM samples data (GOSS – Gradation-based OneSided Sampling) and how it reduces the number of features in the sparse dataset (EFB – Exclusive Feature Bundling) during training.

5. Pickle

The Python pickle module is used to serialize and deserialize Python object structures. This process converts all Python objects (list, dict, etc.) into a byte stream (0 and 1). This process is called pickling or serialization or flattening or shunting. Unpickling is the opposite, reconverting a stream of bytes (from an object such as a computer file or bytes) into an object hierarchy.

6. Joblib

Joblib is a function library built entirely in Python by Scikit-learn developers. It focuses only on Python-based persistence and feature optimization.It provides lightweight pipelining in Python development services. Joblib offers help in persisting any data structure or machine learning model.

7. Keras

Keras is a convenient, minimalist Python library for deep learning that runs on Theano or TensorFlow. It was designed to make the implementation of deep learning models for research and development as fast and easy as possible (Quanqi et al., 2011). It runs on Python 2.7 or 3.5 and runs seamlessly on GPUs and CPUs with the underlying framework. It will be released under the tolerant MIT license.

8. Web Application Framework - Flask

Flask is a convenient and lightweight WSGI web application framework built with a small core and an extensible philosophy. This is a micro-framework that does not include an ORM (Object Relational Manager) or similar functionality. Designed to get you started quickly and easily, it can adapt to complex applications. Flask is based on the WSGI toolkit and the Jinja2 template engine.This section gives an overview about different data processing techniques like imputation, scaling and sampling. It also helps to understand the working of machine learning algorithms like random forest, adaboost and compares their benefits.

9. Data Preparation

The real-world data contains various normalities and missing values which is not supported for proper analysis and to gain useful information. So, data preparation is the first essential component of any kind of data analysis like text, numerical analysis etc.

10. Imputation

The value of the record may be missing, which causes problems with some machine learning algorithms. Before modeling the predictive task, the input data must identify and replace the missing values in each column (Costa and Nascimento, 2016). This is known as the assignment of missing data, or abbreviated as assignment. Here are some of the assignment methods:

- Simple Imputation
- Mean Imputation
- Median Imputation
- Mode Imputation
- Regression Imputation
- Iterative Imputation

Among these Mean Imputation, Median Imputation and Iterative Imputation techniques are implemented in this work.

11. Scaling Techniques

Scaling refers to putting the values within the same range or same scale in order that no variable is dominated by the opposite. It helps the machine learning algorithms to improve the ability of prediction model. For example, algorithms that fit models that use weighted sums of input variables, such as linear regression, logistic regression, and artificial neural networks (deep learning), are affected. Data scaling is often achieved by normalizing or standardizing real-valued input and output variables. Two preferred techniques for scaling numerical data prior to modeling are normalization and standardization. Normalization scales each input variable individually to the range 01. This is the range of floating-point values with the highest accuracy. Standardization subtracts the mean (called centering), divides by the quality deviation to shift the distribution, and scales each input variable individually so that the mean is 0 and the standard deviation is 1. There are different scaling techniques namely,

- MinMax Scaling
- Standard Scaling
- MaxAbs Scaling
- Robust Scaling
- Quantile Transformer Scaling
- Log transform etc.,

Of these scaling techniques the MinMax and Standard scaling techniques are deployed and their results are compared.

11.1 Min-Max Normalization: This method re-scales a feature or observation with distribution value between zero and one, Equation (1).

$$X_{new} = \frac{X - \min(X)}{\max(X) - \min(X)} \tag{1}$$

11.2 Standardization: It is a really effective method which re-scales a value in order that it's distribution with 0 norm and variance equals to one, Equation (2).

$$X_{new} = \frac{X_i - Xmean}{S\tan dard\ \ deviation} \tag{2}$$

12. Types of Machine Learning Algorithms

An ML algorithm is a procedure that executes on prepared data and is used for building a deployment-ready machine learning model (Pestana-Viana et al., 2019). They provide computer systems the ability to learn automatically from the input data so that when the system is exposed to new data it can predict the output accurately.

The major types of machine learning algorithms are:

- Supervised Machine Learning Algorithms
- Unsupervised Machine Learning Algorithms
- Semi-supervised Machine Learning Algorithms
- Reinforcement Machine Learning Algorithms

13. Supervised Machine Learning Algorithms

A supervised learning algorithm is an algorithm that involves a direct administrator of the operation. In this case, the user tag takes sample data of the main structure and sets the strict criteria for the algorithm to work. From a computer point of view, this process becomes more or less routine (Costa and Nascimento, 2016). The basic purpose of supervised learning is to expand the scope of labeled data and make predictions.

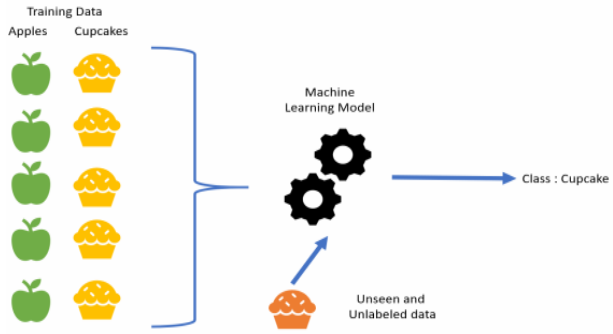

Fig 8. Supervised Learning

Figure 8 depicts a pictorial representation of Supervised Learning. It shows simple classification between green apple and cupcake obtained from seen and labeled data. Supervised machine learning involves two classic methods: classification and regression.

- Classification is the process in which incoming data is labeled according to collected data patterns and manually trains the classifier to recognize certain types of objects and segment them accordingly. The system is capable of discriminating between types of information, performing optical, image or binary character recognition (whether or not a particular small piece of data fits specific requirements in the form of an image or not). "Yes or no").
- Regression is the process of identifying patterns and calculating predictions about adjacent outcomes. The system must understand numbers, their values, groups (e.g. height and width), etc.

The most widely used supervised algorithms are:

- Linear Regression;
- Logistical Regression
- Random Forest;
- Gradient Boosted Trees;
- Support Vector Machines (SVM);
- Neural Networks;
- Decision Trees;
- Naive Bayes;
- Nearest Neighbor
- Adaptive Boosting
- Gradient Boosting

Among these, Random Forest classifier, Adaptive boosting and Gradient Boosted decision tree are the chosen machine Learning algorithms in this paper.

14. Unsupervised Machine Learning Algorithms

Unsupervised learning is learning that does not include the direct control of the user. If the main point of supervised machine learning is that one knows the result and one has to sort the data, then in the case of an unsupervised machine learning algorithm, the desired outcome is undefined and undefined. A huge difference between these two methods is that supervised learning uses only labeled data while unsupervised learning thrives on unlabeled data.

The unsupervised machine learning algorithm is utilized for:

- exploring the structure of the information;
- extracting valuable insights;
- detecting patterns;
- implementing this into its operation to increase efficiency.

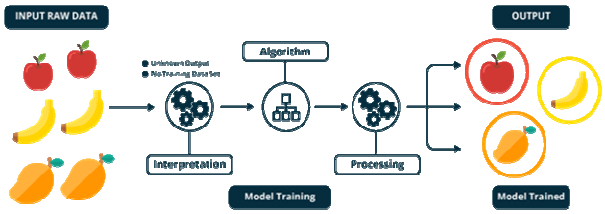

Fig. 9. Unsupervised Learning

In Figure 9, a diagrammatic view of Unsupervised Learning is shown. It depicts how raw data involving apple, banana and mango can be processed and trained to provide a trained model output. In other words, unsupervised machine learning describes information by going through it and understanding it. Unsupervised learning algorithms use the following methods to describe data:

- Clustering:it is data discovery used to divide them into meaningful colonies (i.e. clusters) based on their internal patterns without prior knowledge of the group's identity information. Identifiable information is declared by the similarity of individual data points and aspects of its differentiation from the rest (which can also be used to detect outliers).
- Dimensionality reduction: There is a lot of noise in the input parameter. ML algorithms use dimensionality reduction to reduce this noise while distilling the relevant data.

The most widely used algorithms are:

- k-means clustering;
- t-SNE (t-Distributed Stochastic Neighbor Embedding);
- PCA (Principal Component Analysis);
- Association rule.

15. Semi-supervised Machine Learning Algorithms

Semi-supervised learning algorithms denote a central ground between supervised and unsupervised algorithms. In integration, the semi-supervised model attaches some functions of both supervised and unsupervised algorithms. Figure 10 gives a pictorial representation of Semi Supervised Learning. In simple words, it shows how it utilizes few labeled data initially and retrains the obtained model with unlabeled data in a loop fashion to attain an improvised model.

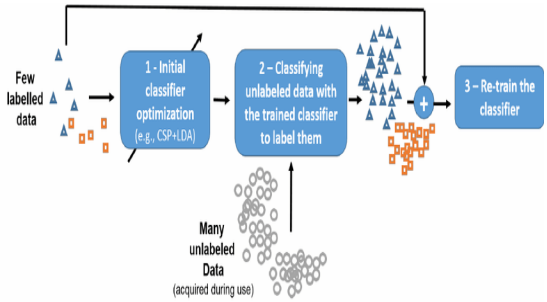

Fig 10. Semi Supervised Learning

Semi-supervised algorithms working:

1. Semi-supervised machine learning algorithms use a limited set of labeled data samples to reshape the operating plan (i.e. train itself).
2. The restriction leads to a partially trained model that will later get the job of labeling the unlabeled data. Due to the limitation of the data sample, the results are considered as dummy labeled data.
3. Finally, the labeled and pseudo-labelled datasets are combined, creating a separate algorithm that combines the descriptive and predictive aspects of supervised and unsupervised learning.

Semi-supervised learning uses classification methods to detect data resources and a clustering process to divide them into distinct parts.

16. Reinforcement Machine Learning Algorithms

Reinforcement learning refers to machine learning artificial intelligence (AI) that involves developing a self-sustaining system through contiguous chains of trial and error, improved upon a combination of data. labeled and interacted with incoming data.

Advanced ML uses a technique known as discovery/mining. The technique is very simple, the results can be observed and the reaction considers the results of the first action. At the heart of reinforcement learning algorithms are the reward signals that occur when performing specific tasks. In a way, the reward signal acts as a navigator for reinforcement algorithms. These algorithms use unlabeled data along with some labeled data to classify a new unlabeled text document. Figure 11 shows the reinforcement learning process.

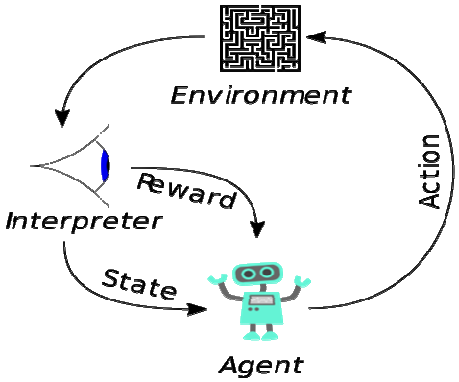

Fig 11. Reinforcement Learning

Most common reinforcement learning algorithms are:

- Q-Learning;
- Temporal Difference (TD);
- Monte-Carlo Tree Search (MCTS);
- Asynchronous Actor-Critic Agents (A3C).

17. Selected Algorithms for the Proposed Methodology

The main aim of this work is to predict if there is a fault in the Air Pressure System or not. Here, the classifier algorithm is provided with labelled data to learn the pattern from it and further with the knowledge gained from this prediction on new data is be made. Therefore, the problem statement at hand is of type supervised learning. Among the various supervised learning algorithms that are discussed above Random Forest, Adaptive Boosting, Gradient Boosting and baseline Neural Network are implemented in this paper. The advantages of using these algorithms are discussed in this section.

17.1 Random Forest

Tree-based algorithms are popular machine learning methods used to solve supervised learning problems. Tree-based algorithms are flexible and can solve any kind of numerical analysis problem like classification or regression. Random Forest is one of the most popular tree-based supervised learning algorithms. was developed by Breiman, L as shown in Figure 12. It is a synthetic learning algorithm that consists of multiple decision tree classifiers and the output type is collectively determined by these individual trees. A decision tree is a tree-like domain used to define a course of action (Pestana-Viana et al., 2019; Costa and Nascimento, 2016). Each branch of the decision tree represents a possible outcome, event, or response. When the quantity of trees in the forest rises, the fallacy in generalization error for forests intersects.

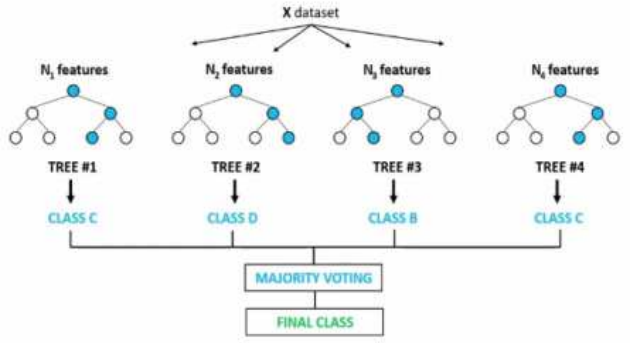

Fig 12 Random Forest Classifier Illustration Tree

Random Forests have many significant advantages. For example, it can handle height data without selecting a function; it reduces the risk of overfitting, it can alleviate this problem by averaging the prediction results from different trees. This gives random forests higher prediction accuracy than a single decision tree; the trees are independent of each other during training and the implementation is also quite simple. In addition, it provides a high level of accuracy. The Random Forest algorithm works well in large data sets and makes fairly accurate predictions.

The random forest algorithm performs the following steps:

Step-1: Select random M data from the training dataset.
Step-2: Create the decision trees associated with the selected subset of data.
Step-3: Choose the number N for decision trees that you want to build.
Step-4: Repeat Step 1 & 2.
Step-5: For new data points, identify the predictions of each decision tree, and assign the new data points to each category.
Step-6: The category with majority votes is taken as final prediction

17.2 Adaptive Boosting Classifier

The first normalization of boosting that saw huge success in application was Adaptive Boosting or AdaBoost for short. The weak learners in AdaBoost are decision trees with a single split, called decision stumps for their briefness. AdaBoost does its job by weighting the observations, attaching more weight on difficult to classify values. New weak learners are appended sequentially that concentrates their training on the more complex patterns. Predictions are made by majority vote of the weak learners' predictions, weighted by their individual precision. The most successful kind of the Adaptive Boosting algorithm was for binary classification problems and was called AdaBoost.M1. The significance of this technique is that it aims at exploiting the discrepancy between models by giving the mislabelled examples higher weights. A weak classifier (decision stump) is prepared on the training data using the weighted samples. Only binary (two-class) classification problems are preferred, so each decision stem makes one output on one input variable and outputs a +1or -1value for the first- or second-class value.

Once finished, pool of weak learners each with a low stage value is left. Predictions are made by calculating the mean weights of the weak classifiers. For a new input instance, each weak learner calculates a predicted value as either positive or negative. The predicted values are weighted by each weak learner's stage value. The prediction result for the ensemble model is considered as the sum of the weighted predictions. If the sum is +ve, then the first class is predicted, if -ve the second class is predicted.

17.3 Hyperparameters

The model hyperparameter is a configuration that is outside the model and its value cannot be estimated from the data. They are often used in processes to help estimate model parameters. A machine learning algorithm is tuned to a particular item so that it makes the most ingenious predictions.

The set of hyperparameters used are listed below,

- n_estimators - number of trees in the forest
- max_features - max number of features considered for splitting a node
- max_depth - max number of levels in each decision tree
- Learning rate - controls how much to change the model in response to the estimated error each time the model weights are updated.
- Number of Epochs - One Epoch is when an entire dataset is passed forward and backward through the neural network only once.
- Activation Function - a function that is added into an artificial neural network in order to help the network learn complex patterns in the data. Two of the frequently used Activation Functions are Sigmoid, Relu.

18. Performance metrics

In machine learning and statistical classification, confusion matrices, also known as error matrices, play an important role in computing performance metrics. A confusion matrix is a graph commonly used to describe the performance of a classification model (or "classifier") on a set of test data for which the true values are known. It allows visualizing the performance of an algorithm. It allows easy identification of confusion between classes, for example; one layer is often mislabeled with the other. Performance measures like accuracy, precision, recall, F1measure, Kappa score were calculated from confusion matrix.

19. Confusion Matrix

A confusion matrix is a summary of the predicted outcomes of a classification problem. The number of true and false predictions is aggregated with the counts and broken down by class. This is the key to the confusion matrix. The confusion matrix shows the ways in which the classification model is confused when making predictions. It provides insight not only into the errors generated by the classifier, but more importantly into the types of errors made.

Definition of the Terms:
Positive (P): result is positive (for example: is an apple).
Negative (N): result is not positive (for example: is not an apple).
True Positive (TP): result is positive, and is predicted to be positive.
False Negative (FN): result is positive, but is predicted negative.

True Negative (TN): result is negative, and is predicted to be negative.
False Positive (FP): result is negative, but is predicted positive.

20. Classification Rate/Accuracy

Classification Rate or Accuracy is given by the relation, Equation (3)

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{3}$$

However, there are accuracy issues. It assumes equal cost for both error types. 99% accuracy can be excellent, good, poor, or terrible depending on the problem.

21. Recall

Recall can be defined as the ratio of the total number of correctly classified positive examples divided by the total number of positive examples. High recall indicates that the class is correctly recognized (a small number of FNs). The recall is used to know how accurate the classification model is. The formula of recall is given as Equation (4)

$$\text{Recall} = \frac{TP}{TP + FN} \tag{4}$$

22. Precision

To get the exact value, divide the total number of correctly classified positives example equals the total number of positive examples predicted, as shown in the equation below. High precision indicates that an example labeled as positive is indeed positive (a small number of FPs), Equation (5).

$$\text{Precision} = \frac{TP}{TP + FP} \tag{5}$$

High recall, low precision: This means that most of the positive examples are correctly recognized (low FN) but there are a lot of false positives.
Low recall, high precision: This shows that missing a lot of positive examples (high FN) but those predict as positive are indeed positive (low FP).

23. F1 - measure

Instead of 2 measures (accuracy and recall), it would be helpful to have one measure that represents both. The F1 measurement uses the harmonic mean instead of the arithmetic mean because it punishes extreme values more. The F1 measurement will always be close to the smaller accuracy or recall value and the formula given in Equation (6).

$$\text{F - measure} = \frac{2 * \text{Re}call * \text{Pr}ecision}{\text{Re}call + \text{Pr}ecision} \tag{6}$$

4. Implementation process

Google Colabs, a cloud-based technology is used to implement the fault classification. So, Specification of the software and hardware of the local machine is not need to be considered for fault ML model creation. The processor of Run-Time Environment is set to Central Processing Unit (CPU), the CPU refers to the cloud CPU, not the CPU of local machine. The Web application based on Flask web framework is developed using Pycharm Integrated Development Environment. The exploratory analysis and visualization of dataset, challenges resolved in data preparation, ML model building and User Interface design are briefly explained in this section.

4.1 Data Visualization

Data visualization gives us a clear idea of what information means by providing it with visual context through a map or chart (Kohavi, 2001). This makes the data more natural to the human mind and thus makes it easier to identify trends, patterns, and outliers in large data sets.

4.1 Dataset Overview

There are different systems associated with the vehicles continuous operation like engine system, fuel system, brake system etc. The data can be generated from these systems using On-board Diagnosis tool (OBD) or other data acquisition methods (Johanson and Karlsson, 2007). The data frame shown in Figure 13 has the features like Intake manifold pressure, Engine speed and power as a part of Engine Subsystem.

```
trip_df.head()
```

	Time	Acceleration	Lat	Lon	Man_pressure	RPM	Speed	Fuel_Flow	Power
8	1900-01-01 00:00:00.500	0.01	42.29723	-71.76543	9.7	844.0	0.0	0.353	2.0
9	1900-01-01 00:00:00.600	0.02	42.29723	-71.76543	9.7	844.0	0.0	0.353	2.0
10	1900-01-01 00:00:00.600	0.01	42.29723	-71.76543	10.0	832.0	0.0	0.353	2.0
11	1900-01-01 00:00:00.700	0.02	42.29723	-71.76543	10.0	832.0	0.0	0.353	2.0
12	1900-01-01 00:00:00.900	0.03	42.29723	-71.76543	10.0	832.0	0.0	0.353	2.0

Fig 13. Data Frame with Engine parameters

4.2 Interactive data plot

Interactive data visualization refers to the use of modern data analysis techniques that allow users to directly manipulate and explore graphical representations of data. Data visualization uses visualization tools to help analysts effectively and efficiently understand the meaning of data (Quanqi et al., 2011). Interactive data visualization techniques enhance this concept by incorporating interactive tools that make it easy to change the parameters of the data visualization, allowing users to see more details, generate information new details, create compelling questions, and capture the full value of information. Interactive data visualization allows vehicle users to interact with vehicle system data in ways not possible with static graphics, such as interactive big data visualizations (Kohavi, 2001). Interaction is the ideal solution for large amounts of data with complex data stories being the exact real-life scenarios of vehicle system component data, providing data discovery, isolation, and visualization data in time periods.

```
df.iplot(x="Time",y="RPM",colors="Blue",xTitle='Time',yTitle='RPM')
```

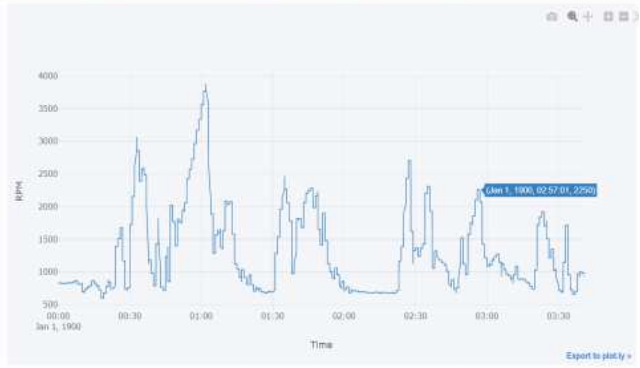

Fig 14 Time vs. Engine RPM plot

To draw plotly graphs in an IPython notebook without connecting to an server, iplot is used.Figure 14 shows a sample Time vs Engine RPM interactive plot.

4.3 Fault Model Creation

Error diagnosis methods can be classified into the following three approaches: data-driven, model-driven, and knowledge-based. A data-driven approach is preferred when system monitoring data is available for nominal and degraded conditions (Sankavaram et al., 2010). One of the common ways to perform data-driven fault diagnosis is to use neural networks and statistical machine learning techniques to classify the data into nominal sane and error classes or into fault classes difference.

4.4 Dataset Overview

The dataset includes data collected from heavy trucks under normal use (Dua and Graff, 2019). The system in question is an Air Pressure System (APS) that produces pressurized air that is used in various truck functions, such as braking and gearshift applications. The active class of the data set indicates the failures of the components of the APS system. Negative overlays are made up of vehicles with non-APS related errors. The training dataset consists of 60,000 data points and 171 features, one of which is a class label. The repository also has a separate test dataset consisting of 16,000 data points (Dua and Graff, 2019). Features are a combination of sensory data and characteristic data of the histogram bin. Object names in the graph are anonymized for proprietary reasons. 59,000 data points belong to the negative class and the remaining 1,000 points belong to the positive class. This tells us that we are dealing with a very unbalanced data set and that is usually the kind of data we would expect in a real world scenario.

4.5 Exploratory Analysis

Exploratory data analysis is performed to see the correlation between features and uses a dimensionality reduction technique to examine how the data is distributed in the 2D space. After importing the required library files, the dataset can be viewed, which validates 60,000 data points and 171 features, including feature classes. The observed APS error data frame is shown in Figure 15.

```
x = pd.read_csv(r'/content/drive/MyDrive/Project/aps_failure_training_set.csv',na_values=["na"])
print(x.shape)
x.head()
```

(60000, 171)

	class	aa_000	ab_000	ac_000	ad_000	ae_000	af_000	ag_000	ag_001	ag_002	ag_003
0	neg	76698	NaN	2.130706e+09	280.0	0.0	0.0	0.0	0.0	0.0	0.0
1	neg	33058	NaN	0.000000e+00	NaN	0.0	0.0	0.0	0.0	0.0	0.0
2	neg	41040	NaN	2.280000e+02	100.0	0.0	0.0	0.0	0.0	0.0	0.0
3	neg	12	0.0	7.000000e+01	66.0	0.0	10.0	0.0	0.0	0.0	318.0
4	neg	60874	NaN	1.368000e+03	458.0	0.0	0.0	0.0	0.0	0.0	0.0

Fig 15. APS fault Data frame head

Python Code 1:
“Import pandas as pd
data = pd.read_csv(r'/content/drive/MyDrive/aps_failure_training_set.csv',na_values=["na"])”

Python Code 1 reads the csv data using the read_csv function of python library pandas. In the class label, ‘negative’ is used to denote non faulty and ‘pos’ is used to indicate the fault representing data. For better understanding and usage, ‘neg’ and ‘pos’ are replaced by 0 and 1 respectively to define the classification problem as binary.Python code 2 shows an user defined function ‘get_correct_label ()’ that receives the text form of labels and returns the numerical form as 0 and 1.

Python Code.2:
“def get_correct_label(y):
return y.replace(['neg','pos'],[0,1])
data['class'] = get_correct_label(data['class'])”
As the labels are anonymized, the main features cannot be separated from the dataset.The first label is designated as ‘aa_000’ where ‘aa’ denotes particular histogram parameter feature and ‘000’ after the underscore indicates the nth bin of the histogram feature. The bin division is explained using graph in the following subtopics. Python code 3 imports seaborn module and plots a count plot of the labels using the barplot function.

Python Code 3:
“Import seaborn as sns
sns.barplot(x['class'].unique(),x['class'].value_counts())
plt.title('Class Label Distribution')
plt.xlabel('Class Label')
plt.ylabel('Count')
plt.show()”

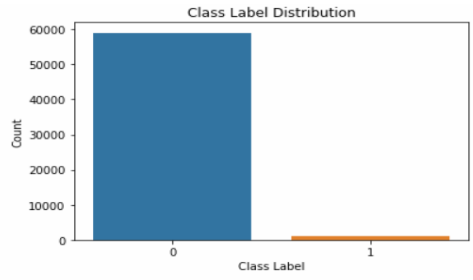

Fig 16. Class label Distribution

The class distribution graph in Figure 16 shows a serious case of data imbalance, total 60,000 training points, about 1,000 points belong to the negative class and 59,000 points belong to the positive class. From this observation, it is interpreted that the minority class datapoints needs to be up-sampled or choose a classifier which counteracts this imbalance.

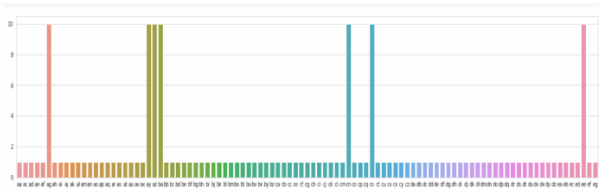

Fig 17 Histogram Bin Distribution Graph

It is assumed that some features are chart bin information and the prefix (the letter before the `_`) is the identifier and the suffix is bin_id (Identifier_Bin ID). Furthermore, all objects of the same graph have the same prefix. This information is used to find features that contain information about the chart area. It can be seen that there are 7 feature sets, each with 10 buckets. In other words, there are 7 charts divided into 10 panes each. For example, "ag" identifiers include ag_000, ag_001, ag_002, ag_003, ag_004, ag_005, ag_006, ag_007, ag_008, and ag_009. From Figure 17 above, the histogram identifiers can be extracted as follows: [`ag`, `ay`, `az`, `ba`, `cn`, `cs`, `ee`].

4.6 Data Preprocessing

Real world data is often missing values like "Nan". The cause of missing values can be due to data corruption or data retention errors. Missing data processing is essential while preprocessing the dataset because many machine learning algorithms do not support missing values. There are several ways to handle missing values. Some of them are:

- Impute missing data for continuous variable
- Impute missing data for categorical variable
- Other Imputation techniques
- Using Algorithms that support missing values
- Prediction of missing values
- Imputation using Deep Learning Library — Datawig

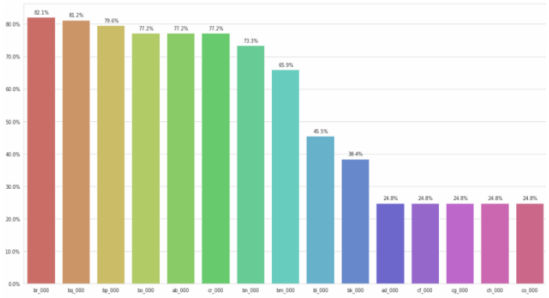

Fig 18 ‘Na’ -Missing value Distribution plot for labels

Upon analyzing the missing value distribution plot in Figure 18, it can be observed that most of the columns are having missing values, if all these feature columns with missing value are removed there will be datapoints with very less data for training the model. This could lead the model to overfit. Hence, all the features in the train dataset where the number of missing values is more than 70 percentile, columns having 42k missing values in the train dataset are alone removed. Seven features (‘br_000’, ‘bq_000’, ‘bp_000’, ‘bo_000’, ‘ab_000’, ‘cr_000’, ‘bn_000') had more than 70% of its values as Nan.Once all these columns are removed, the number of features is reduced from 171 to 164 columns.

For features with missing values less than 5%, those rows will be discarded. 128 features with less than 5% missing values. As a result, rows (4027 rows) that include missing values for these features will be deleted. For objects with missing values from 5 to 15%, these missing values are calculated as mean, and for the rest of objects with missing values % from 15 to 70%, the technique of imposition based on a model is used.

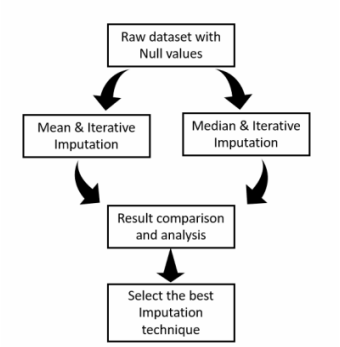

Fig 19. Imputation selection process flow

Fourteen features had 5-15% of their missing values and were passed through sklearn's simple transfer function and the missing values were entered using the "median". Then, for the features with 15-70% missing values, an iterative model-based imposition technique called Multivariable Chain Equation Impliance Function (MICE) is performed. The figure 19 depicts the process flow used to select the imputation technique. The median imputation technique is identified as better choice for missing value imputation. The figure 20 shows the process flow used to select the scaling technique.

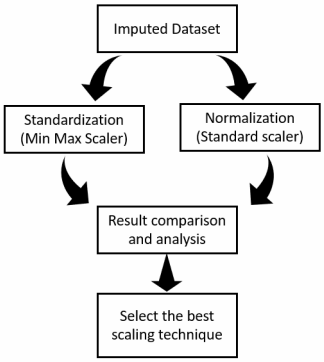

Fig 20. Scaling technique selection process flow

Python Code.4:

```
“from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import StandardScaler
def scale_data(df):
scaler = MinMaxScaler() # scaler = StandardScaler()
    df = pd.DataFrame( data = scaler.fit_transform(df) , columns = df.columns )
    return df , scaler”
```

Python code 4 performs MinMax Scaling and Standard Scaling using the MinMaxScaler and StandardScaler functions imported from sklearnlibrary.The figure 21 depicts the data frame variation before and after the application of standardization technique.

	aa_000	ac_000	ad_000
0	76698.0	2.130706e+09	280.00000
1	33058.0	0.000000e+00	191672.70588
2	41040.0	2.280000e+02	100.00000
3	12.0	7.000000e+01	66.00000
4	60874.0	1.368000e+03	458.00000

5 rows × 162 columns

	aa_000	ac_000	ad_000
0	0.001147	5.538069e-08	0.318041
1	0.000009	1.220252e-08	0.000036
2	0.000005	0.000000e+00	0.000000
3	0.018994	9.999999e-01	0.000516
4	0.014897	0.000000e+00	0.317721

5 rows × 162 columns

Fig 21.Data frame Before and After MinMax Scaling Technique

Thus, Standardization scales the given value between 0 and 1 which can be visualized from the above two different data frames.

76698.0	2.130706e+09	280.00000	-0.464137	-0.373066	-0.006088
33058.0	0.000000e+00	191672.70588	-0.616719	-0.373066	-0.006092
41040.0	2.280000e+02	100.00000	-0.456530	-0.373066	-0.001028
12.0	7.000000e+01	66.00000	-0.361438	-0.373066	-0.001053
60874.0	1.368000e+03	458.00000	-0.650491	-0.373066	-0.006092

Fig 22. Data frame Before and After Standard Scaling Technique

Thus, the normalization scales the given value between -1 and 1 which can be visualized from the above two different data frames. Figure 22 shows the difference in the data before and after performing standard scaling.

4.7 Feature Engineering

The need for feature engineering arises when there are multiple features to be concentrated for creation of improvised model based on specific machine learning algorithms. In general, it makes the dataset, compatible with the machine learning algorithm requirements and aids in improving the performance of machine learning models.Since the obtained APS fault dataset's label has been anonymized for proprietary reasons, only few feature engineering techniques can be performed.

As observed from the class feature distribution plot in figure 6.5, this dataset is highly imbalanced. Synthetic Minority Oversampling Technique (SMOTE) is employed to tackle this problem (Sonak et al., 2016). SMOTE configures the dataset with respect to class feature and oversamples the minority class. Python code 5 balances the dataset by creating duplicates of minority class points.

Python Code 5:

```
"from imblearn.over_sampling import SMOTE
def balance_data(df,label):
    over = SMOTE(sampling_strategy=0.3)
    under = RandomUnderSampler(sampling_strategy=0.5)
    steps = [('o', over), ('u', under)]
    pipeline = Pipeline(steps=steps)
    df, label = pipeline.fit_resample(df, label)
    df = pd.DataFrame(df)
    label = pd.DataFrame(label)
    return df, label"
```

The Figure 23 shows the class distribution plot after SMOTE technique deployment. It operates based on the sampling frequency passed as an argument to the smote function in imblearn python library.

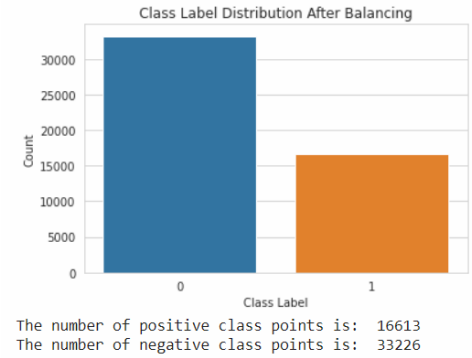

Fig 23. Class label distribution plot after balancing the class feature

There was one feature (cd_000) that had a single value for all data (standard deviation = 0). Since it will not add much value to our model performance, it can be removed. Finally, the total number of features used for training the machine learning models is 162.

4.8 Experimenting with ML Models

The Pre-processed data is passes through various models, hyperparameter tuning is performed, and each of the ML model is evaluated based on performance metrics like Accuracy, Precision, Recall, F1 Score and Confusion Matrix. The different models that are tried out here are:

- Adaptive Boosting
- Random Forest

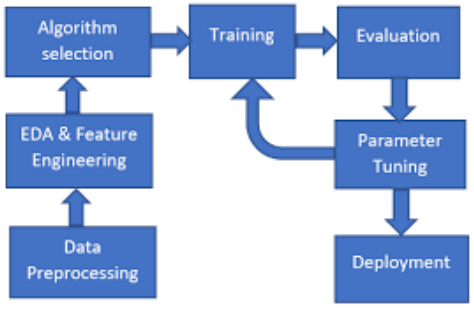

Fig 24.Process Flow for Model Generation

The process flow of building a model is shown in Figure 24. During training, the input data is used to gradually improve our model's ability to predict whether there is an error in the air pressure system. After completing the training, the model is evaluated to see how well the models perform on the test data. Evaluation allows us to test our model against data that has never been used for training (Costa and Nascimento, 2016). This metric allows us to see how the model might perform against data it hasn't seen yet. This is meant to represent how the model might behave in the real world. A good rule of thumb that I use to evaluate training is an 80/20 or 70/30 split.Once the evaluation is done the parameters are tuned to see if the performance of the model improves further. After tuning the parameters, the model id trained and tested again. This step is repeated until a best outcome is reached. Finally, the model can be deployed in an application.

4.8.1 User Interface Model and Output

This section depicts the User Interface (UI) model created for the deployment of ML fault prediction and visualization of Vehicle System parameters for better outcome and analysis.

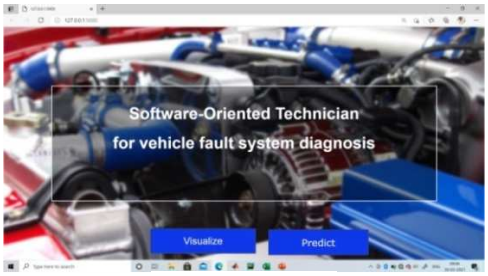

Fig. 25 Home Page

The home page of the Software-Oriented Technician web application is shown in figure 25.Hyper Text Markup Language (HTML) and Cascaded Style Sheets (CSS) have been used for designing the web pages. In this page,the user can choose to visualize the engine sensor data stored in database and predict the fault system using Machine Learning model.

Fig. 26 Visualization Input Page

If the user clicks the visualize button in the home page, the user will be redirected to the visualize data input page as shown in Figure 26. In this page, the user should specify the path of the Vehicle system data file which the user wants to visualize for further analysis of system specific parameters like Engine RPM, Intake Manifold pressure (IMAP) etc. The input file should be in the format of excel sheets or comma separated value (csv).

	AE	AF	AG	AH	AI	AJ	AK
1	SAE.ECT (ti	SAE.ECT Â°	SAE.ECT Â°	SAE.MAP (i	SAE.MAP ir	SAE.MAP k	SAE.RPM (1
2	Engine Coo	Engine Coo	Engine Coo	Intake Mar	Intake Mar	Intake Mar	Engine RPM
20	101	194	90	1053	9.7	33	1112
21	101	194	90	1053	9.7	33	1112
22	101	194	90	1481	10	34	1112
23	101	194	90	1481	10	34	1112
24	101	194	90	1481	10	34	1541
25	101	194	90	1481	10	34	1541
26	101	194	90	1481	10	34	1541
27	101	194	90	1481	10	34	1541
28	101	194	90	1931	10	34	1541
29	101	194	90	1931	10	34	1991
30	101	194	90	1931	10	34	1991

Fig. 27. Visualization Dataset

The dataset used for visualization is acquired from the vehicles with the help of (On-board Diagnosis) OBD tools and stored in Kaggle free source database (Cephasaxbureto, 2017). The Figure 27 pictures the snapshot of the engine parameters found in the dataset. The dataset contains sensor parameters like intake air temperature, exhaust Carbon di-oxide level, fuel density, engine speed etc. Python code 6 imports graph_objects function from plotly python library and creates an interactive plot.

Python Code 6:

```
“import plotly.graph_objects as go
fig.update_layout(
updatemenus=[go.layout.Updatemenu(
        active=0,
        buttons=list(
           [dict(label='Show All',method='update',
args=[{'visible': [True, True, True, True]},{'title': 'All Parameters',
                  'showlegend': True}])])])
fig.show()”
```

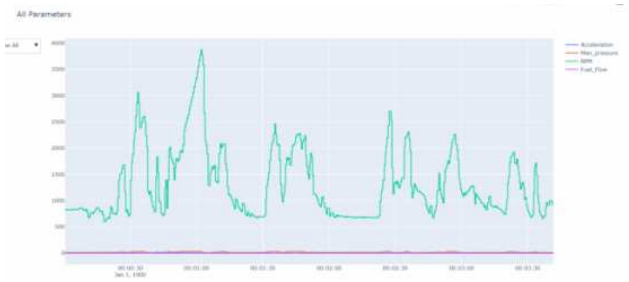

Fig. 28 Visualization Output

After specifying the input file, it has been programmed to preprocess the input dataset and select the features which are very essential for the analysis of engine system. In this paper, it selects the engine system sensor features such as RPM, acceleration, IMAP and fuel flow rate. The ploty python library is used to generate the interactive plot shown in the figure 28.

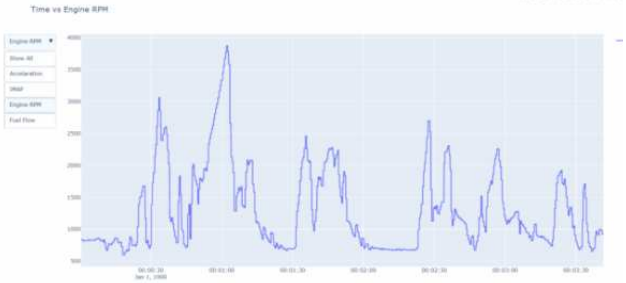

Fig. 29. Time vs Engine RPM Graph

The Figure 29 shows the four different plots with above mentioned parameters as their Y-axis and Time feature is taken as X-axis for all graphs. The interactive plot helps in identifying the maximum value and minimum value between the specified range. A dropdown feature has been appended in order to help the user to switch among multiple plots.

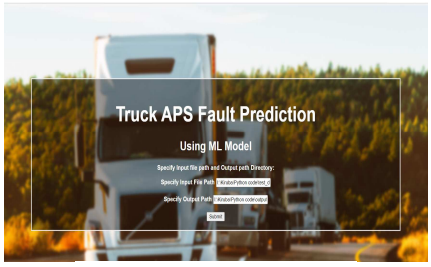

Fig.30. Prediction Input Page

When the user clicks the predict button in the home page, he will be redirected to the predict page depicted in the Figure 30. In this predict page, the users have to specify the paths of the file which holds the vehicle fault unlabeled dataset and folder location where the fault classified file needs to be generated. The file path is stored in the form and used in the fault classification python module. In the backend, the inputted file is read and processed with the model saved as pickle file which stores the weights of the imputation, scaling and ultimate classifier model. The preprocessed data is given to save baseline model for fault classification. The above explained process is achieved through python code 7.

Python Code.7:

```
“aps_data = pd.read_csv(path["inFile"], na_values=["na"])
model = joblib.load('E:\\python code 2\\models\\model.pkl')
veh_name = aps_data
veh_name['pred_class'] = model.predict_classes(aps_data)
veh_name.to_csv(path['outFile'] + '\\output_' + str(timestr) + '.csv',index=False)”
```

	A	B	C	D
1	Vehicle_name	act_class	aa_000	ac_000
2	Vehicle_01	0	-0.651459208	-0.370934234
3	Vehicle_02	1	0.29538202	-0.370934055
4	Vehicle_03	1	5.160773583	-0.370934035
5	Vehicle_04	0	-0.651388189	-0.370934165
6	Vehicle_05	0	-0.438588764	-0.370933957
7	Vehicle_06	1	1.657492029	-0.370934035
8	Vehicle_07	1	5.234110618	-0.370934035
9	Vehicle_08	0	-0.458558079	-0.370932809
10	Vehicle_09	0	-0.428613791	-0.370934035
11	Vehicle_10	1	2.123965672	-0.370934263

	A	B	C
1	Vehicle_name	act_class	pred_class
2	Vehicle_01	0	No Fault
3	Vehicle_02	1	Fault
4	Vehicle_03	1	No Fault
5	Vehicle_04	0	No Fault
6	Vehicle_05	0	No Fault
7	Vehicle_06	1	No Fault
8	Vehicle_07	1	Fault
9	Vehicle_08	0	No Fault
10	Vehicle_09	0	Fault
11	Vehicle_10	1	Fault

Fig. 31.Input file vs. Prediction Output

The Figure 31 depicts the difference between the input raw data file and the file generated as output in the specified path. As shown in the above figure, the input file has vehicle name, actual class and 171 features. The output file has a column “pred_class” which displays the predicted label as ‘Fault’ or ‘No Fault’. The predicted class can be compared with actual class in which ‘0’ and ‘1’ denotes ‘No Fault’ and ‘Fault’ respectively. This output file act as report which helps the users to classify the faulty system and access the working condition of their vehicles.

5. Performance analysis and comparison

Analyzing and comparing the performance of the ML classifiers helps in identifying a best predicting algorithm based on various metrics such as accuracy, precision, recall and F1 score. In this section, parameter tuning, comparing the performance metrics obtained after preprocessing steps, comparing the different classifiers with metrics such as accuracy, F1-meassure and the results of ML classification are briefly explained.

5.1 Python Implementation

The general python code used to explore the best parameter for the classifiers and confusion matrix to calculate the performance metrics are presented in this section. Python code 8 a tuning function is defined that performs HyperParameterTuningusingGridSearchCV for the given model and its parameters. It iteratively performs the training for specified classifier with different parameter values like learning rate, base estimator etc. and gives the best parameter as output.

5.2 Python Code 8

```
“from sklearn.model_selection import GridSearchCV
```

```
def tuning( x , y , model , params , cv=10 ,verbose=10 ):
clf = GridSearchCV(estimator= model,param_grid= params,scoring= 'f1_macro',
                   cv= cv, verbose= verbose,n_jobs= -1)
clf.fit( x , y )
   return clf.best_params_ , clf.best_score_"
```

Python Code 9 plots the Confusion Matrixbased on the true and predicted class labels. The 2x2 matrix is delivered as output since it is a binary classification problem.

Python Code 9

```
"from sklearn.metrics import confusion_matrix
def plot_confusion( y_test , y_hat ):
cf_matrix_test = confusion_matrix(y_test , y_hat)
group_names = ["TN","FP","FN","TP"]
group_counts = ["{}".format(value) for value in cf_matrix_test.flatten()]
   labels = [f"{v1}\n{v2}" for v1, v2 in zip(group_names,group_counts)]
   labels = np.asarray(labels).reshape(2,2)
sns.heatmap(cf_matrix_test, annot=labels, fmt='', cmap='Blues')
plt.show()"
```

Python Code 10 gives the classification report which includes precision, recall and F1 score and the accuracy score of a model.

Python Code 10:

```
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
print(classification_report(Y_test,predictionsRF))
acc = accuracy_score(Y_test,predictionsRF)
```

5.3 Adaptive Boosting Algorithm

The result observed from employing the adaptive boosting algorithm to create a fault detection model is portrayed below. The Table 1 shows the default parameter and their values of the sklearn's Adaboost classifier function.

Table 1. Default parameter values of adaboost classifier

S.No	Parameter	Value (default)
1	Base_estimator	None
2	N_estimator	50
3	Learning rate	1.0
4	Random_state	None

The confusion matrix of the adaptive enhancement classifier is shown in Figure 32. The confusion matrix is used to visualize important predictive analyzes such as recall, specificity, accuracy, and precision exactly. Confusion matrices are useful because they provide direct comparison of values such as true positives, false positives, true negatives, and false negatives.

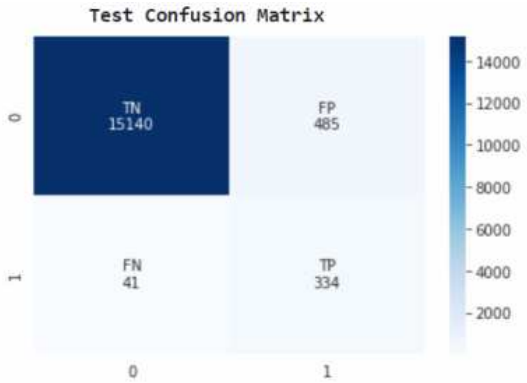

Fig 32. Confusion Matrix for Adaboost Classifier model

Table 2. Best parameter for adaboost after tuning

S.No	Parameter	Value
1	Base_estimator	None
2	N_estimator	700
3	Learning rate	0.1
4	Random_state	42

Hyperparameter tuning is performed for Adaptive boosting algorithm and the best parameters obtained are tabulated in Table 2.

5.4 Random Forest

The Performance observed from employing the Random Forest algorithm to create a fault detection model is portrayed below. The Table 3 shows the default parameter and their values of the sklearn's Random Forest classifier function.

Table 3. Default parameter values of random forest classifier

S.No	Parameter	Value(default)
1	n_estimators	100
2	max_depth	None
3	n_jobs	None
4	verbose	0

The confusion matrix of Random Forest classifier in the Figure 33 shows the true positive, true negative, false positive and false negative predicted values with respect to the test dataset.

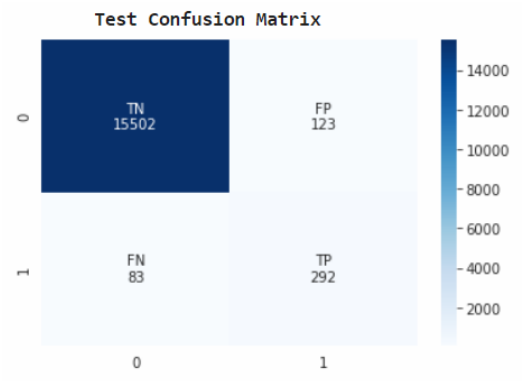

Fig 33. Confusion Matrix for Random Forest Classifier model

Table 4. Best parameter for RF after tuning

S.No	Parameter	Value
1	N_estimators	700
2	max_depth	125
3	n_jobs	-1
4	verbose	0

Hyperparameter tuning is performed for Random Forest algorithm and the best parameters obtained are tabulated in Table 4.

5.5 Performance Comparison

The Random Forest, Adaptive boosting classifiers are used for creating ML APS fault model, the results are tabulated for checking the improvement in the algorithm. To find the better classifier, Random Forest, Adaptive boosting classifiersare compared with the accuracy, precision, recall and shown in the Table 5.

Table 5. Performance metrics comparison after imputation

ML Algorithm / Classifier	Imputation	Precision	Recall	F1-Score
Random Forest	Mean	0.65	0.67	0.57
	Median	0.78	0.80	0.75
Adaptive Boost	Mean	0.54	0.81	0.45
	Median	0.75	0.78	0.73

Inference obtained from the Table 5 is that median imputation produces better results for nearly all algorithms used. This is because the data consist of a lot of outliers i.e., some observation points vary from other observations and median imputation handles outliers better when compared to mean imputation.

Table 6. Performance metrics comparison after scaling

ML Algorithm / Classifier	Scaler	Precision	Recall	F1-Score	Accuracy
Random Forest	MinMax	0.78	0.98	0.91	0.81
	Standard	0.95	0.71	0.82	0.78
Adaptive Boost	MinMax	0.87	0.95	0.85	0.79
	Standard	0.61	0.75	0.76	0.75

Imputed dataset is scaled using MinMax and Standard scaler techniques. Inference obtained from the Table 6 is that the classification results obtained from data that is scaled using MinMax scaler outperforms the standard scaled data. On an average 4.5% increase in accuracy can be seen when MinMax scaling technique is used.

Table 7. Result comparison between chosen classifiers – Before parameter tuning

ML Algorithm / Classifier	Accuracy	Precision	Recall	F1-Score
Adaptive Boost	0.81	0.71	0.76	0.75
Random Forest	0.85	0.59	0.80	0.88

The Table 7 depicts the performance comparison between chosen classifiers. It can be inferred that the best performing model is the Random Forest model with an accuracy of 85% and a recall value of 0.80 and has the highest f1 score 0.88. The above result analysis is tabulated without performing of hyperparameter tuning technique and it is to be noted that performance of each model can be improved further by tuning the parameters.

Table 8. Result comparison between chosen classifiers – After parameter tuning

ML Algorithm / Classifier	Accuracy	Precision	Recall	F1-Score
Adaptive Boost	0.89	0.75	0.91	0.77
Random Forest	0.91	0.87	0.94	0.89

From the Table 8, it can be inferred that Adaptive Boosting shows the least accuracy of about 89%. The accuracy of the Adaptive Boosting, and Random Forest has been improved by 8%, and 6% respectively after tuning the parameters.

6. Conclusions and future scope

A visualization module has been built to display interactive plots of vehicle sensor data. Four classification algorithms to solve the problem of determining if a vehicle has a specific APS component failure are studied and results are tabulated for deciding the best classifier model. The challenge of dealing with raw data which holds highly unbalanced data and a large number of missing values are handled during data preparation. Various data pre-processing techniques for missing value imputation and scaling are performed and the best imputation and scaling techniques are identified. Median Imputation presented better results when compared to mean and mode imputation. An average of 4.5% increase in accuracy was seen when MinMax scaling technique was used to scale the data. From the experimental results it can be summarized that, the Random Forest Model performs well for this classification problem when compared to other three models. It has the highest accuracy of 91% and recall value of 0.94.

The Web application, outcome of this proposed work not only serve as the perfect companion to minimize the cost of time and money involved in unnecessary checks done for fault system detection in trucks but also aids to quickly detect and isolate the faulty system avoiding failure propagation which may lead to even more hazardous circumstance. The design of the proposed system is quite simple compared to Quanqi et al. (2011) and also utilizes histogram bin data which is a popular form of data among heavy duty vehicles. As it is a unique kind of dataset, there are not many existing analyses work performed on this kind of histogram dataset. Despite these limitations in resources, the result looks impressive and further can be improved using neutral networks to attain better trained models. This paper only deals with the fault associated with engine and air pressure systems in

trucks. Further, it can be appended with other major vehicle systems like transmission, chassis, after treatment systems etc. to make it a one stop solution for all kinds of vehicle faults and maintenance issues.

References

Cephasaxbureto, 2017, OBD-II Dataset, Retrieved February, 2021, https://www.kaggle.com/cephasax/obdii-ds3.

Costa C.F. and Nascimento M.A., 2016, IDA 2016 Industrial Challenge: Using machine learning for predicting failures, *4th IEEE International Conference on Industrial Informatics*, Singapore, pp. 1081-1086.

Dua, D. and Graff, C. 2019. UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. *University of California, School of Information and Computer Science*, Irvine, CA

Gomes, I.P., Wolf, D.F, 2021, Health monitoring system for autonomous vehicles using dynamic bayesian networks for diagnosis and prognosis, *Journal of Intelligent and Robotic Systems*, Vol. 101, pp. 19-24. https://doi.org/10.1007/s10846-020-01293-y

Gosain, A. & Sardana, S. 2017, Handling class imbalance problem using oversampling techniques: A review, *IEEE International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, 13-16 September, Udupi, India, pp. 79-85. https://doi.org/10.1109/ICACCI.2017.8125820

Johanson M. & Karlsson L. 2007, Improving vehicle diagnostics through wireless data collection and statistical analysis, *IEEE 38th Vehicular Technology Conference, Baltimore, MD, USA*, pp2184 - 2188. https://doi.org/10.1109/VETECF.2007.458

Kohavi, R. 2001, Data mining and visualization, *In Sixth Annual Symposium on Frontiers of Engineering National Academy Press, DC*, pp. 30-40.

Nair V. and. Koustubh, B. P. 2017, Data analysis techniques for fault detection in hybrid/electric vehicles, *2017 IEEE Transportation Electrification Conference (ITEC-India).* https://doi.org/10.1109/itec-india.

Obodoeze F.C., Okoye F.A., Ifeyinwa O.N. 2018. Design and implementation of a vehicle fault detection system (FDS) with online and SMS fault reporting: Case study of ford motors, *American Journal of Engineering Research*, Vol. 7, pp. 53-64.

Pedregosa F., Varoquaux G., Gramfort A., Michel V., Thirion B., Grisel O., Blondel M., Prettenhofer P., Weiss R., Dubourg V., Vanderplas J., Passos A., Cournapeau D., Brucher M., Perrot M., Duchesnay É. 2011, *Scikit-learn: machine learning in python*, Journal of Machine Learning Research, Vol. 12, No. 85, pp. 2825−2830.

Pestana-Viana D., Gutiérrez R.H.R., de Lima A.A., Silva F.L.E., Vaz L., Prego T.D.M. & Monteiro U.A. 2019, Application of machine learning in diesel engines fault identification, In: Cavalca K., Weber H. (eds), *Proceedings of the 10th International Conference on Rotor Dynamics – IFToMM. IFToMM 2018. Mechanisms and Machine Science*, Vol. 61. Springer. pp 74-89. https://doi.org/10.1007/978-3-319-99268-6_6

Quanqi W., Jian W. and Yanyan W. 2011, Design of vehicle bus data acquisition and fault diagnosis system, *2011 International Conference on Consumer Electronics, Communications and Networks (CECNet), Xianning, China, 16-18 April 2011*, pp. 245-248. https://doi.org/10.1109/CECNET.2011.5768929

Ramentol, E., Caballero, Y., Bello, R., & Herrera, F., 2012, "SMOTERSB*: a hybrid pre-processing approach based on oversampling and Undersampling for high imbalanced datasets using SMOTE and rough sets theory, *Knowledge and Information Systems*, Vol. 33, No. 2, pp. 245-265. https://doi.org/10.1007/s10115-011-0465-6

Sankavaram C., Kodali A., Pattipati K.R. and Singh S. 2015, Incremental classifiers for data-driven fault diagnosis applied to automotive systems, *IEEE Access*, Vol. 3, pp. 407-419. https://doi.org/10.1109/ACCESS.2015.2422833

Sankavaram, C., Kodali, A., Pattipati, K. & Singh, S. & Bandhyopadhyay, P. 2010, Event-driven data mining techniques for automotive fault diagnosis, *21st International Workshop on Principles of Diagnosis*, pp. 1-8.

Sonak, A., Patankar, R., & Pise, N. 2016, A new approach for handling imbalanced dataset using ANN and genetic algorithm, *IEEE International Conference on Communication and Signal Processing (ICCSP)*, pp. 1987-1990.

Svärd C., Nyberg M., Frisk E., Krysander M. 2014, Data-driven and adaptive statistical residual evaluation for fault detection with an automotive application, *Mechanical Systems and Signal Processing*, Vol. 45 pp. 170–192. https://doi.org/10.1016/j.ymssp.2013.11.002

Xie P., Du L., Zhou B., Yu Y., Du H., Cui L., Xu H. 2019, Design and implementation of vehicle data real-time acquisition system, *Journal of Physics Conference Series 1176*, Vol. 5, 052059. https://doi.org/10.1088/1742-6596/1176/5/052059

Xie C., Wang Y., MacIntyre J., Sheikh M., Elkady M. 2018, *Using sensors data and emissions information to diagnose engine's faults*, *International Journal of Computational Intelligence Systems*, Vol. 11, pp. 1142-1152. https://doi.org/10.2991/ijcis.11.1.86

Biographical notes

M. Kiruba Thomas and S. Sumathi are currently in the Department of Electrical and Electronics Engineering, PSG College of Technology, Coimbatore , India