Unsupervised Machine Learning and Data Mining Techniques for Telematics Data to Uncover Fault Behaviour

Gaurav Khadse[1], Ravi Jambhale[2], Deepa Tavargeri Adiga[3], Prasanna P Tagare[4], and Nilesh Powar[5]

[1,2,3,4,5] *Cummins Inc. Survey No. 21, India Office Campus, Tower A, Balewadi, Pune, MH, 411045, India*

gaurav.khadse@cummins.com
ravi.jambhale@cummins.com
deepa.adiga@cummins.com
prasanna.tagare@cummins.com
nilesh.powar@cummins.com

ABSTRACT

The automotive industry has focused on monitoring the health and performance of vehicles to help customers improve uptime and reduce downtime with planned maintenance. Technologies like telematics are for continuous data flow. The data depicts details about the functioning of various components in the engine and subsystems and the Fault Codes (the diagnostic troubleshooting codes) associated with them. One traditional method used by the service representatives to address these Fault Codes is to follow the recommended troubleshooting trees. With the electronic engine and subsystem performance interdependencies, it gets challenging addressing the same. Also, the grouping of the Fault Codes is unknown if the Fault Codes that occurred are related to a specific cause. We implemented unsupervised machine learning and data mining techniques to address such issues. First, we enforced the co-occurrence theory that helps us understand the Fault Codes that occur together and exhibit dependencies and relations. Second, we implemented clustering algorithms to know groups/categories of Fault Codes based on their functional states. These studies provide insights into the failures of the components and their conditions. The study also helps in resolving the problems experienced by the engines and subsystems. Moreover, these methods address the issues in the early stage and help technicians improve uptime (early repairs and diagnostics). Further, this paper presents the results of the experiments aligning to the domain needs.

1. INTRODUCTION AND MOTIVATION

With the dawning of technological alternatives, automotive industries and their customers have adapted advanced means to manage and fix engine and subsystem failures at the sooner stages of the onset of faults. In doing so, vehicles' engine health data is captured and analyzed to uncover valuable business insights and track the nature and cause of the engine and subsystem failures. Engine health data depicts the detailed functioning of various engine components and subsystems and the associated failures. The failures are also known as Fault Codes - the diagnostic troubleshooting codes.

Cummins has its own designed Engine Control Modules (ECMs) to capture the performance parameters and Fault Code data. ECM captures both private and public category performance parameters. These Public parameters and Fault Codes are captured according to SAE J1939 (SAE J1939 Standards, 2022) standard. ECM can capture some key performance parameters in different formats, e.g., a snapshot at the instance of the occurrence of the Fault Code, a specific time duration aggregated form, and a continuous flowing form, depending on ECM's designed capability. With the advancements in wireless technology such as Telematics, ECMs can transmit data continuously. Such continuous engine health data is captured and utilized to develop analytical solutions for optimized diagnostics, prognostics, and maintenance of vehicles.

Telematics is a communication technology for the automobile industry based on information flowing to and generated from vehicles via wireless networks. Telematics combines the systems of wireless communications, information management, and in-vehicle computing to allow vehicle owners to use wireless communication functions to exchange and convey information and provide drivers and passengers with personalized information services. (Neumann, 2018).

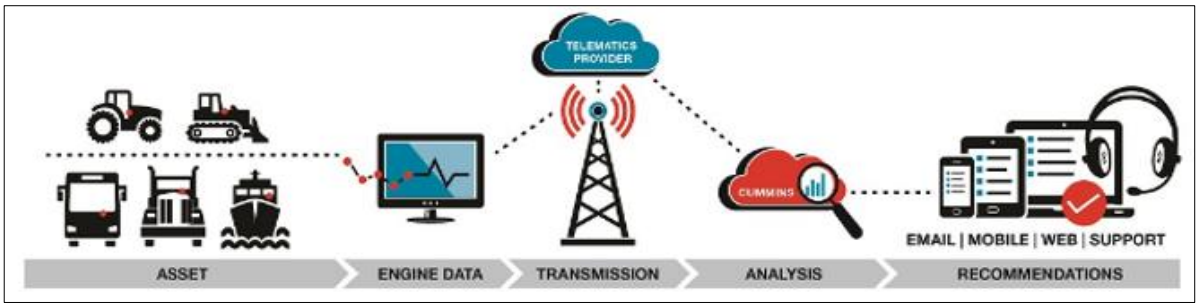

Figure 1. Generalized Architecture for Data Aggregation in a Telematics System.

Figure 1 depicts the generalized architecture for data aggregation in a telematics system. To diagnose the faults captured in the data of the engines and subsystems, based on their technical experiments and field repairing experience, engineers define troubleshooting trees for the routine and historical Fault Codes. With new initiatives and product launches, unhandled Fault Codes lack definitions and relevant troubleshooting trees. Sometimes, Fault Codes in multiple issues need to be traced back on large troubleshooting trees. Manual traversal and tracing of the fault in complex troubleshooting trees induce human errors.

To overcome these challenges, we developed analytical models that inform possible underlying issues and help the engineers focus on specific repairs by isolating the faults. The model further helps to

1. Achieve fast repair and
2. Improves the uptime of the vehicle.

In this paper, we are presenting two efforts to deal with the Fault Codes. The first is to isolate the Fault Codes and understand their dependencies through an unsupervised machine learning approach (Neumann, 2018). The other one is to use a supervised multi-label classification approach. Here we focus on studying critical Fault Codes for which the field team expect recommendation for fast repair based on the historical telematics data.

We analyze the sequencing of Fault Codes and their combinations to define the possible issue and provide a recommendation. We execute the analytical models on the continuously flowing data and provide recommendations for the respective engine and subsystems. The field service team visualizes the recommendations on dashboards and further decides on actions suitable for the Fault Code. To summarize, the paper proposes approaches to replace the traditional ways of following the recommended troubleshooting trees for Fault Codes.

2. PAST LITERATURE REVIEWS

Traditionally, Telematics has been used to track the position of vehicles with the help of the Global Positioning System (GPS). With the help of Advanced Analytics and Cloud Computing techniques, fuel saving and fuel monitoring techniques are explored. (Mesgarpour, 2013). With the advancement in technology and accessing Controller Area Network (CAN) bus information remotely using Telematics devices, driving behavior monitoring, and identifying individual drivers from their driving signatures, have also been explored (Wang, 2017). Some methodologies have also been explored to provide diagnostics and prognostics sides of Telematics. Those methodologies are the stepping stones for this paper to have appropriate troubleshooting trees further to handle complex, uncategorized Fault Codes and their behavior.

Additionally, work on the comparative methods on the careful feature engineering by domain experts and the automatic feature selection techniques provided an added advantage in selecting features and parameter values. It also helped in extracting structure from the data (Jordan Perr-Sauer, 2020). The impacts of various features and methods to find structure in unstructured data have come in handy for exploring the diagnostics of Fault Codes.

One of the most popular supervised learning tasks is multi-class classification, which involves more than two sets of labels. In multi-label problems, each example is associated with more than one target label. This method can be classified broadly into two groups, i.e., algorithm adaptation and problem transformation. The algorithm adaptation methods extend specific learning algorithms to handle multi-label problems directly, whereas the problem transformation methods are algorithm-independent. They transform the multi-label classification task into one or more single-label classifications, regression, or ranking tasks (Dhatri Ganda, 2018). In this paper, only problem transformation methods are discussed.

Evaluation of a multi-label classification algorithm is mostly difficult because multi-label prediction has an additional notion of being partially correct. One trivial way around would be to consider them incorrect and extend the accuracy used in the single-label case for multi-label prediction (Read) (Dhatri Ganda, 2018). Otherwise, other metrics can be used

for multi-label classification namely, precision, recall, and F1-score. The F1 measure is the harmonic mean of precision and recall and is a popular evaluation measure for information retrieval. (Grigorios Tsoumakas, 2011) (Buhmann).

Figure 2. PowerBI Dashboard.

3. APPROACH

3.1. Data Source and Definition

The data source we used to analyze the Fault codes is Telematics data. The data is also called heartbeat data since the data logging frequency is 1 Hz. The data is received continuously with key performance parameters. The Fault Codes appear in the form of a list as one of the columns in the data. Every row in the data depicts the state of the vehicle in terms of key performance parameters and a list of Fault Codes - mapped to its respective engine serial number, as shown in Table 1. The Fault Code list consists of different Codes, with each Fault Code designed to represent a specific problem in a vehicle. There can be a Fault Code representing the condition, for instance - *Aftertreatment Diesel Particulate Filter differential pressure* being above the normal operating range.

Every Fault Code has a defined severity level. However, here in our work, the Faults Codes are given equal importance to study their co-occurrences. The structure of the data is depicted in Table 1.

Table 1. Structure of Data.

ESN	Key Performance Parameters	Fault Code list
E1	n features	[FC4, FC2, FC6]
E2	n features	[FC4, FC3]

The acquired data for every vehicle consists of Engine Serial Numbers and the parameter values specific to that particular Engine. We considered 139 parameters that had good data quality, dropping the ones with missing values beyond the implementation of imputation strategies, Subject Matter Experts exclusion rules (due to the presence of more than 25% null values), and parameter range guidelines further to the inputs from Subject Matter Experts. The data consists of 2871 Fault Code List records along with parameters. There are 61 unique Fault Codes, and 39 out of 61 are Triggered Fault Codes. The final data for the analysis consists of a total of 104 parameters, including the key performance parameters and a list of Fault Codes of ESNs. The parameter selection based on domain knowledge helped have relevant parameters and accurate results according to domain needs.

3.2. Methodologies

In this paper, we use two methodologies to categorize similar Fault Codes and analyze co-occurring Fault Codes and their frequency of co-occurrence. To find groups of similar Fault Codes, an Unsupervised Machine Learning approach is implemented. To perform troubleshooting and study the co-occurrence of the Fault Codes and their frequency of co-occurrence, a Supervised Machine Learning approach is used.

3.2.1. Unsupervised Machine Learning Approach

To group the similar Fault Codes, Clustering - an Unsupervised Machine Learning approach was used; since the clustering techniques help combine similar entities to profile the attributes of the different categories. The clustering algorithms automatically helped recognize the patterns in the parameters and the associated list consisting of Fault Codes and to further analyze the data without the target label.

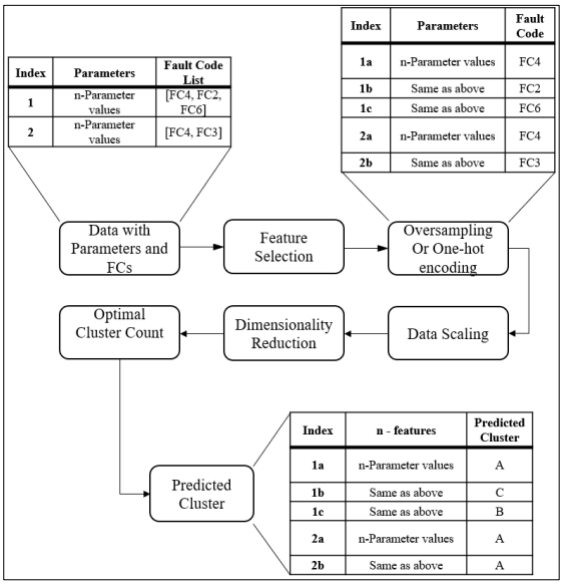

Figure 3. Unsupervised ML Approach – Clustering.

Figure 3 depicts the operational blocks for the unsupervised machine learning approach. The analysis is done by keeping ESNs out of context, making data independent of ESNs.

We prepared the data for analysis in two steps. Step 1 - oversampling with respect to the list of Fault Codes, and step 2 - One-hot encoding on a list of Fault Codes.

In the first step, as shown in figure 3, every row is duplicated for the number of Fault Codes (in the Fault Code list) times such that each row gets associated with an individual Fault Code. Hence, if a row consists of three Fault Codes in the list, then the row is duplicated thrice. For example, row index 1. is broken into three rows, namely, 1a, 1b, and 1c, since three Fault Codes are present in a Fault Code List column for index 1. The parameters remain the same for all the new rows as the original row with index 1. however, the Fault Code column is populated with three different Fault Codes in rows 1a, 1b, and 1c, based on the original row index 1.

Fault Code List	FC1	FC4	FC5	FC6	FC7
FC1	1	0	0	0	0
FC4, FC5, FC6	0	1	1	1	0
FC4, FC7	0	1	0	0	1

One-Hot Encoding of Fault Code List

Sample Parameter Values						FC1	FC4	FC5	FC6	FC7
0.24	0.25	0.35	0.7	0.87	0.33	1	0	0	0	0
0.16	0.8	0.63	0.21	0.10	0.11	0	1	1	1	0
0.21	0.16	0.01	0.01	0.12	0.21	0	1	0	0	1

Resultant Data

Figure 4. Clustering with One-hot encoded Fault Codes.

In the next step, we implemented one-hot encoding on the Fault Codes list to create extra columns depending on the total unique Fault Codes. Figure 4 shows the one-hot encoded representation of a Fault Codes list, where a unique Fault Code is treated as a categorical feature. The resultant data now consists of the parameter values and the additional one-hot encoded columns. Also, as there is no ordinal relationship among Fault Codes, we considered one-hot encoding where the presence of the Fault Code is represented with a binary value.

Having multiple features, with one having a large difference between the points and the other having a small difference, will have features with some large distance to be the driver of the distance, mostly. For example, the odometer reading in miles and features like voltage have values way smaller than the odometer reading. The 1-mile distance might not be as significant as the 1-volt value of any electronic sensor. Moreover, there are mixed numerical data, where each feature is entirely different from the other one. Thus, it is important to have the parameter values on a single scale. The Clustering on the normalized data works better than it does on the feature values on different scales. Min-Max scaler from scikit-learn is used to perform the normalization.

Most clustering methods depend on different types of distance metrics such that points close to each other tend to belong to the same cluster, whereas the points farther from each other belong to different clusters. The clustering algorithm is one of the algorithms that suffers through the curse of dimensionality (Matthew R. Boutell, 2004). If the nearest neighbor is to be found, it can be found using various available distance measures. But in high dimensions, some different situation arises. If the ratio is to be calculated between the closest and the farthest data points, it approaches 1. In such a case, the data points can be called uniformly distant from each other. This situation is observed in almost every other distance metric; however, it is more prominent in Euclidian than the Manhattan distance metric. Thus, all the data points are uniformly distinct from each other, making the separation

seemingly meaningless. Therefore, the dimensionality reduction technique comes to the rescue, and it is an important step before determining the optimal number of clusters. The number of features can be reduced without losing the important information from the data.

The dimensionality reduction technique used in our experiment is Principal Component Analysis. The covariance matrix was also calculated to see if the parameters are varying from the mean concerning each other, or if there is some relationship among the parameters. The calculation of the Eigenvectors and Eigenvalues of the co-variance is important to determine the principal components. The principal components are the dimensionality-reduced features or parameters we get from the original data; these are uncorrelated and contain the maximum possible information from all the original parameters. They represent the direction of the data that tend to explain maximum variance. The principal components are arranged in descending order such that the first component contains the maximum amount of information or the largest possible variance. The second component is perpendicular and uncorrelated to the first component and has the next largest possible variance. The principal components shouldn't be looked at to have a meaning since they are the linear combination of the original parameters.

Further, an object with three principal components is created and the normalized data is passed to fit the Principal Component Analysis object. The top features for the resultant three Principal Components turned out to be different.

Since partitioning of clustering is our target clustering method, we decided to go with the K-means algorithm. The elbow method is one of the best methods to find the optimum number of clusters. WSS (within the sum of squares) is used to find the optimal number of clusters on the data consisting of parameters and Fault Codes. WSS is nothing but the squared distance between each point of the cluster and its centroid.

For a range of K, we applied the K-means algorithm and calculated the square of Euclidean distance of each point from its cluster center and added it to the current WSS. Finally, we plotted the WSS concerning different numbers of K, i.e., clusters. From Figure 5, we can see that the bend occurred at K = 4, and the inertia fell compared to the situation at K = 5; thus, K = 4 is an optimal number of clusters.

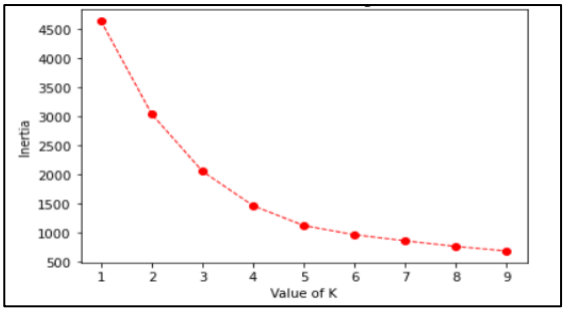

Figure 5. K-means Clustering algorithm - Optimal value of K.

The average Silhouette method for finding the optimal number of clusters is also explored to eliminate ambiguity in the Elbow method. Silhouette coefficients for each data point are needed to be calculated to see how much the data point is like its own cluster compared to other clusters. The average Silhouette approach is used to measure the quality of the clusters. The more the score better the quality of the clusters. First, we computed the Silhouette Coefficients for each data point and then the mean of all the samples to find the Silhouette score. There are three steps involved in calculating the Silhouette coefficient: computing the average distance of a point with all other points in the same clusters (*a(i)*), the average distance of a point with all the points in the next nearby cluster (*b(i)*), and the silhouette coefficient (*c(i)*).

The silhouette coefficient is calculated using

$$c(i) = \frac{b(i) - a(i)}{\max\left(b(i), a(i)\right)}$$

The values of *c(i)* are averaged out to have the Silhouette score.

The plot of the Silhouette coefficient concerning the cluster label is generated to have the visual representation to decide the optimal number of clusters. The graph in Figure 6 represents a measure of how close each data point is close to the points in the neighboring clusters. The value of this measure lies between -1 to 1. The definitions according to the documentation on the official scikit-learn website are as follows:

- The Silhouette coefficient of +1 indicates that the sample is far away from the neighboring clusters.
- The Silhouette coefficient of 0 indicates that the sample is on or very close to the decision boundary between two neighboring clusters.
- The Silhouette coefficient <0 indicates that those samples might have been assigned to the wrong cluster or are outliers.

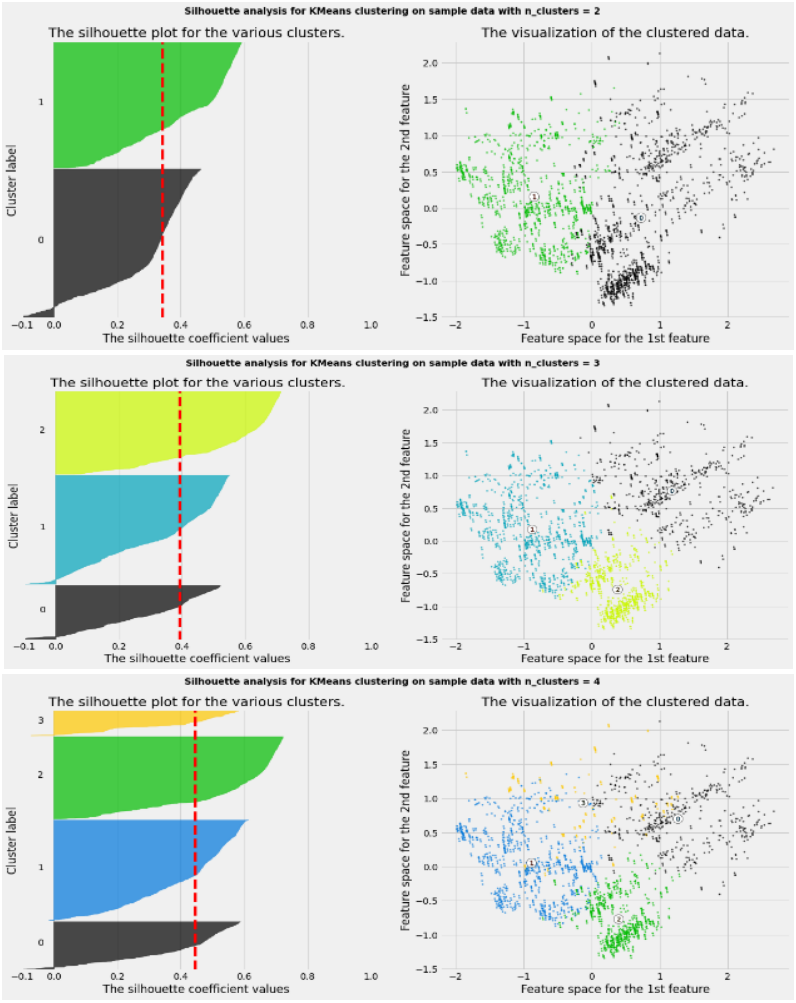

Figure 6. Silhouette Coefficient Analysis.

Silhouette score for various numbers of K-clusters is calculated. From Figure 7, the maximum average Silhouette score is for n_clusters = 4. Thus, using this method as well, the same number of clusters we found to be optimal.


```
For n_clusters = 2 The average silhouette_score is : 0.3443722003154834
For n_clusters = 3 The average silhouette_score is : 0.39372246117026766
For n_clusters = 4 The average silhouette_score is : 0.44598351145346077
For n_clusters = 5 The average silhouette_score is : 0.4379224407053931
For n_clusters = 6 The average silhouette_score is : 0.4387952620914384
For n_clusters = 7 The average silhouette_score is : 0.40653704919924255
For n_clusters = 8 The average silhouette_score is : 0.40701838675192287
For n_clusters = 9 The average silhouette_score is : 0.39016607758393756
```

Figure 7. Average Silhouette score for different numbers of clusters.

After choosing four clusters as the optimal number of clusters, the prediction is done on the Fault Codes, thereby generating a cluster label for all the available Fault Codes. Some of the Fault Codes appeared to belong to more than one cluster since the parameter values associated with those Fault Codes are different.

The number of the Fault Codes (not unique) belonging to different Cluster labels can be seen in Figure 8.

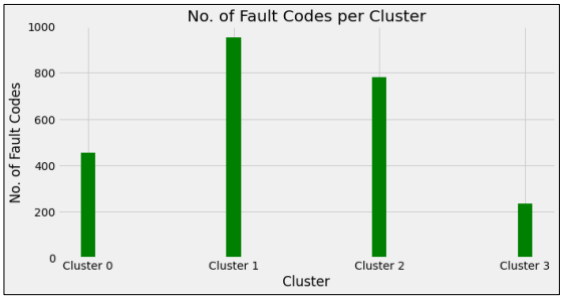

Figure 8. Number of Fault Codes per Cluster.

3.2.2. Supervised Machine Learning Approach: Multi-label Classification

Multi-label classification is one of the supervised classification approaches we used to predict the list of Fault Codes using the parameters from the data. The supervised machine learning algorithms require training data well tagged with the target labels. This algorithm learns from labeled training data to help predict outcomes for unforeseen data.

The data for the experiment includes the list of Fault Codes. For each record, the list of Fault Codes gets generated. The respective list of Fault Codes may involve more than two Fault Codes for a single instance. Those Fault Codes can be considered as target labels for the respective set of parameters from the data. So, at a time, the dataset has more than two classification labels. In such a case, only multi-label classification can help predict the relevant results for the respective instance.

Different methods are used in the multi-label classification approach, such as Problem Transformation and Adapted Algorithms. The target label is transformed from a multi-label problem to a single-label problem in the Problem Transformation Algorithm. As the name suggests, Adapting Algorithm performs directly multi-label classification.

The Problem Transformation method is carried out in different ways such as:

- Binary Relevance
- Classifier chain

Binary Relevance is a simple technique that treats each target label as a single class classification problem. For example, consider the below case, where parameters are independent variables, and C is the target label.

Parameters	C1	C2	C3
n-features	0	1	1
n-features	0	0	1
n-features	1	1	1

Parameters	C1
n-features	0
n-features	0
n-features	1

Classifier 1

Parameters	C2
n-features	1
n-features	0
n-features	1

Classifier 2

Parameters	C3
n-features	1
n-features	1
n-features	1

Classifier 3

Figure 9. Binary Relevance Classifier.

In Binary Relevance, a single problem with multiple classes is split into different single class classifiers, as shown in figure 9.

Whereas, in the Chain Classifier, the first classifier gets trained on an independent variable in the classifier. Then each next classifier is trained on the next independent variable and all the previous classifiers' target labels.

Figure 10 delineates parameters and C (dependent variable). It also shows that a classifier chain would transform this problem into a different single-label problem.

Parameters	C1	C2	C3
n-features	0	1	1
n-features	0	0	1
n-features	1	1	1

Parameters	C1
n-features	0
n-features	0
n-features	1

Classifier 1

Parameters + C1	C2
n-features 0	1
n-features 0	0
n-features 1	1

Classifier 2

Parameters + C1 + C2	C3
n-features 0 1	1
n-features 0 0	1
n-features 1 1	1

Classifier 3

Figure 10. Chain Classifier.

This classifier chain is like binary relevance, the only difference being it forms chains to preserve label correlation.

In addition, there is another technique to classify multi-label data, i.e., Label Powerset Technique. This classifier is trained on the unique combination of the labels found in the train data, but this approach has its limitations. When the unique combinations of the target labels are fewer, then the performance of this approach is better than other models. But, if unique combinations are high, it is difficult to train the model. The complexity of the model increases, which gives less accuracy.

Each approach has its advantages and limitations. A binary relevance classifier works with individual target labels that do not consider the target labels dependencies on each other. The Label Powerset technique requires a smaller number of unique target label combinations, but there are many unique combinations of target labels in the given data. It will affect the performance of the label powerset. In the classifier chain, dependencies of Fault Code are considered. Also, there is no restriction on unique combinations of the target label, and it fulfills the paper's objective. Further, the classifier chain algorithm is discussed in detail.

In Figure 11, the blocks represent the process flowchart of supervised multi-label machine learning technique implementation. In the previous section, data and data source was explained in detail. According to that, the key parameters are 103, and the Fault Code list is considered the target label for the analysis and model development.

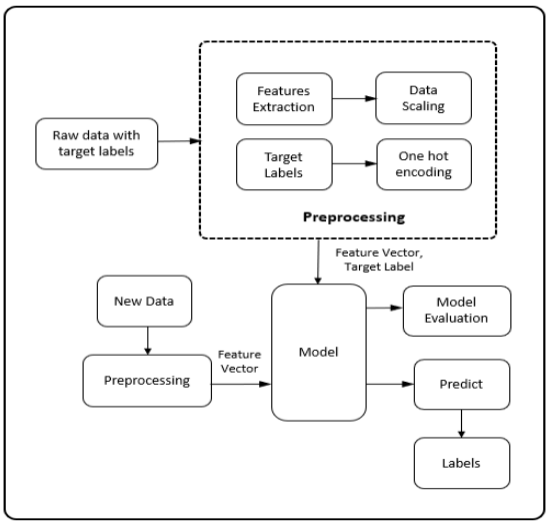

Figure 11. Supervised classification learning model development flowchart.

In the preprocessing, 67 parameters get extracted from 103 parameters using the feature extraction technique. This is used to reduce data from high dimensions to low dimensions, which is helpful for a better understanding of data and, in turn, helps reduce the input parameters of the model in other words, it mitigates the curse of dimensionality. A high correlation filter and recursive feature elimination technique are used to drill down the parameters. In the high correlation filter method, a high correlation parameter gets dropped from the list of parameters. The respective correlation threshold value is considered based on the thumb rule used in the correlation analysis. Recursive feature elimination is the feature selection technique used to remove features that do not substantially affect the target variable or prediction of output. In other words, this method removes the weakest features from the data. For normalizing the data parameters into a standard scale, data scaling is one of the processes in the pipeline of the model.

In the available data set, the target label is in the form of a list comprising Fault Codes, such as [‘FC1’, ‘FC2’, ‘FC5’, ‘FC10’]. Converting the Fault Code list to one-hot encoded data plays an important role. Most machine learning algorithms did not work with categorical labels directly. They required input and output records in the form of numbers. So, One-hot encoding can convert such categorical lists into a numerical format using 0 and 1 binary digits. After these preprocessing steps, scalable data get generated into a standard format.

In model training and the model evaluation parts; the model is trained and evaluated on such scaled data. Model evaluation techniques are used to check model performance. It is pretty easy to identify its performance using precision, recall, accuracy, etc. But in multi-label, it is not easy to locate the proper subset of the target label to the predicted set. In multi-label classification, the result can be fully correct, partly correct, or fully incorrect (Matthew R. Boutell, 2004). To overcome this problem, a few metrics are used to measure the performance of the multi-class classifier. In that, for micro and macro averaging precision and recall, a label-based measure is used, and for subset accuracy, the accuracy_score function is used (Dhatri Ganda, 2018). The drawback of this measure is that multi-class classification problems have a chance of being partially correct, but here the partially correct matches are ignored. So, for the performance evaluation of the model, we considered all the metrics which gave more confidence.

After training the model, new data comes into the picture. New, unseen data is pre-processed the same way as the training dataset. After the pre-processing, the data is fed to the model for prediction. This process flow can be depicted in Figure 11.

4. RESULT AND DISCUSSION

In our work, the clustering method is used to find the group of fault codes with similar behavior, and multi-label classification is used to predict the fault codes for a given record and to indicate the sets of co-occurring fault codes for the given form. These proposed approaches are helpful for fault code diagnostics in the telematics data.

The clustering approach helps in identifying the group of Fault Codes, which, in turn, helps the domain experts identify and name the clusters and compare one occurrence of Fault Codes with the other. It also helps study the common parameter values belonging to a specific cluster. Thus, an in-depth analysis of Fault Codes and the associated parameters is possible by grouping them and studying the chain of occurrences. The clusters of Fault Codes are shown in Figure 13.

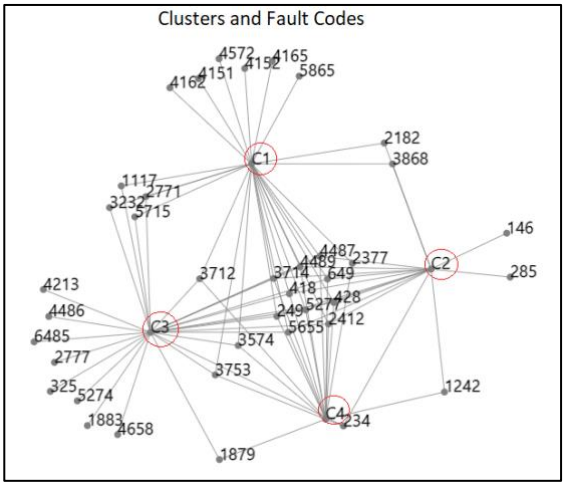

Figure 13. Cluster and Fault Codes.

The multi-label classification method is started from the raw data. Feature extraction, data scaler, and one hot encoding APIs convert raw data into model data. As we move on to preprocessed data passes to train the model, and after model evaluation, we found that multi-label classification algorithms performed well on the telematics data. Using the tree-based model as the base model of the multi-label classification model as per the API requirement model gives better validation consequences. The weighted F1 score and average accuracy measure the model performance. In the final model of classification, the XGBoost classifier was used as the base model. 39 target labels were used to develop respective models, along with the required parameters.

Based on the prior understanding of the data and the model, the results were almost close when data were fitted in binary relevance classifier and classifier chain. After close observation, the classifier chain shows better results than the binary relevance classifier, shown in Table 2 below. Among these two methods, the binary relevance classifier does not provide any relation between the target label; instead, the classifier chain gives it.

Table 2. Model performance metrics.

Model	Accuracy	F1_score
Binary Relevance Classifier model	85 to 87%	85 to 87%
Classifier Chain model	87 to 90%	87 to 90%

This multi-label classification approach is also used to find the co-occurrence of the Fault Code. The representation of such Fault Code occurrence is shown in Figure 14. For example, FC4, FC5, FC8, FC11, FC12 and FC13 occurred in set_11.

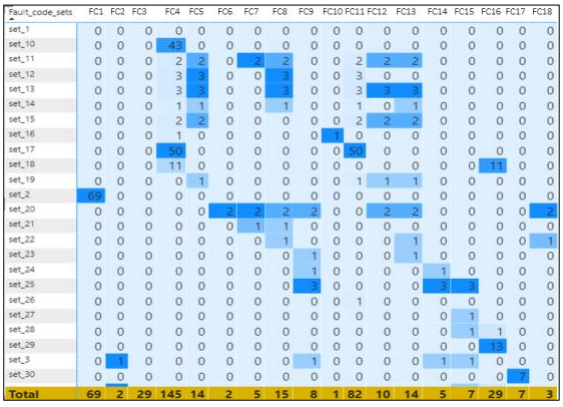

Figure 14. Fault code co-occurrence chart.

5. CONCLUSION

In the case of the Clustering approach, categories of similar Fault Codes are formed. These categories will be provided names by the Subject Matter Experts. The approach has also helped in the detailed analysis of Fault Codes. It has helped to know distinct and common Fault Codes in different Categories and their functional behavior. Clustering helped subject matter experts to profile the attributes of different groups, organize the volume of data, and achieve intrinsic grouping of the unlabeled telematics data.

Moreover, a supervised multi-label machine learning approach has helped in troubleshooting Fault Codes. It has helped analyze co-occurring Fault Codes, along with the frequency of occurrence. It also helps in guiding Subject Matter Experts to understand Fault Code behavior, their dependencies, and the significance of co-occurrence. Additionally, identification of implicit and functional relationships between co-occurring Fault Code can be achieved. For the given set of telematics parameters, the approach helps identify the Fault Codes that are likely to co-occur. For example, when the FC2 occurs, the following troubleshooting tree is followed:

- Check for primary Fault Codes
- Check whether the Engine speed sensor is malfunctioning
- Check whether the turbocharger oil seal is leaking oil into the air intake/exhaust system
- ECM calibration revision history check

In *the Check for primary Fault Codes* step, before troubleshooting FC32, troubleshooting any Fault Code that has occurred within the last 25 engine operating hours from the cluster FC32 belongs to, is carried out.

Co-occurrences of Fault Codes help identify relationships within the data, illustrating not just which codes appear together but how often they appear, providing a means of assessing the prominence of the combination. Also, such an analysis helps subject matter experts to be informed of multiple possibilities that are inherently not apparent.

6. FUTURE WORK

We would like to extend the work for analyzing specific Fault Codes that could further help the organization cash in the benefits of uncovering Fault Code behavior and their dependencies. It will eventually help the organization in better detection of root causes for faults. Eventually, it will help to

gain financial advantages through optimized diagnostics.

ACKNOWLEDGMENT

We are especially appreciative of the technical insight and guidance provided over the course of this project by all the directors from Cummins. Mrs. Subhalakshmi Behera provided us with key guidance on planning all the activities and helping us understand business requirements. Mr. Virendra Parte provided us with key insights regarding data extraction and storage. All these colleagues were key contributors to this effort. Finally, the authors thank Cummins, Inc for the opportunity to publish this work.

REFERENCES

Buhmann, A. P. (n.d.). *Classification of Multi-labeled Data: A Generative Approach.* 8092 Zurich, Switzerland: Institute of Computational Science.

Charu C. Aggarwal, A. H. (2001). On the Surprising Behavior of Distance Metrics in High Dimensional Space. *Springer-Verlag Berlin Heidelberg*, 420-434.

Dhatri Ganda, R. B. (2018). A Survey on Multi-Label Classification. *RTPL*, 19-23.

Domingos, P. (2012). A Few Useful Things to Know about Machine Learning. *communications of the ACM*, 50(10).

Grigorios Tsoumakas, I. K. (2011). Random k-Label sets for Multi-Label Classification. *IEEE*.

Jordan Perr-Sauer, A. D. (2020). *Clustering Analysis of Commercial Vehicles Using Automatically Extracted Features from Time Series Data.* NREL.

Matthew R. Boutell, J. L. (2004). Learning multi-label scene classification. *Pattern Recognition*, 1757-1771.

Mesgarpour, M. (2013). *Overview of Telematics-Based Prognostics and Health.* Springer.

Neumann, T. (2018). The Importance of Telematics in the Transport System. *TransNav: International Journal on Marine Navigation and Safety of Sea Transportation*, 12(3):617-623.

Perr-Sauer, J. (2020). *Clustering Analysis of Commercial Vehicles Using Automatically Extracted Features from Time Series Data.* NREL.

Read, J. (n.d.). *A Pruned Problem Transformation Method for Multi-label Classification.* The University of Waikato, Hamilton, New Zealand.

SAE J1939 Standards. (2022). Retrieved from SAE International: https://www.sae.org/standardsdev/groundvehicle/j1939a.htm

Wang, B. (2017). *Driver Identification Using Vehicle Telematics Data.*

BIOGRAPHIES

Gaurav Khadse is a Data Science professional working as a Senior Data Scientist at Cummins, Inc. He holds a Bachelor of Engineering in Electronics and Telecommunication from Govt. COE (University of Pune) and pursuing a Master of Technology in Data Science and Engineering from Birla Institute of Technology and Science, Pilani. He has been working in the field of Data Science for eight years. He has got the opportunity to explore Automobile, Telecommunications, and Sports domains. He has a special interest in the Internet of Things and Natural Language Processing. Apart from Data Science, he spends his time building wearable devices based on IoT platforms.

Ravi Jambhale is an Associate Data Scientist, working in Cummins since last year. He has completed a bachelor's in production engineering from Kolhapur University, India. and MTech. in Mathematical Modelling and Simulation from Pune University, India. He has been working in the field of data science since last year and his area of interest is to study and implement machine learning models.

Deepa Tavargeri Adiga is a Principal Data Scientist, working at Cummins. She has a rich experience of 20 years spanned across research, industry, and academia. Deepa is Natural Language Processing (NLP) researcher and currently working on building data sciences solutions for the

automotive industry. Her contributions have been in the domains of Requirements Engineering, Behavioral Sciences, Cognitive Sciences, and Manufacturing utilizing NLP, machine learning, deep learning, and data science methodologies. In her free time, Deepa enjoys reading, painting, gardening, and long walks.

Prasanna Tagare works as Engineering Analytics Tech Advisor at Cummins. He has a rich experience of 20 years in rotating machines like alternators, Electric motors, Internal Combustion engines, and related systems. He holds a bachelor's in electrical engineering from the University of Pune, India. He recently completed the Executive Programme in Engineering Management from SPJIMR Mumbai, India. He has filed a Patent on integrating the Data Analytics Signal to Service.

Nilesh Powar is the Advanced Analytics Director at Cummins. He holds a bachelor's in electronics engineering from the University of Bombay, India, an M.S in Computer Engineering from Wright State University, and a doctoral degree in Electrical and Computer Engineering from the University of Dayton, Ohio. He has over 20+ years of experience in the field of image processing, machine learning, statistical pattern recognition, and system integration. He had worked in the US as a Distinguished Research Scientist for the University of Dayton Research Institute, Dayton, OH, USA. Recent efforts involve data analytics for die-casting, predictive analysis for supply chain management, and video summarization using deep learning.