Master Thesis

Master's Programme in Network Forensics

Forecasting Components Failure Using Ant Colony Optimization For Predictive Maintenance

Thesis in Digital Forensics, 15 credits

Halmstad 2020-06-07

Ankit Gupta , Durlabh Shahi

HALMSTAD UNIVERSITY

ABSTRACT

Failures are the eminent aspect of any machine and so is true for vehicle as it is one of the sophisticated machines of today's time. Early detection of faults and prioritized maintenance is a necessity of vehicle manufactures as it enables them to reduce maintenance cost and increase customer satisfaction.

In our research, we have proposed a method for processing Logged Vehicle Data (LVD) that uses Ant-Miner algorithm which is a Ant Colony Optimization (ACO) based Algorithm. It also utilizes processes like Feature engineering, Data preprocessing. We tried to explore the effectiveness of ACO for solving classification problem in the form of fault detection and prediction of failures which would be used for predictive maintenance by manufacturers.

From the seasonal and yearly model that we have created, we have used ACO to successfully predict the time of failure which is the month with highest likelihood of failure in vehicle's components. Here, we also validated the obtained results.

LVD suffers from data imbalance problem and we have implemented balancing techniques to eliminate this issue, however more effective balancing techniques along with feature engineering is required to increase accuracy in prediction.

ACKNOWLEDGEMENTS

I must express my heartfelt appreciation to my wife for providing me with unfailing support and continuous encouragement throughout my study and through researching and writing this thesis. Also, I would like to thanks Halmstad University, for supporting me during my study.
I am heartily thankful to my supervisor Reza Khoshkangini for his valuable advice and guidance during my research and design of the report. Finally, I would like to thank my thesis partner Durlabh Shahi for his support and commitment while writing the thesis report. This achievement would not have been feasible without them. Thank you.

Ankit Gupta, June 7, 2020

I would like to give my heartfelt gratitude to my parents for their support and care. In addition, I would like to thanks Halmstad University for supporting me throughout my study.
I am greatly thankful to my supervisor Reza Khoshkangini for his continuous advice and guidance throughout our research and construction of the report.

Durlabh Shahi, June 7, 2020

CONTENTS

ACRONYMS

ACO Ant Colony Optimization

LVD Logged Vehicle Data

ACO-FAR Ant Colony Optimization based fault aware routing

MCBM Moore Algorithm Condition Based Maintenance

ACO-RVM Ant Colony Optimization Relevance Vector Machine

ACO-TCSP Ant Colony Optimization based test case prioritization

SVM Support Vector Machine

SI Swarm Intelligence

PCA Principle Component Analysis

1 INTRODUCTION

This thesis has been undertaken to satisfy the requirement of "Masters In Network Forensics" .
Failure of a component can happen at any time in equipment and machines. These failures could result due to many reasons such as mechanical failures, software faults, problems in sensors, actuators, bugs, malicious software, intrusions, etc. Consistent occurrence of faults may lead to an increase in maintenance cost, affect the production quality, assembly line, claim rates by the customers, and so on. All of this add up to be a significant loss for the manufacturer. Therefore detection of faults is a necessity for manufacturers. Continuously improving technologies in the field of informative and computing serve this growing need for fault detection. The question today is not only detection of the faults but detecting them faster to avoid faulty products being shipped to customers, increasing claim rates and loss to the manufacturer and their brand value. Fault Detection is very crucial in every industry today because it helps in high savings of overall cost. Not only that, as faults can occur in areas that can affect human lives, so it is really crucial for an overall safety-related aspect to find the faults as soon as possible to avoid life-threatening incidents. This need to find faults early has given rise to predictive maintenance, fault detection is an integral part of predictive maintenance. Predictive maintenance is a method that helps us to predict the point of breakage of equipment based on evaluation of parameters concerning components of machinery. It helps to estimate when maintenance would be required. It is the most effective type of maintenance in respect to cost savings as it ensures maintenance of those parts that really maintenance. It assist the manufacturer to estimate in which components failure could occur and what mitigation plans could be set up. It prevents small problems to turn into serious failures. It is a method to predict the condition of the equipment so that the component could be changed, just before the occurrence of any failure

1.1 MOTIVATION

Fault detection in terms of predictive maintenance is very crucial in today's time. We decided to research in this area as it is very relevant topic in today's manufacturing scenario. Predictive maintenance helps to detect the possible failures and take action before it can happen to

yield many ripple benefits and saving the manufacturer from many losses from warranty claims, damage to the brand image. It automatically helps to reduce the downtime of the machinery and improve the lifetime of the machinery. It monitors the performance of equipment during regular operation for possible fault detection. It significantly reduce the duration for maintenance of equipment, it can cut the production duration lost to the maintenance. Similarly, it also reduces the overall cost of supplies that had to be ordered immediately if no predictive maintenance strategy was implemented as loss in maintenance is lessened. In the same way,Ant Colony Optimization (ACO) is a powerful algorithm as it has a number of benefits such as it can bring quick discovery of optimal solutions. Due to this reason, it is highly useful for applications with dynamic functionality. In the same way, inherent parallelism is supported in ACO, i.e. engineers could exploit parallel features inherently present in the data. It can adapt to new changes in parameters quite rapidly. Similarly, it helps to solve the Traveling Salesman along with other related problems for optimization. In our research, we have found that people have researched on fault detection in the context of predictive maintenance in various algorithms such as Support Vector Machine (SVM) (Support Vector Machine). However, we have not found much solid research on fault detection in the context of predictive maintenance using Ant Colony Optimization(ACO), which could be useful in the automobile industry.

1.2 HYPOTHESIS

We hypothesize that ACO could be a solution for analyzing the dataset for predictive maintenance. We want to see how effective ACO is for detecting the failure of components in the context of predictive maintenance. We are going to execute this by using ACO to solve the classification problem in the dataset to classify the dataset into healthy and unhealthy ones. The model for classification attempts to classify test dataset into healthy and unhealthy ones based upon the observation from training dataset.
The dataset that we used in this research is Logged Vehicle Data (LVD). We have researched on this data set for Predictive Maintenance using Ant Colony Optimization. The research question for this thesis would be **"To what extent can we predict vehicle quality problem (component failures) exploiting Ant Colony Optimization algorithm?"**

1.3 ANT COLONY OPTIMIZATION

ACO is an optimization technique that falls under swarm intelligence. Swarm Intelligence (SI) is intelligence that occurs through the col-

lective behavior of a self-managing community of systems that may mimic the community of real-world animals such as bats, bees, and ants. In Swarm Intelligence (SI), individual agents take action on their own to perform a straight forward task. However, their actions in the collective swarm are solving complex situations on a macro level. A simple alteration in the behavior of few agents could lead the swarm solving big problems that the individuals are not aware of. ACO could also be termed as a type of swarm intelligence algorithm ACO is a technique for optimization which is a metaheuristic algorithm. Meta-heuristic algorithms are the algorithms that tend to find the optimal solution out of the possible solutions using a heuristic. In this, a heuristic is created to extract the optimal solution for optimization in the situation. This is really beneficial in a situation where there is limited information or limited computational power. It is a technique for optimization which is based upon the social behavior of colonies of ants. It works in a way to find the shortest path for solving the optimization problem. The problem is changed into a graph for ACO's implementation. Here, ants that are artificial in nature mimic the real ones by evaluating various routes and finding the best route. They use a parameter known as a pheromone in the form of mathematical measure,s which are updated at each interval to construct the optimal model. These pheromones help to find the best solution by the method of convergence, where the ants converge on the best path using certain criteria. There is an exchange of information between the ants in the form of pheromone as the majority of ants will converge to that path that has a high proportion of pheromone.

1.4 PROPOSED METHOD

Our proposed system assist in prediction of component failures by solving classification problem using machine learning techniques. our method includes data mining processes such as Data pre-processing of Logged Vehicle Data(LVD), Feature Engineering and Learning, and Modelling based upon an algorithm, which is an ACO variant. This is also shown in Figure 2.

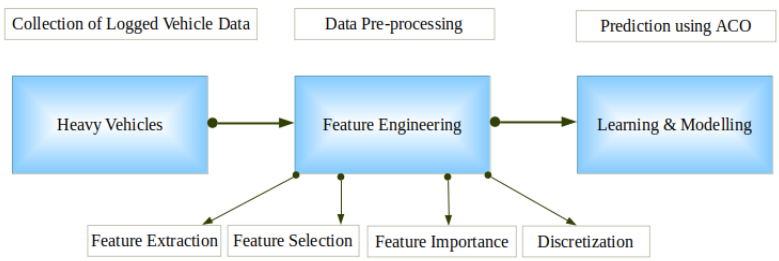

Figure 1: Proposed method

1.5 CHALLENGE

We faced a lot of challenges in our thesis especially in the case of LVD (Logged Vehicle Data) as the data that we had was unbalanced data that means whatever result that we get after training and testing with ACO variant (Ant-Miner) algorithm could not fully be trusted, so we need to overcome this unbalanced dataset problem. We also had a lot of noise in data which we had to overcome.

1.6 CONTRIBUTION

The contribution that we have done in thesis is the implementation of solving the fault prediction task using artificial intelligence techniques along with ACO based Ant-Miner Algorithm It consists of processes such as data cleaning, feature extraction, feature selection etc. In addition to this, we have also given a new method for the pruning of rule 3.3.4.

1.7 OUTLINE

This thesis report is divided into five major sections. Chapter 2 Chapter 2consists of Background, which gives information about the related work of our thesis topic. Chapter 3 Methodology of the thesis is described in Chapter 3, which also describes our method workflow. Experimental Evaluation Chapter 4 is described in Chapter 4. Discussion is given in Chapter 5. Finally, a conclusion is shown in Chapter 6.

2

RELATED WORK

In this chapter, we have described various related work that is relevant to our thesis topic which is detecting component failure for the predictive maintenance using optimization algorithm such as Ant Colony Optimization (ACO).

Many studies have been going on in this area. Mohanapriya et al. [8] have used Ant Colony Optimization based fault aware routing (ACO-FAR) for balancing of the load in the networks that are faulty in processor or chip-set. It is also used for routing based upon the awareness of faults. Here, Ant Colony Optimization (ACO) is used to find the best or shortest path in the network from source to a base node where the concept of artificial ant and pheromone updating is implemented through a node. Three mechanisms are implemented here, which are detection of faults, searching of the path, and selection of the path. Detection of faults helps to pinpoint the faults in the system, which leads to reconfiguration step, then in path searching step, the direction to output channel is found, and then by using Ant Colony Optimization (ACO) , best output channel is determined by analyzing and omitting path's likelihood towards faulty areas.

Bougacha [2] have used Ant Colony Optimization (ACO) in order to determine the best flow of decisions in order to optimize the overall significance of a system for production. Here,Ant Colony Optimization (ACO) is used to optimize the problem of decision making. Ant Colony Optimization (ACO) is being used for maintenance and production scheduling jobs to see its effect on a single machine.Ant Colony Optimization (ACO) is being compared with another algorithm knows as Moore Algorithm in the context of maintenance based upon condition Moore Algorithm Condition Based Maintenance (MCBM). By comparing both of them, it is observed that the utilization of equipment for production is more with ACO for job scheduling than with Moore Algorithm Condition Based Maintenance (MCBM), along with that the duration of waiting job is also lesser than Moore Algorithm Condition Based Maintenance (MCBM) and due to that, the duration of maintenance is also less. It also shows how prognostic information helps to reduce faults and errors and helps in smooth production and maintenance. This is analyzed with the help of ACO being looking at its performance with another algorithm.

Zhiwen Liu et al. [6] implemented a novel way to fuse data utilizing Relevance Vector Machine with the help of ACO for fault detection of the gearbox. In order to reduce the unreliability of sensor data that is based upon vibrations, an intelligent technique is used that is based

upon RVM and is termed as Ant Colony Optimization Relevance Vector Machine (ACO-RVM) . Higher accuracy is achieved with the implementation of this technique in the detection of faults than RVM with the implementation of simple cross-validation with the overall accuracy of 96.25 percent compared to 91.25 percent of RVM with simple cross-validation. In this algorithm, after data pre-processing, values of parameters are initialized, which are trained on RVM on the classifier, then ACO search and updates solution using probabilistic decision scheme which yields optimum variables for testing RVM classifier which gives the result of classification. Here ACO helps in the selection of optimal parameters for RVM.
Bharti Suri and Shweta Singhal have used ACO [10] in another domain, which is the prioritization of testing during the maintenance of software to check its usability after various updates. It helps to check the maximum amount of errors in the minimum duration possible. Here, the prioritization techniques are used that will reorganize the test packages for testing programs. These techniques are used in the environment where there is limited time duration. They have analyzed to see how effective is Ant Colony Optimization based test case prioritization (ACO-TCSP) for regression testing. In the experimentation phase, Ant Colony Optimization based test case prioritization (ACO-TCSP) is used to test on various programs and for various limitations of time. Ant Colony Optimization based test case prioritization (ACO-TCSP) is tested with various constraints of time duration. It is observed that Ant Colony Optimization based test case prioritization (ACO-TCSP) performs better in terms of the time duration of execution and level of accuracy at a higher constraint of time. The fastest duration for Ant Colony Optimization based test case prioritization (ACO-TCSP)for test packages is achieved, and greater optimum solutions are found at higher time duration level.
Reza Khoshkangini et al.[4] proposed a system based upon machine learning that can forecast the vehicle failures using the logged vehicle data and warranty claim data. In the paper, the proposed method comprised of integration of data, feature selection, feature extraction, and (training and prediction). Fault detection in relation to predictive maintenance is the core of the paper as given in its title. Experiments have been described in the paper which shows various benefits of the system such as the ability to predict vehicles that will be getting faulty in the future with high accuracy and feature to predict claims related to warranty precisely.
Tran [11]et al. implemented ACO algorithm in order to optimize and smooth the process of Maintenance, Repair and Overhaul. The result obtained from the implementation of this algorithm is also studied in comparison with the performance of another scheduling program where ACO based scheduling method performed well in dynamic scheduling environment.

Ant Colony Optimization (ACO) has been used in various domains in the context of predictive maintenance and fault detection. In the further chapter, we have explained the methodology that we have used in order to process the data to find a solution to the problem that we are researching in this thesis.

3

METHODOLOGY

We have seen various papers for fault detection that are using ACO. Based on that learning, we have set the main goal of our thesis to be to predict fault detection in relation to predictive maintenance with the help of ACO. This thesis will help us to assess the effectiveness of ACO for the classification of faulty vehicle components in the context of predictive maintenance, which is also the reason for choosing this thesis. This section describes the method that we have used in order to experiment and process our data to get the predicted outcome. This chapter is categorized into three main modules such as: Data pre-processing, Model Construction and Learning . Our proposed method workflow for processing our data is majorly structured in these sections. The proposed workflow, which is defined in Figure 2, consists of four main modules.

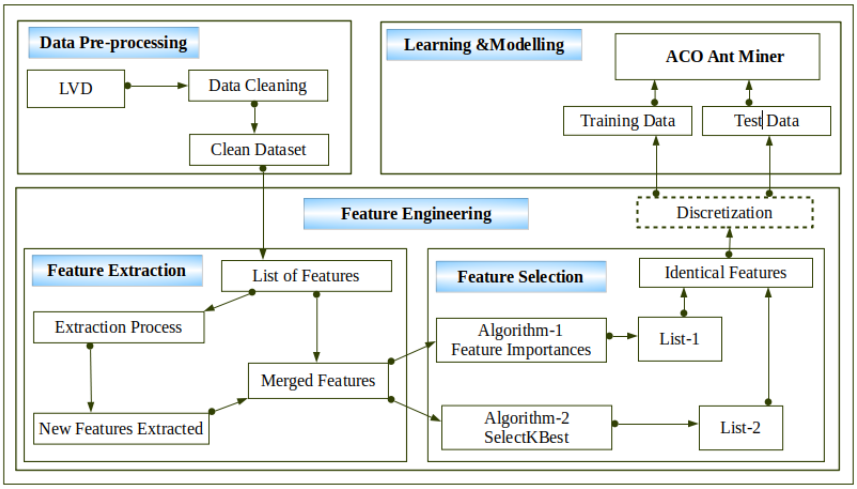

Figure 2: Proposed method

3.1 DATA PRE-PROCESSING

It is a step to filter to the data from inconsistent, wrong data, which are fused with many errors such as missing data, incompatible data types, and so on, which does not give meaningful information. In the case of Data pre-processing, we are trying to prepare the data so that a lot of irregularities such as errors and noise, etc. in the data could be solved, which would later help in further processing. In this, we have searched and found missing values and eliminated them from data to make the data organized. We are using Python for programming all these updates on the data. We are also using a lot of data mining

modules such as pandas, NumPy, and so on in order to refine the data for processing. As we can see in Figure 1, we take the LVD data and try to clean the data in the Data Cleaning phase to obtain little refined data than earlier.

3.1.1 *Feature Engineering*

Then we go to the Feature Engineering phase. Feature Engineering is the engineering of the features which is done in order to make the input of the data compatible with the algorithms of data mining. It also tries to enhance the data so that it will boost the overall accuracy yield by the algorithm .In Feature Engineering, we have used Feature Extraction and Feature Selection. Feature extraction is a method that is used to create the brand new features, which is discussed in the section(). Feature selection, it is the process to remove the irrelevant features from the data. Both of these techniques are further explained.

Feature Extraction

Feature extraction is the technique of reducing the dimension of raw data so that it can be used to process easily and effectively. As we know that large data set consist of a large number of variables that required a lot of computational time and processing. The feature extraction is the method that is used to select the new features from the existing features. It also combines the features to make the brand new features. So, this process helps to increase the dimension of the data set to make new valuable features. The primary motivation for this process is to find out the hidden information from the features. We have extracted the hidden features from our LVD data set. There are a lot of good Feature extraction algorithms such as:Principle Component Analysis (PCA),independent component analysis. We have not applied any feature extraction algorithm as we have the cumulative values in the data set. So, we have done the extraction by finding the difference of samples to extract hidden information from the existing feature of raw data set as shown in the given table and insert the extracted feature into a new column, and a new feature is added to an existing feature. In the next section, we have selected features using the feature selection algorithm SelectKBest.

F1	F1_1	F2	F2_1	. .	Fn	Fn_1
S1	0	S1	0	. .	S1	0
S2	S1-S2	S2	S1-S2	. .	S2	S1-S2
S3	S2-S3	S3	S2-S3	. .	S3	S2-S3
S4	S3-S4	S4	S3-S4	. .	S4	S3-S4
.	.	.	.	. .	.	.
Sn	Sn-1-Sn	Sn	Sn-1-Sn	. .	Sn	Sn-1-Sn

Table 1: Feature Extraction Process

In the given table, F1 to Fn is the existing number of features. S1 to Sn is the n number of samples of the dataset. F1_1 to Fn_1 is the Extracted features added to list of features. S1-S2 to Sn-1-Sn is the number of sample extracted from the existing feature and inserted into the new column as shown in the given Table 1.

Feature Selection

Feature selection is the process of removing irrelevant features and select the best useful feature to make a good predictive model. Feature selection plays a vital role in improving the accuracy of the model. The irrelevant feature can mislead and reduce accuracy. Feature selection before modelling helps to make a good prediction. There are a lot of feature selection algorithm, but we have used the ***SelectKBest*** algorithm. It is one of the well-known feature selection algorithm. In the previous section, we have discussed the feature extraction. Our dataset is quite a large dataset. We have more than 736 features, as shown in the figure 3

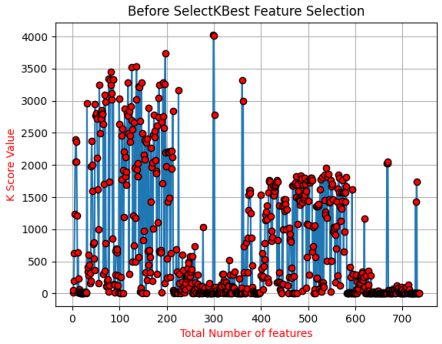

Figure 3: Before Feature Selection

. It is not easy to find relevant features manually. SelectKBest algorithm is good for feature selection. SelectKBest algorithm chooses the features on the basis of their K score, as shown in the figure 4 .

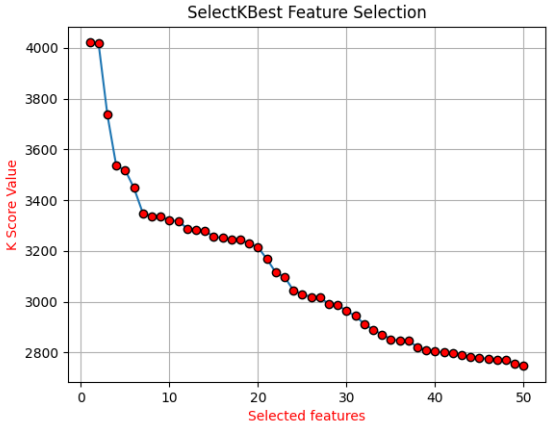

Figure 4: SelectKBest Feature Selection

Features	Range Values
lx_pmu_p1fwm_engine_speed_torque_h_x_index_1	[0-4062.54]
lx_pmu_p1fwm_engine_speed_torque_h_y_index_1	[0-3298.03]
lx_pmu_p1fwm_engine_speed_torque_h_y_index_10	[0-166.30]
lx_pmu_p1fwm_engine_speed_torque_h_y_index_11	[0-143.01]
lx_pmu_p1fwm_engine_speed_torque_h_y_index_12	[0-245.37]
lx_pmu_p1fwm_engine_speed_torque_h_y_index_2	[0-4264.01]
lx_pmu_p1fwm_engine_speed_torque_h_y_index_3	[0-2028.80]
lx_pmu_p1fwm_engine_speed_torque_h_y_index_6	[0-358.68]
lx_pmu_p1fwm_engine_speed_torque_h_y_index_7	[0-301.27]
lx_pmu_p1fwm_engine_speed_torque_h_y_index_8	[0-256.94]
lx_pmu_p1fwm_engine_speed_torque_h_y_index_9	[0-205.98]
lx_psc_p1app_gear_lever_automatic_time	[0-3.139]
lx_psc_p1apu_clutch_number_disengages	[0-1.155]
lx_psc_p1aqd_gearbox_movement_gear1r_gear1	[0-372341.57]
lx_psc_p1aqg_gearbox_movement_gear23_gear2	[0-405145.51]
lx_psc_p1aqh_gearbox_movement_gear23_gear3	[0-262359.76]
lx_psc_p1aqi_gearbox_movement_gear23_neutra	[0-666354.54]
lx_psc_p1aqj_gearbox_movement_split_direct	[0-520543.02]
lx_psc_p1aqk_gearbox_movement_split_indirec	[0-499997.59]
lx_psc_p1aqm_gearbox_movement_range_low	[0-104172.90]
lx_psc_p1aqn_gearbox_movement_range_high	[0-104164.10]
lx_psc_p1arb_economy_mode_time	[0-3.04]
lx_psc_p1ash_coolant_pump_number_of_activat	[0-309093.41]
lx_psc_p1ast_main_log_drive_distance	[0-593601.93]
lx_psc_p1asu_main_log_drive_fuel	[0-208781.62]

Table 2: Illustration of Selected features by using SelectKBest Algorithm.

Features	Range Values
lx_psc_p1asv_main_log_drive_time	[0-8181.05]
lx_psc_p1ata_main_log_coasting_time	[0-1894.41]
lx_psc_p1atd_main_log_economical_time	[0-6879.95]
lx_psc_p1atf_main_log_key_on_time	[0-15659.11]
lx_psc_p1ato_main_log_total_energy	[0-3.18]
lx_psc_p1bbo_total_engine_time	[0-9291.28]
lx_psc_p1bbz_total_engine_revolutions	[0- 7.14]
lx_psc_p1eon_total_number_of_park_brake_man	[0-63330.00]
lx_psc_p1e8y_total_number_of_park_brake_app	[0-64420.00]
lx_psc_p1ivo_air_production_modulator_heate	[0-1.00]
lx_pst_p1b3c_62_elenginegreenareatime_log	[0-2.77]
lx_pst_p1c20_362_vdlcomprdisjunctiontime_log	[0-4.01]
lx_pst_p1c20_363_vdlcomprconjunctionnottraction	[0-2.59]
lx_pst_p1c20_364_vdlcomprconjunctiontractiontim	[0-1.17]
lx_pst_p1eol_1074_mv5	[0-257733.33]
lx_pst_p1e1e_812_tav_comp	[0-7.34]
lx_pst_p1em9_12_pmairdryercartridgeresetvehicl	[0-6.48]
lx_pst_p1em9_13_pmairdryercartridgeresetengine	[0-3.95]
lx_total_vehicle_distance_rule_based	[0-595362.25]
x_pst_p1c20_365_vdlcomprtotaltime_log	[0- 5.13]
age_months	[0-41.00]
age_years	[0-3.40]
index	[0-12.00]
slx_pfe_0008_nbr_of_gearshifts	[0-3.56]
lx_psc_p1bby_total_fuel_consumption	[0-304646.02]

Table 3: Illustration of Selected features by using SelectKBest Algorithm.

The main process of the SelectKBest algorithm is it selects the feature based on their K higher score. The user can easily pass the value of K according to their need, and this algorithm will return the higher K score features. In the next section, we will see the process of ***Feature Importance***.

Feature Importance

Feature importance is the process of finding an important feature from the list of features. If you have a large dataset, in that case, it is not an easy task to find the important feature from your dataset. So, Feature importance is a good process to find a good feature to make a good model, and it helps to increase the performance of the

model. The working of feature importance is based on the higher the score, and it is a more important and relevant feature for the output variable. Random forest and Extra tree classifiers have the inbuilt class of feature importance. We have a good number of features. So, we have selected the important features, as shown in the figure 5. In the next section, we discussed Data Balancing.

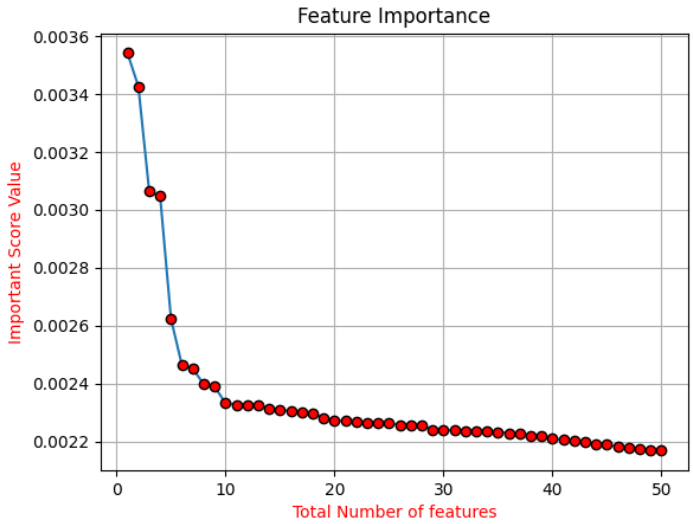

Figure 5: Feature Importance

Data Balancing

Our data had a lot of problems, such as missing data, improper data types. However, we have overcome those in the data cleaning process. Still, the major challenge in the dataset was the data imbalance in the data as there were a significantly greater number of "0" class labels than "1" class labels in class column" as shown in the figure 6

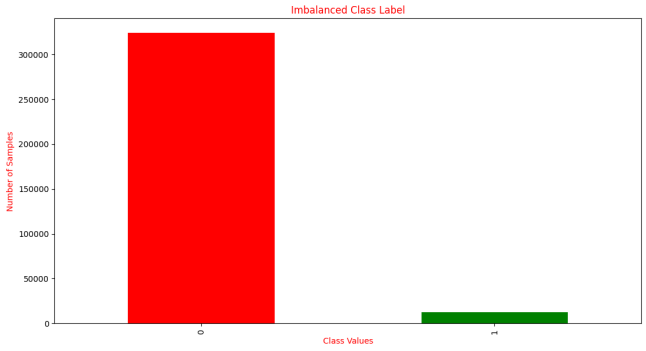

Figure 6: Imbalanced Class values

We have considered "0" class label as healthy and "1" class label as unhealthy. We solved this problem by using random oversampling

where we randomly added samples from minority_class which is "0" class label to balance the dataset. After its implementation , we get following result in figure 7

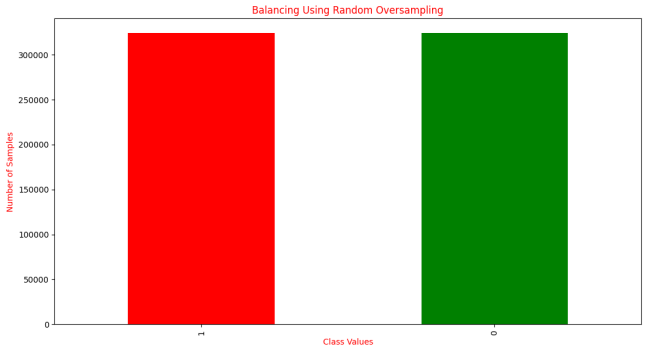

Figure 7: Imbalanced Class values

Discretization

After balancing of data, we have to undergo the process of **Discretization**. The reason for using discretization for processing of data is that we are using ACO for training the test data, and based on the result, we are predicting the test data. However, we also know that ACO is an algorithm that does not process the continuous data well, so for ACO to work on our dataset, we have taken the help of discretization, which will convert the continuous data into categorical data. For discretization, we could use various algorithms, but we have used pandas cut function [7] for discretizing our data. The reason for doing this is that discretizing our data with cut function reduces overall complexity as it is a simple and straight forward process. Through discretization, we converted the continuous data into the categories of "Very_Low", "Low" ,"Medium", "High" and "Very_High". After this, Then we process the data with the algorithm.

3.2 MODEL CONSTRUCTION

In this section, we described how we created the model for failure prediction. We have formed two models. For seasonal model, we have taken the concatenated data of two seasons which are summer and autumn of 2016. It is used as train dataset which is predicted on the summer and autumn months of 2017. In the same way, concatenated data of winter and spring of 2016 is taken as train dataset to get predicted on the winter and spring months of 2017. For yearly model, we have taken the yearly data of 2016 and 2017 as train data and predicted it on the yearly data of 2018.

3.3 LEARNING

In this section, we have described the algorithm (Ant-Miner) that we have used to learn on the LVD data. It processes our data to make the prediction of failure. It has major sections such as: construction of the rule, pruning of rule, heuristic calculation and updation of pheromone as given in the figure 8

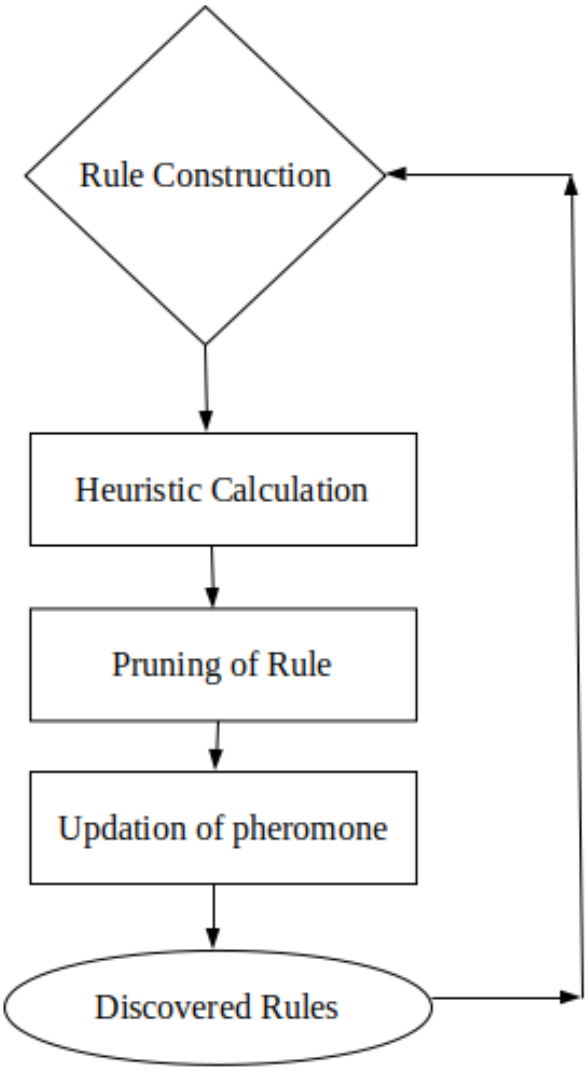

Figure 8: Flow Diagram of Ant-Miner

3.3.1 *Overview*

The algorithm that we have used for processing our data is based upon ACO(Ant Colony Optimization) algorithm. It is a meta-heuristic algorithm that is used to search and find optimal solutions using the heuristic strategy. The core of the algorithm is the artificial ant's foraging behavior for finding the best way to the solution. The success of this foraging behavior depends upon another important element, which is "pheromone." Pheromone is the chemical substance that is left on a path by the ants. The deposition of pheromone becomes higher on the path where the distance to the food source is smaller and more ants began to follow that path where one best path remains in the end. ACO tries to mimic this property of pheromone addition and evaporation, which finally gives the best solution. This is why ACO is good at solving optimization problems.
Dorigo et al. [3] gave the algorithm with ACO in the year 1990, and since then, various variants of ACO have been proposed for solving various problems to satisfy various requirements. H.S. Lopes et al. [9] gave a variant of ACO known as Ant-Miner, where it was compared with other algorithms to assess its performance in terms of accuracy which it clearly surpassed. Ant-Miner is an ACO based algorithm which could be taken as a variant of ACO. This algorithm extracts association rules from the data which helps to classify the data into various labels. Extracting rules for classification seemed like the primary goal of Ant-Miner.The algorithm that we used is a variant of original Ant-Miner.It is different from the original Ant-Miner in

some areas.The algorithm could be understood through Figure 1.

Algorithmus 1 : Proposed Ant-Miner Algorithm

```
TrainSet= total training cases;
Rulelist=[ ](Empty rulelist initialized);
while TrainSet > max_uncovered do
    t=1(ant index);
    j=1(test convergence index);
    Initialize all terms with same pheromone level;
     REPEAT;
    An ant starts with empty list and add term to rule iteratively;
    Prune rule with proposed technique;
    Update by increasing the pheromone in trail followed by ant
      , increasing pheromone according rule quality and
      evaporating pheromone on rest trails;
    if current rule= previous rule then
        j=j+1;
    else
        j=1;
    end
    t=t+1;
    UNTIL( t>=No_of_ants OR j>= Num_rules_converge);
    SELECT best rule among partial rules;
    ADD best rule to discovered rulelist;
    TrainSet=TrainSet-cases correctly covered by best rule;

end
```

Ant-Miner is focused on the foraging behavior to ants to find the solution where association rules are created, and based upon it, classification is done to find the optimum solution. The algorithm initiates with a rule list which is vacant at the beginning; however, it keeps on increasing with rules as the algorithm process the data and keeps on adding the rule. Here, the rule is presented as "IF Term 1 AND Term 2 AND .., THEN target class value".Here each term could be understood as a feature, operator and value. The IF part is known as the antecedent, and the THEN part is known as consequent.
In the algorithm, rules are discovered and are put into a list, which is considered a rule list, which is mined and discovered from the data. The base pheromone level is also initialized. Each iteration of loops in the algorithm finds a rule, and those cases that are already covered by the rule are reduced from the list, and this goes on continuously until maximum cases that are uncovered are greater than max_uncovered level which is a condition to end WHILE loop. There is another loop of REPEAT UNTIL which comprises of important steps of Ant-miner that is the construction of the rule, pruning of rule, pheromone updation and so on.The REPEAT UNTIL loop ends when the number of rules constructed is greater or equal to threshold (No of ants) or j is greater

than or equal to(Num_rules_converge)

3.3.2 *Construction of Rule*

During construction of the rule, individual ant begins with a list which is vacant. Then artificial ant tries to add terms into the rule list, which also correlates with the ant's path. The likelihood of a term to be selected into the rule totally depends upon the heuristic value along with associated pheromone value. A term is added to rule with each iteration, and addition is done until few conditions are met which is either the coverage of any specified rule is less than minimum amount of cases for each rule and same features occur in the rule such as IF(fuel_consumption = high) AND (fuel_consumption = high) . as rules are mined from the data. The probability for adding term into the rule is given by the formula [9] :

$$P = \frac{(\eta_{ij}.\tau_{ij})}{\sum_{i=1}^{a} .x_i. \sum_{j=1}^{bi} \eta_{ij}.\tau_{ij}} \tag{1}$$

η_{ij} heuristic value of term

k is the number of class labels

τ_{ij} pheromone value in ith jth position of (feature, value)

x_i becomes one if ant covers corresponding feature else its zero

a is amount of total features

b_i is values in domain of features

After discovering all the necessary terms, the algorithm then finds the class attribute, which is in the majority among covered cases and assigns it to the rule, which increases the rule's quality.

3.3.3 *Heuristic Calculation*

In the Ant-Miner[5], the value of heuristic is determined by the function of heuristic, which is dependent on information theory. Here, entropy is a measure that helps in measuring the term's quality for classification. The entropy is calculated for the pair of feature and associated value. It is calculated using the formula (2), and the heuristic function is calculated using the following formula 3

$$Entropy_{ij} = -\sum_{w=1}^{k} \left\{ \frac{freq_{ij}^{w}}{|T_{ij}|} \right\} .log_2 \left\{ \frac{freq_{ij}^{w}}{|T_{ij}|} \right\} \tag{2}$$

$$\eta_{ij} = \frac{\log_2 k - \text{Entropy}}{\sum_i^a \cdot \sum_j^{b_i} \text{Entropy}_{ij}} \tag{3}$$

Here,

b_i number of instances in domain of feature in ith position

k is the number of class labels

freq_{ij}^w is frequency of instances with class w in T_{ij} area

T_{ij} is number of instances in T_{ij}

a is amount of total features

Here, If corresponding value of related feature does not fall in training data, entropy is set to maximum of

$$\log_2 \text{freq}_{ij}^w \tag{4}$$

and if the covered cases fall to same target class , then the entropy is set to zero.

3.3.4 *Pruning of Rule*

Pruning [9] is a process in data mining, which removes terms which are not important as the local operation of function for heuristic gives rise to a lot of unnecessary term in the rule. Pruning tries to remove terms that are not important to the overall rule; in fact, the deletion of them could increase the accuracy of them. Pruning is important because it increases the accuracy of forecasting of the rule. Similarly, it also reduces the complexity of the rule. In Ant-Miner, the rule pruning is done iteratively, where the terms from the discovered rules are reduced. The quality given in 7 plays an important role while removing terms as they are removed based on it, which shows their likelihood of improving the accuracy of the rule. Pruning stops by assuring maximum quality in a rule or when only one term is remaining in the antecedent. We have studied various papers that have implemented ways for improving pruning procedure [1]
We have proposed a different pruning approach than the original one [9]. In our approach, pruning is only done if the number of terms in antecedent exceeds a certain threshold that is user-defined. In the original Ant-Miner, pruning was done in a dataset that does not have many attribute columns, so the rule was also not that complex and lengthy, and the computational cost was low. However, we are working in the dataset with a greater number of attributes, so using original rule pruner [9] is computationally expensive. So

we have proposed our approach which is shown in pseudocode 2

Algorithmus 2 : Proposed Rule Pruning Pseudocode

```
Prune=True;
c ( predefined value);
if Prune=True AND Term's Quantity in antecedent > c then
    REPEAT No of Terms in antecedent is randomly reduced;
     UNTIL(No of Terms in antecedent = c)
end
THEN run Ant-Miner original rule pruner
```

As we can see in the algorithm, in the rule pruning segment, we see whether the attributes in antecedent crosses a certain user-defined threshold, a loop is executed, and the loop runs for the number of iterations which is equal to the number of terms in the antecedent. Inside the loop, then the number of terms is randomly selected and reduced. The loop stops when the number of terms is equal to the threshold "c" as given in pseudocode 2. After coming out of the loop, the terms again go through pruning of original Ant-Miner where an individual antecedent is removed with every iteration.Then quality of rule is calculated. If the calculated quality becomes greater than previous quality, then the term is surely removed from the antecedent.This helps to prune the rule.

3.3.5 *Updation of pheromone*

At the beginning of each iteration, base pheromone is initialized which is same for all the terms , so all the terms have equal opportunity of getting selected in the beginning . The formula is given below:

$$\mathrm{Base\_pheromone} = \frac{1}{(\sum_{i=1}^{a} b_i)} \tag{5}$$

b_i represents the values of feature in ith position

a is the number of total features

In case of pheromone updation, pheromone value of those term's that occur in the rule is increased with respect to corresponding quality and similarly those term's pheromone is lessened that does not fall in the rule which is known as pheromone evaporation. Pheromone is updated with given formula:

$$\mathrm{Updated\_pheromone} = \tau_{ij} + \tau_{ij}.Q \ \forall i, j \tag{6}$$

where, Q is the quality and it is calculated with the following formula

$$Q = \frac{TP}{TP + FN} . \frac{TN}{TN + FP} \tag{7}$$

Here,

- TP stands for true positive where cases that are in rule's domain have same class than forecasted one
- FP stands for false-positive where cases that are in rule's domain have different class than forecasted one
- TN stands true negative were cases that are not in rule's domain have class different than forecasted one.
- FN stands for false-negative where cases that are not in rule's domain has the class similar to forecasted one.

The value of Q lies between 0 and 1, and greater Q means greater quality and greater chances being added to the rule for the corresponding term.
Similarly, the decrease of pheromone from the term which is not included in the rule is accounted by dividing the corresponding pheromone value by sum of total pheromones .
The final discovered rules comprise of rules that will be used to predict the test dataset. The creation of default rule also helps in classifying newer cases, if those cases were not covered in the training of train data.

4

EXPERIMENTAL EVALUATION

In this chapter, we have shown various experiments that we have done. Along with it, we have also shown and analyzed the results that we have obtained. This chapter is divided into six sections. The experiments that we have described have used partial amount of LVD. LVD is data that is collected from sensors of thousand of Volvo vehicles. First and second experiment are focused on seasonal prediction however third is focused on yearly prediction and fourth compares the processing with or without Feature Engineering 3.1.1. This data contains 336677 rows and 665 features. The data consists of three data types. There are 442 columns of float64 datatype, 23 columns of integer data type, and 200 columns of object types. There are 6675 unique vehicles; that is, data is extracted from 6675 unique vehicles. LVD contains data of three years starting from January 2016 to May 2019.

4.1 FIRST EXPERIMENT

Here, our objective is to predict the failure of components according to seasons. Here, we differentiate vehicles into good and faulty ones. We focused on the seasonal data and did experiments on it. Our first experiment is divided into two parts. In the first part, We have made the combinational dataset of summer and autumn, here (June, July, and August) are summer months and (September and October) are the autumn months corresponding to the year of 2016 from the LVD. We took it as train data. Then we made the combinational dataset of Summer and Autumn's months corresponding to the year 2017 and took it as test dataset. Then we modeled the training dataset of 2016 using proposed Ant-Miner and predicted on the test dataset of 2017 and obtained the results as shown in the figure 9 and figure 10
For the first part4.1 , we obtained result which is shown in figure 9. In the figure ,we can see the we have predicted on test set of summer-autumn months, 2017 and extracted the result where we have found the highest accuracy is observed in the month of October, similarly, the highest F1 score is observed in the month of October which is given in figure 10. The table 4 shows highest number of rules is in the month of September.

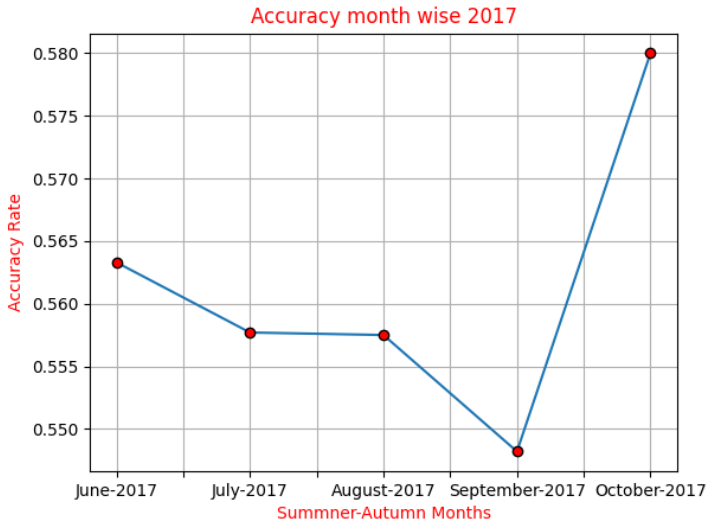

Figure 9: Accuracy summer-autumn month wise of 2017

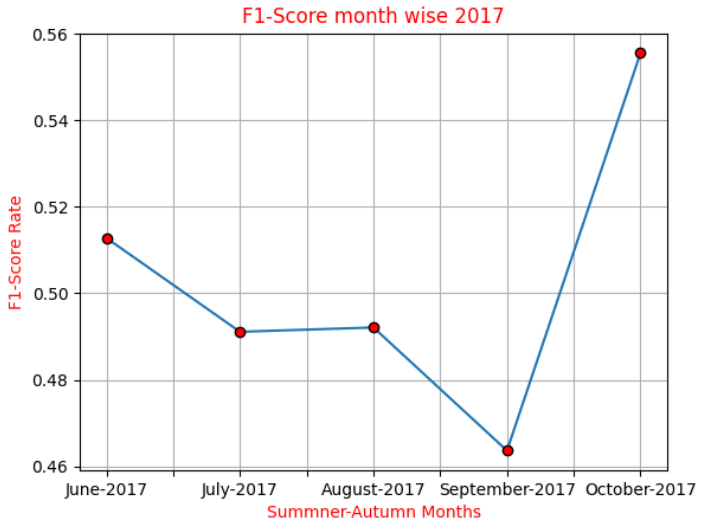

Figure 10: F1 score summer-autumn month wise of 2017

Total Rules	No. of Rules for Class 0	No. of Rules for Class 1	Accuracy %	F1-Score %
16	12	4	56.33	51.28
15	12	3	55.77	49.11
16	12	4	55.75	49.21
22	17	5	54.82	46.36
16	12	4	58.09	55.56

Table 4: Illustration of Rules, Accuracy and F1-score result for the combinational season 2017.

In the second part , we have taken combinational data of winter (November and December), and spring (March, April, and May) related

to 2016 as train dataset, and test dataset is taken as combinational data of winter(November and December) and spring (March, April, and May) related to 2017. Then we used the proposed Ant-Miner on train data of 2016, and based on it, we predict on a test dataset of 2017 and get the result as shown in figure 11 and figure 12.
Regarding second part 4.1, the result has been obtained as given in figure 11 where we have made a prediction on the test set of winter-spring months, 2017 and found that the greatest accuracy is yielded in the month of May, similarly, the highest f1 score is obtained in the month of which is shown in figure 12. The table 5 shows greatest number of rules is in December.

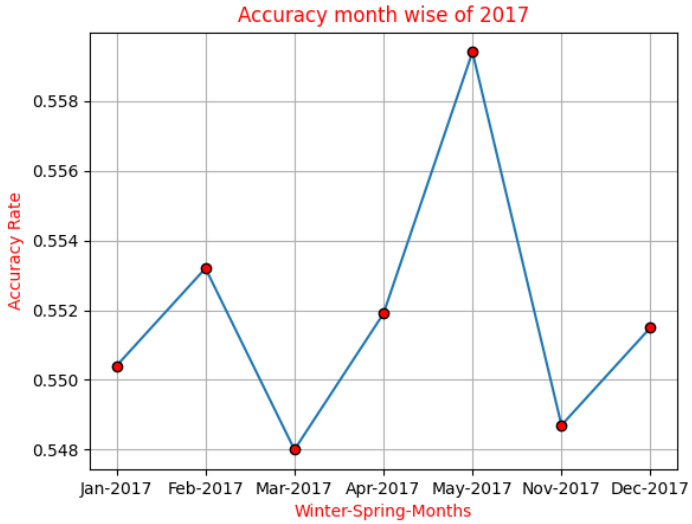

Figure 11: Accuracy winter-spring month wise of 2017

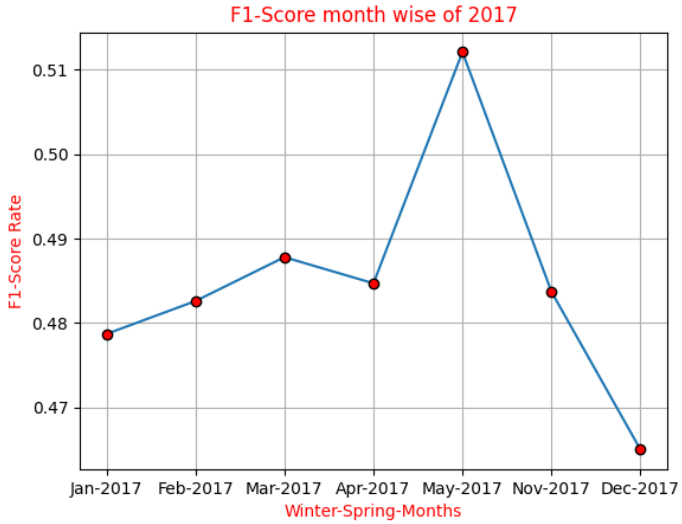

Figure 12: F1 score winter-spring month wise of 2017

Total Rules	No. of Rules for Class 0	No. of Rules for Class 1	Accuracy %	F1-Score %
14	11	3	55.04	47.87
17	14	3	55.32	48.26
13	10	3	54.80	48.78
14	11	3	55.19	48.47
18	15	3	55.94	51.21
16	11	5	54.87	48.37
19	15	4	55.15	46.50

Table 5: Illustration of Rules, Accuracy and F1-score result for the combinational season 2017.

4.2 SECOND EXPERIMENT

The second experiment also focuses on the seasonal prediction of vehicle component failures. It also has two parts. In the first part , we have taken a combination of summer and autumn data of 2016 into a dataset as a training dataset, and then we have taken test dataset as the combination of summer and autumn of 2018 as the test dataset from the LVD. Then we employ the proposed algorithm to train the model and predict it.The result from it is shown in figure 13 and figure 14 .

For the first part 4.2,the result has been obtained as given in figure 13 ,here we have predicted on on train dataset of 2018 and got the outcome where we got the greatest accuracy in the month of June, similarly, the highest f1 score is obtained in the month of September which is shown in figure 14. The months of August and September have the highest number of rules as shown in the table 6

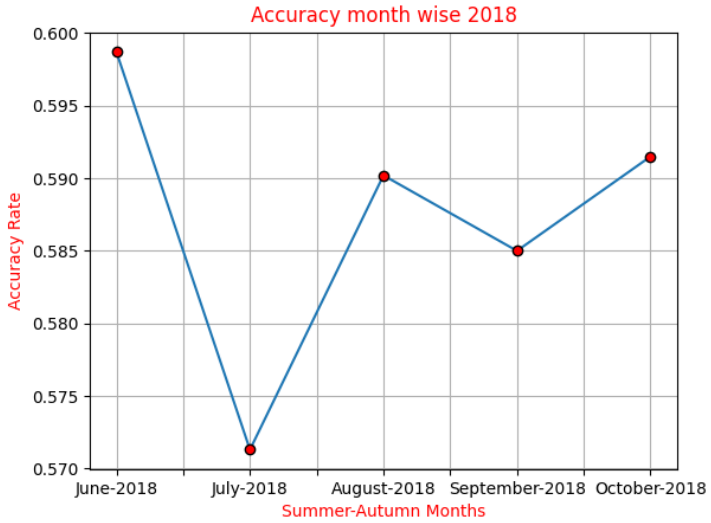

Figure 13: Accuracy summer-autumn month wise of 2018

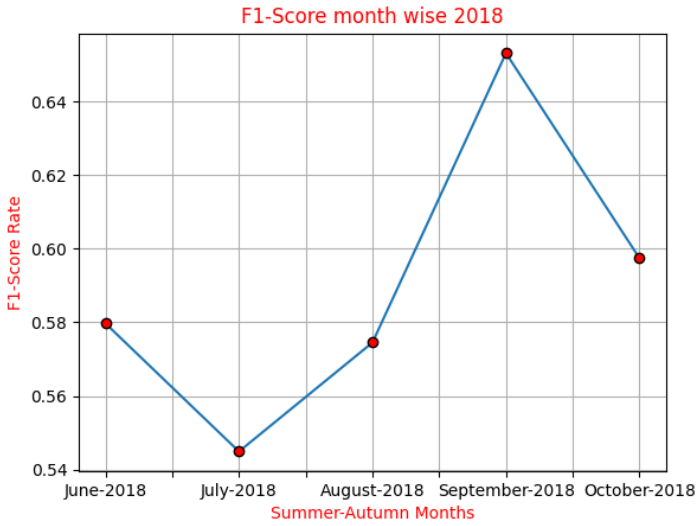

Figure 14: F1 score summer-autumn month wise of 2018

Total Rules	No. of Rules for Class 0	No. of Rules for Class 1	Accuracy %	F1-Score %
12	8	4	59.87	57.97
16	12	4	57.13	54.49
18	13	5	59.02	57.45
18	13	5	58.50	65.31
16	13	3	59.15	59.74

Table 6: Illustration of Rules, Accuracy and F1-score result for the combinational season 2018.

In the second part , we have taken the same train dataset and taken the spring and winter data as test data set and implemented the proposed algorithm on it and gathered result which is given in the figure 15 and 16

For the second part 4.2, we have observed that the greatest accuracy occurs in the month of November as in the figure 15, in addition to that highest f1 score is also obtained in the month of November as given in the figure 16. Greatest number of rules is seen in the month of November as shown in table 7

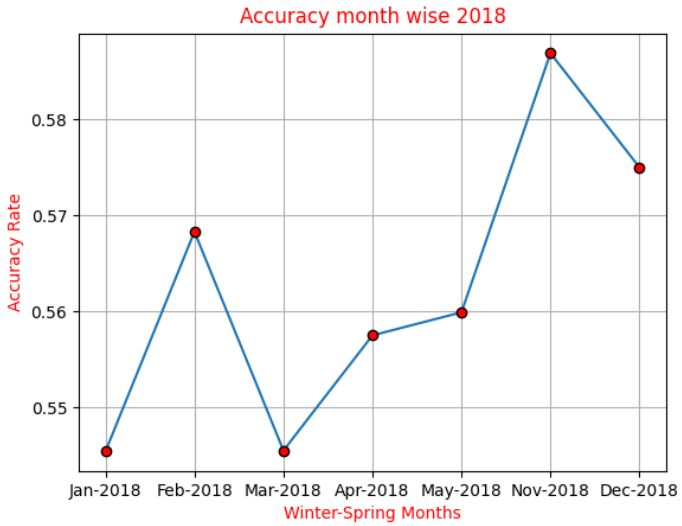

Figure 15: Accuracy winter-spring month wise of 2018

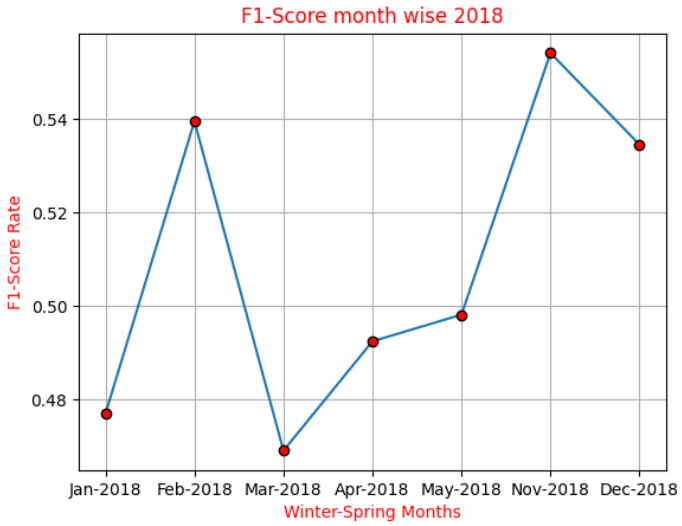

Figure 16: F1 score winter-spring month wise of 2018

Total Rules	No. of Rules for Class 0	No. of Rules for Class 1	Accuracy %	F1-Score %
15	11	4	54.54	47.70
15	12	3	56.83	53.96
15	12	3	54.55	46.91
15	12	3	55.75	49.25
16	12	4	55.99	49.82
18	14	4	58.69	55.42
12	10	2	57.50	53.46

Table 7: Illustration of Rules, Accuracy and F1-score result for the combinational season 2018.

4.3 THIRD EXPERIMENT

The third experiment deals with yearly forecasting of vehicle failures. In this, we have taken the combinational yearly data corresponding to 2016 and 2017 as a training dataset. We have trained the model using the proposed algorithm and made a prediction where the yearly data of 2018 is taken as test dataset, after this we have obtained the result is which is shown in figure 14 In the third experiment4.3, we have obtained the result by making prediction on test dataset which is yearly data of 2018 . The result is shown in figure 17 and 18 where the highest accuracy and f1 score is seen in the month of September. The highest number of rules is given for the month of July as shown in table 8

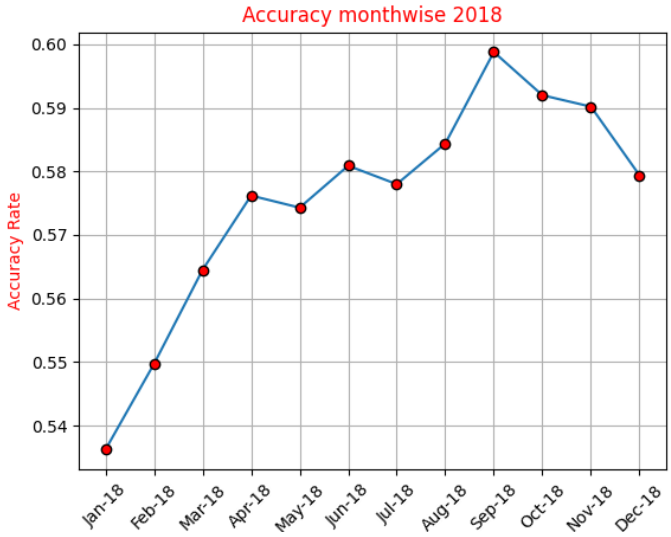

Figure 17: Accuracy in 2018

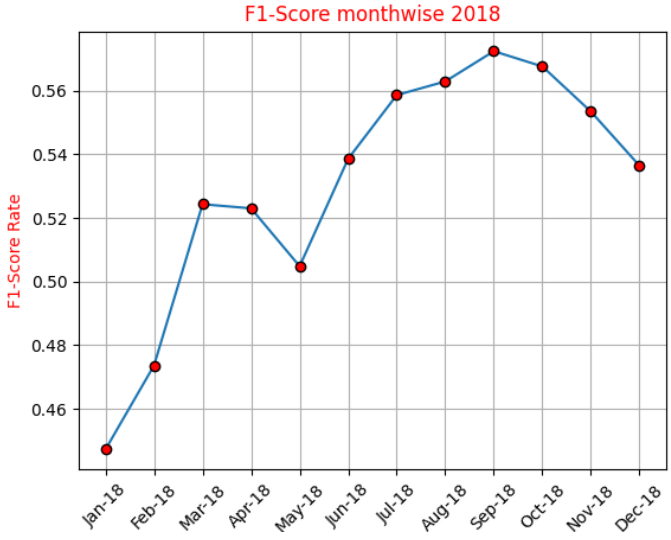

Figure 18: F1 score in 2018

Total Rules	No. of Rules for Class 0	No. of Rules for Class 1	Accuracy %	F1-Score %
15	12	3	53.60	44.72
13	10	3	54.97	47.36
17	14	3	56.45	52.43
19	15	4	57.62	52.30
15	12	3	57.43	50.48
21	17	4	58.09	53.86
22	18	4	57.80	55.86
16	11	5	58.84	56.29
12	9	3	59.88	57.24
20	17	3	59.20	56.76
14	11	3	59.02	55.34
19	16	3	57.94	53.64

Table 8: Illustration of Rules, Accuracy and F1-score result for the year 2018.

4.4 FOURTH EXPERIMENT

The objective of this experiment is to make yearly prediction of 2018 without Feature Engineering. We are doing this basically compare it with Feature Engineering result. We are taking less samples for this experiment as compared to others.
In the fourth experiment 4.4, we have obtained the result where highest accuracy and f1 score is obtained for the month of September2018 as shown in figure 19 and figure 20. However, it has greater computational time in relation to other experiments. Similarly, less number of rules have been obtained in particular months. Its result is given table 9

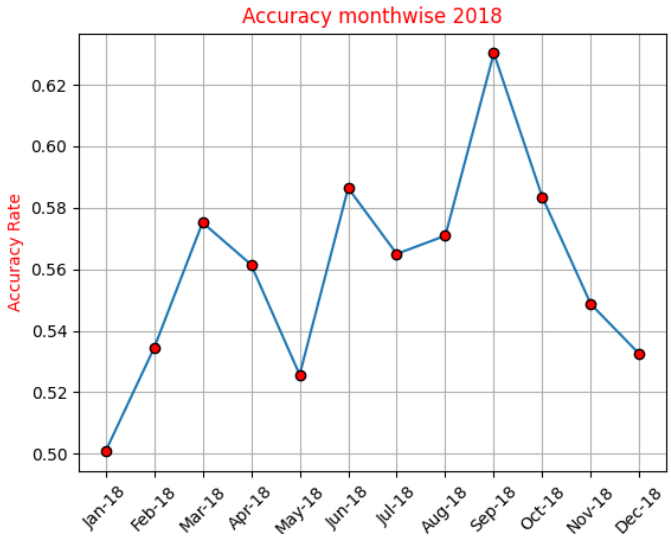

Figure 19: Without Feature Engineering Accuracy in 2018

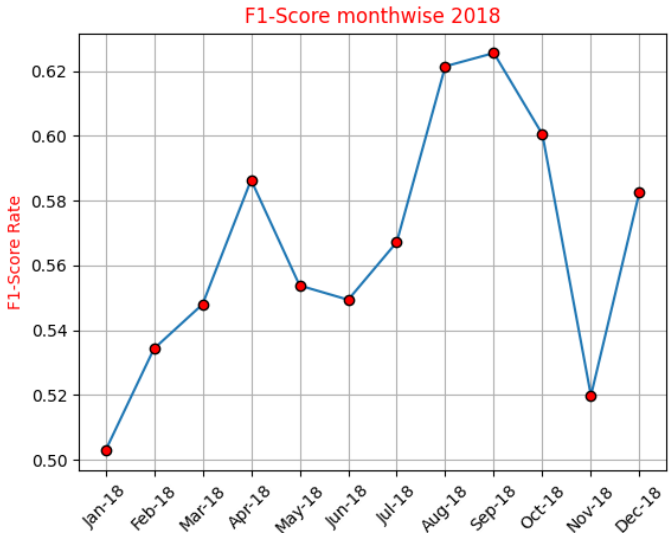

Figure 20: Without Feature Engineering F1 score in 2018

Total Rules	No. of Rules for Class 0	No. of Rules for Class 1	Accuracy %	F1-Score %
4	3	1	50.07	50.28
10	6	4	53.43	53.43
12	8	4	57.52	54.80
9	5	4	56.15	58.64
16	12	4	52.57	55.38
20	15	5	58.64	54.94
9	5	4	56.50	56.72
10	5	5	57.09	62.15
10	8	2	63.03	62.56
8	3	5	58.34	60.05
12	8	4	54.86	51.98
8	4	4	53.24	58.27

Table 9: Illustration of Rules, Accuracy and F1-score result for the year 2018.

In all the experiments, we have taken f1 score as the most important measure in order to analyze the result of the experiments. The reason for choosing f1 score is that our dataset is unbalanced in nature which has complexities. In this type of scenario, we can trust f1 score as it is the weighted average of precision and recall. Therefore, in first seasonal experiment, the greatest likelihood of breakdowns occurs in the month of October(first part)4.1 and May(second part)4.1 of 2017. In second seasonal experiment, the greatest chance of breakage occurs in September(first part)4.2 and November(second part)4.2of 2018. In third experiment highest likelihood occurs in September of 2018 4.3. There is a sudden hike in f1 score and accuracy for the month of September. We think that this hike is due to the imbalance of data. Similar result is obtained in fourth experiment. Furthermore, we have taken the result of fourth experiment as baseline and compared to that our solution gives almost similar performance. However, result could improve with better balancing method in future.

4.5 PARAMETERS

In the Ant-Miner, we have used four parameters as shown in the table

Parameter	Value
Number of ants used	1000
Minimum No. of cases per rule	10
Maximum uncovered cases	10
No. of rules for convergence	40

Table 10: Parameter used in our experiments.

These parameters seemed suitable for experimenting on our dataset as we have noticed this after hundred iterations of processing .

In the next chapter, we have discussed the findings of our research. Implications of this research to various stakeholders are also explained in the next chapter.

5

DISCUSSION

In this chapter, we are discussing the implications of the result of our experiments and how it affects various stakeholders. After analyzing the results from seasonal experiments of first(4.1,4.1) and second(4.2,4.2), we have learned that prediction of failure in components is different in the test set of 2017 and 2018 which means that prediction of breakdowns is not dependent on season in the period between 2017 and 2018. Similarly, after evaluation of results from the third experiment(4.3, we understood that the failure prediction in the yearly test set of 2018 is similar to the seasonal prediction of 2018, where the greatest likelihood of fault occurs in the month of September. Fourth experiment also gives similar output but with great computational cost and number of rules. This could be inferred that there is high probability of breakage in September, as this result has been validated by multiple methods of predictions. Through this, we are also predicting when a failure is going to happen.
The aim of this thesis is to implement a method that would help us to extract the most meaningful information from the complex Logged Vehicle Data (LVD) data, which would help us for the detection of breakdowns for the time managing the maintenance of components in vehicles.The information that we have extracted through our method helps manufacturers to get early information on breakdowns in-vehicle components like the time of failure occurrence. This thesis is trying to solve the problems which are affecting both manufacturers and customers. The manufacturers want to reduce the faults, which will give better quality products to customers. This has two big benefits to the manufacturer. Firstly, it will give early information to the manufacturer about failure before the actual failure. This information will help them prepare for that failure, by implementing countermeasures for them such as: they can focus on those components to overcome their faultiness. They can also order those components early, which also reduces costs in maintenance as they don't need to place an order in a rush. In addition to that, it will also reduce the likelihood of the faulty product being handed to customers and affecting their satisfaction with the manufacturer's brand. Along with it, customers will also benefit from it as early detection and prevention of faults will also lessen the hassle for customers in the form of repairs or placing claims for warranty. Customers can also create genuine loyalty towards the brand as it bestows greater value over time This thesis could be useful for predictive maintenance not only in terms vehicle industry but also to smart cities as just like predicting component failure in-vehicle

components it could also predict the breakdowns in the sensors and equipment in a smart city.
This thesis also has some limitations. Firstly, Logged Vehicle Data (LVD) data consists of multiple variable types, such as integers, floats, and objects. However, Ant-Miner that we have used for processing Logged Vehicle Data (LVD) has problems working with continuous or multiple variable type data. Hence to overcome this, we have discretized3.1.1 the data to convert it into categorical and then process the data. We know that some information is lost during this process. Hence, a hopeful extension of this work could be implementing a better discretization process to overcome information loss or a method that would address multiple variable datatypes for processing. Better data balancing techniques and improvement in feature extraction could also be implemented for getting more meaningful information, which could also be an interesting extension.

6 CONCLUSION

Overall, the problem that we are trying to solve is the detection of faults before they can occur in components of vehicles. We are trying to research on fault detection that will assist in predictive maintenance. We have implemented a method that include processes like data pre-processing , feature extraction , feature selection and Ant-Miner algorithm in order to process the data.We have used a modified Ant-Miner algorithm in order to process Logged Vehicle Data (LVD) and extracted the result. Logged Vehicle Data (LVD) had a lot of issues such as noise, irrelevant values, data imbalance, and so on, but we tried to overcome all of them for extracting meaningful information. However, there are areas where improvements could be made, especially in feature engineering and balancing of data.The result that we have got by processing Logged Vehicle Data (LVD) with our method is that prediction of failure of components is not affected by the seasons in the year 2017 to 2018. Similarly, we validated the seasonal prediction of breakdowns by yearly prediction in the year 2018 with and without Feature Engineering and achieved similar outcome. With its help, We also successfully predicted the month when failure would occur. The scope of this thesis is not only limited to the vehicle industry, but it could also be relevant for doing predictive maintenance for components in smart cities.

BIBLIOGRAPHY

[1] Hayder Naser Khraibet Al-Behadili, Ku Ruhana Ku-Mahamud, and Rafid Sagban. Rule pruning techniques in the ant-miner classification algorithm and its variants: A review. In *2018 IEEE Symposium on Computer Applications & Industrial Electronics (ISCAIE)*, pages 78–84. IEEE, 2018.

[2] Omar Bougacha, Christophe Varnier, Noureddine Zerhouni, and Sonia Hajri-Gabouj. Integrated production and predictive maintenance planning based on prognostic information. In *2019 International Conference on Advanced Systems and Emergent Technologies (IC_ASET)*, pages 363–368. IEEE, 2019.

[3] Marco Dorigo, Alberto Colorni, and Vittorio Maniezzo. Distributed optimization by ant colonies, 1991.

[4] Reza Khoshkangini, Sepideh Pashami, and Slawomir Nowaczyk. Warranty claim rate prediction using logged vehicle data. In *EPIA Conference on Artificial Intelligence*, pages 663–674. Springer, 2019.

[5] Bo Liu, Hussein A Abbas, and Bob McKay. Classification rule discovery with ant colony optimization. In *IEEE/WIC International Conference on Intelligent Agent Technology, 2003. IAT 2003.*, pages 83–88. IEEE, 2003.

[6] Zhiwen Liu, Wei Guo, Zhangchun Tang, and Yongqiang Chen. Multi-sensor data fusion using a relevance vector machine based on an ant colony for gearbox fault detection. *Sensors*, 15(9):21857–21875, 2015.

[7] Wes McKinney. *Python for data analysis: Data wrangling with Pandas, NumPy, and IPython*. " O'Reilly Media, Inc.", 2012.

[8] N Mohanapriya and C Priya. A fault tolerant router with optimized an colony algorithm. In *2016 International Conference on Advanced Communication Control and Computing Technologies (ICACCCT)*, pages 167–170. IEEE, 2016.

[9] Rafael S Parpinelli, Heitor S Lopes, and Alex Alves Freitas. Data mining with an ant colony optimization algorithm. *IEEE transactions on evolutionary computation*, 6(4):321–332, 2002.

[10] Bharti Suri and Shweta Singhal. Analyzing test case selection & prioritization using aco. *ACM SIGSOFT Software Engineering Notes*, 36(6):1–5, 2011.

[11] Le Vu Tran, Bao Huy Huynh, Humza Akhtar, et al. Ant colony optimization algorithm for maintenance, repair and overhaul scheduling optimization in the context of industrie 4.0. *Applied Sciences*, 9(22):4815, 2019.

Ankit Gupta has completed Bachelors of Technology in Information Technology

,

Durlabh Shahi has completed Bachelors in Computer Information System

PO Box 823, SE-301 18 Halmstad
Phone: +35 46 16 71 00
E-mail: registrator@hh.se
www.hh.se