Flexible Methodology for diagnosing DTC codes

James Mitchell and Ryan Fortney

A report

submitted in partial fulfillment of the

requirements of the degree of

Master of Business Analytics

Iowa State University

2017

Project Committee:

Sree Nilakanta, Chair

Heike Hofmann, Committee Member

**Abstract**

Quality is one of the core values for a company like John Deere and is vital to its future. Quality can be viewed as a differentiator to create a competitive advantage for a company. Within the 21st century, farmers are constantly desiring increased quality from their equipment and thus the need for proactive monitoring to decrease machine downtime.

The goal of this project is to generate a generic tool or algorithm that can search through data that is collected and stored in a Machine Knowledge Center (MKC) in order to discover a faster or easier method of diagnosis of the source of failures thru the determination of key parameters that separate a good/bad population.

The net result would be less time being spent by dealers and engineers in trying to diagnose problems and provide a more informed information set for better analysis. Other net benefits would be lower warranty costs and more prescriptive repairs which would reduce customer downtime out in the field and increased customer satisfaction.

This project looked at two different approaches in marking machine data as good or bad. Then a decision tree was tuned and trained before testing on a larger set of information. The results were inconclusive as to what telematic information could be causing the specific failure code being looked at. However, the processes that were used in this project could be used in a real world setting to reduce the human overhead cost while investigating the root cause by narrowing down in a quick fashion what the potential area of focus should be.

**Introduction**

For over 180 years, John Deere been manufacturing products that help farmers both in and out of the field. Within the 21st century the need for proactive monitoring to decrease machine downtime has come to the forefront as America’s farms continue to shrink and the growth of corporate farming increases. This need has grown with the introduction of our interim Tier 4 (iT4) and Final Tier 4 (FT4) engines.

The purpose of this project is to take the data that is being collected and stored in the Deere Machine Knowledge Center (MKC), use techniques learned within the Business Analytic’s coursework and apply them in order to attempt to discover a faster or easier method of diagnosis of the source of failures. The net result would be less time being spent for dealers, the Dealer Technical Assistance Center and engineers in trying to diagnose the problem through trial and error, and provide a more informed information set for better analysis. Other net benefits for John Deere would be lower warranty costs and more prescriptive repairs which would reduce customer downtime out in the field and increased customer satisfaction.

Working with the engineering group it was identified a specific product family that has a large number of the 1347.7 diagnostic codes per hour. The self propelled sprayer line is one of two lines that are exhibiting a trouble code rate higher than 2% which is above the Deere threshold quality metric threshold. The 1347.7 DTC indicates a generic failure or fault within the machine.

**Data**

The data collected in the project consists of two different sets. The first set we hold off for our training or test set. This will allow for quick analysis and proper tuning of algorithms before applying to the full set of information. The data contained is a random sample over a 3 month period which contains 3 major sections of information. The first section being the machine id, this has been anonymized by the engineering team so that the customer's identity is protected and unknown throughout the project. The second major section of information is the telematic information. This includes measurements over a given 30 minute period for the machine such as barometric pressure, inlet air temperature, engine rpm speed, and fuel consumption. In all this section contains 120 different measurement variables. There is a second dataset by machine that has an entry for each time a DTC is detected, irregardless if it is a 1347.7 or not and finally a third dataset has what could be called meta information about the vehicle. This includes information such as model, manufacture date, engine model, fuel injector part numbers, and others. For a complete listing of variables please see Appendix B.

To begin with, in our analysis we looked at the data in order to get an understanding for what we were looking at. The first transformation made was to the DTC code where it is going to be treated as a class variable. Any DTC code of 1347.7 was converted to bad and all other converted to neutral. The next transformation made was around the coolant temperatures. What was done is the temperature of the coolant measurements were converted into a time spent in the range to a percentage of overall time spent. This allows for a better idea on how much time the coolant was spent within a specific measurement range. The final transformation made to the data to consolidate all columns between the four large samples of data so that it was contained in a singular sheet and the associated vehicle meta information attached to it. It was opened in R and exported to CSV for easier portability into WEKA.

There were two approaches taken in order to analyze the data. Each approach follows the same steps but how the errors were marked was different. With approach 1, each DTC of 1347.7, all lines for a specific machine id were marked as bad the rest were considered as neutral. Marking them as good is not a correct assumption as the base of a DTC is that it is an error, however for this analysis we wanted to focus on just one specific code. Approach 2 was more granular and exact to try and narrow down the results. It’s DTC errors were set to the specific 30 minute period they occurred in and if another reading came across at the same time a duplicate entry was made, one good and one bad. In the end approach 1 resulted in a dataset with 1135 or 2% errors marked and approach 2 resulted in 59 or .09% errors out of a possible 59226 of the dataset.

When talking with the SME of data, these approaches are in line with how they have attempted to analyze the data, how do you mark the errors. Do you mark errors based on the machine or do you mark it based on when it happens and all other times is considered no errors. Throughout the project, the SME group will be our main source of information and data pulls.

**Approach 1**

**Initial Analysis**

To start the analysis to determine how good the algorithms work one needs to start with a baseline measurement. This measurement was taken by using the decision tree or J48 classifier on the full set of data before attribute selection or performance tuning was performed, Figure 1.

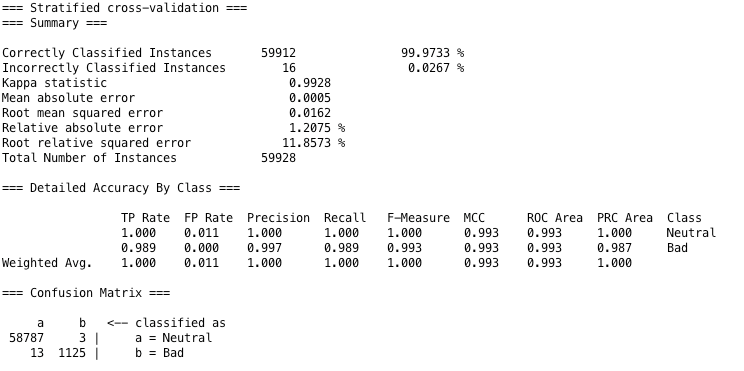


Figure 1: Analysis approach 1 initial baseline

As can be seen in figure 1 the error rate is less than .03%. This is probably due to the large discrepancy in minority vs majority class in the data. When presented with this issue, there are some steps that can be taken to help overcome some the class imbalance. The first option is to implement a class balancing technique called SMOTE, where synthetic data is injected based on its nearest neighbors to help balance the class at the cost of precision. Since precision is extremely high, this would be a good candidate for rebalancing through SMOTE first. A second is to resample the data, which would create copies of the data. Each approach is acceptable, however with less than 2% of the minority class represented the method utilized was SMOTE. It should be noted that this should only be performed on the training set not on the test set or the full data set itself. If done in this manner the results can be severely modified and could undermine the answer that is needing to be answered.

**Attribute Selection**

One very important step within the process especially with a large amount of attributes is to determine if it makes sense to reduce the number of attributes. This can be done through a technique called feature selection. The process involved has you going through all the attributes and determining which ones actually add the most value or information gain. The route chosen for this project was a three step analysis where three different evaluators were run and then a consolidated list of attributes among all three were used to give us the final set features.

The two attribute evaluators used were CorrelationAttributeEval and InfoGainEval. When run an aggregation of the two was made where the ranking threshold was greater than .01 This produced a feature set of 48 attributes including the class variable. They can be found in figure 2 below:

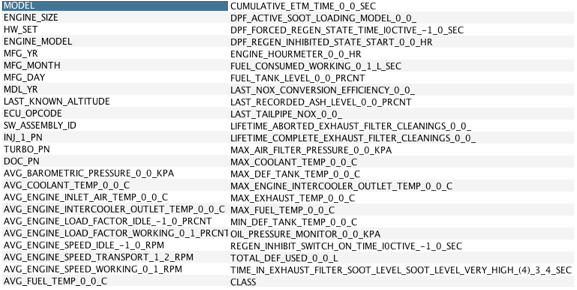


Figure 3: Attributes with best information gain for first approach.

The final step in the process is to keep only the attributes found above and the class value. To remove these, WEKA has a built in filter called Remove that can be used by simply inputting the attributes you want to keep and selecting the inverse selection = TRUE option.. The attributes to be kept will generate our main set of data that we can do the training and testing sets as described above.

With all the attributes selected we separated a randomly selected set of data from all of the data provided. Using WEKA this task can be performed using the Filter -> Unsupervised -> Instance -> RemovePercentage function. Because the dataset was large, we chose a 25/75 split, where 25% was our training set and 75% would be our test set.

**Parameter Tuning**

After increasing the minority class through SMOTE the classifier needs to be optimized. This is done in order to find the best step size and confidence variable. Within WEKA this task can be quickly done by adding a search pattern within it that gets passed to the classifier until it finds the optimal pattern to use. This can be used when doing the final training exercise with the proper confidence level. CVParameter selection is the classifier to use with the the J48 decision tree as the sub classifier that we want to pass variables too. As can be seen below (figure 3), we chose to search from a .01 to .5 confidence to find the best confidence interval for our data



Figure 3: WEKA J48 parameter tuning via CVParameter Selection

Once completed this resulted in an optimal confidence of .1325 for the J48 classifier. With that known, training the set of data with these parameters is required so that the decision tree can learn the dataset before we run it on the test set that we reserved.

**Training**

The final step in the entire process before running against the main dataset is to train the dataset. This process is the culmination of all the steps above which will give the precision, accuracy, recall of the dataset. These measures will tell how well the entire process worked and also predict how well the fit is for final model and how it will perform against the test set of data withheld. The results of the trained model appears to have improved its classification which is to be expected as the machine learns the rules, figure 4.

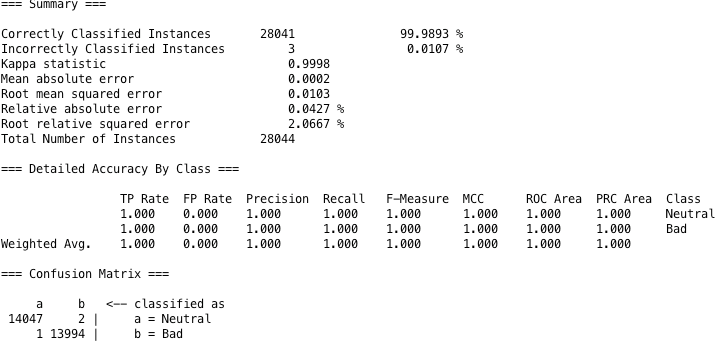


Figure 4: WEKA output of trained model.

With the model trained the next step is to see just how well the model model will work against the test set that we withheld previously. When running this and choosing test set from within WEKA using the same classifier the results can be seen in figure 5.

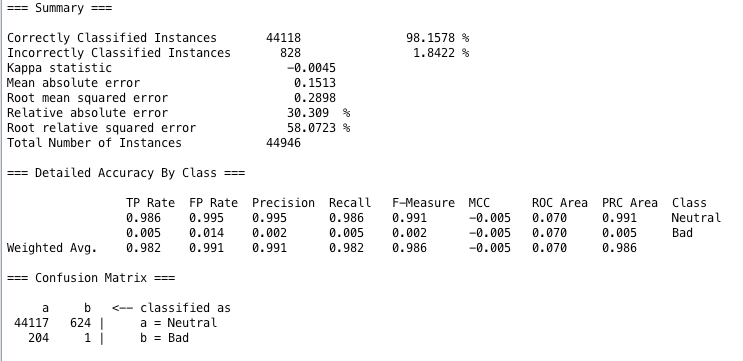


Figure 5: Trained model against the test set.

One thing that stands out clearly for this approach is that the model completely failed at classification. Out of 1135 bads that we started with it correctly classified 1, a final decision tree can be found in Appendix C. The question raised is what could possibly have caused the issue? One, it's quite possible that the amount of bad classifications is extremely low which caused the classifier to have issues. Another possible answer is the variables used from the telematics are providing false information. With that said, the next step was to see what would happen by working with the second analysis to see if it will yield any different results or end up with the same issues.

**Approach 2**

**Initial Analysis**

For analysis 2 a cursory review of the data was taken before doing any major analysis. This was done to see if maybe something stuck out. In doing that, when looking at the data after loading it into WEKA, there are some interesting pieces of information that stick out right away.

1. The Sprayer model R4030 appears to carry the majority of the error records figure 6:

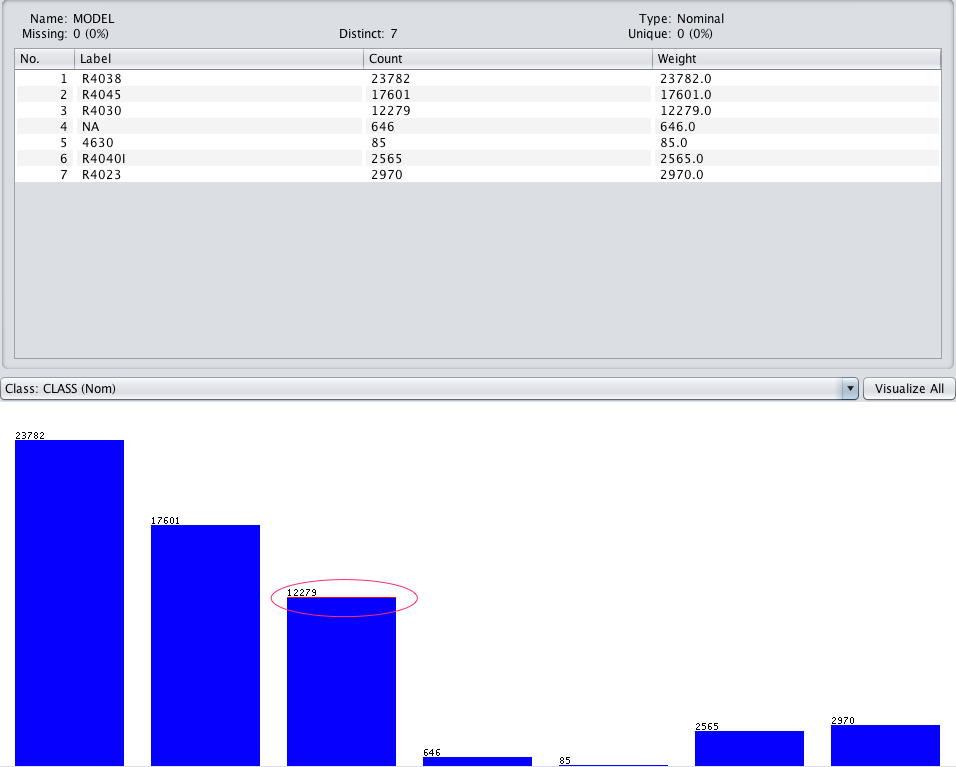


Figure 6: Vehicle model to error chart.

2) Engine model 6068HN059 seem to exhibit most of the errors, figure 7.

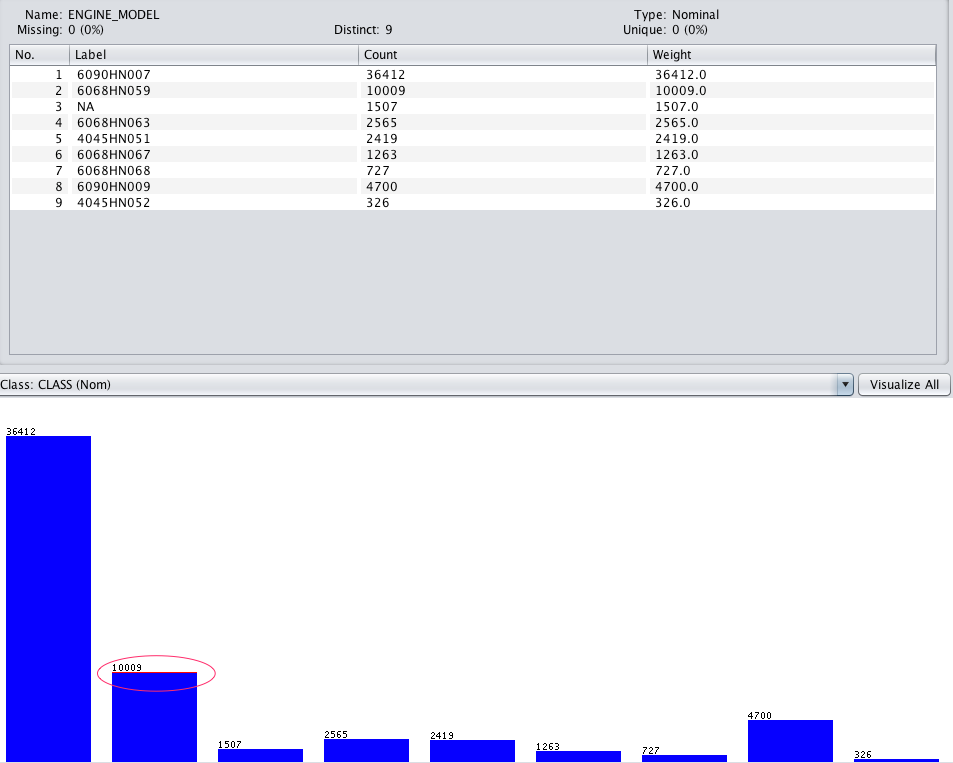


Figure 7: Engine model to bad indicator chart.

3) A specific year (2014) contains all the errors, figure 8.

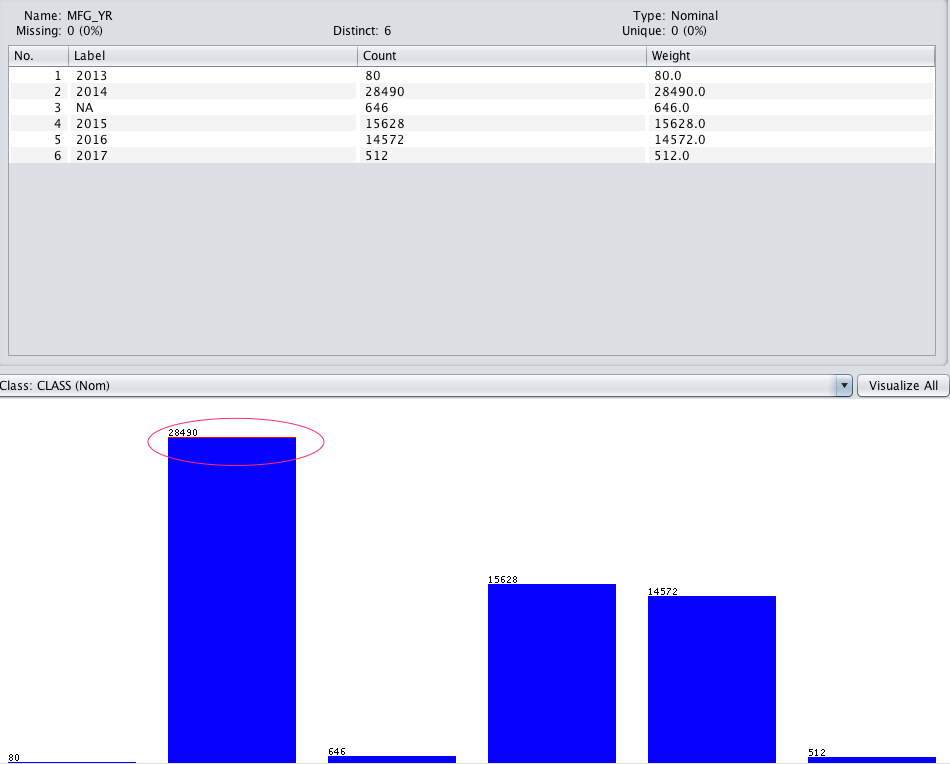


Figure 8: Engine manufacture year to bad record chart.

4) The months of March and June also have all the errors, figure 9.

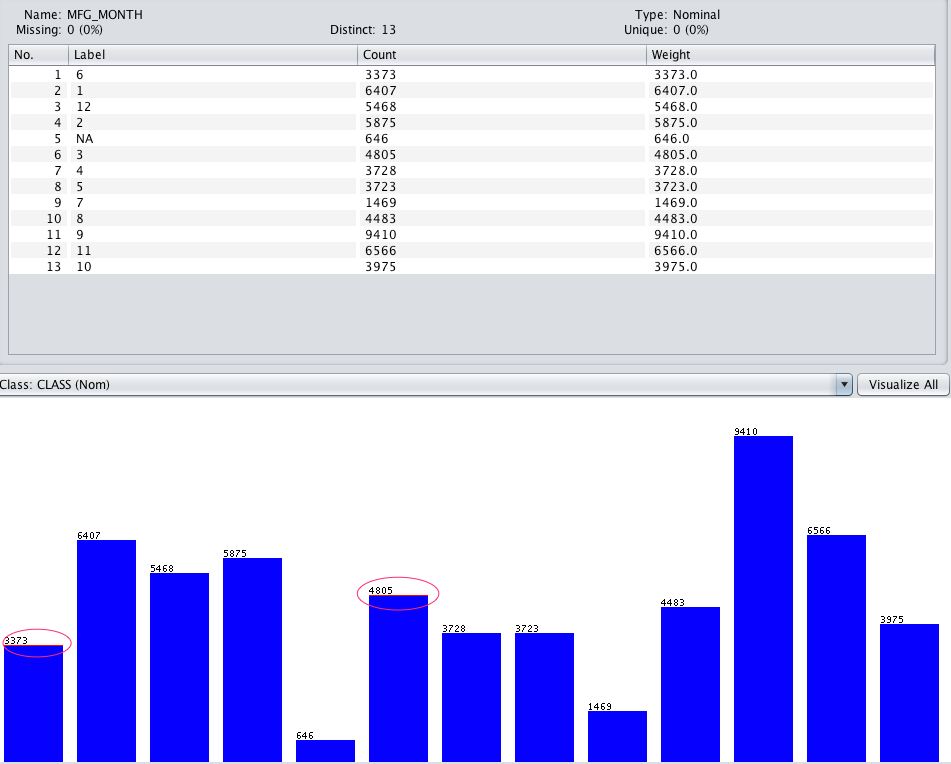


Figure 9: Manufacture month to bad record chart.

With that said it could be viewed that this is enough information to assume that these failures are paired down to a specific engine model manufactured in 2014 and consider the problem resolved. However because warranty can cost quite a large amount of money from part replacement, customer downtime, DTAC time, and other personnel time, knowing the root cause of the problems is needed. This raises a few questions, what is causing these failures? Is it attributed to environmental conditions or how the engine is being used? With the errors being marked to a specific time period rather than all entries as performed in approach 1. The remainder of the analysis looks into these questions to try and figure out if there is any other correlation to the failures.

**Attribute Selection**

As with approach 1 attribute or feature selection is very important, with 186 columns of data there are many that more than likely do not contribute. The methods used across the entire set of data were used to look at information gain and attribute correlation as the main methods. Then selection was made based on aggregation of those elements that were above a threshold of .001 infogain. This number was changed for this analysis because the normal threshold of .01 would have provided zero results across the two evaluators. In the end this resulted in the following subset of 31 parameters, figure 10:

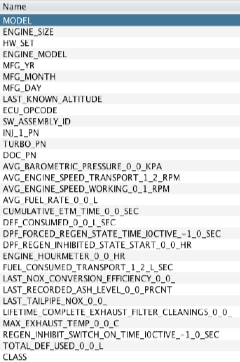


Figure 10: Attributes left after attribute selection

**Baseline results**

Before the attributes were pruned a baseline was taken to determine if attribute selection helped or hurt the overall performance of the decision tree. The results show that overall it correctly classified 99.87% of the instances as shown in figure 11.

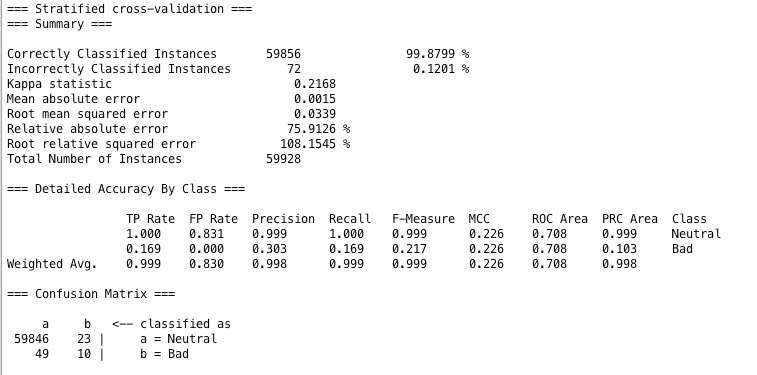


Figure 11: Initial model results pre-parameter tuning.

While this looks sufficient the issues are within the details itself. Looking closer and recalling there were 59 known errors in the entire dataset, only 10 values or 20% of the total errors were classified correctly, giving a 80% miss rate.

As with analysis 1 a 25/75 split was created for a training and testing set. Parameter tuning was the same and resulted in the same confidence of .1325 to be used for the J48 decision tree.

Once the training set was tuned and run through WEKA, the key measurements and confusion matrix look well defined, figure 12.

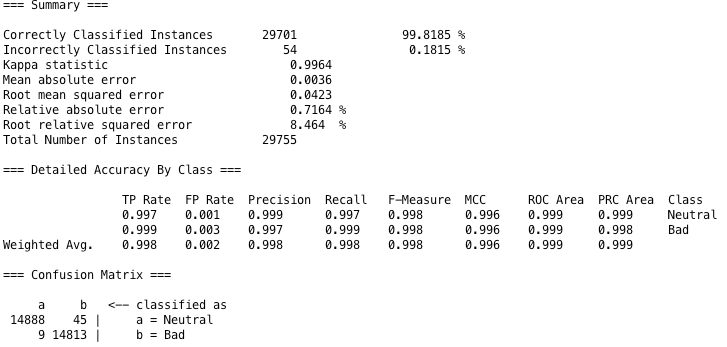


Figure 12: Training set initial results.

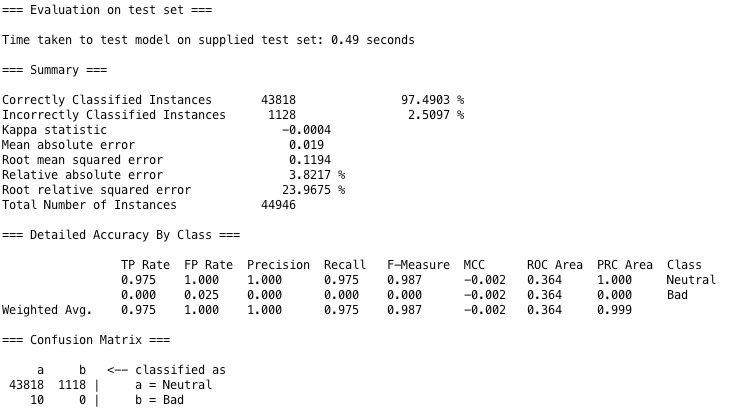
With the trained classifier looking to have minimal errors the expectation is that the test set will result in better results and a final decision tree. However, as can be seen in figure 13, there is a misclassification of about 2.5% and a 100% failure in classifying the bad class condition. 

Figure 13: Test set J48 classification

In the end for this dataset the training and testing failed to produce any results. The biggest reason for this is that the sample of bad data is too small and random for good analysis.

**Final Conclusion**

In conclusion after running the data through two different approaches with how errors are indicated it clearly shows that the results are inconclusive from what telematic information could be causing the issue. This is no surprise as the team of engineers that have looked into this are coming up with the same problem. With that said though, there is some information that was gained that may be unknown such as that the issues are constrained to a certain sprayer model, engine year and month, and even engine model. From a manufacturing standpoint this cannot be viewed as a loss as it is a very good starting point to begin tracking down manufacturing and supplier base information. The processes that were used can be deployed when trying to narrow down root cause analysis as currently there is a lot of time and resources trying to find this information quickly. So the side benefit of the practices are potentially a quicker investigative response time.

Some items that also contributed to the lack of results could be inconsistent data collection. For example each set of information that was pulled was randomly sampled, the telematic information being captured was different for all 4 sets. In working with the SME this is something that is problematic because as new software is released different data points are being collected.

Potentially in working with this set in the future, rather than combine all sheets together so that the dataset contains every column of data, an aggregate should be done. This could keep the information from being skewed by data that was not collected for other machines. Not only this but looking at the data itself there are many data points that were not collected at all within a specific time period.

To further this project more there are a few items that could be done differently that may produce different results.

1. Standardize the content of the data being collected. In example, make sure all data used contains the same attributes.
2. Remove those rows of information that do not contain complete information. This could reduce the skewing of information and give a better list of attributes to go off of.
3. Focus on the DTC time slices themselves. By doing this one could possibly see trends in that if condition A occurs condition B will always occur.
4. Do not collect random samples of information. With the data being collected in random intervals there is a possibility that key elements were missed due to the randomness. Collecting all data over a specific time period may expose different information or results.

The above list can be used as a checklist for continuing to improve upon this project and also improve upon processes currently being used within Deere. The impact of the data quality is very noticeable and trying to improve upon this can help with future investigations.

**Appendix A - Definitions**

ECU – An ECU or engine control unit is a self contained microcomputer that controls functions on an engine such as fuel/air mixture ratio’s and other functions needed for an engine to run efficiently. It is also responsible for monitoring and communicating any issues.

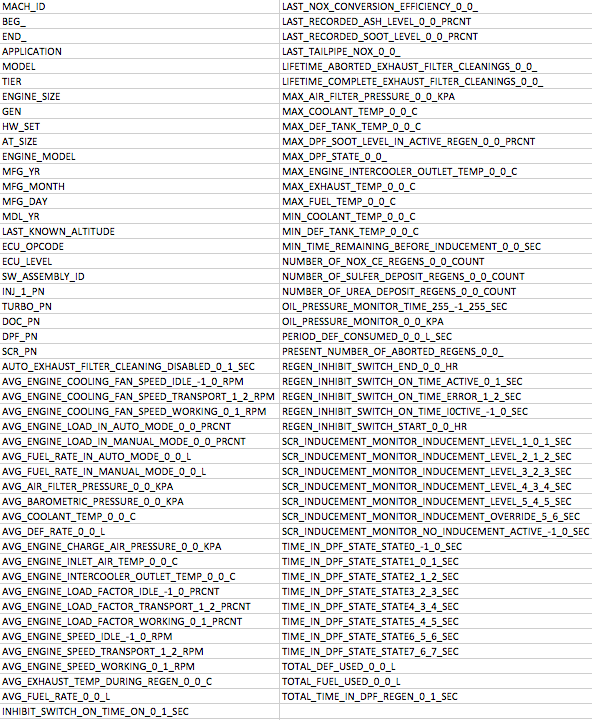
ServiceAdvisor™ – a system implemented by John Deere in order to provide technicians a method which they can diagnose problems with vehicles quicker.

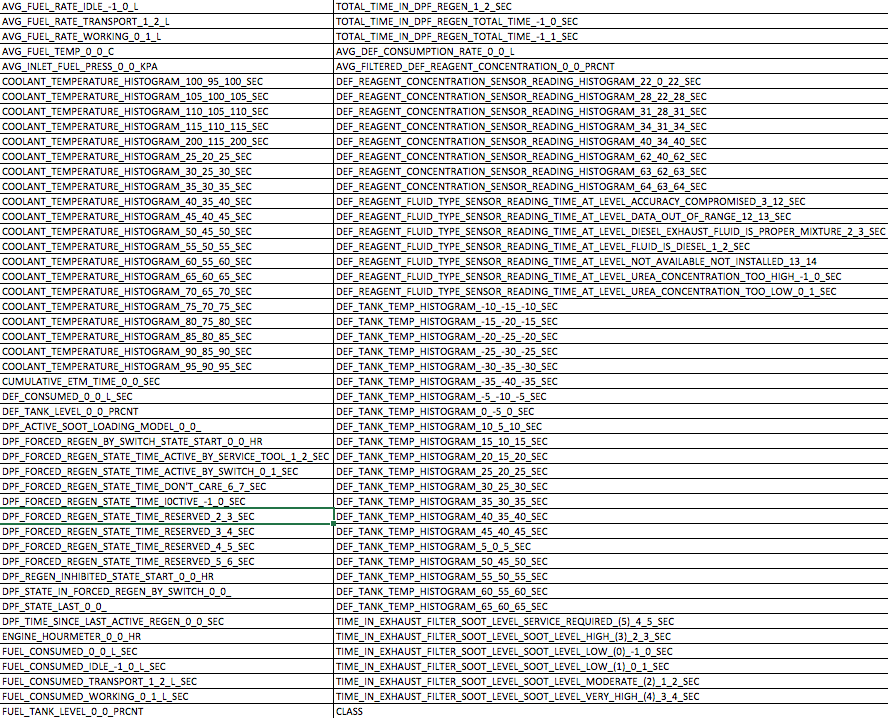
DTAC – Dealer Technical Assistance Center is a group made up of Deere technicians to assist the dealer network with more advanced diagnostic troubleshooting.

iT4 – Interim Tier 4 is an engine solution developed as a go between Tier III and Final Tier 4 government regulations as an effort to reduce off highway diesel engine emissions.

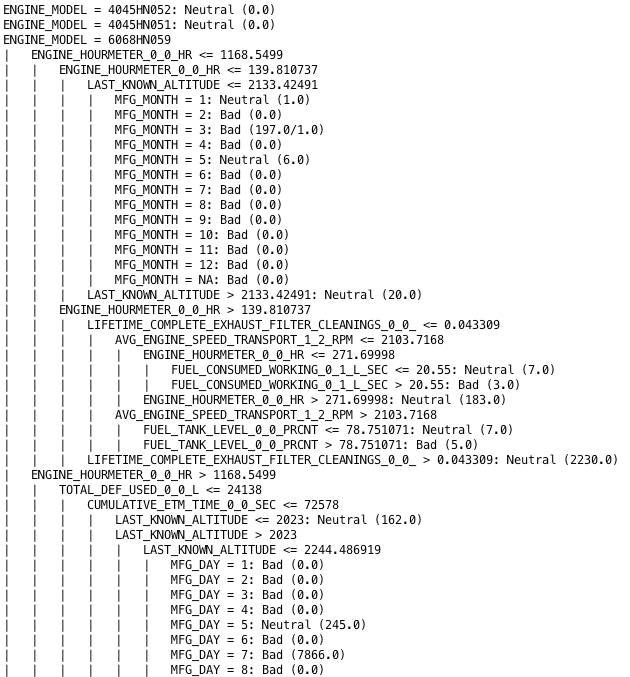
FT4 – Final Tier 4 is the final engine solution that is in full compliance with the Environmental Protection Agencies standards on emission reduction for off highway diesel engines.

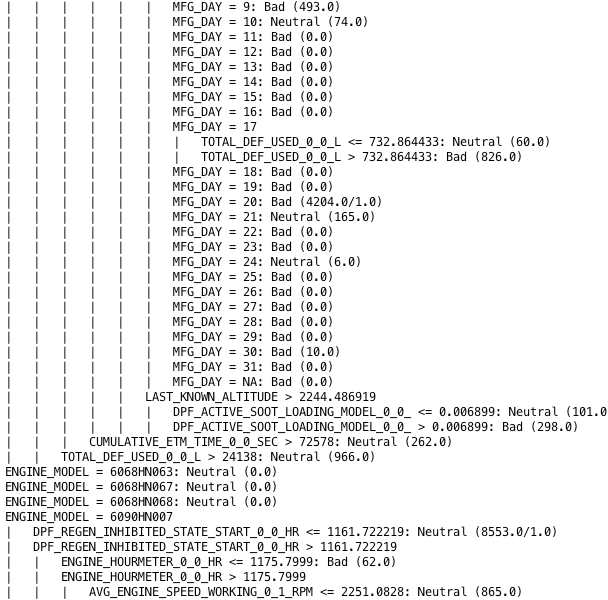
**Appendix B - All attributes in dataset**

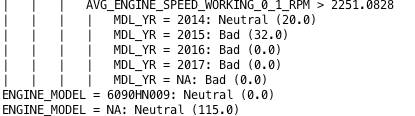




**Appendix C - Approach 1 Final Decision Tree**







**Appendix D - Approach 2 final decision tree**



