

**North South University**

Department of Electrical & Computer Engineering

**Project: Phase 1 (Cse445.02)**

**Title of the project**: APS Failure at Scania Trucks

**Team Name**: The Technocrats

**Date of submission:** 03/07/19

**Group Members**:

1. Md. Masudur Rahman (1631189042)

2. Saraf Sumaita Hasan (1631258042)

3. Md Rifat Hasan (1620259042)

4. Rehnuma Sharmin (1620739042)

**Table of Contents**

1. Introduction ---------------------- 01
2. Background ---------------------- 01
3. Data Understanding and visualization ---------------------- 04
4. Data preprocessing ---------------------- 06
5. Model implementation ---------------------- 09
6. References ---------------------- 10

**Introduction**

Our project revolves around the trucks manufactured at Scania. Heavy load vehicles like these form the backbone of the transportation system in the industrial sector. But with frequent use, they become more and more prone to malfunctions. Hence, it is of paramount importance to ensure the well-being of these trucks so that all activities remain uninterrupted and no loss is incurred. So the ultimate goal here is to minimize the maintenance costs as much as possible. This can be achieved if we can predict beforehand, any possible failure and take necessary precautions.

The problem can originate from any unit of the truck but in this case we are particularly interested in keeping track of the Air Pressure System (APS). It is responsible for generating pressurized air which is used in several functions like braking, gear changing etc. For any given truck, we want to be certain about the cause of its possible breakdown; which can either result from the failure of a component in the APS or outside of it. Basically, we are dealing with a classification problem.

In our dataset, we have the positive class and the negative class. Positive class indicates that the failing component belongs to the APS and the negative class indicates otherwise.

Calculating Total cost for miss-classification (Cost Metric):

-Given a failure belongs to the negative class but the predicted class is positive, the resulting cost will be cost\_1 (cost\_1 = 10 which is the cost of unnecessary checking/servicing)

-Given a failure belongs to the positive class but the predicted class is negative, the resulting cost will be cost\_2 (cost\_2 = 500 which is the cost of letting loose a faulty truck and the possible damage resulting from it)

Total cost calculation of a prediction model:

Total cost = Cost\_1 \* Number Instances (False positive) + Cost\_2 \* Number Instances (False negative)

The sum of (cost\_1 times the number of instances of the positive class failure) and (cost\_2 times the number of instances of the negative class failure).

A good prediction model can cut down costs to a great extent by highlighting only the trucks which require fixing. Blind tactics like checking every truck or no truck at all can be easily avoided.

**Background**

1. Paper: Prediction of Failures in the Air Pressure System of Scania Trucks using a Random Forest and Feature Engineering

Authors: Christopher Gondek, Daniel Hafner, and Oliver R. Sampson

University of Konstanz, Germany (October 2016)

The mentioned people decided to approach this problem with Random Forest. They took a close look into the data with the means of visualization schemes like box-plot, correlation matrices, scatter plots and radar charts.

As for data preparation, they used a combination of data cleaning, feature engineering and feature selection.

They figured that 82% of the values were missing for every feature and used the mean value to fill the gaps. They believed that a normalization was not necessary as their chosen classifier i.e. Random Forest performed the best among several tested classifiers.

Starting with 282 features resulting from feature engineering, they decided to carry out feature selection having known the existence of correlated features. This was done in two steps:

(i) Ranking the features according to their expressiveness

They started off by selecting a random 200 features from the 282 features and predicted the class by using a Random Forest. The precision of the results along with features were recorded.

After repeating this 2000 times, they calculated the mean precision value for each feature and ranked them in a descending order, am\_0 being the most expressive feature.

(ii) Performance testing of the feature sets of different sizes:

Here, they made use of the ranked features from the previous step, by creating feature sets of varying sizes to calculate the costs of the prediction model. A 10-fold cross validation was used to predict the class, after training with Random Forest. They determined that the average costs per data of only one feature was about 3.1 and decreased rapidly. The cost fluctuation between 10 to 282 features was very low i.e. between 0.85 and 0.6. Hence they could conclude that using many features was not necessary and so narrowed down the features list.

Moving to modeling, Random Forest always tries to minimize the prediction error and assumes all wrong predicted classes to be equally costly. This is obviously a problem here because cost\_2 is 50 times the cost\_1 which makes a huge difference. As a solution, they set a threshold for the prediction confidence for every feature set and changed it in steps of 1%. The predicted class was set to positive whenever the confidence was equal to or lower than the threshold. They figured that 95% was the best threshold in most cases and provided a graph for the costs they got from using it. They used 210 features to achieve the best prediction possible.

Finally, they provided numerical evidence of their approach bringing better results than other naive approaches. Taking every class to be negative gives a mean cost of 8.33 and taking every class to be positive gives a mean cost of 9.83, whereas with this approach they attained average costs of around 0.6 which reduces the mean cost by a factor of 13.9. A significant reduction in the total cost is seen.

1. Paper: Improving predictive maintenance classifiers of industrial sensors’ data using entropy. A case study

Authors: Eleonora Peruffo

This paper explores and compares the performances given by the various classification trees using different entropies (the measure of information present in a dataset) applied to the Scania trucks dataset. As claimed by the author, a C5.0 model is the best performing tree in this case.

Besides this one, other entropy classification trees such as Renyi and Tsallis have also proven to be useful in improving the classification of the minority class in the dataset without significantly affecting the classification of the majority class.

After looking at the different decisions taken by the data scientists who worked on this dataset before, the following judgements could be made:

* -Deleting the missing values rows completely and creating synthetic examples can be a good choice since it not only omits the incomplete cases but also results in a better distribution of the classes. But doing this reduces the dataset to only 591 instances which is a disadvantage.
* -In the case of Scania dataset, feature engineering resulted in 282 features (16 new features for each histogram) .This approach is said to be complicated because it required feature selection afterwards. Metafeature engineering applied to the Scania dataset (Cerqueira et al.; 2016) has also been regarded as an advantageous solution.

Some conclusions drawn from previous studies:

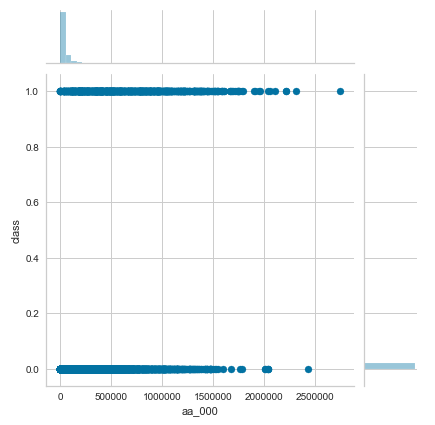
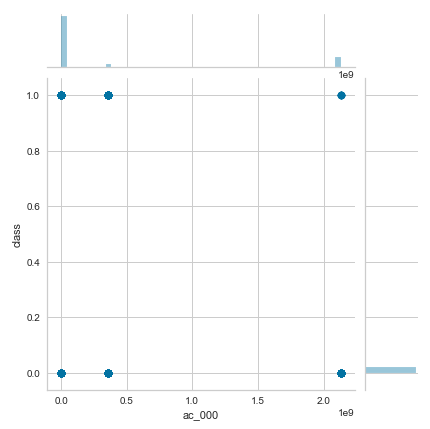
* Random Forest worked the best (Ferreira Costa and Nascimento; 2016; Gondek et al.; 2016)
* Boosted trees worked better than Random Forest (Cerqueira et al.; 2016)
* The use of an unsupervised algorithm, K-NN (Ozan et al. (2016)) will not be the best approach since it has its limitations.

**Data Understanding**

The data given to us is a high dimensional dataset which presented by the Industrial Challenge for IDA 2016. The dataset contains a test and a training set. The data consists of a subset of all available data, selected by experts (In order to guarantee the quality of the predictive model). The attribute names of the data have been anonymized for proprietary reasons. It consists of both single numerical counters and histograms consisting of bins with different conditions. The summary of the datasets are given below:

* Training dataset :
  + Number of Features : 171
  + Records : 60,000
  + Features : Anonymized
  + Type : Labeled (positive or negative) – Imbalanced(59000 – Negative , 1000- Positive)
  + Associated task : Classification
* Testing dataset :
  + Number of Features : 171
  + Records : 16000
  + Features : Anonymized
  + Type : Labeled

For the visualization, we try to generate joint plots (class vs other features) but as the number of features is very large so the sample of graphs are shown here:

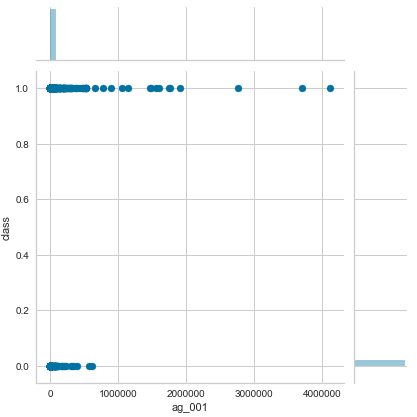
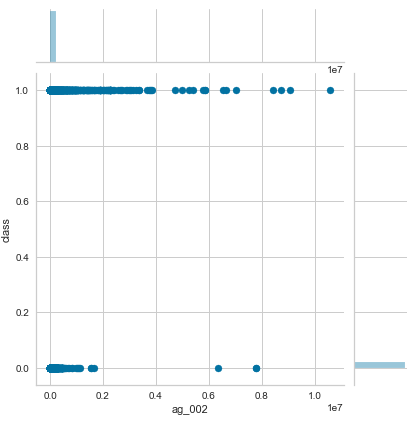
 

Fig. 1: Jointplots of the attributes for the Pos/Neg (1/0) Class and columns

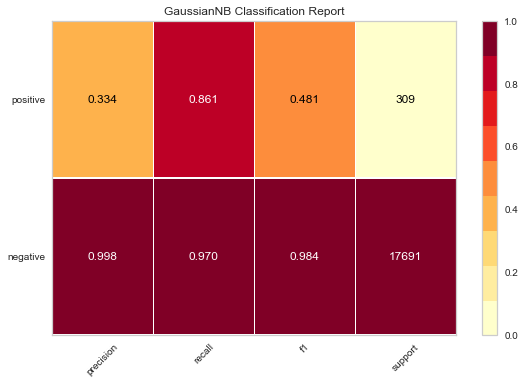


Fig. 2: Classification Report for positive and negative classes showing precision,

Recall, f1 and support

**Data Preprocessing**

After evaluating the dataset, we have observed a large amount of missing values. The missing values can be illustrated by using a **heatmap:**

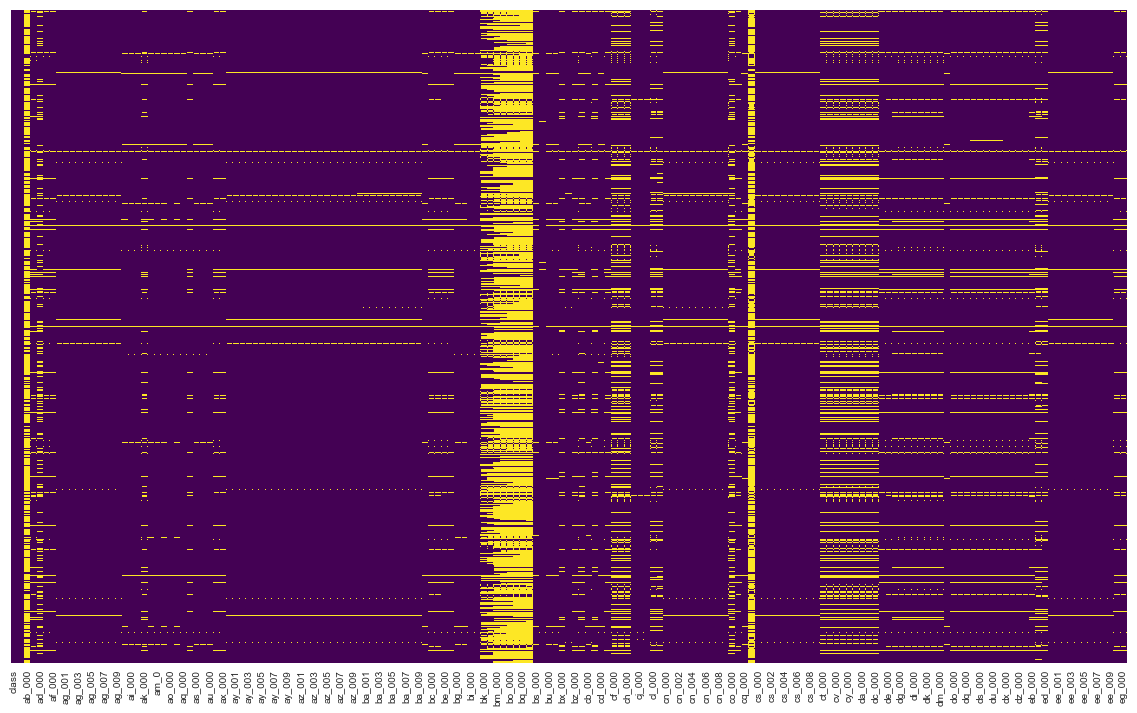


Fig. 3: Heatmap to illustrate the missing values (Yellow lines indicate the missing values)

So, we can see there are a lot of missing values (Yellow lines). So for handling these missing values we are going to follow some approaches:

1. Removing the column having more than 10,000 missing values and rest of the missing values with their mean
2. Removing the column having 80% missing values
3. Keeping all the features and replacing the missing values with their mean [Ref1 ]
4. Later, we will try to implement some feature engineering as well

So, for now we have gone through the first approach. So for that we have first calculated the columns having more than 10000 missing values:

**missing\_data = training\_data.isna().sum().to\_frame()**

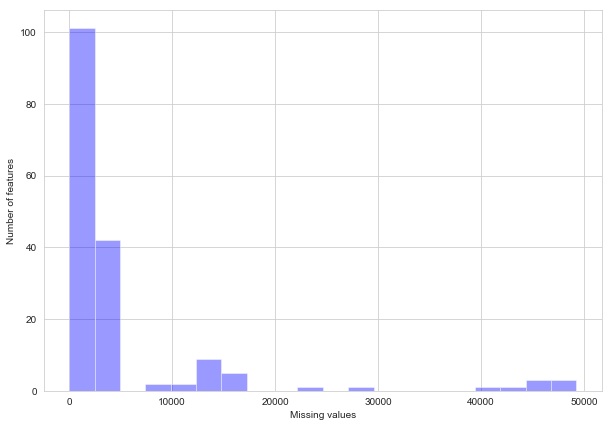


Fig. 4: Visualization of the missing values per column (Distplot)

So, by observing we can say that, a large number of features contain more than 10000 missing values. So for the next step, we have drop these columns. After dropping the heatmap becomes:

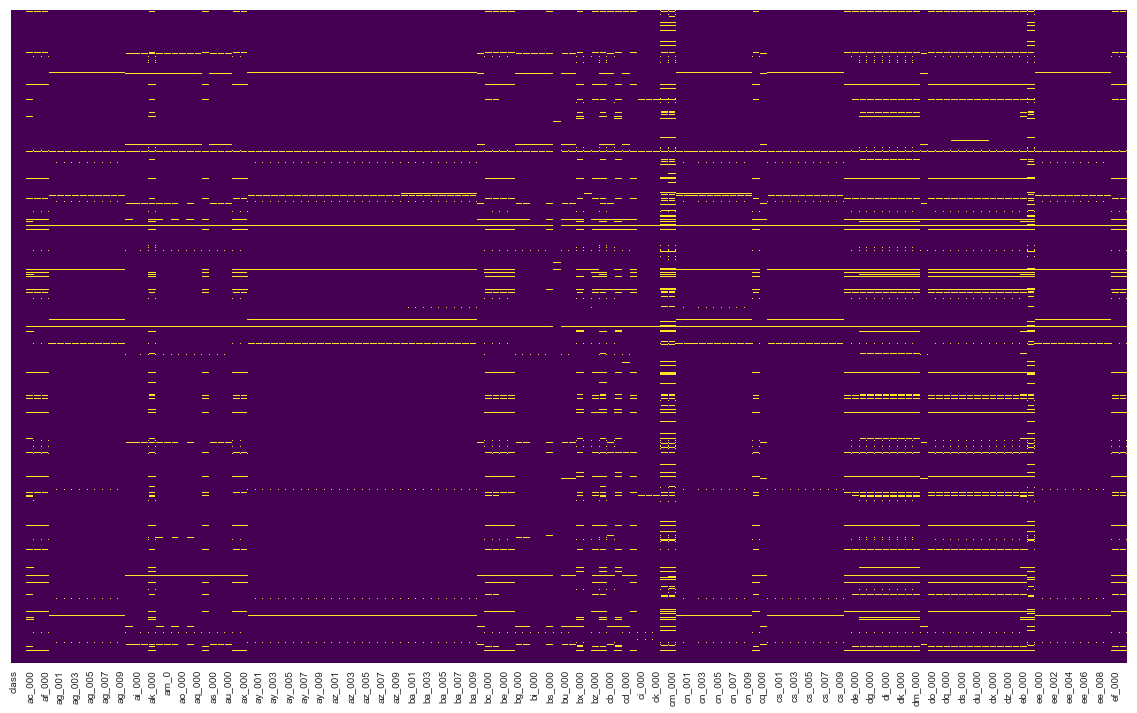


Fig. 5: Heatmap to illustrate the missing values (Yellow lines indicate the missing values) after dropping features containing more than 10000 missing values

This situation is much more improved than the previous one. But still there is some missing values. We have handled these by taking their mean as their value:

**training\_data.fillna(training\_data.mean(),inplace=True)**

After that the heatmap visualization is:

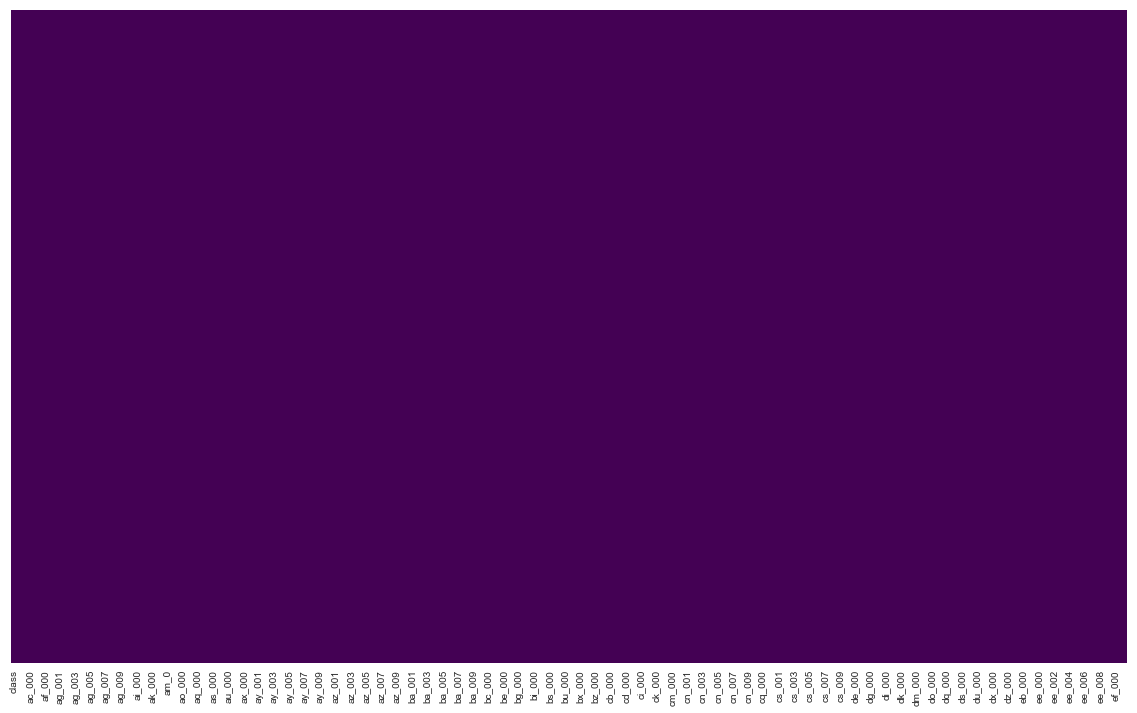


Fig. 6: Heatmap to illustrate the missing values (Yellow lines indicate the missing values) after

replacing the missing values with their columns’ mean

As we can see no yellow lines that means all the missing values have been handled and we can move forward. But still there is one step left which is to handling the categorical variables (Class: pos/neg).

We are going to replace them with a dummy variable (pos-1 and neg-0):

**training\_data = training\_data.replace('neg',0)**

**training\_data = training\_data.replace('pos',1)**

Now the data is clean enough and is ready for the implementation.

**Model Implementation**

For our first step, we have implemented Logistic Regression Model for our dataset. Later we will try to improve our estimation by implementing more models.

**Logistic Regression:**

For this model firstly we have separated out training set into predictors and response. In our dataset, the only response is the ‘**Class**’ column. So we have separated this column from the others:

**X = training\_data.drop('class',axis=1)**

**y = training\_data['class']**

So, the equation can be written as y **= f(x)**.

Although we have our separate testing dataset, but for checking we are going to split our training dataset into train and test dataset in (70%-30%) portion:

**X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=101)**

Now, we have to fit the model for the logistic regression:

**logmodel.fit(X\_train,y\_train)**

Now, it is the time for prediction:

**prediction = logmodel.predict(X\_test)**

To evaluate our prediction we will need a classification report. The classification repost states:

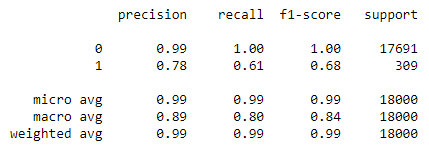


Fig. 7: Classification report

By observing the report, we can see that the negative class prediction has a precision of 99% which is pretty standard whereas the positive class prediction stands with a precision of 78% which is average. For more evaluation we have used the confusion matrix to estimate the cost:



Fig. 8: Confusion Matrix

By observing the confusion matrix, we can see that we have Number of Type 1 faults which is **false positive (fp)** of 53 which is quite considerable. On the other hand, Type 2 faults which is **false negative (fn)** is 122 which is considerably large and directly affecting our **cost score** as type 2 faults have the higher expense.

So, about the model, we can conclude by saying that, although we have a good precision and good estimate for the Type 1 fault but still the Type 2 fault is lagging the score behind. So for further exploration we need to focus more about handling the missing values in another approach and to choose another model which eventually going to create a balance between the type 1 and 2 faults by also considering the individual expense.

**References:**

1. Christopher Gondek, Daniel Hafner, and Oliver R. Sampson: Prediction of failures in the Air Pressure System of Scania Trucks Using a Random Forest and Feature Engineering
2. Eleonora Peruffo: Improving predictive maintenance classifiers of industrial sensors’ data using entropy
3. Hyun Kang , Department of Anesthesiology and Pain Medicine, Chung-Ang University College of Medicine, Seoul, Korea: The prevention and handling of the missing data