Analysis Report

Cause-of-failure classification

APS Failure at Scania Trucks Data Set

by,

Deborshi Goswami

Graduate Student

Dept. of Industrial and Enterprise Systems Engineering

University of Illinois at Urbana Champaign

Objective

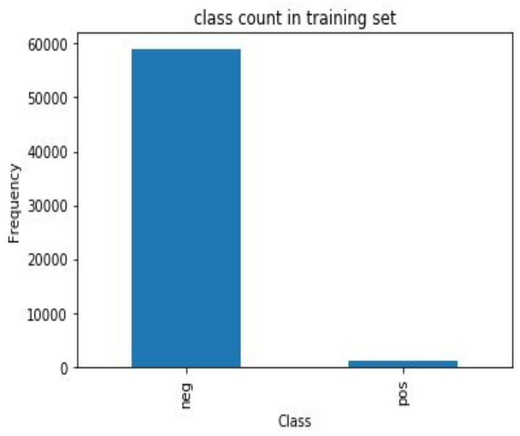
The Air Pressure System(APS) of heavy Scania trucks generate pressurized air that are utilized in critical systems such as braking and gear changes.[1] Naturally, APS failures are not only incredibly costly to the company employing Scania trucks, but also pose a legal liability in terms of property damage and casualties, caused due to failures of the braking mechanism for instance. It is therefore, in our interest to be able to tell whether a truck can fail due to an APS system failure based on measurements of the truck's condition. The problem is a binary classification problem to classify the samples into APS and non-APS related failures. The aim of this analysis was two fold:

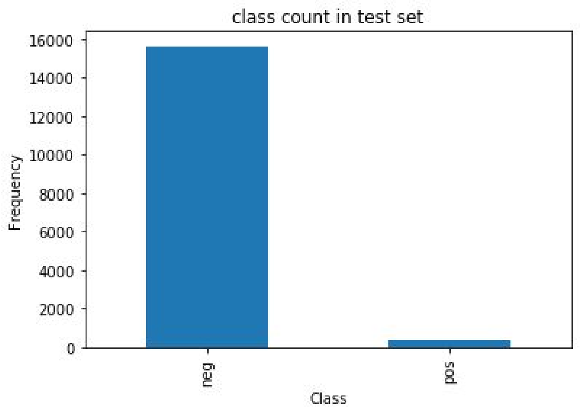
a. Analyzing the performance of three different supervised learning algorithms in classifying APS failures.
b. Obtaining the best possible classification performance on the testing set i.e. to minimize the cost of mis-classification.

Inferences from Data

The first observation to be made is the high dimensionality of the data-set with 171 attributes. It is quite likely that a number of these attributes do not have strong correlations with the attribute 'class' which contains the classification labels. This paves the way for reduction of dimensionality. However, the dimensionality reduction approaches like PCA would not be wise as PCA does not necessarily maximize the inter-class variance and so we run the risk of losing information. Linear Discriminant Analysis or its sister algorithms however could provide the best subspace to maximize inter-class variance. However initially, one must refrain from transforming data to a subspace without understanding how the data behaves. There could be just a few attributes which completely determine the classification outcome and therefore dimensionality reduction through transformation could be sub-optimal. One algorithm that can be eliminated due to high dimensionality is Logistic Regression(LR). As noted by Zhang and Oles[2][3], LR is unsuitably slow for high dimensional data and may lack regularization.

The second most obvious observation is the class imbalance. The number of negative samples outnumber the positive samples by 50 to 1 in the training set. This could bias our model towards predicting negative classes more often. Since the cost of a false negative is high, the class imbalance has to be dealt by adding class weights during the model fitting. As seen below, the class imbalance is extremely glaring in both, the test set as well as the training set. However, we can see that the ratio of negative to positive samples is the same over the training and testing set. This indicates that samples in both sets were chosen from the same distribution.





A relatively hard to notice but, interesting observation is that in the dataset description, attributes are often histogram bins where each name, such as ‘ab’ represent a physical quantity being measured and each bin represents the values to be fed into the histogram. This means that ‘ab_000’ and ‘ab_001’ are essentially same data being described but put into different bins. This has been explored further in the feature engineering section.

Data Cleaning and Preprocessing

Eliminating columns with high missing values

A better approach to reduce the dimensionality than described previously would be to eliminate unnecessary attributes completely. A number of attributes in the data have a very large number of missing values. Traditional techniques of imputation would not yield good results and could even lead to information loss. Such attributes may be eliminated. For this dataset a very low threshold was set for eliminating columns. If more than 2.5% of the data points in a column were missing then they were removed. This value was obtained after tuning of the model.

Imputation for removal of Missing Values

Attributes where the number of missing values was lesser than the defined threshold were appropriately imputed using the means of every column. Mean imputation seemed more natural for this dataset as a median imputation would find values that are affected by outliers in a column.

Feature engineering

As mentioned earlier, features have an identification such as ‘ab’ and each identification has a bin number such as ‘001’, ‘002’, etc. For this analysis, these attributes have been grouped into one feature with one identification number by taking the mean of the values of each bin. The rationale for this is simple, as each of these values represent bins of a histogram they represent the same feature. Having multiple attributes represent the same feature may get in the way of classification rather than aid it.

Standardization

Each column of the training dataset was standardized to a its mean and unit variance. This was a design choice based on rules of thumb in data science. Care was taken to ensure that the test data set was also standardized to the same mean and variance as the training data set.

Rounding

In order to increase the chances of getting repeated values in the data, all entries were rounded off to 2 decimal places. The value was obtained after tuning the performance of the model to find the number of the decimal places that yielded the lowest cost. Rounding off was performed with the aim of aiding the classifier in identifying prominent features, for example, in decision trees. However, too much rounding resulted in too many repeated values therefore leading to underfitting. Rounding also saved significant computation time.

Shape of Data sets

The shape of the the two sets before and after preprocessing have been given below:

DataSet	Before Preprocessing	After Preprocessing
Training Set	60000 x 171	60000 x 37
Testing Set	16000 x 171	16000 x 37

Model Selection

Support Vector Machine

A good starting point for the data analysis was to choose Support Vector Machine to obtain an initial baseline for the cost estimate. SVM's success in the machine learning world has been prolific, making it applicable in everything from facial recognition to audio-classification. Since it accommodates quite easily for non-linearities using the kernel trick, SVM provides a good way of understanding the difference in performance between a linear model and non-linear models. Especially when the behaviour of data is unknown the, 'rbf' kernel normally is a good place to start the analysis procedure.

Random Forest

This model was chosen with the intuition of it being the most suitable model for this application. Being an ensemble model i.e., it aggregates multiple decision trees to build a more powerful one, the random forest is a very strong learner. Since it is quite likely that every feature of the Data is not useful, the RF method provides a good way of performing classification using the best subset of features or at least close to the best subset of features.

Bagging CART

Given the assumption of Random Forest being the optimal classifier it would be interesting to test another ensemble method which bags multiple decision trees. While algorithms like CART are sensitive to training data, the Bagging CART model effectively chooses the most frequent outcome out of a number of CART models which is where the term 'Bagging' comes from. Given that it is an ensemble method, it is known that it will perform better than CART. However its performance against Random Forest is less obviously differentiable and is interesting to test.

Model Fitting

Throughout the model building process, the final test set was left untouched until there was enough surety that the models were performing optimally.

80/20 Split

For the initial training and validation of each model an 80 to 20 split was maintained between the training and validation set. Each set is sampled at random from the training data without replacement. In general an 80/20 Split is better when computation time is a concern. Although tuning of the model has to be effectively manual, a split instance saves time over a 10-fold validation as it is in essence a single fold validation.

Cross Validation

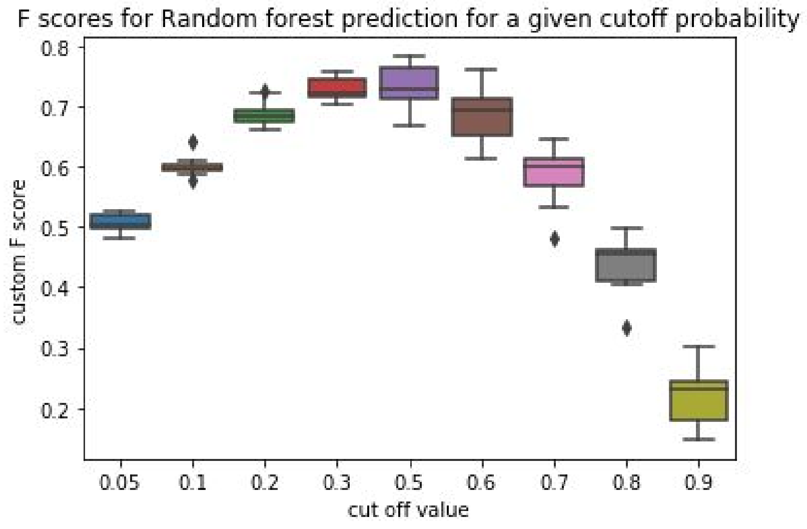
Towards the later stages of the analysis, a 10-fold cross validation was implemented on the training set in order to tune for the best parameters.

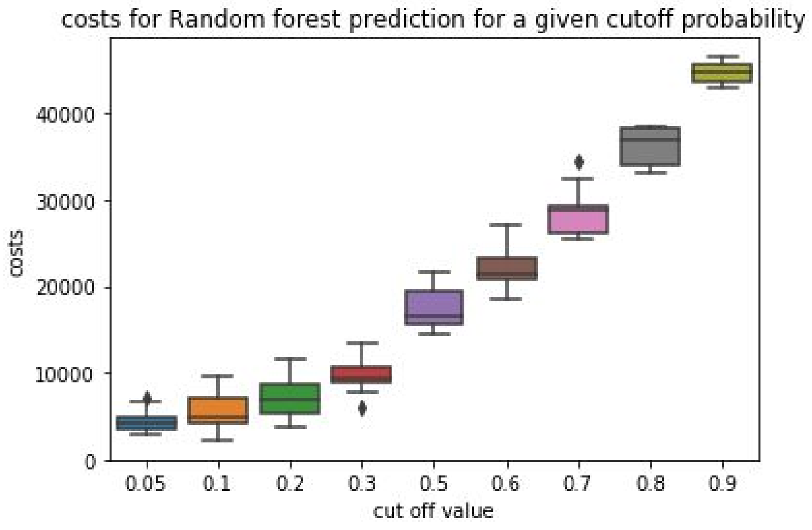
Calibration of Cut-off Probabilities[4]

This was a significant feature during the validation stage of the model and was designed with the aim of compensating for the class imbalance. Normally, classification algorithms predict anything with a probability of over 0.5 as a positive class and probabilities under 0.5 are classified as the negative class. However, the problem with this approach is that it doesn't take into account the class imbalance present in the data. The APS dataset, if naively trained on a model, is more likely to predict a negative than a positive class. Therefore the effective probability required to predict a sample as positive is very high. Hence, the probability threshold for positive classification has to be lowered. The idea was to tune the probability cut-off during the fitting and validation phase to minimize the cost of mis-classification. This has been emphasized later in the results section. There is another benefit to Calibration which has been addressed in the next section.

Evaluation of Performance

There were two performance metrics to choose from, the F1 score of predictions and the cost. Both metrics were evaluated before proceeding to the model fitting and classification process. The F1 score normally represents the ideal balance between Precision and Recall of a classification algorithm by computing the harmonic mean of the two. However, for this problem there was an additional consideration lent by the dataset description in the form of Cost. If a faulty APS system is misclassified it has a much worse consequence than if a working APS system is classified as faulty. At worse the false positive classification gets sent to a mechanic and hence the cost of the error is low. The cost ratio of false negative(FN) to false positives (FP) is 50:1. The effect of erroneously choosing F1 as the performance metric is emphasized below using the Random Forest Algorithm.





As seen above, the best F1 score is obtained at probability cut off of 0.5. This means at a cut off of 0.5 there is a good balance of false positives and false negative, or rather the ideal balance. But it does not necessarily minimize the cost of misclassification. As we can see, the cost reduces uptil a cut off of 0.05 where it starts to stagnate. This particular model was already pre-tuned to compensate for class imbalance using class weights which is also the reason why we see the F1 score peaking at 0.5. Even then we can see that maximizing the F1 score does not minimize the cost. This is true for all model parameters and not just the probability cut off. Hence, the cost is better metric to use during performance optimization.

Parameter Tuning

SVM

The only algorithm in our selection that does not naturally provide probability estimates is SVM. The estimates must be derived from the distance of a predicted sample from the decision boundary. This may prove to be an unreliable estimate. Hence the SVM model was tuned by iterating through different values of important parameters and then minimizing the cost. The important parameters considered were the type of Kernel and the class weights (to compensate for class imbalance.) The kernels considered were linear, polynomial, and the radial basis function. The performance on each of these kernels provided a good understanding of the non-linearity in the model. The best parameters are listed below:
Kernel : Polynomial
Class weight: Neg → 1 to Pos → 500
Also due to the high computation time of SVMs, the train test split model was chosen to tune the algorithms.

Random Forest and Bagging CART

Both of these models were tuned using the K-fold cross validation. Parameters considered for initial tuning of RF were ‘number of decision trees’ to consider for model building, the maximum subspace of attributes to consider and the class weights. Other parameters seemed to have a negligible effect on the model. The number of trees in the Bagging CART model was the important parameter considered.

The final tuning and optimization however was performed for the probability cut-offs. This optimized the cost for class imbalance as well. Hence, the error-rate, or rather, the final cost of misclassification was much lower on these models as compared to SVM.

Results and Comparison

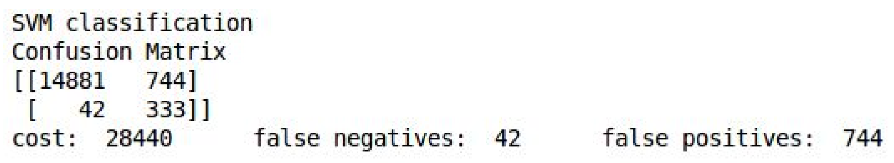
Computation Time

Although given the size of the dataset and considering more powerful processors exist, this metric of performance was relatively unimportant. However, since the computation time to a large extent determined the methodology of parameter tuning (SVM could not be cross validated), one would be remiss to not pay attention to it. The Random Forest algorithm was the least computationally expensive quite likely because it was considering a subset of the whole feature space to train on. Bagging CART and SVM were much more computationally expensive with SVM taking a disproportionately large amount of time to converge during training.

Cost and Mis-classification

SVM

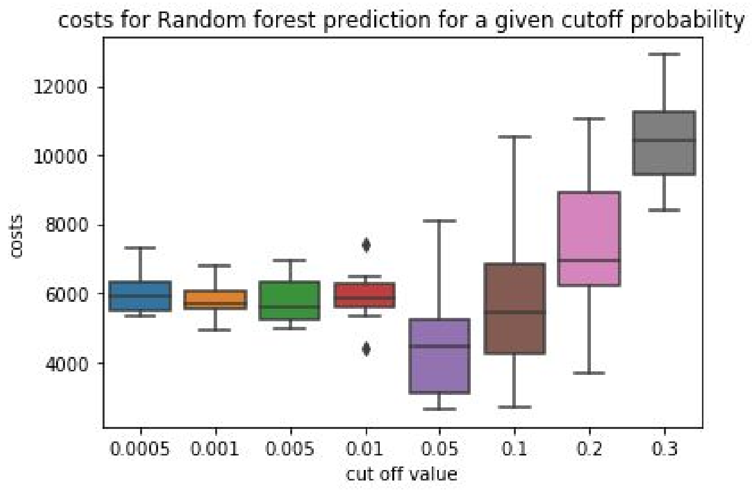
While SVM provided a good starting point for classification, it proved to be not the best one. This was also partly due to the fact that SVM could not be optimized completely for cost due to its high computation time. The model was fitted on the training set and predictions were made on the test set to yield the following result:



```
SVM classification
Confusion Matrix
[[14881   744]
 [   42   333]]
cost:  28440     false negatives:  42     false positives:  744
```

Random Forest

Shown below is the optimal performance cut off probability of RF.

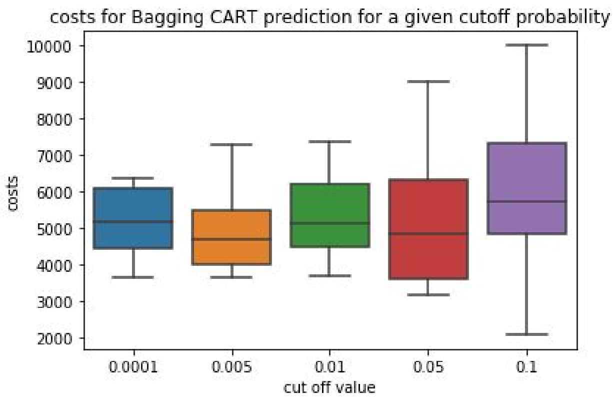


The model was fitted on the training data and then predictions were made on the test set to yield the following results.

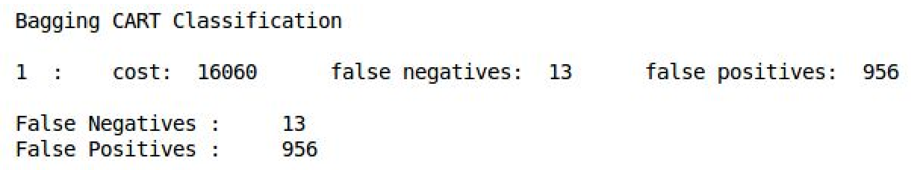
```
Random Forest Classification

1  :    cost:  15050     false negatives:  21     false positives:  455

False Negatives :     21
False Positives :     455
```

Bagging Cart

The best performance cutoff of Bagging CART can be seen below:



The interesting part about the result obtained below is that while Bagging CART performs worse than Random Forest in terms of net cost, it in fact makes lesser false negative predictions. This could have been because the class imbalance was overcompensated for.



```
Bagging CART Classification

1  :    cost:  16060      false negatives:  13      false positives:  956

False Negatives :     13
False Positives :     956
```

Conclusion

Given the time frame for this analysis, the amount of tuning and experimentation possible to be done on different models was less than optimal. Given the results and given a well preprocessed data, the best performer out of the three models chosen was the Random Forest in terms of both misclassification cost as well as computation time. However, further testing is to be done to determine the clear winner between Random Forest and Bagging CART in terms of accuracy of predictions, even though the RF beats Bagging CART in computation time.

Citations

[11] UCI Repository : APS Failure at Scania Trucks
https://archive.ics.uci.edu/ml/datasets/APS+Failure+at+Scania+Trucks

[2] Logistic Regression for Data Mining and high dimensional Classification, *Paul Komarek*
http://repository.cmu.edu/cgi/viewcontent.cgi?article=1221&context=robotics

[3] Text Classification Based on Regularized Linear Classification Methods
https://link.springer.com/content/pdf/10.1023%2FA%3A1011441423217.pdf

[4] Dealing with Unbalanced Class, SVM, Random Forest, Decision Tree
http://bigdata-madesimple.com/dealing-with-unbalanced-class-svm-random-forest-and-decision-tree-in-python/

General References

[5] Scikit Learn Software and Documentation
http://scikit-learn.org/stable/

[6] Stack-overflow
www.stackoverflow.com

[7] Kaggle
www.kaggle.com