Project Implementation

Group 23

Overview of Problem

This project is focused on classifying the root cause for the breakdown of heavy Scania trucks; whether they are caused by failure in Air Pressure system (APS) or not. The APS is utilized in various critical functions, such as braking and gear changes, and its failure renders a truck out of service. The dataset presents a binary classification problem, where the positive class constitutes a specific component failure of the APS and the negative class indicates a failure unrelated to the APS. The objective of this binary classification problem is to minimise the false classifications by correctly identifying and classifying the failure type (APS/non-APS).

Dataset Description

The dataset has been collected from heavy Scania trucks and sourced from UCI Machine Learning Repository. It consists of 76,000 observations, pre-partitioned into training and test sets of 60,000 and 16,000 examples respectively. There are 171 attributes, the names of which have been anonymised for proprietary reasons. The data types consist of single numerical counters, seven histogram variables (consisting of bins with different conditions), and one factor attribute for the Class.

The challenges of this dataset include: (1) extreme class imbalance (<2% positive class), (2) large size and the confidential nature of the dataset, (3) the high proportion of missing values (NA's), zeroes and outliers, and (4) the motivation to minimise Total Cost to Scania, by minimising misclassification.

Initial Data Analysis

Our exploration has been centred on the training set to avoid information leakage to the test set. In Figure 1, the barplot shows the class imbalance of the training set and the second chart displays the distribution of the values in column aa_000, which is endemic of more than 90% of all attributes for this dataset. Here, the majority of values are zeroes, while non-zero values have a high variance. Only 11 attributes have a more linearised distribution like bn_000 and ca_000 below.

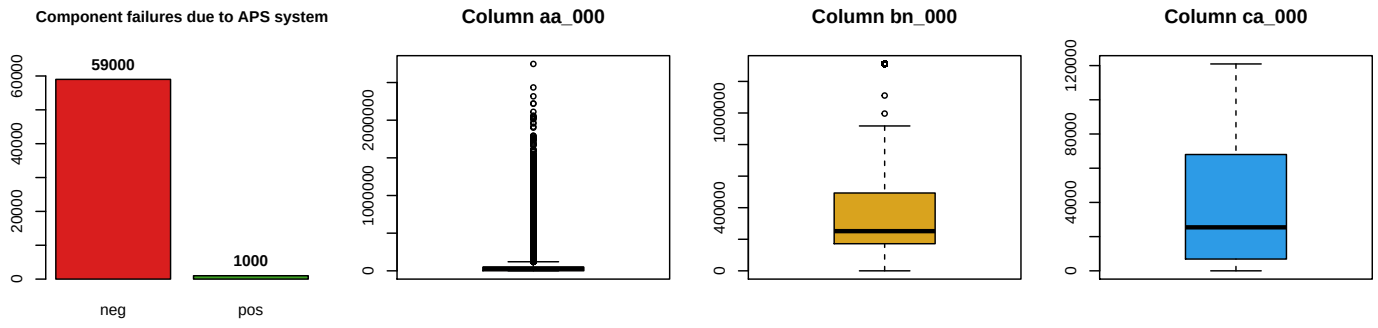

Figure 1: **Figure 1. Data distribution of APS truck dataset**

From Figure 2, the histograms show the proportion of missing and zero values. Whilst only eight attributes contain more than 60% missing values, 49 columns contain more than 60% zero values. All columns also contain mild (outside 1.5 IQR) and extreme (outside 3 IQR) outliers. This is because, as stated earlier, more than 90% of attributes are skewed towards zero, with a high variance in non-zero values.

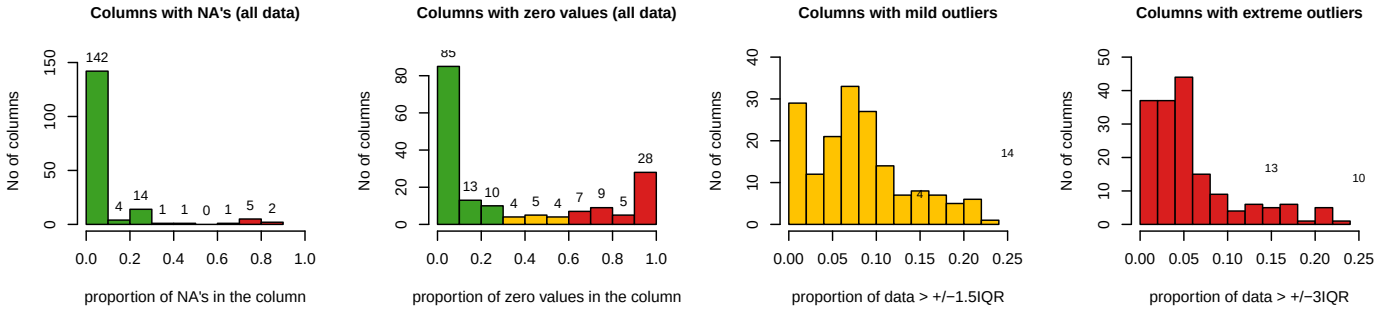

Figure 2: **Figure 2. Proportion of NA's and Zero's in the data**

Data Cleaning

Using a large number of features to train a model can reduce the quality of the model. This is because data points in a large-dimensional space can be very far away from each other, resulting in poor model predictions. Additionally, not all features are truly associated with the response variable and some may be highly correlated with each other. Using these noise or redundant features can lead to overfitting and increase the error on the test data (James, Witten, Hastie, and Tibshirani, 2013). Therefore, the first step in our data cleaning was to remove the NAs and reduce the number of features from 170 to a more reasonable number. First, column "cd_000" was removed as it contained only two values (1,209,600 and NA). Then, 51 columns with more than 60% of missing values were also removed, as the large number of NAs in these columns can introduce bias and lead to deterioration of our models.

We also removed 74 features which have high Pearson correlations with the other features in the dataset (r>0.7). We then performed forward and backward selection methods (using 'leaps') to further reduce the number of features to 20 (as shown in Figure 3, whilst according to 'leaps', the optimum number of features is 36, from 20 features onwards the reduction in Mallows Cp had flattened out).

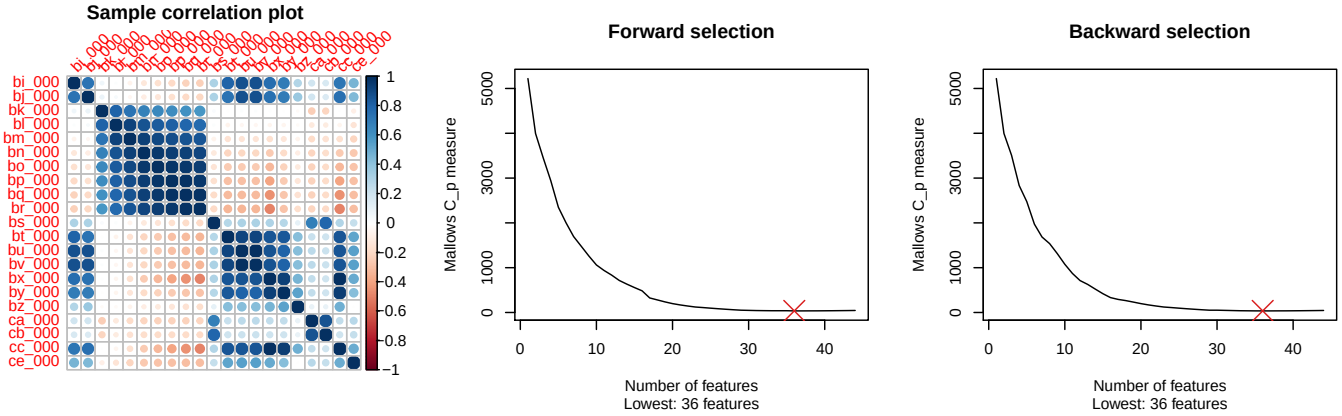

Figure 3: **Figure 3. Sample Corellation Plot and Forward/Backward Feature Selections**

Despite removing the above columns, there were still 39,062 rows remaining with at least one missing value. Three methods were tested to impute these missing values, with examples of scatter plots for the values in columns az_003 against bk_000 shown in Figure 4 below.

From figure 4, Amelia II did not perform well on this dataset as it generated a large number of negative values, likely due to its assumption of multivariate normality (Honaker, King and Blackwell, 2019). The median approach on the other hand, was influenced less by the highly skewed data distribution, but it resulted in many zero values and could cause the uncertainty in the missing value data to be lost (Wishart, 2020). The last approach, MICE, generated more varied values with no negatives because the Classification and Regression Trees (CART) method used in MICE is quite robust against outliers, and can handle both multi-collinearity and skewed distributions quite well (van Buuren, 2018).

Given the high correlation between many pairs of columns in the training set, another sensible approach is to use the Principal Component Analysis (PCA) for feature reduction. Since the data is already anonymised, there is no loss of interpretability here as would normally occur with PCA. The scatterplot in Figure 5 below shows the two classes plotted

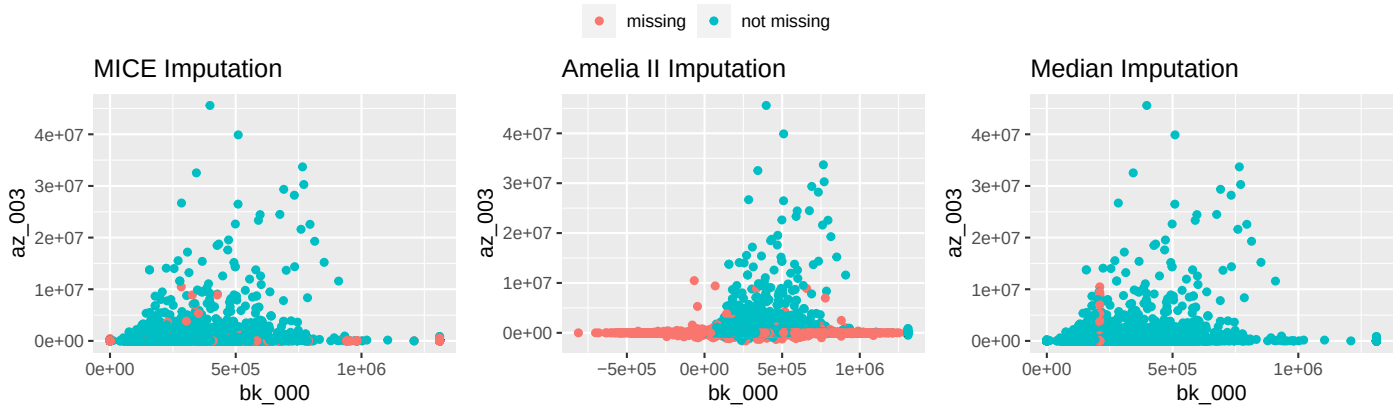

Figure 4: **Figure 4. Comparison of missing data imputation methods**

using the first two principal components. There is some structure in the negative classes, though there is no clear distinction of positive classes. The cumulative variance plot on the right hand side shows over 95% of the variance can be explained using only the first six principal components, meaning the number of features can be reduced quite significantly.

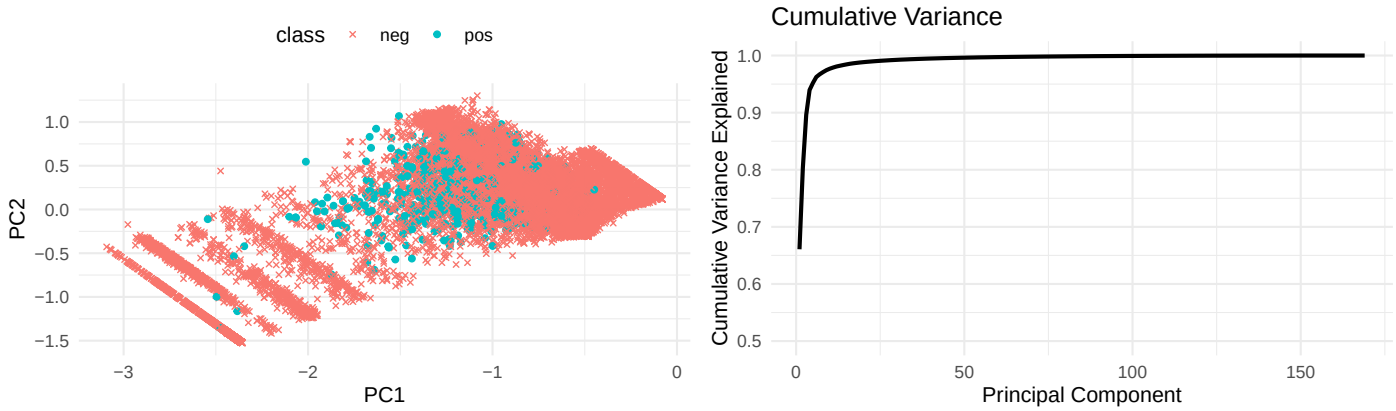

Figure 5: **Figure 5. PCA Data Plot and Cumulative variance**

Lastly, in order to address the class imbalance issue, we used the SMOTE algorithm to up-sample the "pos" class and down-sample the "neg" class in the training set (we kept the test set imbalanced as any new data during testing and implementation would also be imbalanced). Using k=5 (five nearest neighbours are used to generate new examples in the "pos" class), perc.over=3000 (this decides how many extra data points from the "pos" are generated) and perc.under=100 (this decides how many extra data points from the "neg" are selected for each case generated from the "pos"), we obtained 30,000 "neg" classes and 31,000 "pos" classes.

Methodology and Modelling

Train/Test split

The dataset provided by Scania was already split into training and test sets. Initial data analysis, hyperparameter tuning and model training were performed on the training set, while the test set was used for evaluation and final model selection.

Feature Engineering

As mentioned during the data cleaning stage, we removed all features except the 20 that we believed would best explain the data to produce a dataset denoted by Truck20. We also performed Principal Component Analysis on the training data to produce a PCA dataset, where we kept the first six principal components, which explained 95% of the variance in training data. Both datasets were used to train our models to compare the two pre-processing methods. Since the distribution of

classes is highly imbalanced, we also balanced the training set using SMOTE, as described earlier. The balanced Truck20 and PCA datasets were saved as csv files and used for training all models in order to maintain consistency.

Model Selection

For a robust comparison between models we have used repeated cross validation with 10 folds and 3 repeats on the balanced training datasets. Hyperparameter(s) of the models were tuned using caret::train, which sets up a grid of tuning parameter combinations and evaluates the performance of each fold and hyperparameter combination. The best hyperparameters were then used to fit the final models on the full training set and later to predict the test set. The models considered are Logistic Regression (LR), Linear Discriminant Analysis (LDA), k-Nearest Neighbours (kNN), Decision Tree (DT) and Random Forest (RF).

Model Evaluation

For evaluation and comparison purposes, all models were evaluated on the following metrics:

Total Cost: The correct classification of APS system failures is paramount from a cost perspective. Scania has provided a cost metric, defined as: Cost = 10FP + 500FN. A false positive (false APS fault) requires unnecessary workshop checks keeping trucks off the road, and false negatives (APS fault not recognized) could cause potential breakdown while in use. Since the severity of potential breakdowns is much higher than unnecessary checks, the cost of a false negative is weighted much more highly. As this metric has been provided specifically for use with this application, it will be the primary metric considered, and will be used for hyperparameter selection. Misclassified datapoints act to increase the total cost, so the objective is to minimize this metric. Also of note is that this metric is not scaled relative to the number of datapoints, so the test set will almost always have a higher cost than the train set, since it has more samples.

Area Under Curve (AUC) / Receiver Operating Characteristic (ROC) curve: ROC is a probability curve and AUC represents degree or measure of separability. It indicates how well the model is capable of distinguishing between two classes. Higher AUC values mean the model is better at predicting the classes. The ROC curve is plotted with the true positive rate (Sensitivity) against the false positive rate (Specificity).

F1-score: This metric is used when the false negatives and false positives are important parameters for evaluation, which is the case with this dataset. The F1-score is the harmonic mean of Precision and Recall and gives a better measure of the incorrectly classified cases in cases such as this where the data is imbalanced, when compared to standard accuracy.

Model Training

As mentioned earlier, five machine learning algorithms were used to train and predict the root cause for the breakdown of heavy Scania trucks, ie whether it is caused by failure in APS or not.

Logistic Regression

Logistic regression models the probability that a sample belongs to a particular class. It uses the logistic function to compute the probability and the maximum likelihood method to estimate the coefficients, so that the predicted probability of a class corresponds as closely as possible to the actual class. In this dataset, the class is categorized as 'pos' if the estimated probability is higher than 0.50, and 'neg' otherwise. We use logistic regression as the base model for our model comparison because it is fast to train, works well with high-dimensional data and can produce good predicted probabilities.

Dataset	Hyperparameters	Best Parameters
Truck20	cost = c(0.1, 1, 5, 10, 15, 20)loss = c("L2_primal", "L1")epsilon = c(0.000001, 0.0001, 0.001, 0.01, 0.1)	cost=5loss = "L2_primal"epsilon=1e-04
PCA	cost = c(0.1, 1, 5, 10, 15, 20)loss = c("L2_primal", "L1")epsilon = c(0.000001, 0.0001, 0.001, 0.01, 0.1)	cost = 10loss = "L1"epsilon = 1e-06

Table 1. Logistic Regression Hyperparameter Tuning

We applied regularized logistic regression in caret using method='regLogistic'. This method has three hyper-parameters to tune: the loss function, epsilon (tolerance for stopping criteria) and cost. The cost parameter penalizes the complexity

of the model, reducing the variance, and consequently the noise components at the expense of some bias. This can prevent overfitting of the training data and perform better in generalization of the unseen test data. As can be seen in Table 1, we tested a wide range of parameter values to ensure the best parameters are chosen and the cost results can be seen below.

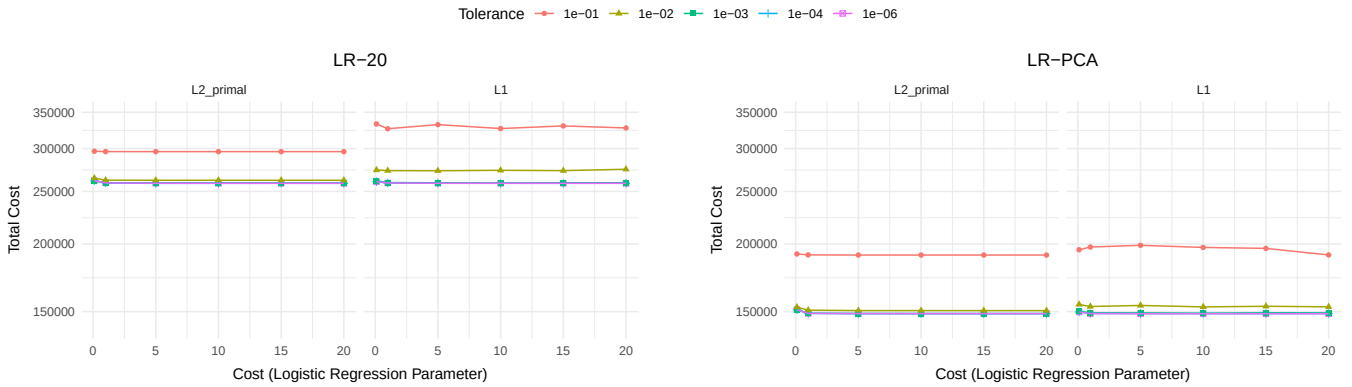

Figure 6: **Figure 6. Logistic Regression Hyperparameter Tuning Results**

Figure 6 shows the total cost is mostly constant for various values of the hyperparameters, except for tolerance. A large tolerance value can cause the algorithm to stop early (even before convergence), resulting in bad classification, whereas a small value might mean slow convergence, but better classification result. For both datasets, small tolerance values seemed to have worked well, giving the lowest validation costs.

Linear Discriminant Analysis (LDA)

Linear Discriminant Analysis utilizes Bayes' theorem to find the probability of a response class for a given set of predictors. It assumes that the distribution of the predictors is approximately normal (Gaussian) in each of the classes and the covariances among the predictor variables across all classes are equal. It models the distribution of the predictors separately in each of the response classes, ie given class, and then flip these around into estimates of the posterior probability of the classes using Bayes' theorem.

There are no hyperparameters to tune for the caret LDA, and therefore, the default parameters have been used for fitting the model. It can be seen from the initial data analysis that most of the data is extremely skewed towards zero with wildly different variances. Therefore, LDA is not expected to produce a good result for this dataset. However, it is still included for its simplicity, short training time and as a point of comparison.

k-Nearest Neighbours (kNN)

k-Nearest Neighbours classifies a sample based on the classes of its closest neighbours; the assigned label will be the majority class held by the points near it. The model works on the assumption that similar data points are always close together. It does not make assumptions on the model form, making it more flexible than parametric methods such as Logistic Regression or LDA. However, it can have high computational cost as it needs to recompute the distances from the new sample to all training data points to make a prediction.

Dataset	Hyperparameters	Best Parameters
Truck20	k = c(3, 5, 7, 9)	k=3
PCA	k = c(3, 5, 7, 9)	k=3

Table 2. kNN Hyperparameter Tuning

The hyperparameter for kNN is k, which is the number of the nearest neighbours to consider. A lower k has fewer neighbours from which the prediction will be based; it will be more flexible and have lower bias, but this may come at a cost of higher variance. A higher value of k on the other hand, may use more neighbours to derive class of the new sample; however, it may have less variability but higher bias. As seen above, to find the optimal value of k, which is not too low and too flexible that it overfits the training data and not too high that it underfits, a sequence of k=3-9 with increments

of 2 were used on the training set.

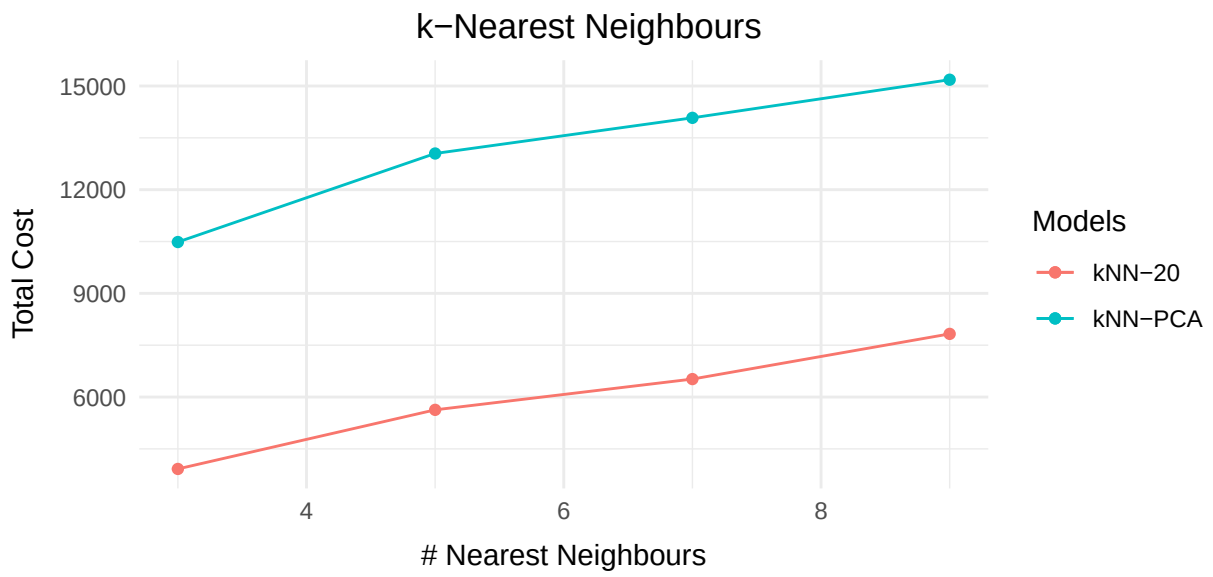

Figure 7: **Figure 7. kNN Hyperparameter Tuning Results**

Figure 7 shows the resulting hyperparameters for kNN were the same for the two datasets, with both having the lowest Total Cost at k=3. This is a bit unexpected as lower k's can often overfit the training data by focusing too much on individual training samples rather than finding trends in the data.

Single Decision Tree

Decision trees use a series of simple binary decisions to generate a prediction. Although they are often outperformed by other methods, decision trees have the advantage of producing models that can be easily interpreted and displayed as a dendrogram, as shown in Figure 8 below. In this case, a decision tree could be incorporated by Scania and used by non-technical people to determine the nature of a fault in the APS.

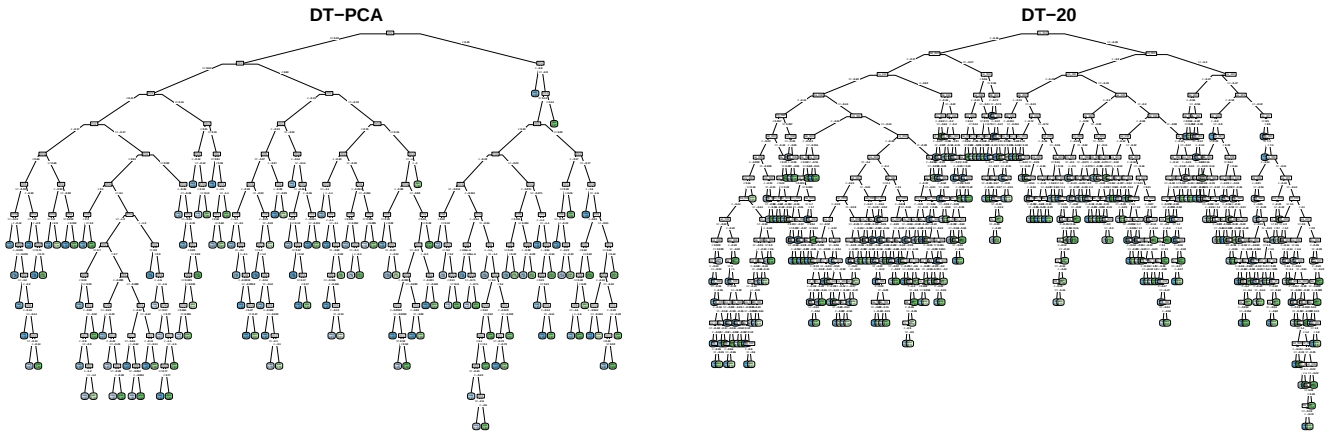

Figure 8: **Figure 8. Decision Tree for Truck20 (left) and PCA (right) datasets**

In caret, decision trees are implemented using the 'rpart' package, which uses a complexity hyperparameter cp. This controls the size of the tree by introducing a penalty based on the number of splits in the tree (Therneau & Atkinson, 2019). A tree that is too large is likely to focus too much on the specifics of the training data and thus overfit, while a tree that is too small will not capture all of the structure of the data and may underfit. In order to find the optimal value of cp, a sequence of 20 cp values from 0.0002-0.1 were tested.

Dataset	Hyperparameters	Best Parameters
Truck20	cp = c(0.0002, . . . , 0.01) (20 values)	cp = 8.4743e-05
PCA	cp = c(0.0002, . . . , 0.01) (20 values)	cp = 1.2915e-04

Table 3. Decision Tree Hyperparameter Tuning

Figure 9 shows the effect of the complexity hyperparameter cp is similar across the datasets. Although the PCA dataset gives a lower cost metric across most complexity values, it comes at the cost of the loss of interpretability, as the principal components have no physical meaning. This means one of the significant advantages of decision trees is lost. However, since the data is anonymised anyway, this is not a material issue. If Scania were to use this model, they might opt to use the non-PCA data for interpretability benefit.

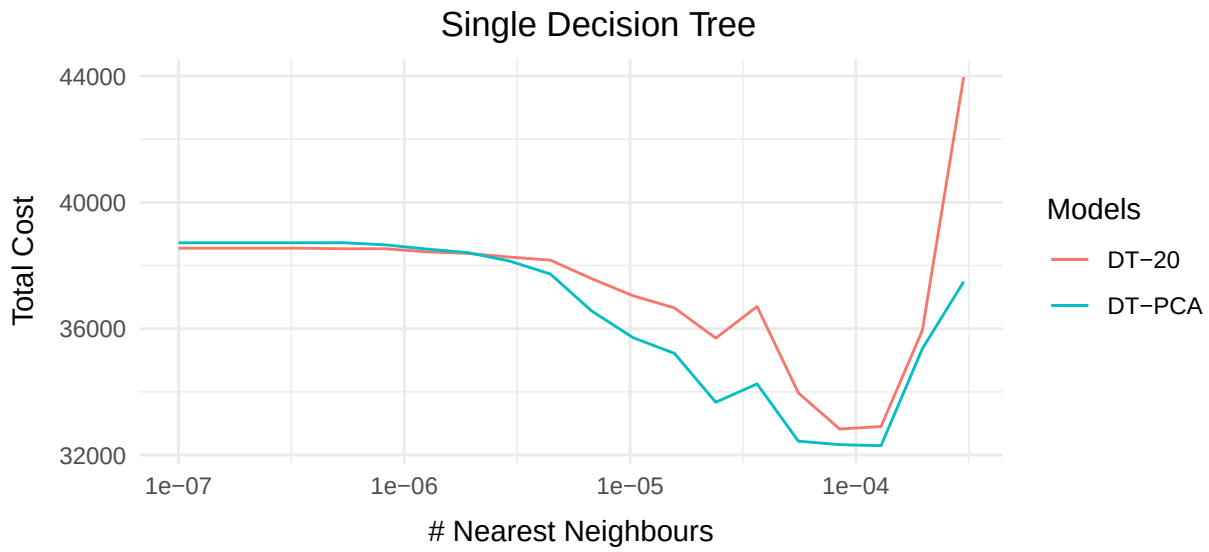

Figure 9: **Figure 9. Single Decision Tree Hyperparameter Tuning Results**

Random Forest

Random forest is an ensemble of decision trees which are trained using the bagging method. The general idea is to use bootstrapped samples, which introduces variability on the training data, to train multiple simple trees. It then aggregates the votes from the trees to decide on the final class of the test sample. Random forest improves the bagging model by further decorrelating the trees (using a random subset of different features for each split). This will generate a more diverse and decorrelated set of trees, as most trees will no longer use the same dominant predictors at earlier splits. One of the drawbacks however, is that the resulting trees can no longer be visualised, hence it is less interpretable than a single tree.

Dataset	Hyperparameters	Best Parameters
Truck20	mtry = c(3,5,7)ntree = c(700, 900, 1100, 1500, 1800)maxnodes = 38:42	mtry = 5ntree = 1100maxnodes = 42
PCA	mtry = c(3,5,7)ntree = c(700, 900, 1100, 1500, 1800)maxnodes = 38:42	mtry = 3ntree = 1100maxnodes = 42

Table 4. Random forest Hyperparameter Tuning

We applied random forest in caret using method= 'rf'. As seen above, the parameters we have tuned are maxnodes, which sets the maximum number of terminal nodes in the forest; mtry, which is the number of variables randomly sampled as candidates at each split (a smaller value will be more effective if a lot of predictors are correlated); and ntree, which is the number of trees grown in the random forest. Increasing the number of trees does not always result in overfitting as averaging the prediction from a large number of trees actually produces a more stable output which reduces the variance,

though it makes the training considerably slower. The cost results of the hyperparameter tuning is shown below.

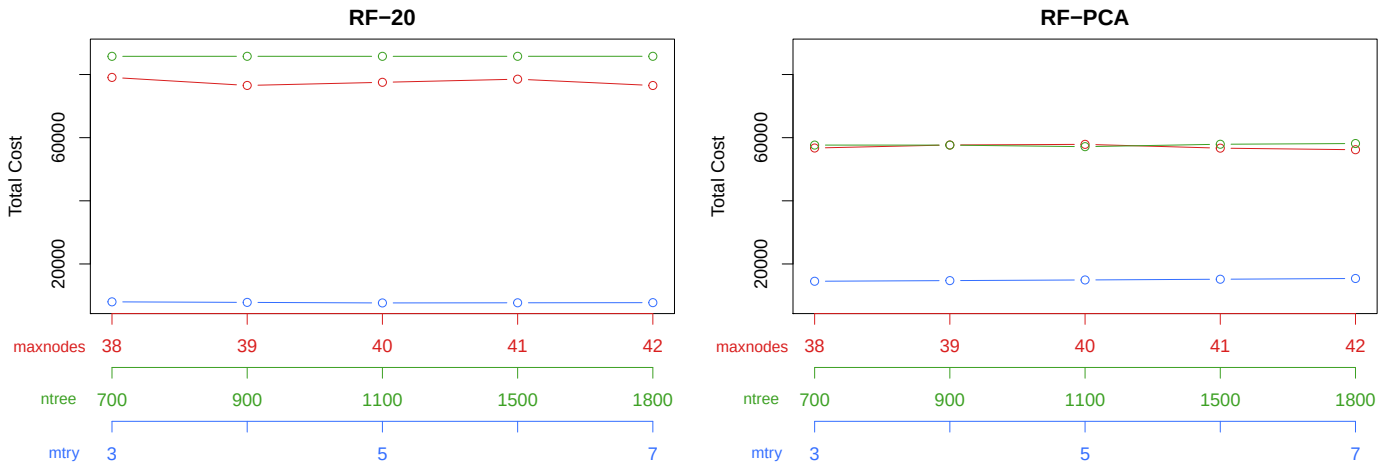

Figure 10: **Figure 10. Random Forest Hyperparameter Tuning Results**

Figure 10 above indicates maxnodes is the main hyperparameter that dictates the value of the Total Cost metric, while the Total Cost is mostly similar for the other hyperparameter values. In this case, increasing the maximum number of terminal nodes in the forest improves the validation results.

Training Results

Model	Run Time	Accuracy	F1-score	Sensitivity	Specificity	Mean Cost	Stdev Cost
LR-20	37.69 mins	0.8948	0.8895	0.8336	0.9580	2,589,100	10,069
LR-PCA	10.79 mins	0.9316	0.9308	0.9058	0.9582	1,472,050	6,446
LDA-20	0.50 min	0.8424	0.8223	0.7176	0.9713	4,381,110	12,145
LDA-PCA	0.19 min	0.8917	0.8840	0.8124	0.9736	2,915,930	9,746
KNN-20	20.78 mins	0.9915	0.9917	0.9994	0.9834	14,490	1,206
KNN-PCA	6.89 mins	0.9817	0.9823	0.9952	0.9678	83,660	2,782
DT-20	0.36 mins	0.9799	0.9803	0.9864	0.9732	218,540	15,552
DT-PCA	0.69 mins	0.9690	0.9699	0.9817	0.9560	297,210	13,695
RF-20	18.12 hours	0.9347	0.9358	0.9374	0.9319	990,420	5,767
RF-PCA	4.13 hours	0.9512	0.9523	0.9585	0.9436	659,920	2,818

Table 5. Training results summary

The best performing model when evaluating on the training data is kNN with the Truck20 dataset. From Table 5, not only did kNN perform well on the Total Cost metric, which is the amount we are trying to minimize, it also outperformed other models on all metric scores, yielding the highest accuracy, F1-score, sensitivity, specificity and standard deviation of Total Cost for both datasets. This means the kNN model was able to predict both the negatives and positives with high accuracy on the training data. The best hyperparameter for KNN is k=3 however, which is quite low. By using only three neighbours to predict the classes of new data, it is likely that it may have overfit the training data, given the test results below.

The highest total costs were from logistic regression and LDA, both of which generate linear decision boundaries. Given the performance of the flexible non-parametric models (kNN, Decision Tree and Random Forest) compared favourably to the parametric models (logistic regression and LDA), it can be surmised that the dataset is quite complex and not in the forms assumed by the two parametric models used.

Test Results

Model	Accuracy	F1-score	Sensitivity	Specificity	ROC-AUC	Total Cost
LR-20	0.9571	0.4853	0.8613	0.9594	0.9582	32,330
LR-PCA	0.9588	0.5126	0.9253	0.9596	0.9818	20,320
LDA-20	0.9648	0.4973	0.7413	0.9702	0.9433	53,150
LDA-PCA	0.9713	0.5746	0.8267	0.9748	0.9733	36,440
KNN-20	0.9552	0.4453	0.7653	0.9598	0.8933	50,270
KNN-PCA	0.9579	0.4791	0.8267	0.9610	0.9295	38,590
DT-20	0.9502	0.4329	0.8080	0.9537	0.9439	43,220
DT-PCA	0.9489	0.4611	0.9333	0.9492	0.9572	20,430
RF-20	0.9324	0.3969	0.9467	0.9320	0.9780	20,590
RF-PCA	0.9460	0.4552	0.9627	0.9456	0.9806	15,500

Table 6. Test results summary

In order to generalize the model, we didn't upsample or downsample the test dataset, so the model should behave properly with unseen data, which is highly imbalanced. Table 6 shows that all models produced reasonably high accuracy and specificity on the test datasets. As expected, these models were able to detect the negative classes with high accuracy (>0.93). However, they all produced lower scores when predicting the positive classes, resulting in low F1-scores (<0.60). Since the data is highly imbalanced, it is expected that the abundance of "neg" classes would contribute a high proportion of false positives if it makes a wrong prediction as compared to the true positive data, thus making the F1-Score [F1 = TP / (TP + 0.5(FP+FN))] a sub-optimal value. However, given Scania estimated that the cost of missing a faulty truck is higher than the cost of an unnecessary check by a mechanic, we consider the low F1-scores to be quite trivial here.

The model with the lowest test cost was random forest with the PCA dataset. The ensemble of weakly correlated decision trees reduced the variance of the random forest prediction. Choosing a good set of hyperparameters may have also helped it to have an acceptable amount of bias. As would be expected, having a large number of trees did not lead to overfitting. Whilst random forest did not have the highest score for accuracy, F1-score, or specificity, it did the best on sensitivity. To recall our goal, which is to minimize the Total Cost [Total Cost = 10FP + 500FN], it implies that the cost is mostly influenced by the number of False Negatives. Thus, having the best sensitivity score [Sensitivity = TP / (TP+FN)] is also attributed to lowering the total cost as the sensitivity increases as the number of false negatives decrease.

All Metrics

Figure 11 shows a parallel coordinate plot for each of the models and datasets, evaluated by the accuracy, F1-score, sensitivity, specificity and the ROC-AUC. All scenarios give high accuracy due to the imbalanced distribution of classes. As previously noted however, the F1 scores are low for all models. This is because they are inversely proportional to the Total Cost, which is unsurprising given the class imbalance.

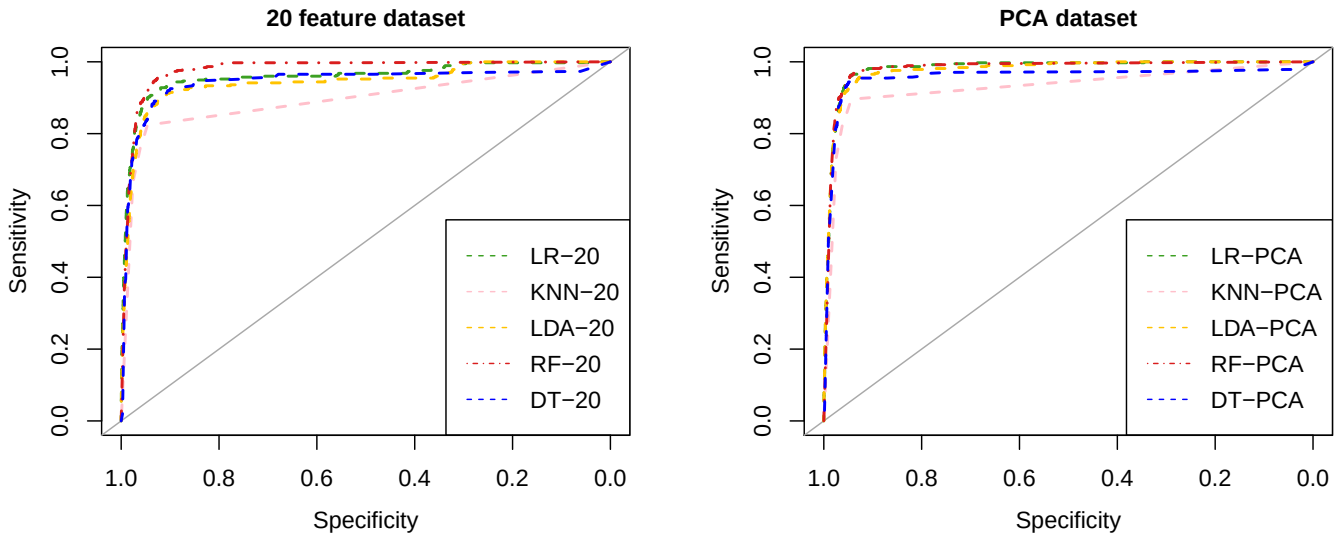

Figure 11: **Figure 11. ROC curve comparing all models**

In Figure 12, the ROC curve on Sensitivity vs Specificity shows that all models have similarly high specificity (false positive rate), but vary on the sensitivity (true positive rate) due to the imbalanced data. The two kNN models in particular, showed noticeably lower sensitivity compared to other models. This confirms our earlier finding that, using k=3, kNN may have overfit the training data, hence performed poorly on the test set.

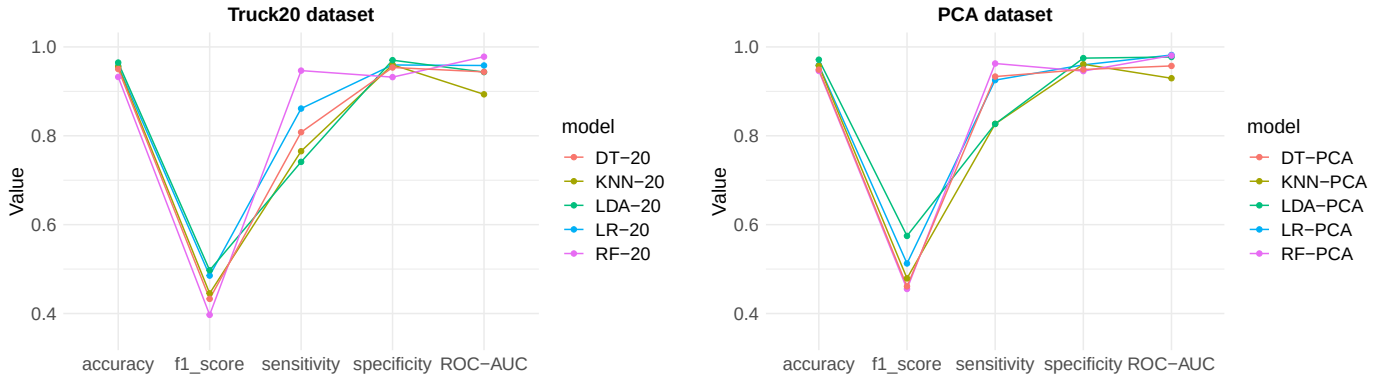

Figure 12: **Figure 12. APS test data metrics comparison**

Lastly, the primary metric being considered is the total cost, which also best shows the difference between the two datasets. The total costs for each model and dataset are plotted in Figure 13 below.

Figure 13: **Figure 13. APS test data total cost metric comparison**

For each model, the PCA dataset produced a lower total cost compared to the Truck20 dataset. When performing data cleaning on the Truck20 dataset, many columns were dropped to reduce the number of features and remove highly correlated features. In doing so, we may have thrown away valuable information or accidentally introduced bias into the data (Wishart, 2020). With the PCA dataset on the other hand, we were able to reduce the number of features significantly by representing the data in a different space using only the first six principle components, whilst still retaining over 95% of the variability in the data. For this reason, the PCA models seemed to have performed better than the Truck20 models.

Conclusion

For both datasets, random forest gives the best total cost, although it is worth noting that the time for training and predicting the random forest models (shown in Table 5) were also significantly higher than the others. Depending on the application of this model by Scania, this may or may not be acceptable. If this system is to be used in each individual truck as it is running in real time and new samples come in quickly, the time to generate a prediction may not be acceptable. Both logistic regression and decision tree had reasonably low total costs with lower runtime. However, the standard deviation of the total costs across the folds for decision tree is notably high (see Table 5), meaning it may not reliably produce good predictions on unseen data. Therefore, in this case, logistic regression is a better option, offering a respectable total cost at a fraction of the runtime, while still maintaining low uncertainty in the cross validation results.

References

1. Scania CV AB 2016, APS Failure at Scania Trucks Data Set, https://archive.ics.uci.edu/ml/datasets/APS+Failure+at+Scania+Trucks
2. James, Witten, Hastie, and Tibshirani 2013, An Introduction to Statistical Learning with Applications in R, Springer New York Heidelberg Dordrecht London
3. Honaker, King and Blackwell 2019, AMELIA II: A Program for Missing Data, version 1.7.6, https://cran.r-project.org/web/packages/Amelia/vignettes/amelia.pdf
4. Wishart 2020, Week 7: Missing Data and class imbalance, Semester 2, 2020, University of Sydney
5. van Buuren, Stef 2018, Flexible Imputation of Missing Data, viewed 19 October 2020, https://stefvanbuuren.name/fimd/sec-cart.html
6. Therneau and Atkinson 2019, An Introduction to Recursive Partitioning Using the RPART Routines, Mayo Foundation, https://cran.r-project.org/web/packages/rpart/vignettes/longintro.pdf

-