APS Failure and Operational Data for Scania Trucks

Introduction

- The dataset consists of data collected from heavy Scania trucks in everyday usage.
- The system in focus is the Air Pressure system (APS) which generates pressurized air that is utilized in various functions in a truck, such as braking and gear changes.
- The dataset's positive class consists of component failures for a specific component of the APS system
- The negative class consists of trucks with failures for components not related to the APS
- Our goal – design classifiers

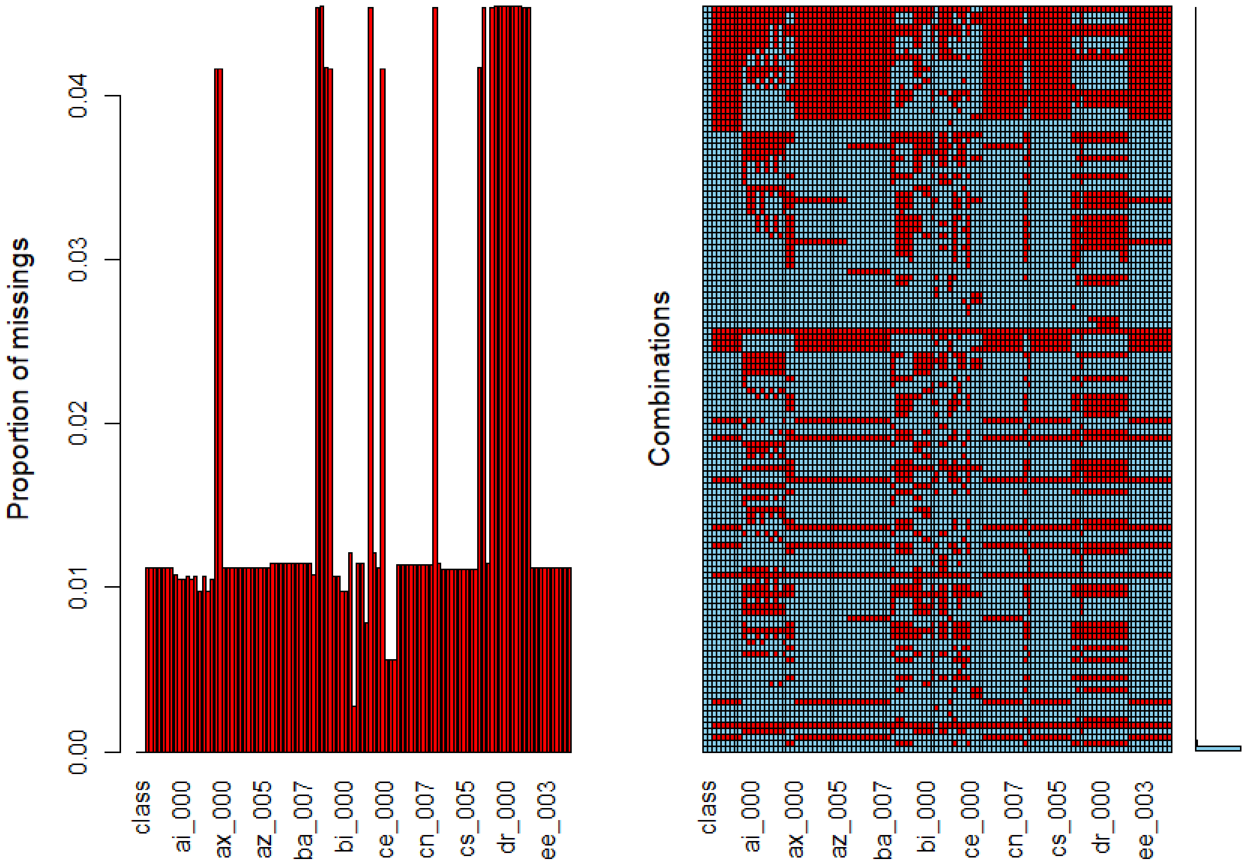
Preprocessing

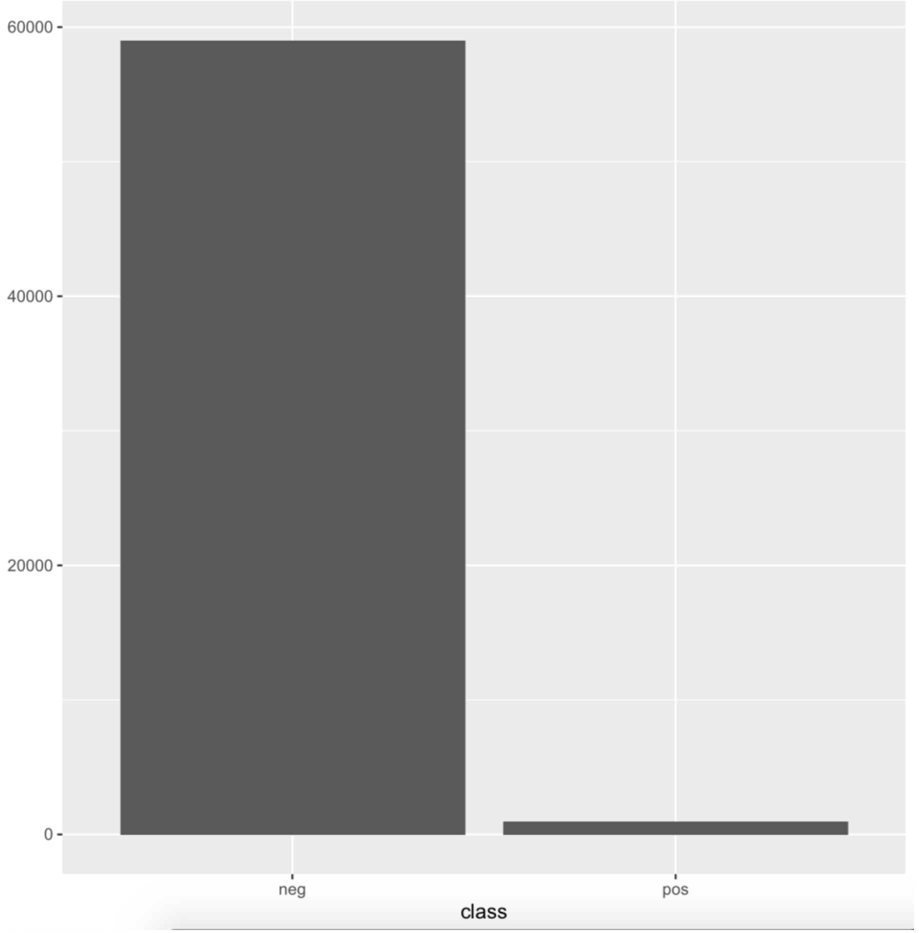
Remove **columns where over 5% of instance values are** missing values

Remove **columns where over 90% of instance values are zero**

Change missing values to the mean of the column

Data Visualization





Train & Test Split

Split the dataset into 60% training and 40% test with stratified split

```
# Split dataset 60% train and 40 % test
set.seed(3456)
trainIndex <- createDataPartition(data$class, p = .6, list = FALSE, times = 1)
train <- data[ trainIndex,]
test <- data[-trainIndex,]
```

Methods

NAÏVE BAYES

RANDOM FOREST

LOGISTIC REGRESSION

MULTILAYER PERCEPTRON

DECISION TREE

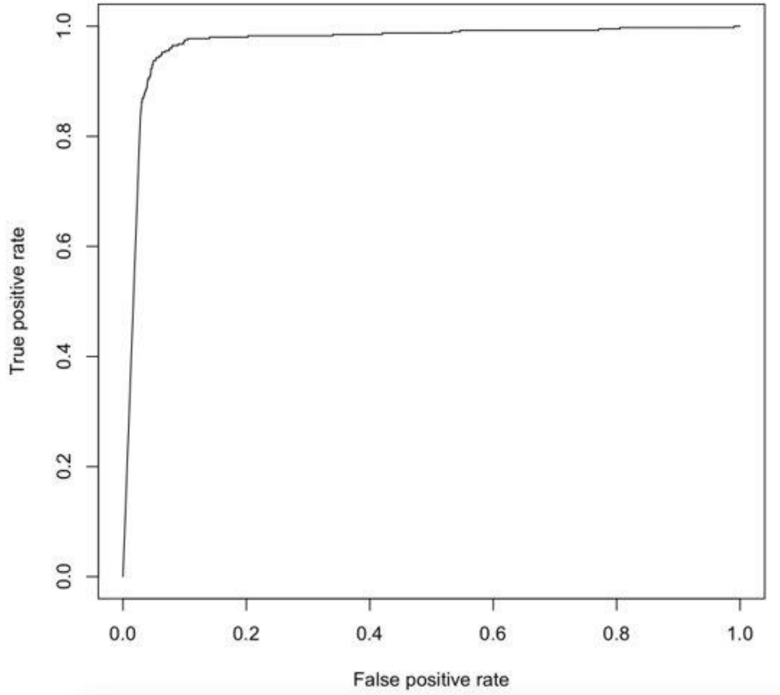
Methods
Naïve Bayes

```
Naive Bayes Classifier for Discrete Predictors

Call:
naiveBayes.default(x = X, y = Y, laplace = laplace, probability = TRUE)

A-priori probabilities:
Y
       neg        pos 
0.98333333 0.01666667 

Conditional probabilities:
     aa_000
Y          [,1]     [,2]
  neg  49241.67 110144.7
  pos 649999.28 421728.5

     ae_000
Y          [,1]      [,2]
  neg  5.995908  99.62148
  pos 16.829267 228.26675

     af_000
Y         [,1]     [,2]
  neg 10.05033 163.5419
  pos 54.52735 641.7894
```

Results

Naïve Bayes

Accuracy rate:	96.59%
Recall:	82.25%
Precision:	31%
F1_score:	45.46%
Total cost:	37090



Predict\Actual	True	False
Positive	341	759
Negative	59	22841

Naïve Bayes

Strength :

1. Assume all predictors are independent --> Fast
2. Able to handle high-dimensional data

Weakness:

1. NOT all predictors are independent --> less accurate

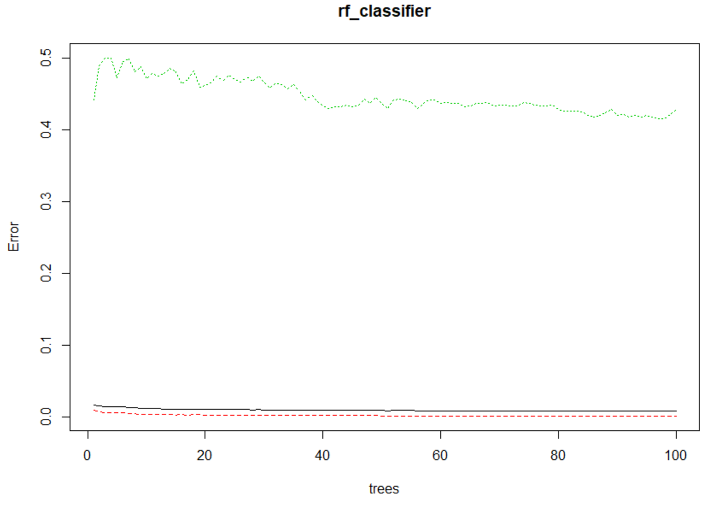
Methods
Random Forest

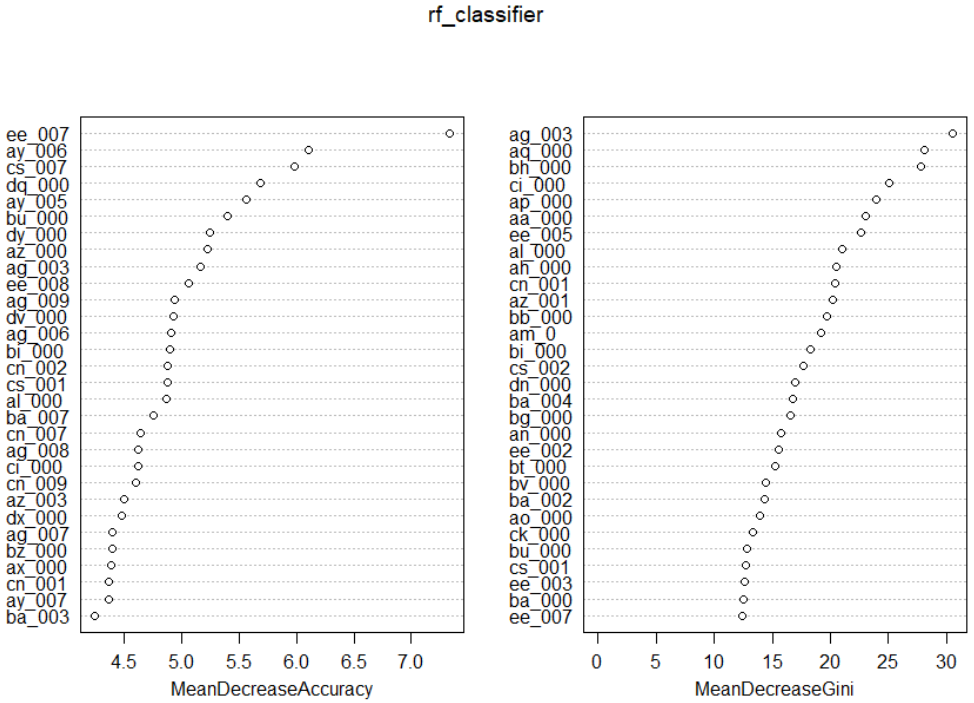
```
> print(rf_classifier)

Call:
 randomForest(formula = class ~ ., data = train, ntree = 100,      mtry = 2, importance = TRUE)
               Type of random forest: classification
                     Number of trees: 100
No. of variables tried at each split: 2

        OOB estimate of  error rate: 0.87%
Confusion matrix:
      neg pos class.error
neg 35344  56 0.001581921
pos   257 343 0.428333333
```

Results

Random Forest

Accuracy rate : 99.17 %
Recall : 60 %
Precision : 86.02 %
F1-Score : 70.69 %
Total cost : 80390





Confusion matrix

	Predictions	
Targets	2 (Positive)	1 (Negative)
2 (Positive)	240	39
1 (Negative)	160	23561

Random Forests

Strength :

1. Can handle very large dataset relatively faster
2. Output importance of variable
3. Random forest can solve both type of problems that is classification and regression
4. Maintains accuracy when large proportion of the data are missing

Weakness:

1. Very little control on what the model does

Methods
Logistic Regression

```
Call:
glm(formula = class ~ ., family = binomial("logit"), data = train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-5.0793  -0.0670  -0.0583  -0.0542   4.3944

Coefficients: (1 not defined because of singularities)
              Estimate Std. Error z value Pr(>|z|)
(Intercept) -6.423e+00  1.536e-01 -41.814  < 2e-16 ***
aa_000       1.351e-05  3.594e-06   3.759 0.000170 ***
ag_003       4.724e-07  6.752e-07   0.700 0.484152
ag_004      -1.101e-06  5.798e-07  -1.900 0.057481 .
ag_005      -8.128e-07  5.820e-07  -1.397 0.162526
ag_006      -8.538e-07  5.825e-07  -1.466 0.142754
ag_007      -7.789e-07  5.792e-07  -1.345 0.178671
ag_008      -9.641e-07  6.327e-07  -1.524 0.127604
ag_009      -7.809e-07  6.026e-07  -1.296 0.194991
ah_000       5.273e-07  1.939e-06   0.272 0.785634
ai_000       4.531e-07  1.412e-07   3.209 0.001331 **
aj_000      -1.173e-06  7.394e-07  -1.586 0.112668
al_000       5.268e-07  7.997e-07   0.659 0.510051
am_0        -1.302e-08  3.065e-07  -0.042 0.966123
```

Results

Logistic Regression

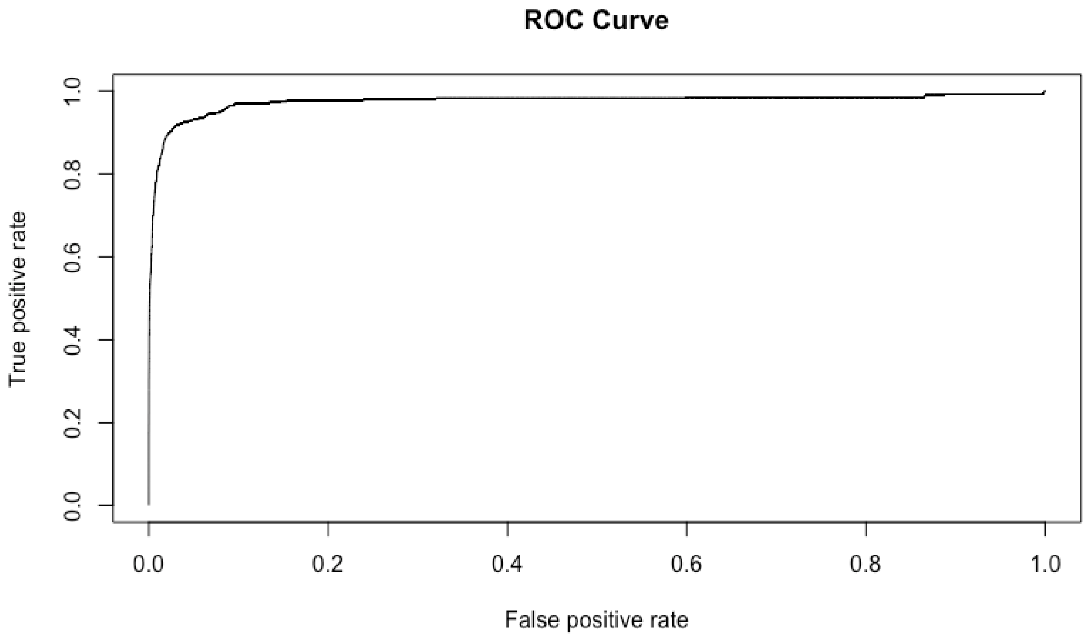
Accuracy rate: 99.05%

Recall: 62.5%

Precision: 76.45%

F1_score: 68.78%

Total cost: 75770



Predict\Actual	Negative	Positive
Negative	23523	150
Positive	77	250

Logistic Regression

STRENGTHS

1. More informative output than others
2. Efficient to train
3. No scaling required

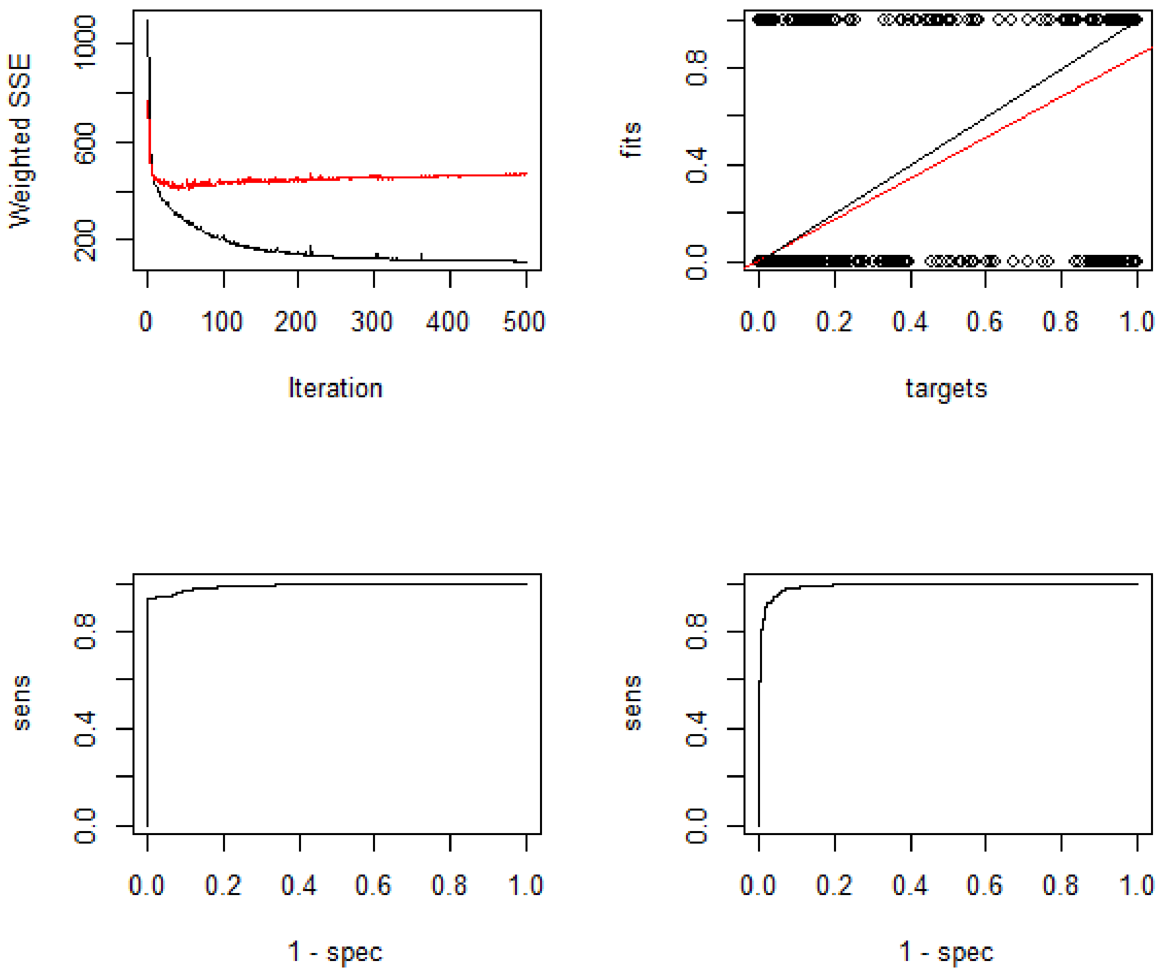
WEAKNESS

1. Assumption of linearity
2. Independent observations required
3. Used to predict discrete functions
4. Overfitting the model

Methods
Multi-Layer Perceptron

```
> print(mlpModel)
Class: mlp->rsnns
Number of inputs: 106
Number of outputs: 2
Maximal iterations: 500
Initialization function: Randomize_Weights
Initialization function parameters: -0.3 0.3
Learning function: Std_Backpropagation
Learning function parameters: 0.02
Update function:Topological_Order
Update function parameters: 0
Patterns are shuffled internally: TRUE
Compute error in every iteration: TRUE
Architecture Parameters:
$size
[1] 10

All members of model:
 [1] "nInputs"              "maxit"                  "initFunc"       "initFuncParams"
 [5] "learnFunc"            "learnFuncParams"        "updateFunc"     "updateFuncParams"
 [9] "shufflePatterns"      "computeIterativeError"  "snnsObject"     "archParams"
[13] "IterativeFitError"    "IterativeTestError"     "fitted.values"  "fittedTestValues"
[17] "nOutputs"
```

Methods
Multi-Layer Perceptron Visualization



Results

Multi-Layered Perceptron

Accuracy rate: 99.27%

Recall: 79.38%

Precision: 69.97%

F1_score: 74.38%

Total cost: 34090

Predict/Actual	Negative	Positive
Negative	23571	66
Positive	109	254

Multi-Layered Perceptron

STRENGTHS

1. Ideal for complex training problems
2. Each layer has adaptive weights
3. All attribute values could be put into numeric values which made it easy to model
4. Suited for large number of data points
5. Once modelled, quick to execute predictions

WEAKNESS

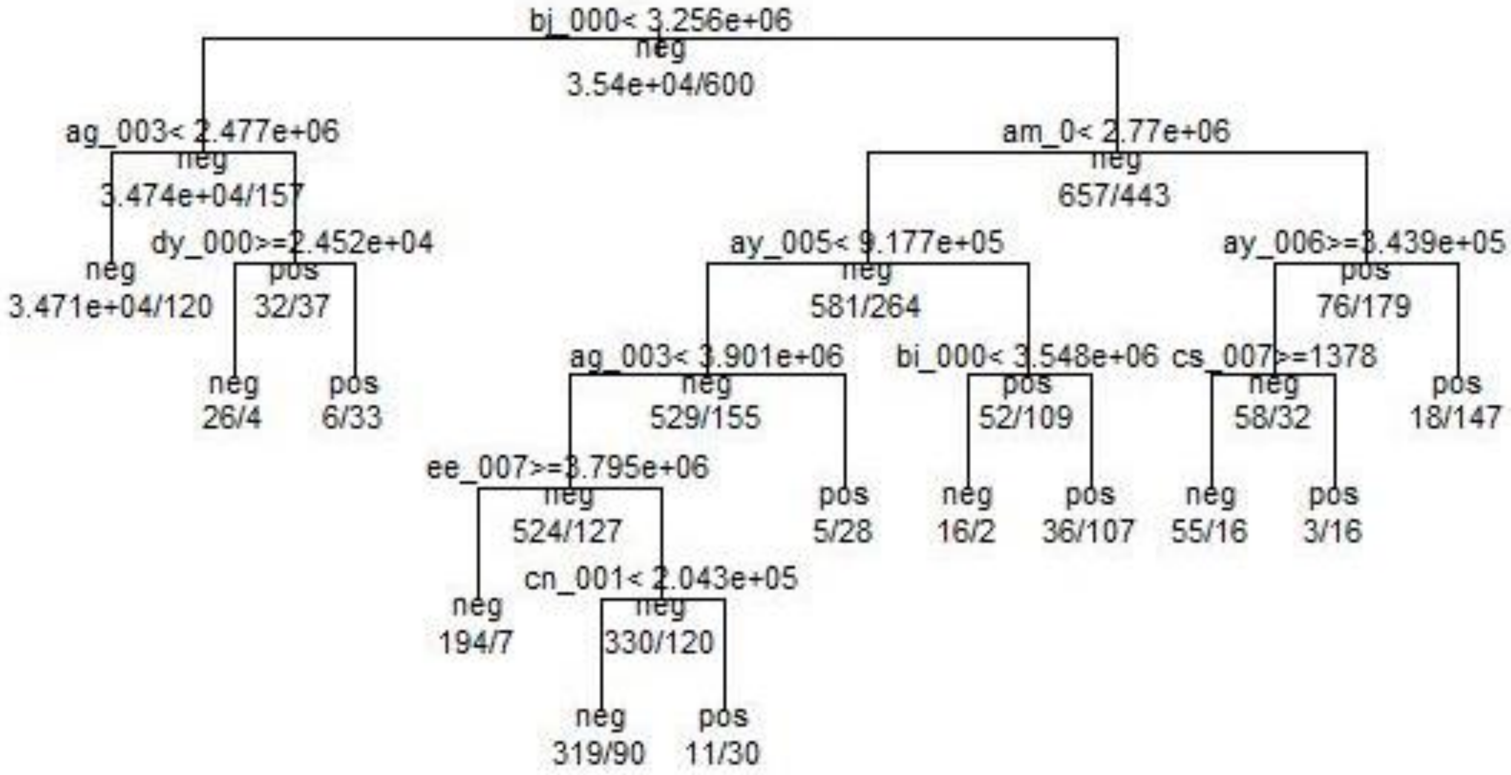
1. More iterations mean more training time needed.
2. Data needs to be inputted in a specific way.
3. Relies heavily on the training data.
4. A blackbox approach

Decision Tree

Fit decision tree to training dataset

```
dtree_classifier <- rpart(class ~.,
data = train, method = "class")

print(dtree_classifier)
```

```
node), split, n, loss, yval, (yprob)
      * denotes terminal node

 1) root 36000 600 neg (0.983333333 0.016666667)
   2) bj_000< 3255874 34900 157 neg (0.995501433 0.004498567)
     4) ag_003< 2476919 34831 120 neg (0.996554793 0.003445207) *
     5) ag_003>=2476919 69  32 pos (0.463768116 0.536231884)
      10) dy_000>=24518 30   4 neg (0.866666667 0.133333333) *
      11) dy_000< 24518 39   6 pos (0.153846154 0.846153846) *
   3) bj_000>=3255874 1100 443 neg (0.597272727 0.402727273)
     6) am_0< 2770482 845 264 neg (0.687573964 0.312426036)
      12) ay_005< 917691 684 155 neg (0.773391813 0.226608187)
        24) ag_003< 3901484 651 127 neg (0.804915515 0.195084485)
          48) ee_007>=3794501 201   7 neg (0.965174129 0.034825871) *
          49) ee_007< 3794501 450 120 neg (0.733333333 0.266666667)
            98) cn_001< 204310 409  90 neg (0.779951100 0.220048900) *
            99) cn_001>=204310 41  11 pos (0.268292683 0.731707317) *
        25) ag_003>=3901484 33   5 pos (0.151515152 0.848484848) *
      13) ay_005>=917691 161  52 pos (0.322981366 0.677018634)
        26) bi_000< 3547695 18   2 neg (0.888888889 0.111111111) *
        27) bi_000>=3547695 143  36 pos (0.251748252 0.748251748) *
     7) am_0>=2770482 255  76 pos (0.298039216 0.701960784)
      14) ay_006>=343861 90  32 neg (0.644444444 0.355555556)
        28) cs_007>=1378 71  16 neg (0.774647887 0.225352113) *
        29) cs_007< 1378 19   3 pos (0.157894737 0.842105263) *
      15) ay_006< 343861 165  18 pos (0.109090909 0.890909091) *
```



Visualizing Decision Tree

Results

Decision Tree

Accuracy rate: 98.97 %
Recall: 58.25 %
Precision: 74.20 %
F1_score: 65.26 %
Total cost: 84310

Predict/Actual	Negative	Positive
Negative	23519	167
Positive	81	233

Decision Tree Approach

STRENGTHS

1. Does not require normalization of data
2. Does not require scaling of data
3. Missing values in the data also does not affect the process of building decision tree
4. Intuitive and easy to explain to stakeholders

WEAKNESS

1. Higher time to train the model
2. More Complex
3. Relatively expensive

Discussion & Conclusion

Method	**Accuracy rate**	**F1-Score**	**Cost**
Naïve Bayes	96.59%	45.46%	37090
Random Forest	99.17%	70.69%	80390
Logistic Regression	99.05%	68.78%	75770
Multilayer Perceptron	**99.27%**	**74.38%**	**34090**
Decision Tree	98.97%	65.26%	84310

References

1. Medium. 2020. *Top 5 Advantages And Disadvantages Of Decision Tree Algorithm*. [online] Available at: <https://medium.com/@dhiraj8899/top-5-advantages-and-disadvantages-of-decision-tree-algorithm-428ebd199d9a> [Accessed 10 June 2020].

2. Kumar, N., 2019. *Advantages And Disadvantages Of Logistic Regression In Machine Learning*. [online] Theprofessionalspoint.blogspot.com. Available at: <http://theprofessionalspoint.blogspot.com/2019/03/advantages-and-disadvantages-of.html> [Accessed 10 June 2020].