Stockholms universitet

Hiding in the Trees

A case study of filtering approaches to Insurance fraud classification using tree based Gradient Boosting

Oliver Murquist

Masteruppsats i försäkringsmatematik
Master Thesis in Actuarial Mathematics

Masteruppsats 2019:11
Försäkringsmatematik
September 2019

www.math.su.se

Matematisk statistik
Matematiska institutionen
Stockholms universitet
106 91 Stockholm

Matematisk statistik
Stockholms universitet
Masteruppsats **2019:11**
http://www.math.su.se/matstat

Hiding in the Trees

A case study of filtering approaches to Insurance fraud classification using tree based Gradient Boosting

Oliver Murquist*

September 2019

Sammanfattning

With easy to access complex machine learning methods that are powerful right out of the box, the risk of misuse and overuse increases. Are simpler methods like Gradient Boosted trees and Random Forests intuitive and flexible enough to compete? Previous studies made on the subject indicate that slow learning models produce the best performance in terms of prediction and robustness. The results of this study were partly contradictory to that. A gradient boosting model configured to be a fast learner was found objectively better than the other models fitted, however most of that success could be tied to model complexity rather than learning rate. Additionally both gradient boosted trees and random forests displayed great difficulties handling imbalanced data. Overall the analysis concluded that a considerable portion of performance could be attributed to tailoring the model configuration to the outcome of an initial analysis as well as to the specific application environment.

*Postadress: Matematisk statistik, Stockholms universitet, 106 91, Sverige. E-post:oliver.murquist@gmail.com. Handledare: Chun-Biu Li.

Acknowledgements

Primarily I would like to extend my sincere gratitude to my supervisor Chun-Biu Li who has provided me with constructive criticism and motivation during this project, as well as encouraging me to return and finish the thesis after it was put on hold. I would also like to thank my family and friends for supporting me the whole way through.

Contents

1 Introduction

Decision tree diagrams, they are intuitive, easy to interpret and apply. Compared to more advanced machine learning methods, however, what they gain in simplicity they lose in performance. Combined with the recent surge in popularity of deep learning and Artificial Intelligence as well as enormous progress for "out of the box"-application of more advanced models, tree based methods are easily overlooked. The general consequence of this is a diminishing need to understand the underlying mathematics and an increased risk of misuse among business applications. In this thesis we will consider one application where we deem this risk to be noteworthy, namely Insurance Fraud detection. On the surface the setting is simple binary classification of fraudulent vs legitimate claims, with many intuitive and easily measurable feature variables. Underneath it is much more complex simply because the nature of fraud is to avoid detection, as apart from classification of weather or disease. The cherry on top is that the corporate interests in this particular problem are almost solely monetary, which moves focus even more toward model performance. According to the report "Försäkringsbedrägerier i Sverige 2017"[9] made by Svensk Försäkring, it is estimated that general insurance fraud costs $3-6$ billion SEK per year. The rate of fraud is estimated to be $5-10\%$ yet only 500 million SEK in payouts were rejected as a result of fraud-investigation. These investigations are generally done manually by case workers, and as such, out of the total 3 million claims in 2017, only 7000 were investigated. In this thesis we will consider two strategies to implement a screening process that all claims go through in order to increase the rate of fraudulent claims in the subset that in turn is investigated. For a more controlled environment we will narrow our scope to automotive insurance fraud since that category is estimated to covering 45% of all insurance fraud.

2 Theory

In this section the theory and foundation that the methods used are built upon is described. In some cases expansions of specific parts of interest for a given method is also provided. The order in which we define things is based on the hierarchy of background knowledge required to define gradient boosted trees. Since the application focuses on binary classification the theory will mostly cover that specific case.

2.1 Machine learning

During the 1980s and primarily 1990s a general influx of data driven statistical models made its way into mainstream mathematical statistics. Many of the entry level machine learning methods like *random forests*, *boosted trees* and *support vector machines* where either invented or greatly improved upon during this time. The change of focus was probably due to a general increase in ways to acquire and process large amounts of data with computers. A major interest for scientists was the ability to analyse, draw conclusions and generally "learn" from these large data-sets without any significant prior knowledge of the event or situation they described.

Basic *classification trees* are most commonly traced back to "Classification and Regression Trees"[2] from 1984 by Brieman et al. and were used due to being adaptive, non-parametric and providing easily interpretable results. Consequently the model structure was sensitive to changes in data and thus suffered from high variance. Expansions like Tin Kam Ho's "Random Decision Forests"[7] in 1995 and Jerome H. Friedman's "Stochastic Gradient Boosting"[3] in 1999 greatly improved accuracy and generality of tree based methods while only slightly impairing interpretability, the latter more so than the former. While those papers are the original works, our core reference for theory regarding those machine learning methods is "The Elements of Statistical Learning"[4] from 2017 by Hastie, Tibshirani and Friedman, as well as "An Introduction to Statistical Learning"[5] from 2013 by James, Witten, Hastie and Tibshirani.

2.2 Classification Trees

Figure 1 depicts a classification tree. The top displays our data consisting of two features variables X and Y as well as a class Color. This tree partitions the feature space, and subsequently data, into a set of four "terminal" regions $\{R_m; 1 \leq m \leq 4\}$ illustrated at the bottom of the figure. Classification of a new observation is done decending down the tree and by taking most frequent class

of observations in the resulting region, as depicted by the color of each box in Figure 1. If there is a draw the overall most frequent class is chosen.

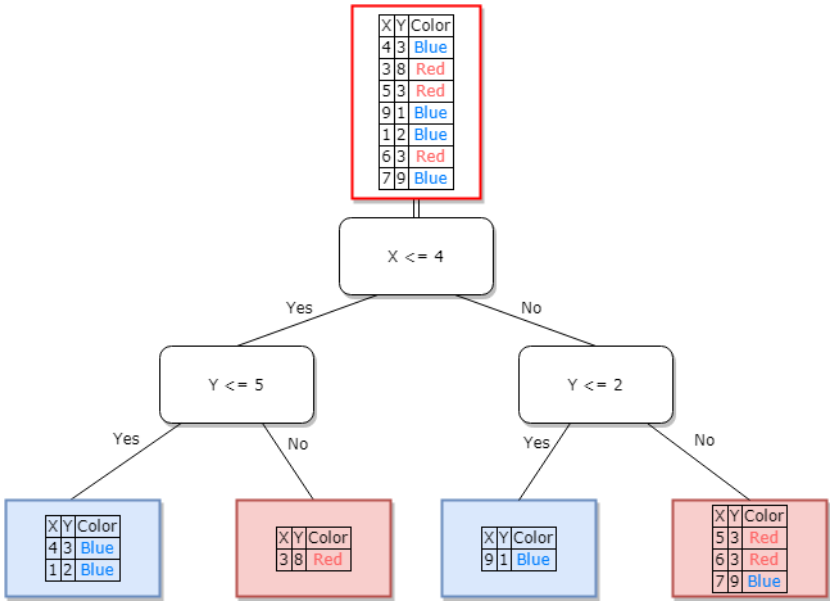

Figure 1: An illustrated example of a decision tree with depth 2 and 4 "terminal regions".

For a more detailed description we first need to generalise the base assumptions given in the previous example. For this section we will restrict ourselves to binary classification, meaning that an observation can belong to one of two classes 1 or 0. Consider a set of N observations of a stochastic variable X with l features, denoted as $x_i = (x_{i,1}, x_{i,2}, ..., x_{i,l})$ $i = 1, 2, ..., N$. Each observation belongs to a true class k_i from the set of classes $\{1, 0\}$. We use notation R for a continuous l-dimensional sub-region of the feature space and N_R as the number of observations in data contained in R.

For any region R we can perform three main actions. Firstly, we can perform a binary split of feature j at value s if j is continuous, or at subset s if j is a factor feature. The split divides the region in two and is defined by the resulting sub-regions

$$\text{Continuous feature: } R_-(j,s) = \{X|X_j \leq s\} \text{ and } R_+(j,s) = \{X|X_j > s\}$$
$$\text{Factor feature: } R_-(j,s) = \{X|X_j \in s\} \text{ and } R_+(j,s) = \{X|X_j \notin s\}.$$

Secondly, we can calculate the ratio of observations in R with true class k by

$$p_k = \frac{1}{N_R} \sum_{x_i \in R} I(k_i = k).$$

Thirdly, we can use a function of p_k to measure how evenly classes are distributed in the region. This is called (class)impurity and has an inverse relation to class homogeneity in that region. There are a few different functions that can be used, but in this thesis we will use Entropy which for multiple classes is defined as

$$Q(R) = -\sum_k p_k \log(p_k).$$

In our binary case we have the relation $p_0 = 1 - p_1$ and thus

$$Q(R) = -(p_1 \log(p_1) + (1 - p_1)\log(1 - p_1)).$$

This is a symmetric function with respect to p_1, since $0 \leq p_1 \leq 1$, as is visible in Figure 2.

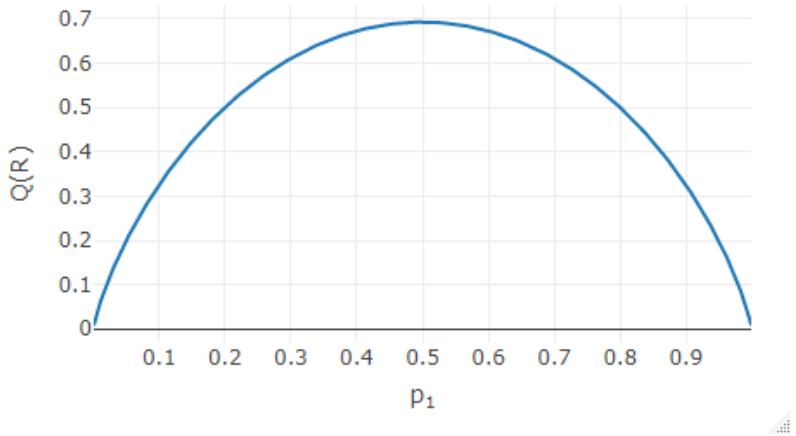

Figure 2: (Binary)Entropy of a region with respect to p_1. The impurity of a region strictly increases when moving toward even class distribution within said region.

The impurity measure has its minimum value when maximum separation is attained and only one class is present in the region, i.e. $p_1 = 1$ or 0. Conversely the maximum value is attained when classes are evenly distributed , i.e. $p_1 = 0.5$. As such this can be used to quantify impurity of any given region R.

Utilising these tools we want to construct a partition of M "terminal" sub-regions of the feature space, $\{R_1, R_2, ..., R_M\}$, that minimises joint impurity. Optimally this would result in a set of regions where all observations in each belong to the same class. Why this is favourable becomes more evident later when defining how we estimate the true class of new observations.

A "greedy" algorithm to construct such a partition is initiated by assessing all possible splits of the complete feature space and performing the one resulting in lowest overall impurity. This process is repeated recursively for the two sub-regions, and in turn their sub-regions, so on and so forth. To prevent over-partitioning we can insert a stopping criterion such as a maximum value for M, or a minimum change in Entropy between R and the sub-regions created by splitting R.

Algorithm 1: Greedy Partitioning algorithm

1. Initialise with $R =$ the complete feature space.
2. Find binary split $(\hat{j}, \hat{s}) = \underset{(j,s)}{argmin} \left[\frac{Q(R_-)}{N_{R_-}} + \frac{Q(R_+)}{N_{R_+}} \right]$
3. (a) If stopping criterion is met, R is considered a "terminal" region.
   (b) Else repeat from 2 for each of $R = R_-$ and $R = R_+$.
4. If all remaining regions meet the stopping criterion the set of terminal regions is the desired partition.

This is a "greedy" optimisation in the sense that we only consider the immediate change in purity when selecting a split. The main downside occurs when data has one or more dominant feature variables. By definition these dominant features will provide immediate separation between classes, but run the risk of overshadowing more effective interactions between minor features. While it is possible to test future splits in advance in hopes of countering this behaviour. It quickly becomes computationally expensive for a single tree, let alone when fitting thousands at a time.

The last piece needed to turn a partition into a classification tree is a method of estimating the true class of new observations. This is more commonly referred to as classification.

For an arbitrary partition $\{R_1, R_2, ..., R_M\}$ of the feature space, the conditional

probability of an observation x_i having true class 1 given that $x_i \in R_m$ is

$$P(k_i = 1 | x_i \in R_m) = \frac{P(k_i = 1, x_i \in R_m)}{\sum_{k=0}^{1} P(k_i = k, x_i \in R_m)}$$

This can be estimated by the ratio p_1 of data points in R_m with true class 1

$$p_1 = \frac{1}{N_{R_m}} \sum_{x_j \in R_m} I(k_j = 1).$$

Like we previously stated, since there are only two classes we get relation $p_0 = 1 - p_1$. Classification of a new observation $x \in R_m$ is done using some classifier $\hat{f}(x)$, the most simple being majority vote which is defined as

$$\hat{f}(x) = \underset{k}{argmax}(p_k).$$

A flaw in this classifier can be illustrated by comparison to an alternative. Given a threshold value $0 \leq P_1 \leq 1$ we can define

$$\hat{f}(x) = \begin{cases} 1 & \text{if } p_1 > P_1 \\ 0 & \text{otherwise} \end{cases}.$$

Intuitively $P_1 = 0.5$ is equivalent to majority vote. A high threshold reduces the risk of incorrectly classifying an observation as having true class 1 at the cost of increased risk for the opposite. This is useful when your primary goal is to distinguish characteristics or observations of class 1. Conversely, a low threshold could be used initially to weed through large data-sets in preparation for an expensive but more accurate classifier.

In summary, construction of a classification tree is done by first partitioning the feature space and then applying a classifier to new observations dependent on those partitions. The end result is an estimation of the true class of each new observation.

2.3 Bagging

Bagging is an expansion of the previous procedure in which we apply the concept of bootstrap to reduce model variance. A classification tree fit using a subset of data(a training set) has stochastic error e when performing classification on the remaining data(a test set). Given a set of T trees each fitted to an independently sampled sub-set of data, the overall mean error of classifications would have reduced variance $\frac{Var(e)}{T}$. This is by property of the mean of T independent

equally distributed stochastic variables.

Since each tree constructs its own partition based on the sampled data provided, a mean of classifications cannot be calculated for set feature space region. With notation $R^t(x_i)$ to specify the region attained when descending tree number t using observation x_i. The conditional probability of x_i having true class 1 given that $x_i \in R^t(x_i)$ is

$$P(k_i = 1 | x_i \in R^t(x_i)) = \frac{P(k_i = 1, x_i \in R^t(x_i))}{\sum_{k=0}^{1} P(k_i = k, x_i \in R^t(x_i))}.$$

Similarly to the single tree case we can estimate this probability with

$$p_{t,1} = \frac{1}{N_{R^t(x_i)}} \sum_{x_j \in R^t(x_i)} I(k_j = 1)$$

where $p_{t,0} = 1 - p_{t,1}$. We are then left with a set of T trees, each with an estimated probability $\hat{P}(k_i = 1 | x_i \in R^t(x_i)) = p_{t,1}$. The majority vote classifier for Bagging gives the class which has the highest mean estimated probability

$$\begin{aligned} \hat{f}(x_i) =& \underset{k}{argmax} \left(\frac{1}{T} \sum_{t=1}^{T} \hat{P}(k_i = k | x_i \in R^t(x_i)) \right) \\ =& \underset{k}{argmax} \left(\frac{1}{T} \sum_{t=1}^{T} p_{t,k} \right). \end{aligned}$$

Given a probability threshold value $0 \leq P_1 \leq 1$ we can define the threshold classifier for bagging as

$$\hat{f}(x_i) = \begin{cases} 1 & \text{if } \frac{1}{T} \sum_{t=1}^{T} p_{t,1} > P_1 \\ 0 & \text{otherwise} \end{cases}.$$

2.4 Random Forest

While Bagging alters the tree classifier to be dependent on a set of T trees, it does not affect each individual partition in any way other than using different sampled subsets of data. Thus a set of dominant features may severely limit variability of each tree's feature space partition. Random forest is an expansion of the Bagging method in which only a random sample of features are considered when evaluating each split. This is illustrated in Figure 3, where we have three trees with three splits each and consider two out of three feature variables X, Y, Z for each split. This procedure only changes how the partition is constructed, and thus uses the same classifier as Bagging.

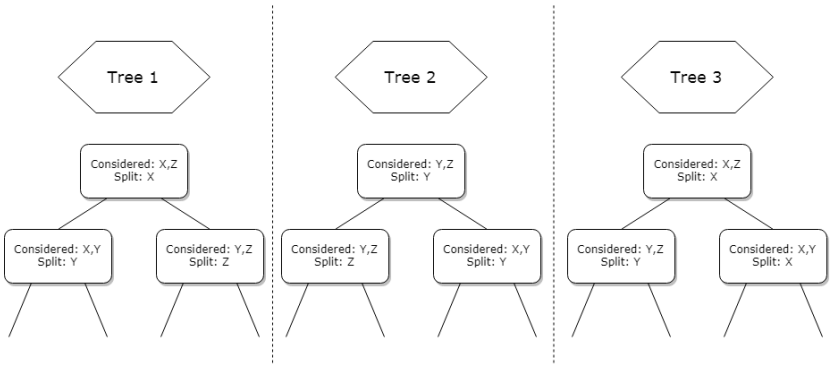

Figure 3: The above illustrates an example of how only considering a sample of features at each split can affect the tree structure.

The main drawback of these two bagging-based methods is that increasing the number of trees only reduces variance of the error and not the expected value of it. Despite being referenced to as machine learning methods, they do not learn in the same sense as other methods. The models do not take advantage of information about previously incorrect classifications for future tree fits. The main gain however, is that we do not risk over-fitting data when increasing number of trees.

2.5 Boosting

Consider terminology "weak learner" to mean a classifier that is slightly better than random guessing, and "strong learner" to mean one that is well correlated to the true classification. Boosting is an algorithm that combines a set of weak learners to create a strong one. To derive this algorithm for classification we start by considering a different way of expressing our goal.

Data is a set of N observations of a stochastic variable X with l features, denoted as $x_i = (x_{i,1}, x_{i,2}, ..., x_{i,l})$ $i = 1, 2, ..., N$. Each observation belongs to a true class k_i which is one of two possible classes. We wish to model the posterior probabilities $p_k(x_i) = P(k_i = k|X = x_i)$ while ensuring that their sum equals one and that they are constrained to the interval $[0, 1]$. After that we want to classify observations based on those probabilities. Apart from not specifying a condition that the model should be a linear function of x, our goals fit the theory of Logistic regression's binary case. In a quick summary, binary logistic regression works by fitting an additive function $F(x_i)$ to the logit transformed posterior probability of one class

$$\text{logit}(p_1(x_i)) = \log\left(\frac{p_1(x_i)}{1 - p_1(x_i)}\right) = F(x_i)$$

while using relation $p_2(x_i) = 1 - p_1(x_i)$ to calculate the same for the other class. Consider this setup and let $L(F)$ be an arbitrary differentiable loss function describing loss during classification of x_i using $F(x_i)$ on training data

$$L(F) = \sum_{i=1}^{N} L(k_i, F(x_i)).$$

Without a specified loss function the fact that we compare a class to a logit transformed probability might seem odd. We will come back to this when addressing the classification case of gradient boosting specifically.

Intuitively the loss function is a measurement of how bad a given classifier is. Our goal is then to minimise $L(F)$ but finding an analytical solution if possible is very difficult. However, the problem can be altered to be one of parameterisation instead by regarding a vector containing the logit probability of each observation, $\mathbf{F} = \{F(x_1), F(x_2), ..., F(x_N)\}$, as parameters for the loss function. The parameter optimisation problem becomes

$$\hat{\mathbf{F}} = \underset{\mathbf{F}}{argmin}(L(\mathbf{F})).$$

We can approach this numerically by employing an initial guess $\mathbf{F}_0 = \mathbf{h}_0$, and updating it stepwise with a vector $\mathbf{h}_t$ at each step according to

$$\mathbf{F}_T = \sum_{t=0}^{T} \mathbf{h}_t.$$

The numerical minimisation of $L(F)$ then boils down to selection of the incremental vector $\mathbf{h}_t$. Tracing back to the definition of boosting we see that this numerical approach illustrates the concept well. Using a set of weak learners $\{\mathbf{h}_t : t = 0, 1, ..., T\}$ we combine them using a sum to create a strong learner $\mathbf{F}_T$. So far we have re-formulated the initial true class estimation problem into a numerical estimation problem for the logit transformed class probabilities.

2.5.1 Gradient Boosting

We approach the selection of $\mathbf{h}_t$ by applying the concept of gradient decent. According to this concept the way to find a local minimum of a function is to stepwise move an arbitrary length ρ along the negative gradient of the function. Applying this to minimisation of $L(F)$ we move length ρ_t along the negative gradient of the loss function at each step, i.e. $\mathbf{h}_t = -\rho_t \mathbf{g}_t$. Components of the

gradient vector are defined as

$$g_{i,t} = \left[\frac{\partial(L(k_i, F_{t-1}(x_i))}{\partial(F_{t-1}(x_i))}\right]$$

and the step length is given by

$$\rho_t = \underset{\rho}{argmin}(L(\mathbf{F}_{t-1} - \rho \mathbf{g}_t)).$$

Following the previously stated numerical minimisation, the vector $\hat{\mathbf{F}}$ is updated at each step according to

$$\mathbf{F}_t = \mathbf{F}_{t-1} - \rho_t \mathbf{g}_t.$$

Taking a closer look at $\mathbf{F}_t$ at any given step t we note that it is a vector of estimates of $F(x_i)$ for each point in training data. As such it, and consequently also the gradient, is only defined for points in training data. Estimating the class probabilities for a new observation with unknown true class is therefore not possible with the current procedure. This is a major issue due to classification of new observations being the main reason we construct a model. However, like other functions it is possible to approximate the gradient with values close enough to it. A proposed solution to the issue is then to fit an inner model that as accurately as possible estimates the gradient at any given point. The bulk error of our gradient boosting procedure is therefore based on the accuracy of that model fit.

2.5.2 Gradient Boosting using Regression Trees

For this method we consider regression trees as model for the previously discussed gradient definition issue. The proposed solution then becomes fitting a regression tree to training data using the gradient as response. Previously we have not defined the regression tree method, but it is very similar to the classification tree method described in section 2.2. The main differences are that a different impurity measure is used, and that the model estimate for any given terminal region is the mean of the response variable for training data points in that region. We refer the reader to *The Elements of Statistical Learning* [4] or *An Introduction to Statistical Learning* [5] for the exact differences.

With a solution to the gradient definition issue, the last missing piece before we can define a compact algorithm is a choice of loss function $L(k_i, F(x_i))$. For binary classification we use the binomial negative log-likelihood(or binomial

deviance) which using our set of notations is defined as

$$L(k_i, F(x_i)) = -I(k_i = 1)\log(p_1(x_i)) - (1 - I(k_i = 1))\log(1 - p_1(x_i))$$

where $p_1(x_i)$ has the inverse logit relation to $F(x_i)$

$$p_1(x) = \text{logit}^{-1}(F(x_i)) = \frac{e^{F(x_i)}}{1 + e^{F(x_i)}}.$$

To derive the gradient we simplify the first equation using the second to get

$$L(k_i, F(x_i)) = -I(k_i = 1)F(x_i) + \log(1 + e^{F(x_i)})$$

which conveniently illustrates that rather than penalising incorrect classifications, we calculate a baseline loss and subtract the logit probability for observations with true class 1. Taking the derivative of this function with respect to $F(x_i)$ gives us the gradient components

$$g_{i,t} = -I(k_i = 1) + \frac{e^{F(x_i)}}{1 + e^{F(x_i)}} = -(I(k_i = 1) - p_1(x_i)).$$

With this we are ready to construct the algorithm for gradient boosting using regression trees. We will first state the algorithm and then elaborate on each component.

Algorithm 2: Binary Gradient Boosting using Regression Trees

1. Initialise with $F_0(x_i) = \text{logit}(\frac{1}{N}\sum_{x_j} I(k_j = 1))$, $i = 1, 2, ..., N$.
2. For steps $t = 1$ to T:
   (a) Let the posterior probability function for class 1 be
   $$p_1(x) = \frac{e^{F_{t-1}(x)}}{1 + e^{F_{t-1}(x)}}$$
   (b) Compute the negative gradient component for each $i = 1, 2, ..., N$
   $$-g_{i,t} = \left[\frac{\partial(L(k_i, F_{t-1}(x_i))}{\partial(F_{t-1}(x_i))}\right] = I(k_i = 1) - p_1(x_i)$$
   (c) Fit a regression tree using $-g_{i,t}$ as response values, resulting in the set of terminal regions $\{R_{j,t} : j = 1, 2, ..., J_t\}$
   (d) For each of the J_t regions $R_{j,t}$, calculate the change $h_{j,t}$, which is mutual for all observations in the region, approximated by a single Newton step
   $$h_{j,t} = \frac{1}{2}\frac{\sum_{x_l \in R_{j,t}} -g_{l,t}}{\sum_{x_l \in R_{j,t}} |-g_{l,t}|(1 - |-g_{l,t}|)}$$
   (e) Update current estimate with the change in step t by
   $$F_t(x) = F_{t-1}(x) + \sum_{j=1}^{J_t} h_{j,t} I(x \in R_{j,t})$$
3. The final estimate is a sum of T updates
   $$\hat{F}(x) = F_T(x) = F_0(x) + \sum_{t=1}^{T}\sum_{j=1}^{J_t} h_{j,t} I(x \in R_{j,t}).$$

The resulting estimates are as we recall logit transformed probabilities, and needs to be transformed back before performing classification. In our analysis we will classify observations using the same Threshold classifier as previously defined for bagging based methods. Now we will extend the motivation behind each line in the algorithm.

On line 1 we initialise the process with guessing that the posterior probability of belonging to class 1 is the natural ratio of class 1 observations in data.

Line $2(a)$ is relatively self-explanatory and on (b) we prepare the first step of gradient decent by calculating the negative gradient vector of the loss function. To be able to use the final model for classification of new observations, at (c) we apply the proposed solution of the gradient function issue. After step T we will have an ensemble of T fitted trees, each partitioning the feature space into terminal regions $\{R_{1,t}, R_{2,t}, ..., R_{J_t,t}\}$.

Line (d) is where it gets complicated. Like previous tree based methods that we have presented, all observations contained in an arbitrary terminal region are treated the same. The next stage of the numerical optimisation strategy is to move length ρ along the negative gradient, where the length is given by solving

$$\rho_t = \underset{\rho}{argmin}(L(\mathbf{F}_{t-1} - \rho \mathbf{g}_t)).$$

Estimating the negative gradient of an arbitrary observation, known or unknown, can be done in many ways with the most intuitive one being the average negative gradient of training data in the same region. However, the optimal length is still unknown. Some sources like the regression case in Hastie et al. (2017, page 361) [4] simply state "compute...' and present the above equation with added constraints for the feature space region. Later on page 387 in the same source Hasite et al. simply state the resulting estimated change at step t for k-class classification case without any comment on how it is derived or approximated. In Friedmann (1999, page 9) [1] it is stated that the estimated change $h_{j,t}$(there with notation $\gamma_{j,m}$) is approximated by taking a single Newton step, but not how or why.

Moving on to the final line of the algorithm we present the final model estimate $\hat{\mathbf{F}}(x)$. A more intuitive way of picturing this estimate is to consider the products of the algorithm. After step T we have a set of $\sum_{t=1}^{T} J_t$ regions $\{R_{j,t} : j = 1, 2, ..., J_t \ , \ t = 1, 2, ..., T\}$ each with a computed change $h_{j,t}$. An arbitrary unknown observation x will be contained in exactly T of those regions and thus have relation to exactly T change values. The logit transformed probability estimate of x being of class 1 is then the sum of those change values as well as addition of the initial guess $F_0(x)$.

2.5.3 Learning rate and Stochastic Gradient Boosting

After Friedmann's initial discovery of the method, he presented two additions to the model for improved generalisation. The first one was "Learning rate / Shrinkage" which aims to decrease overfitting. At step (e) in the algorithm we

introduce learning rate parameter λ to get

$$F_t(x) = F_{t-1}(x) + \lambda \sum_{j=1}^{J_t} h_{j,t} I(x \in R_{j,t})$$

The parameter shrinks the change of each additional tree and thus controls how fast convergence to the final estimate is. Balancing this allows for adding more trees to get a slow but sure convergence.

The second addition was adding a stochastic element to the method by fitting each tree to a sub-sample of training data. In our application we use a sample size 70% each fit. The sample size can be adjusted to allow for more or less variation but the most common values seem to be around $50 - 75\%$.

2.5.4 Different approaches

To summarise, Boosted trees work by sequentially adding trees to correct previous classification errors. Using terminology "tree depth" for the maximal number of subsequent splits in a given tree, consider two independent trees fit to the same data. One with a high value of tree depth and one with a small value. The large tree will generally perform better and result in a lower error due to being able to separate data more extensively. Consequently it runs the risk of overfitting data. Now consider two gradient boosted models

Few large trees: High tree depth, Low number of trees

Many small trees: Low tree depth, High number of trees.

Few large trees will generally have low initial error, and potentially correct a large portion of that error next iteration and so on and so forth. However, to not risk correcting errors that are the product of overfitting we keep the number of trees(iterations) low, i.e. aiming at a quick convergence. *Many small trees* on the other hand would initially produce a large error and only correct a small portion of it at each iteration. Since we do not run the same risk of overfitting, a large number of trees can be added with a small decrease in error at each step, i.e. aiming at a slow convergence. How viable these two approaches are is situational and directly depends on actual data.

2.6 Imbalanced data

Some applications require classification of data where class distribution is significantly skewed. An example of that is diagnosing rare diseases. Consider the case of having data with two classes and the class ratio being $1 : 9$. A single

tree classifying all observations as the majority class will have an error rate of 10%, but consequently will not correctly classify a single minority observation. Suppose a second tree has a higher error rate, but identifies all minority observations. Taking the pure mathematical perspective and disregarding our subjective priority of classes(eg. in disease diagnosis), the first tree is always more favourable.

2.6.1 Weights

By weighting each observation according to which class it belongs to, we can effectively increase the impact of a correct minority classification. Defining weights dependent on minority class weight w_M by

$$w_i = \begin{cases} \frac{w_M}{N_{Minority}}, & \text{if } k_i = Minority \\ \frac{(1-w_M)}{N_{Majority}}, & \text{if } k_i = Majority \end{cases}$$

leads to w_M interpreted as "total weight of minority". For example, using $w_M = 0.9$ would roughly be the same as having class distribution ratio $9:1$ instead of $1:9$. With $w_M = 0.5$, classifying all observations as minority would be considered as favourable as the opposite. In simple terms, with w_M we can control which class the model will prioritise in regard to correct classification.

2.6.2 Priorities

The "problem" of few correct minority classifications(denoted C_m) arises only after introducing this prior subjective class priority. Suppose we optimise model construction with respect to minority classifications, the resulting model will be the one with lowest error among models with highest possible C_m. For perspective, consider the situation where class ratio is $1:9$ and one model classifies all observations as minority. It will have 100% Hit-rate($\frac{C_m}{N_m} = 1$) but 90% error rate. According to class priority this will very good model, but in the case with fraud detection, incorrectly classifying 9 times more cases than correctly seems bad. Due to this we define a second set of priorities: low False-positive to True-Positive rate, denoted r_{FP}. Why are we interested in r_{FP}? Because in the same way correctly detecting fraud saves an insurance company money, investigating a legitimate claim costs money. We can with some certainty assume that these priority-sets are negatively correlated for imbalanced data. Each insurance company has their own definition of the optimal balance point, we however lack insight into such a organisation and have to approach the problem in our own way.

Defining two model-types where the aim is maximising C_m unbound/bound by r_{FP} respectively

Aggressive : Prioritising maximisation of C_m over r_{FP}.

Conservative : Prioritising minimisation of r_{FP} over C_m.

we cover each side of the spectrum and can analyse behaviour of those models separately.

2.7 Model performance measures

As previously explained the specific application in this thesis uses alternative optimisation criteria and thus requires custom performance metrics to cover different sets of priorities.

2.7.1 Custom Metrics

We have two main quantities to measure: Hit-rate and False-positive to True-Positive ratio. More formal mathematical definitions are

$$h = \frac{\sum_{i=1}^{N} I(\hat{f}(x_i) = Fraudulent) I(k_i = Fraudulent)}{\sum_{i=1}^{N} I(k_i = Fraudulent)}$$

$$r_{FP} = \frac{\sum_{i=1}^{N} I(\hat{f}(x_i) = Fraudulent) I(k_i = Legitimate)}{\sum_{i=1}^{N} I(\hat{f}(x_i) = Fraudulent) I(k_i = Fraudulent)}$$

where $\hat{f}(x_i)$ is the model classification of observation x_i. Important to note is that in our analysis, we can not alter the inner optimisation of any applied machine learning method. This is a consequence of using a R package with predefined implementations. What we can and will do is vary model parameters to optimise model selection with respect these custom metrics. As such, any single tree is optimised according to general classification error within the restrictions of applied algorithm and set of parameter values. The main flaw of this approach with two optimisation metrics h and r_{FP} is that we can not directly quantify their relation and thus not easily find a balance between them.

2.7.2 Profit equation

In an attempt to connect defined metrics in search of harmony between them, we backtrack and consider the specific application. Previously we stated that money saved by an insurance company is directly dependent on Hit-rate h, but simultaneously negatively dependent on r_{FP}. Applying basic economics, a

profit(or savings) equation can be formulated. Assuming the following quantities are known: N =total number of reported claims, r_F =overall rate of fraudulent claims, c_c =average claim size and c_i =average cost of investigating a claim, the mentioned equation becomes

$$\begin{aligned} average\ savings &= N \cdot r_F \cdot h \cdot (c_c - c_i) \\ average\ costs &= N \cdot r_F \cdot h \cdot r_{FP} \cdot c_i \\ average\ profit &= N \cdot r_F \cdot h((c_c - c_i) - r_{FP} \cdot c_i). \end{aligned}$$

Note : The derived equation works under the assumptions that c_i and c_c are not dependent on class or feature values.

3 Data

Data used in this paper originates from an Oracle database [10] containing sets for data mining education. Each observation is an Auto insurance *claim* filed to a non-disclosed US insurance company during years 1994 to 1996. By insurance *claim* we refer to the event which triggers consequences constituted by an insurance policy. Each claim in data has been subject to investigation of whether it is a legitimate claim or a $fraudulent$ one. By $fraudulent$ claim we refer to any event caused or action made with malicious content, such as an orchestrated event, a fictional event or an event where reported information has been tampered with or exaggerated. In the report "Försäkringsbedrägerier i Sverige 2017"[9] made by Svensk Försäkring, it is estimated that general insurance fraud costs $3-6$ billion Swedish crowns per year. Almost 7000 claims were investigated in 2017, resulting in saving nearly 500 million crowns in potential payouts.

3.1 Data overview

The data consists of 15 420 Auto insurance claims spanning years 1994 to 1996. The overall rate of fraudulent claims in data is 6%. Each claim contains information on 33 variables concerning the event or the insurance policy.
A brief overview of variable groups is presented below.

Group : Variables(specific factor-levels).

Time : Day, week, month, year of incident and submission of claim.

Vehicle : Vehicle type, make, price interval, age, number of supplements.

Incident : Incident area, number of vehicles, Fault(policy holder/3rd party), police report filed(Y/N), witnesses(Y/N).

Policy holder : Sex, age, driver rating, past number of claims, marital status, time since address change.

Policy : Policy vehicle type, base policy(collision/liability/all perils), rep number, deductible, days policy - accident/claim, agent type(internal/external).

Fraud : Was the claim found to be fraudulent? (Y/N).

3.2 Data summary

A vast majority of mapped variables are naturally ordered or grouped factors. While the option to specify their order exists, we would actually lose information and limit our options if we choose to do so. A binary split of the ordered feature space would translate to dividing levels "above" and "below" a certain order. The number of possible splits is then equal to levels minus one. However, a split of the unordered feature space is the set of two disjoint sets, together containing all factor levels. To demonstrate this we consider the relation between Vehicle Price and the ratio of fraudulent claims found in Figure 4. Using ordered factors, levels "less than 20 000" and "more than 69 000" would always need at-least two splits to isolate, and would never belong to the same partition. Using unordered factors, isolating all levels with a fradulent ratio above 6% to the same partition would only require one binary split.

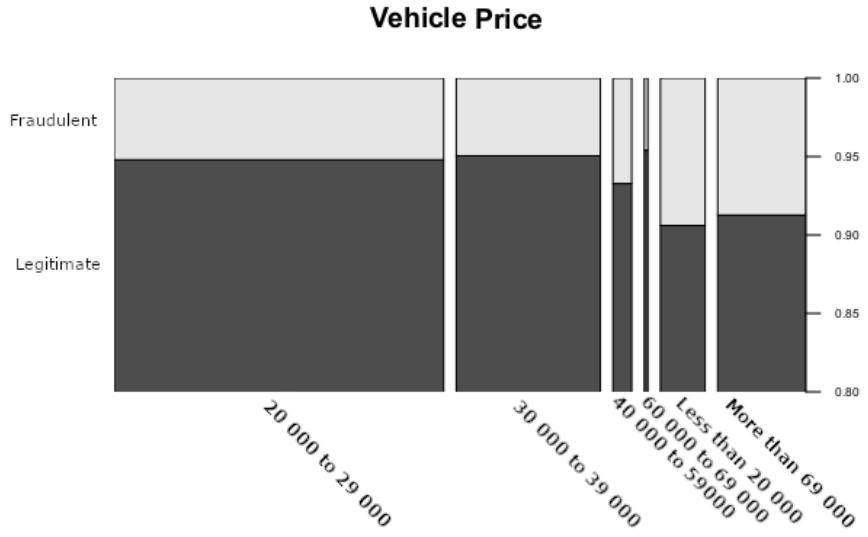

Figure 4: The relation between Vehicle Price(levels) and the ratio of fraudulent claims. Were the width of bars relate to proportion of observations for each level, and where the Y-axis has been restricted at 0.8. It is evident that there are different ratios of fraudulent claims for different levels.

The vast majority of features have no clear relation to fraud if we look at ratios within levels. However a handful have one or two diverging factor levels where fraudulent claims are over-represented with regards to the general ratio of about 6%. Figure 5 illustrates such a feature.

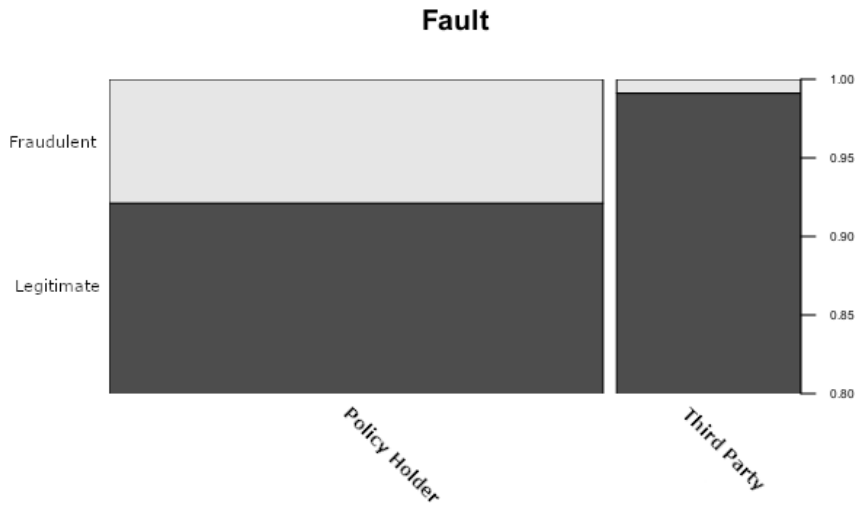

Figure 5: The relation between which the Faulty party was and the ratio of fraudulent claims. The ratio of fraud is greatly reduced for cases where fault lies in a third party. The width of bars relate to proportion of observations for each level, and the Y-axis has been restricted at 0.8.

Since tree based methods lack distribution-specific assumptions and generally are very robust when it comes to non-pruned data, we move on to model assessment without further data analysis.

4 Pre-analysis and Model assessment

Continuing from propositions made in section 2, we have two priority sets derived from expectations of a hypothetical insurer, and two approaches to application of the Boosting method. Considering all combinations we get four model compositions corresponding to different demands.

Table 1: Combinations of model types presented in section 2.5.

Name	Abbreviation	Hit-rate	r_{FP}	Depth	#Trees
Aggressive Large	AL	High	High	High	Low
Aggressive Small	AS	High	High	Low	High
Conservative Large	CL	Low	Low	High	Low
Conservative Small	CS	Low	Low	Low	High

Model selection is then a performance comparison problem of four independent models. In turn those are each a multivariate optimisation problem dependent on the total fraudulent class weight w_F, singular tree depth d, the number of trees T, learning rate λ and fraud-classification threshold P_F. However 5-way numerical optimisation of a method that can take 1-3 minutes to apply is not a reasonable approach. Before parameter optimisation can be attempted, removal or grouping of some parameters must be performed, alternatively dependencies between them must be found. We start by identifying the purpose and effects of each parameter, then tether similar ones to each other.

For a single tree, d has a positive correlation with information gain, which when generalised to additive iterations of boosting intuitively should have the same effect. T and λ are also connected to information gain, and are by definition closely related, with slow learning models requiring high number of trees to reach convergence. Since d directly positively affects tree size it is trivial to see that combinations high d, low T and low d, high T fit opposite parts of the spectrum for "few fully grown trees vs many small trees". If in turn λ can be dependently calibrated by T, approximate optimisation of all three parameters is plausible and isolated to the method approach issue.

We are then left with finding a solution to "high Hit-rate vs low false positive ratio" using parameters w_F and P_F. Initial testing exemplified by Figure 6 indicate that early iterations approximately follow

$$\hat{f}(x) \approx w_F$$

and while the weights are consistent throughout the algorithm, their main effect seems to be increasing the fraud classification probability of all observations. On the contrary P_F has the negative latter effect on classifications, meaning their interaction could counteract the unnecessary inflation of classification probability. Figure 7 illustrates the interactive effects w_F and P_F have on both Hit-rate and rate of false positives to true positives. Not using weights is approximately equivalent to using w_F equal to the natural rate of fraudulent cases in data, 7%.

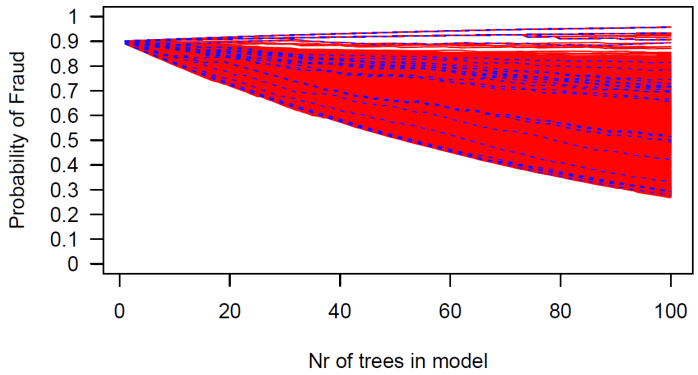

Figure 6: Probability of classifying a claim as fraudulent using a model with $w_F = 0.9$. Blue lines are truly fraudulent claims, red are legitimate. Legitimate claims have seemingly uniform distribution on [0.3, 1]

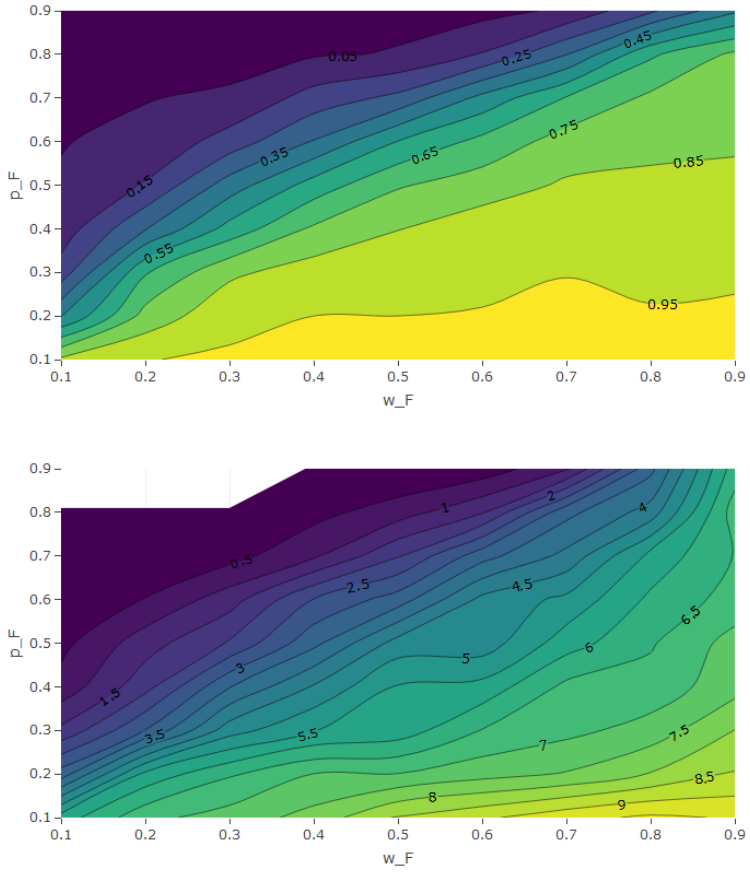

Figure 7: Hit-rate(top) and False-positive ratio(bottom) of model with d=6, T=100, $\lambda = 0.1$. w_F is the fraudulent class weight and P_F the probability threshold. Both metrics have very similar behaviour above the diagonal, but clearly differ below.

Note that the rates have a similar, but not identical relation to parameters. Figure 7 also seems to indicate that the "high Hit-rate vs low false positive ratio" problem can be approached by using high or no w_F and supporting it with high and low P_F respectively to counteract probability inflation. The result of initial parameter estimation is the following four models

Table 2: Initial parameter composition based on theory and priority sets.

Name	w_F	d	T	λ	P_F
Aggressive Large	0.9	6	500	0.01	0.75
Aggressive Small	0.9	2	1500	0.005	0.75
Conservative Large	-	6	500	0.01	0.40
Conservative Small	-	2	1500	0.005	0.40

In the next section we compare these independently calibrated models to evaluate importance and effects of "few fully grown trees vs many small trees" and "high Hit-rate vs low false positive ratio".

5 Results

In this section the results of our analysis will be presented with some commentary to highlight important details. The theory applied is defined and explained in section 2 and thus will not be expanded on further, with the exception of entirely new information. All applications of machine learning methods are performed with the "gbm" and "random forest" packages in "R". It should be noted that the implementation of decision trees in "gbm" is a "right bound recursive growth algorithm". In pseudo code that is

Algorithm 3: Right bound Recursive growth

1. Find binary split for current partition R_m.
2. If not (stopping criterion): Put R_L as first element of "que", repeat alg from 1 with $R_m = R_R$.
3. If (stopping criterion): Set R_m to terminal region, repeat alg from 1 with R_m =first element in "que".
4. If (que empty): Stop.

Since our trees are restricted by depth rather than number of terminal regions, this type of algorithm allows for considering a large number of varied feature interactions.

5.1 Model performance

Figure 8 and 9 depicts the variation in defined metrics stemming from stochastic properties attained by the boosting method when a sub-sample of the overall training set is used in each boosting iteration. The figure presents shape, with average values for mean and standard deviation located in Table 3.

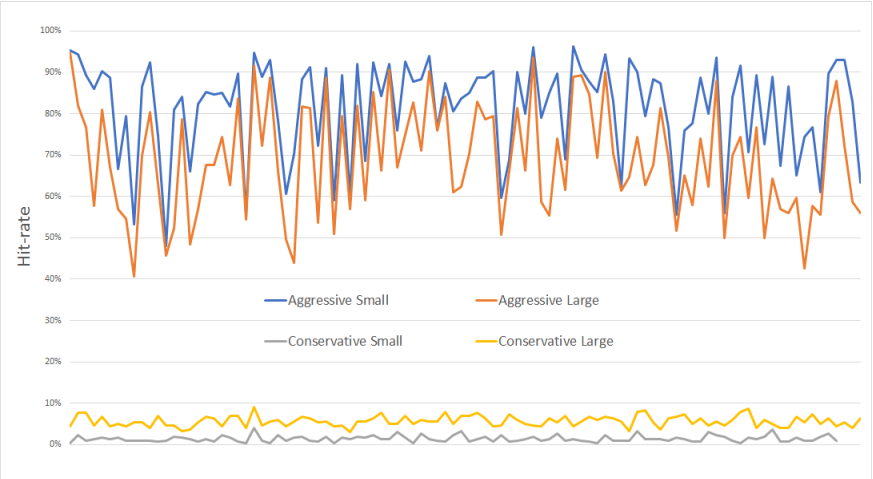

Figure 8: Hit-rate for 100 samples of the four initial models. Randomly sampled test-set between each of the 100 sets. For aggressive models the Hit-rate is very volatile but model hierarchy seems unaffected.

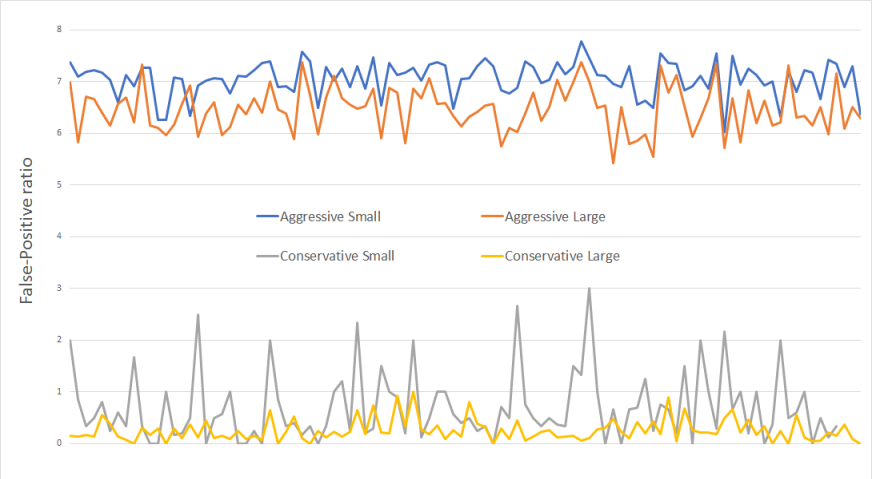

Figure 9: False-positive rate for 100 samples of the four initial models. Randomly sampled test-set between each of the 100 sets. Conservative small model stands out by having a large amount of spikes.

Table 3: Mean and standard deviation of Hit-rate and False-positive rate for the four initial models, each with a sample size of 100.

Model	$\overline{h}$	$sd(h)$	$\overline{r_{FP}}$	$sd(r_{FP})$
Aggressive Large	$68,95\%$	$13,33\%$	6,47	0,44
Aggressive Small	$81,37\%$	$11,62\%$	7,06	0,33
Conservative Large	$5,68\%$	$1,30\%$	0,26	0,21
Conservative Small	$1,44\%$	$0,79\%$	0,70	0,67

As hypothesised aggressive models have higher h at the cost of higher r_{FP}. While CL is objectively better than CS, the same can not with certainty be stated for AL and AS. Standard deviation of the False-positive rate is stable apart from CS, although conservative models have substantially higher relative values. Hit-rate performance for conservatives is surprisingly low. The table percentages represent 15 and 5 correctly identified claims out of 300 in our test-set. This could be an indication that we shrink individual tree contribution too much relative to the number of trees, and consequently stop the converging process early. Testing shrinkage(λ) values 0.1 for small and 0.05 for large models results in Table 4. These values are closer to Friedman's upper limit discussed in Section 2.5.3.

Table 4: Mean and standard deviation of Hit-rate and False-positive rate , each from a sample size of 100.

Model	$\overline{h}$	$sd(h)$	$\overline{r_{FP}}$	$sd(r_{FP})$
Aggressive Large V2	$29,12\%$	$14,09\%$	1,86	0,36
Aggressive Small V2	$42,46\%$	$14,53\%$	4,92	0,53
Conservative Large V2	$17,27\%$	$6,25\%$	0,71	0,23
Conservative Small V2	$9,38\%$	$3,39\%$	1,25	0,48

The change has improved conservative models considerably, on the other hand the two aggressive have halved in Hit-rate and even had their standard deviation increase one and three percent units. Regarding Large vs Small, neither looks significantly better than the other apart from CL and CS. The only apparent overall difference so far is that small models tend to the extreme points of the spectrum more than large ones.

By inspecting the probability of an observation being fraudulent and calculating the overall median as well as interesting quantiles within each class, we get a notion of which probability threshold values P_F are appropriate. The inspected models here are the initial aggressive and the adjusted conservative(V2). We use initial aggressive since the performance represents their original priority set

more than adjusted(V2). From Table 5 we observe that higher Hit-rates for conservative models require significantly lower threshold than the one used. We also observe that an aggressive model aiming at 25% Hit-rate would classify 5% of legitimate claims wrong, which combined with class imbalance 16:1 results in $r_{FP} \approx 3$.

Table 5: Median and 75%-/95%-quantiles for probability of an observation belonging to class Fraudulent, given that the true class is Fraudulent or Legitimate. Letting $P_F = q_{75}(\text{Fraud})$ would for example result in 25% Hit-rate.

Model	median Fraud	q_{75}(Fraud)	median Legit	q_{95}(Legit)
AL	86%	94%	19%	95%
AS	91%	95%	40%	95%
CL V2	15%	30%	1%	14%
CS V2	13%	23%	1%	19%

Since P_F does not affect model fit we can easily calculate h and r_{FP} for any $P_F \in [0, 1]$ without refitting the model. Illustrating the h *vs* r_{FP} trade-off can thus be done trivially with a line graph in Figure 10.

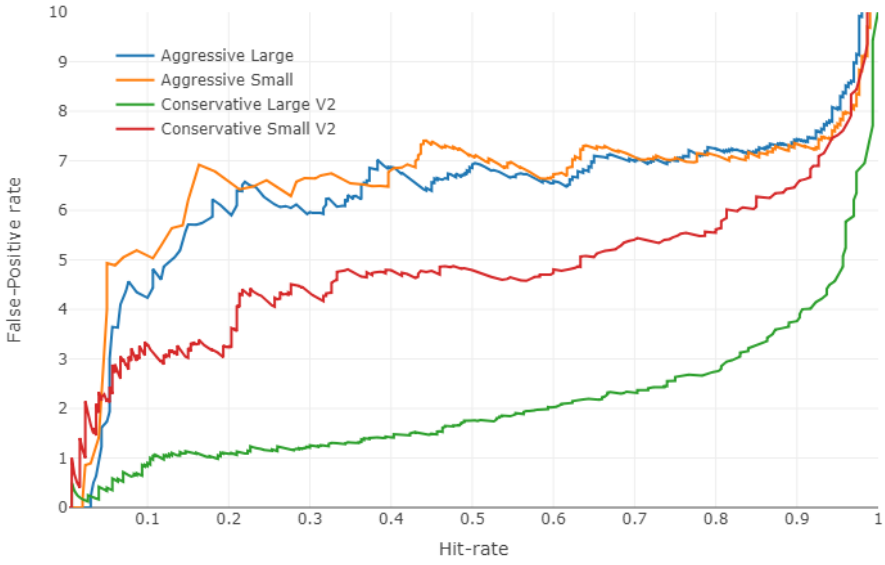

Figure 10: Hit-rate vs False-positive ratio for threshold values in [0,1]. The optimal case is where a line's lowest rightmost point is at [1,0] meaning perfect classification with no False-positives.

From this illustration we get valuable information about the trade-off shapes. As an example we observe that Hit-rate for CL V2 can be increased from 20% to 30/40% without affecting r_{FP} much. Apart from information about individual

performance the graph provides model rankings for particular intervals. Any line positioned below another within a given interval on the x-axis has objectively better metric performance than the other. Previously we could not rank aggressive vs conservative due to incomparable metrics. In Figure 10 the latter ranks above the former, with CL V2 being superior in all situations.

The performance and versatility of threshold value choice for CL V2 allows us to focus the further analysis on this model type only. However at this point it is still unknown what r_{FP} and h values are considered "good". Furthermore without a reliable benchmark model or goal values for metrics, our performance analysis and the task of assigning a P_F value becomes a guessing game.

5.2 Cost effective fraud detection

As previous results indicate, metrics used have a positive correlation but opposite optimums, which raises a critical issue. Any two models can only be compared objectively if they share value in one metric, or if both metrics are closer to their respective optimum for one of the models. By introducing the profit-equation defined in Section 2.7.2 we produce a quantifiable relation between metrics. We use notation N =number of claims, r_F = overall rate of fraud, r_{FP} =rate of False-positive to True-positive for model, h =Hit-rate, c_c =average cost of claim and c_i =average cost of fraud investigation. By simplifying that equation we get

$$\begin{aligned}\text{average profit} &= N \cdot r_F \cdot h((c_c - c_i) - r_{FP} \cdot c_i) \\ &= N \cdot r_F \cdot h(c_c - c_i(r_{FP} + 1)).\end{aligned}$$

Bound by this equation objective comparison of models is made possible for all sets of metric values. Immediately we observe that the breaking point for profit is

$$r_{FP} = \frac{c_c}{c_i} - 1$$

which is entirely independent of h. Considering claim cost as the cornerstone we define $c_r = \frac{c_i}{c_c}$ as the Investigative cost ratio. In words a value $c_r = \frac{1}{3}$ would be that on average the cost of investigation is a third of the claim cost. Utilising this ratio, the following simplification can be made

$$\begin{aligned}\text{profit} &= N \cdot r_F \cdot h(c_c - c_i(r_{FP} + 1)) \\ \frac{\text{profit}}{N} &= r_F \cdot h\left(c_c - c_c \cdot c_r(r_{FP} + 1)\right) \\ \frac{\text{profit}}{N \cdot c_c} &= r_F \cdot h\left(1 - c_r\left(r_{FP} + 1\right)\right).\end{aligned}$$

Here $\frac{\text{profit}}{N \cdot c_c} = 0.01$ is interpreted as "on average profiting 1% of claim cost for each of N claims", which is the same as "on average profiting 1% of total claim cost". Using $c_r = \frac{1}{2}$ Figure 11 illustrates the above, with other examples located in appendix.

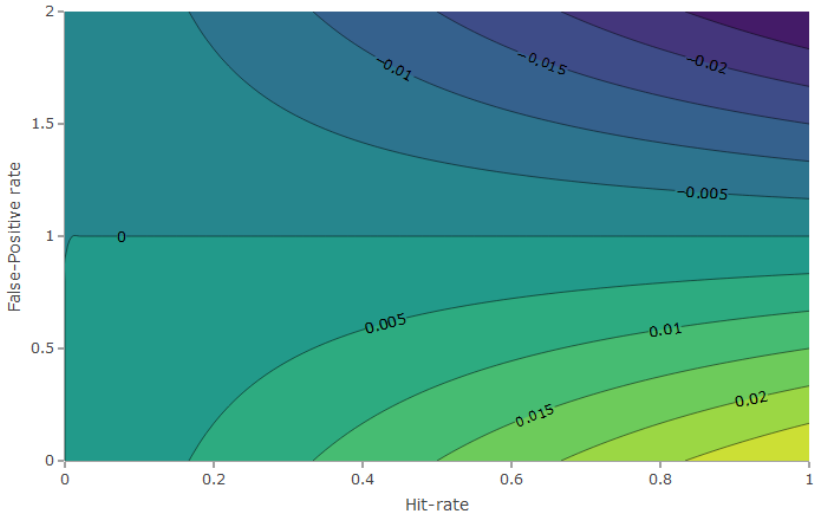

Figure 11: Profit of total claim cost of N claims. Since loss is not of particular interest, the y-axis has been restricted. For metric values of a model this figure gives us an indication of which metric to focus on improving.

Considering model CL V2, for a given threshold we have a pair (h, r_{FP}) which in turn can be used to calculate the profit at a specific c_r. Figure 12 below illustrates this connection with a contour plot. The red line indicates optimal (h, r_{FP})-pair for a specific c_r(based on data from Figure 10).

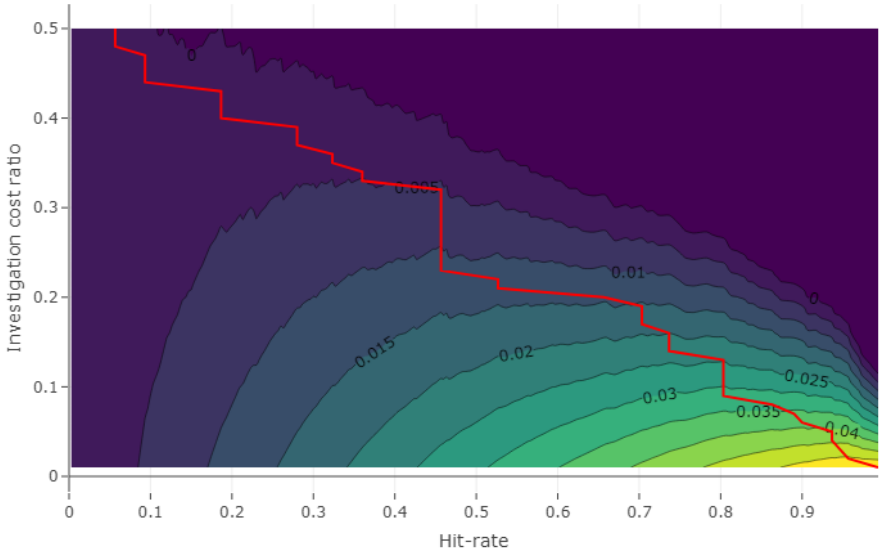

Figure 12: The relation between positive profit, Hit-rate and Investigation cost ratio. Since positive profits are of interest, negative profits are not shown. The matter of choosing what Hit-rate to aim for becomes trivial when the cost ratio is known.

5.3 Class probability convergence

In this section we want to analyse how the boosting method arrives at the results in Section 5.1 by observing the convergence of classification probabilities $p_k(x_i)$. Previously we used $p_k(x_i)$ to calculate and plot the effects that varying P_F has on h and r_{FP}. This time we predict our test-set using a boosting model consisting of the first t trees. We perform predictions with $t = 1, 2, ..., T$ to form a sequence of T predictions for each observation. In a graph with axes t and $p_k(x_i)$, each sequence is represented by a line. The colour of the line indicates the true class that observation belongs to, blue for fraud and red for legitimate. The graph presents us with a number of different characteristics inherent to the model. Amongst minor things it illustrates shape of convergence, separation of classes and separation within classes. Figure 13 consists of such a graph for CL V2. From this point on, any reference to "the model" or "our model" will be referencing to CL V2 unless otherwise specified. These graphs are filtered to only display the 50 highest probabilities from each class. This is to reduce cluttering as the original plot(see appendix or Figure 6) displays 5300 unique lines.

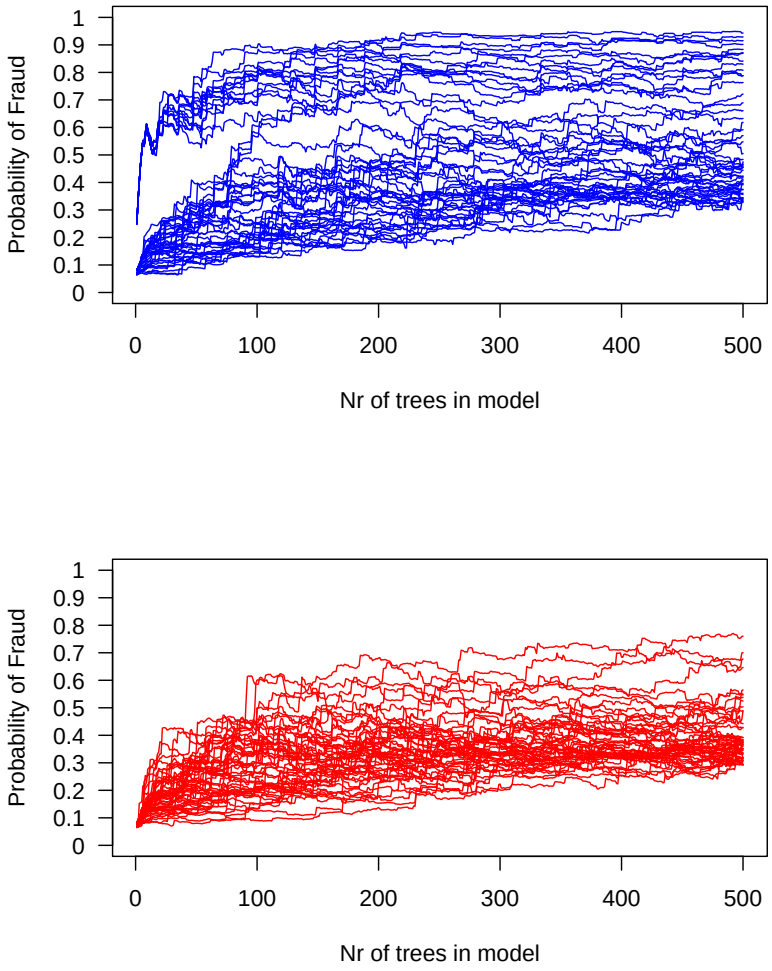

Figure 13: Both figures show the probability of classifying a claim as fraudulent using model CL V2, where each line represents a claim. The top figure are actual fraudulent claims while the bottom are legitimate. Illustrating them like this provides an overview of difference in convergence behaviour between classes.

A cluster of fraudulent claims are identified immediately and populate the top of the top graph trough out the complete algorithm. The rest follow a very similar pattern to legitimate observations in the bottom graph. As the number of trees increase parts of the identified cluster deteriorates while conversely the large lower body improves. Legitimate claims seem to be interchangeable with the large portion of displayed fraudulent.

Slow convergence models performed worse than fast, with the best of them being CS V2. Figure 14 provides two convergence graphs using CS V2. The previous relation in shape between classes is present here as well, but the cluster of fraudulent claims that is identified early has a considerably smaller distance from the main group. The overall similarity between graphs is clearer, and displayed legitimate claims have a lower spread than that of the other class. For the non filtered figure see appendix.

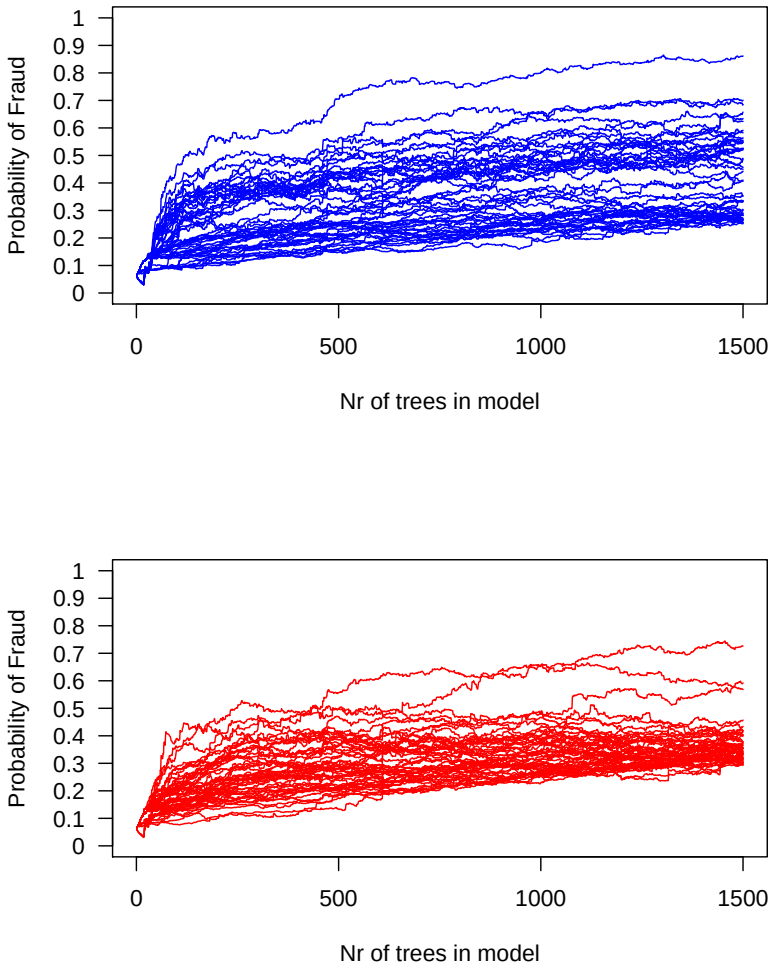

Figure 14: The same type of figures as Figure 13 but for model CS V2.

Generally this second figure appears to be a more compressed and longer version of the first. The question is then what happens if we break the model

boundaries and try a large slow converging model. In figure 15 we have fit a model without weights, using depth of 6, 1500 trees and shrinkage equal to 0.05. That is the same parameter composition as CL V2 but with increased number of trees and lower shrinkage. Legitimate claims do not only have a tighter spread, but also have generally lower probabilities than fraudulent claims. Non-displayed claims are still indistinguishable between classes, but this model looks to have separated classes better than those previously tested.

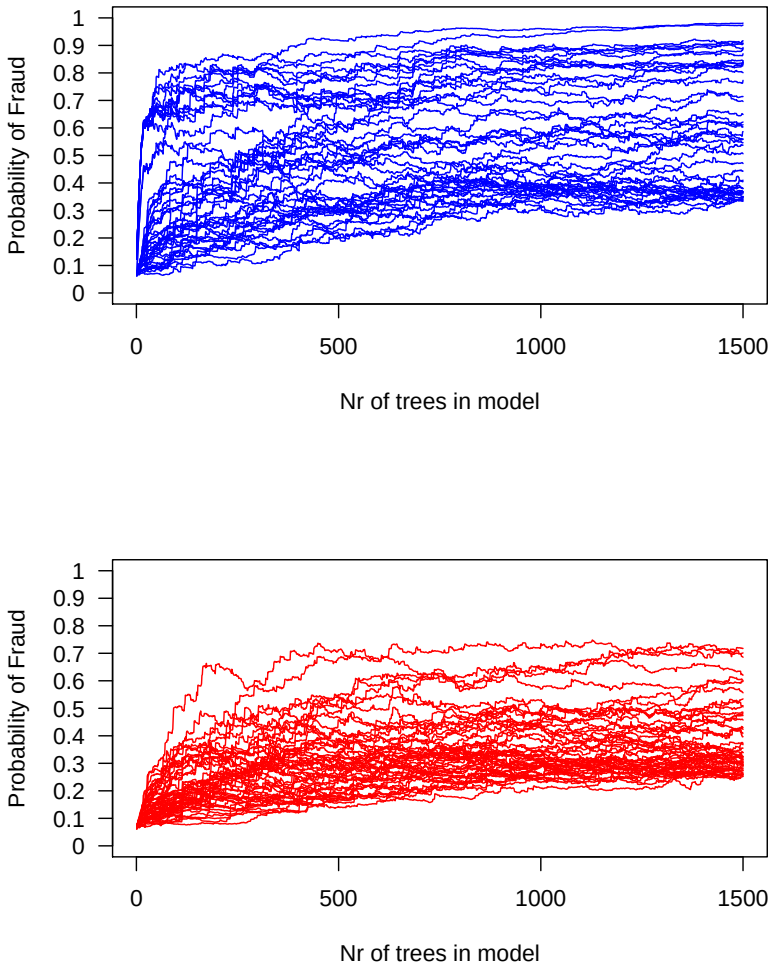

Figure 15: Both figures illustrate the probability of classifying a claim as fraudulent using a model like CL V2 but with slower and longer convergence. The figure is of the same type as Figure 13 and 14.

Inspecting Table 6 and Figure 16 we see that while metrics in the table may indicate better performance, the graph shows that for other threshold values than 0.4, ranking is uncertain.

Table 6: Metrics for Conservative Large models.

Model	$\overline{h}$	$sd(h)$	$\overline{r_{FP}}$	$sd(r_{FP})$	$C_r = 2$
CL V2	$17,27\%$	$6,25\%$	0,71	0,23	$0,18\%$
CL V3	$18,69\%$	$6,92\%$	0,53	0,17	$0,24\%$

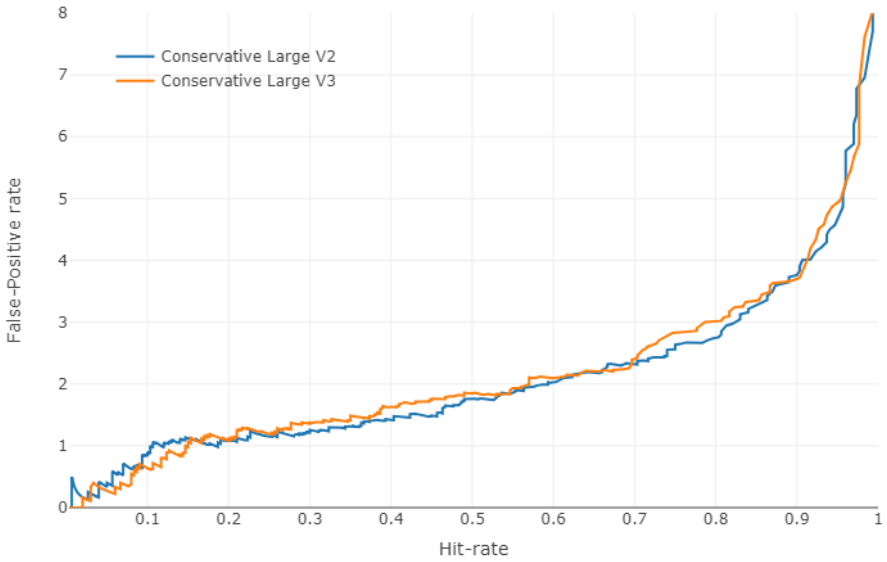

Figure 16: Comparison of Hit-rate vs False-positive ratio for threshold values in (0,1) for CL V2 and a variant with slower and longer convergence.

Inspecting all three convergence figures in closer detail we detect that all lines have constant small jittering. For a given tree that increases a claims probability, the next tree appears to decrease it, the next increase so on and so forth. Even though the net-movement of this process might be increasing, it could cancel out or disrupt a slow convergence over a large set of trees. To analyse this we inspect the structure of singular trees.

5.4 Initial split behaviour

Due to the sheer number of trees individual structure is hard to compare. Instead we look at what feature variable is split initially for each tree. By having all feature variables populate the y-axis and letting tree number in order of initiation be x-variable we can plot points that describe initial splits. In Figure 17

we plot splits of CL V2.

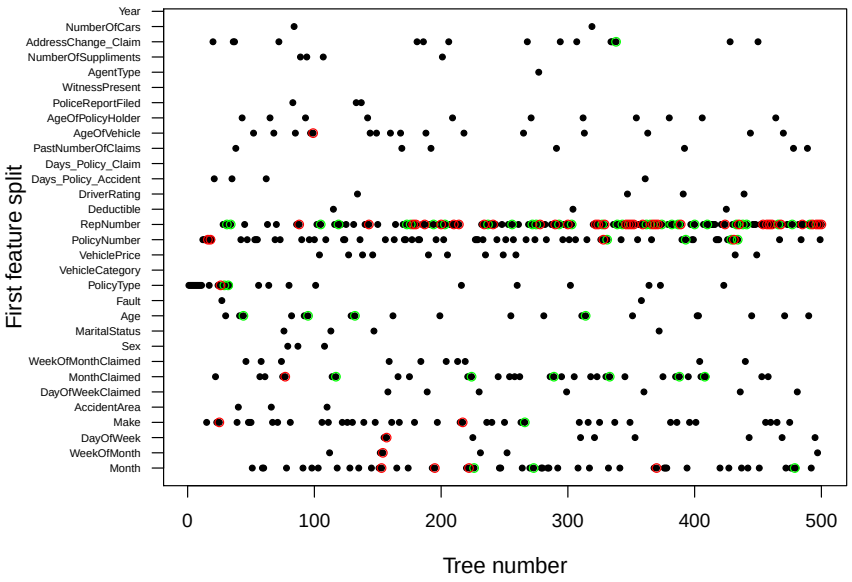

Figure 17: An illustration of which feature is used in the initial split for each tree in model CL V2. A red marker at tree t indicate that the feature was also used initially for tree $t-2$, but no for tree $t-1$. Green markers indicate used for $t-3$ but not $t-1$ or $t-2$. Markers should indicate when jittering occurs.

Policy Type is used heavily in first 20-30 trees while later focus shift to Representative number and Policy Number. We note some clusters of rings although fewer than expected from previous results and hypothesis drawn from convergence plots. Illustrating the same graph for CS V2 in Figure 18 we note a lot more clusters, both red and green. A similarity with Figure 17 is that clusters tend to appear at higher tree numbers and could indicate that convergence has grown to a halt, leaving only random jittering.

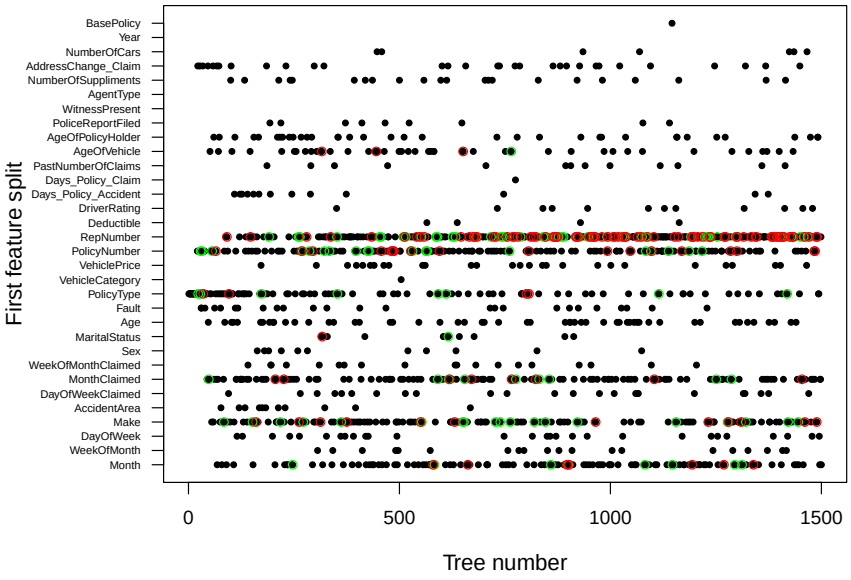

Figure 18: Feature used in initial split for each tree in model CS V2. This is the same type of figure as Figure 17.

While these figures might spawn a hypothesis concerning jittering and its causes, they do not constitute proof of it.

Evidently later stages of convergence do not display the sought after "self-correcting" characteristic of boosting models. The remaining question is then if the boosting method is any different, performance-wise, from the much simpler Random Forest method.

5.5 Random Forest comparison

As explained in Section 2 the Random Forest method works by forcing variation in tree composition and counteracting it with set averaging. Using 1 000 trees and considering 10 features for each split, initial testing suggested that random forest models still benefit from increased depth even after 6. Figure 19 displays a trade-off comparison between two of these random forest models and the Conservative Large model Version 2.

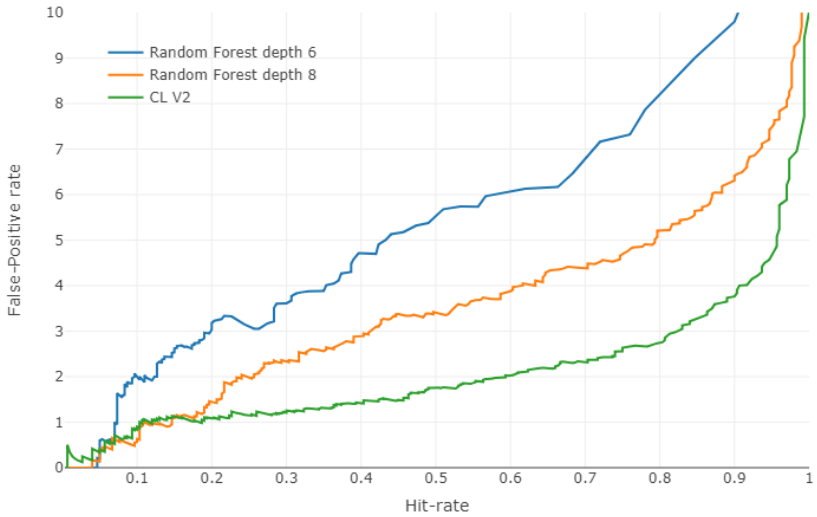

Figure 19: A comparison regarding Hit-rate vs False-positive ratio between two random forest models and the best boosting model, effectively illustrating performance hierarchy.

Evidently boosting still prevails despite convergence issues. Comparing to Figure 10 random forest models perform better than the rest of boosting models, at least for low to medium Hit-rates. Just as with gradient boosting we conclude the analysis with a contour plot of cost-effectiveness for different investigative cost ratios, as well as a line indicating the optimal Hit-rates.

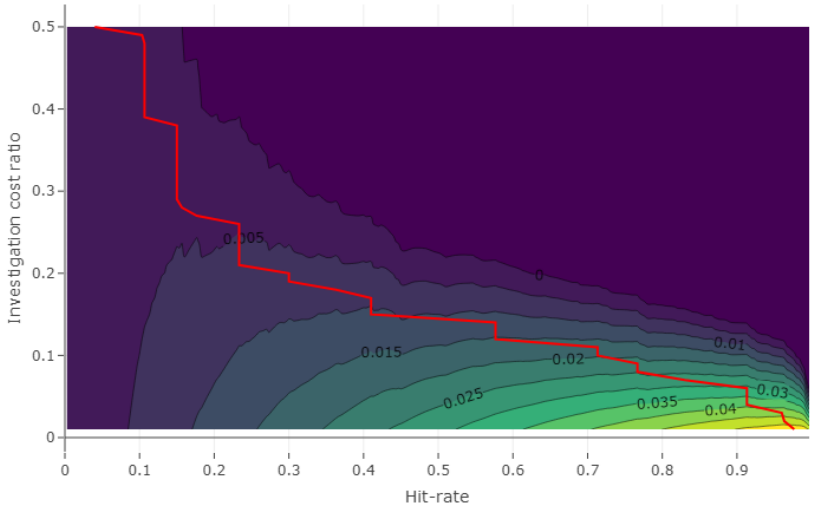

Figure 20: For the random forest model with depth 8 the figure shows profit as a function of Hit-rate and Investigation cost ratio. Only Positive profit contours are displayed.

6 Discussion

In this section we will discuss main results from Sections 5, 3 and the theoretical framework from 2. Discussion will be divided in three parts. First: explaining how theory produces results, how these results would be applied and affect a practical application. Second: what possible extensions could be made to the analysis for future study. And third: corrections and limitations of the performed study.

6.1 Theory and Results

The goal of using machine learning methods in this study was to be able to identify patterns and in some sense outliers from a relatively large set of data. Decision tree methods are easy to apply and while there are more powerful methods available, it is a good place to introduce machine learning as a concept. With that said, this type of learning becomes very data driven and situational when applied to real world problems, which is what we have explored in this thesis.

A concurrent problem throughout the analysis is how to balance and configure parameters. For learning rate/shrinkage λ and number of trees T there is a general logic that when increasing the latter you should decrease the former, and vice versa. For specific values there are different preferences with regards to computation time, model size, scope, situational preferences etc, however the suggestions made by the inventor of this method is used as overall baseline in most sources I have come across. In *Friedman's* first boosting paper[1] from 1999 he prominently uses fewer than 500 trees and $\lambda = 0.1$. When he analyses the trade-off between these parameters on page 14 he suggests that "...the best value for λ depends on the number of iterations T. The latter should be made as large as is computationally convenient or feasible.". For values $\lambda < 0.125$ he observes diminishing returns and number of trees up toward 1000. In 2007 Greg Ridgeway the original developer of the *gbm* package for R, writes in a paper [6] on page 4 that he performs the reversed selection, "In practice I set λ to be as small as possible and then select T by cross-validation.". Later on page 7 he states that "I usually aim for 3 000 to 10 000 iterations with shrinkage rates between 0.01 and 0.001.". The larger amount of iterations compared to Friedman is most likely a combination between more modern computers with higher computational power, and a desire to confidently overshoot the optimal number of trees according to cross-validation.

As for our value selection we picked 1 500 and 500 trees, with shrinkage rates of

0.1, 0.05, 0.01 and 0.005. Shrinkage rates were picked partly with regard to the trade-off and partly to fit sides of the spectrum. Number of trees/iterations were picked to fit model priorities and computational power of my machine. While higher T could be set for fitting individual models, getting averages and fitting a model type many times in a row would require vastly more computational power and time.

Our results with regard to the trade-off between shrinkage and number of trees are somewhat inconclusive in relation to previous findings. Experimenting with slower convergence, i.e. lower λ and higher T, for the Conservative Large model Version 2 resulted in possibly improved performance. The reason why we did not get results like those in source papers could be that we define "improved performance" differently. Source material used overall error rate and cross-validation error as measurements of performance, while we used Hit-rate, False-positive ratio and average profit. The result that slow convergence(*Small*) models were inferior to fast convergence(*Large*) ones is in my opinion not as much an argument for reduced T and higher λ as it is an issue of high vs low tree depth.

Regarding singular tree depth Ridgeway as well as Friedman and his co-authors of *The Elements of Statistical Learning*[4] Trevor Hastie and Robert Tibshirani, all favour low number of terminal nodes like *stumps*(depth=1, 2 terminal nodes). This is due to generally improved performance for slow long-term convergence compared to models with more terminal nodes. In Section 10.11 of ESLII[4] it is stated, with notation J = Number of terminal nodes, that "The generative function is additive(sum of quadratic multinomials), so boosting models with $J > 2$ incurs unnecessary variance and hence higher test error.". Later in the same section authors conclude that "Althoug in many applications $J = 2$ will be insufficient, it is unlikely that $J > 10$ will be required". Due to the implementation of *gbm* for R, we are only able to specify and restrict tree depth d as opposed to J. In our analysis we have used depth 2 and 6 which result in min/max $J = 3/4$ and $7/64$.

From results in Section 5 we get the indication that models with low depth under-perform and are tempted to conclude that our results contradict those previous sources. This is in fact not entirely true. Consider a tree with $J = 8$, it can reach depth 7 to fit a very specific isolated interaction between feature variables. Meanwhile a fully grown tree restricted by $d = 3$ can only reach depth 3 while having the same number of terminal nodes $J = 8$. Our conclusion is then that for this particular application, gradient boosting using trees restricted by low depth value performs worse than using higher values. Considering data

imbalance with 15 000 total observations and splits dividing data with ratio 1:3, we would need depth 3 to get a region with 233 observations. To pass a majority vote for fraudulent, that region would have to include at least 12.7% of all fraudulent claims. In this sense a possible explanation to the poor performance of small models is that we need to be able to partition data more thinly to identify class separating characteristics.

The core characteristic of Gradient Boosted trees is convergence. While we encountered problems during late stages of the process, comparison to Random Forest models indicated that this characteristic indeed was the key to better performance. However, it proved to be secondary to tree depth in terms of importance since the Random forest model with higher depth out-performed remaining boosted models.

As for the actual metric results we found that conservative models tend to have higher relative standard deviation for the rate of False-positives. This is somewhat expected since a model with Hit-rate of 5.7% combined with a False-positive ratio of 0.26 results in 15 true-positives and 4 False-positives using the test-set with 300 fraudulent and 5 000 legitimate claims.

Inconsistency regarding Hit-rates for aggressive models were more surprising. High False-positive rates were expected, however fitting an aggressive model twice and getting a Hit-rate difference of up to 30%-units (Figure 8) was alarming. This lack of consistency would make practical application difficult.

6.2 Practical applications

The aim of using a automatic method for fraud classification is to have an initial filter before manual investigations are made. This filter can be generally be one of two types.

The first aims to reduce the large total amount of claims to a smaller set where all fraudulent cases are included. It could be used in conjunction with manually analysing sets of claims, since it effectively cuts the workload while having low risk of missing fraudulent cases during filtering. Our aggressive models were designed to be of this type. Post filtering we are left with a high rate of both classes, and therefore this filter works best when the cost of investigation/analysis is low and the process is time-effective. Overall aggressive models fit the objectives of this kind of filter rather poorly. They can attain high Hit-rates easily but judging from Figure 10, for rates below 95% both conservative

models have substantially lower False-positive rates. As such, contrary to initial hypothesis conservative models fit this type of priority better than aggressive.

The second type of filter is aimed at having high certainty that those claims who make it through actually are fraudulent. This would mean that while we risk missing committed insurance fraud, we can be very certain that classifications are correct. Our conservative models were designed to fit this type. After filtering we are left with a small set of cases, of which we can allow high investigation-costs. The conservative models fit these goals relatively poorly. Even at low Hit-rates like 5% and 1% approximately a third of the "confidently fraudulent" claims are incorrectly classified..

None of the explored models performed as initially hoped, but that could be due to a lack of prior benchmark rather than low metric scores. The Conservative Large model Version 2 fit both filter types objectively better than the rest, even Random forest models. At a Hit-rate of 96% we could reduce the ratio of fraudulent cases i data from 1:16 to 1:5, significantly lowering the potential workload. Additional information about realistic cost ratios would be required to make a confident assessment if this model would acceptably fit the second filter type.

The true cause of problematic class separation and seemingly random jittering, evident in probability convergence plots, is unknown. However we can form a few hypothesis based on our results.

The first is that insurance fraud detection is a multinomial classification problem as opposed to the binomial problem explored in this thesis. This is plausible due to fraud in general being a behaviour with the goal to adapt and avoid detection. Considering each scheme to commit fraud as a separate class, logically the fewer individuals in a specific class, the harder it will be to classify correctly. This leads to an incentive that individuals who commit fraud would want to diversify them selves from the rest of fraudsters. Naturally insurance fraud would then best be considered as being split in to many classes with unique characteristics. Jittering present in probability convergence plots could possibly be a consequence of attempting to fit multiple diverse patterns to a single class. During later iterations of the boosting algorithm, correcting the fit for a specific fraud-subclass might worsen the current fit of one or more separate sub-classes. If this hypothesis is true, a solution would be to switch to methods that can handle multinomial classification.

A second hypothesis related to the first is that some of the defining characteristics of insurance fraud affect other variables than those present in provided data. Examples could be geographical location and whether or not the parties involved in the crash have some social relation. Consider an extreme case where all fraudulent car incidents are committed in a particular area, but all other parameters are as diverse as in the remaining claims. If we do not include geography as a feature variable we can not split the feature space in any way that this simple characteristic is isolated. It does not depend on tree depth, number of iterations or how class balance. In situations like this or when access to data is limited, machine learning is generally not appropriate due to being heavily data-focused.

Lastly at least a portion of poor performance could be linked to "estimated number of unknown cases in data", in Swedish: "mörkertal i data". It is a problem that arises when the response in data is based on an external or previous classifier. Consider a true data-set with 90 legitimate claims and 10 fraudulent. One model, manual investigation, correctly classifies 90 legitimate and 8 fraud, resulting in 98% accuracy and 80% Hit-rate. Since true values are unknown this model might be assumed as fact. A data set is then constructed from the results, containing 92 legitimate claims and 8 fraud. Fitting a second model to this data might classify the original data correctly, score a Hit-rate of 100% but only 98% accuracy. The two "incorrectly" classified claims are the "unknown cases in data". Any model or method used to classify data always runs the risk of identifying previously unknown cases and consequently suffer a reduced performance score. This is a present problem in modern machine learning as well as applied statistics in general.

6.3 Expansions and Corrections

A major issue throughout our analysis is the lack of knowledge on realistic metric values. Mainly the costs associated with investigation of a claim and the amount insurance companies budget for combating insurance fraud. Sources like the annual report "Försäkringsbedrägerier i Sverige" by *Svensk Försäkring* present numbers like total amount paid for claims, total paid for detected fraud, rate of fraud, etc. but not the amount spent on investigation. Reproducing this study with knowledge of realistic values would allow for more specific model calibration, mainly the probability threshold.

Expanding further on costs a more practical study on effectiveness of real-time filtering would be interesting. Since investigators are most likely full-time

employees and claims with high fraudulent probability most likely occur non-uniformly, workload will be non-uniform as long as the classification threshold is constant. Assigning a dynamic threshold would allow for a more even workload, but consequently affect Hit-rate and False-positive ratio in an unknown way.

7 Conclusion

This study has served as an introduction to tree based machine learning methods, mainly gradient boosting, with a case study of Auto insurance fraud detection. We have presented and explained the background theory of these methods and the thought process behind parameter calibration. The analysis also serves as an example of how one can measure performance when application specific goals do not coincide with the default mathematical optimisation. We provide insight on how to define and interpret custom metrics, as well as how to use them in junction with pre-defined/-implemented statistical methods. The results of this study are mostly in line with previous research of the applied machine learning methods. Results indicate that the boosting method is an improvement from using single trees, and that the correction/convergence aspect of the algorithm works as intended. Regarding slow vs fast convergence this study did not find conclusive evidence that one is better than the other. For this specific application results indicate that deep multi-variable interactions are necessary to suppress the False-positive ratio. Some models presented in this study are shown to be profitable when the cost of investigation a claim is 2/3 of the claim amount to be paid, with more models following for higher values. The main issue and thus possible expansion of this study is realistic information about target values for defined metrics as well as investigation costs.

Appendix

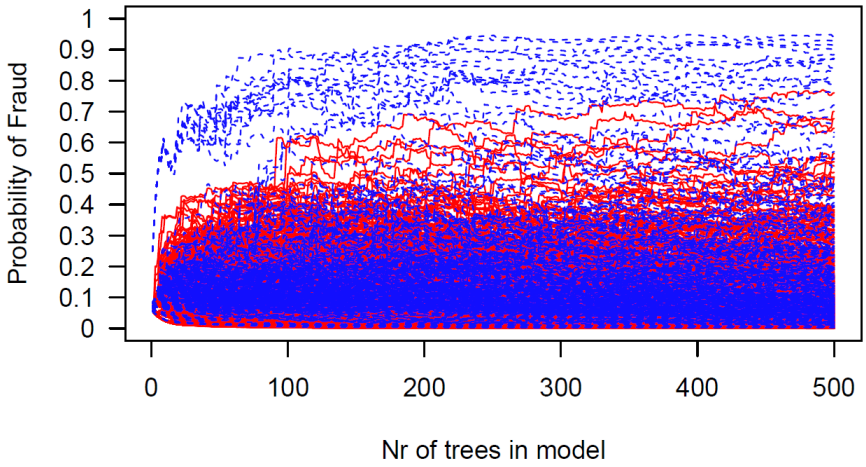

Figure 21: For model CL V2, probability of classifying a claim as fraudulent. Blue lines are actual fraudulent claims, red are legitimate. Each line represents an observation.

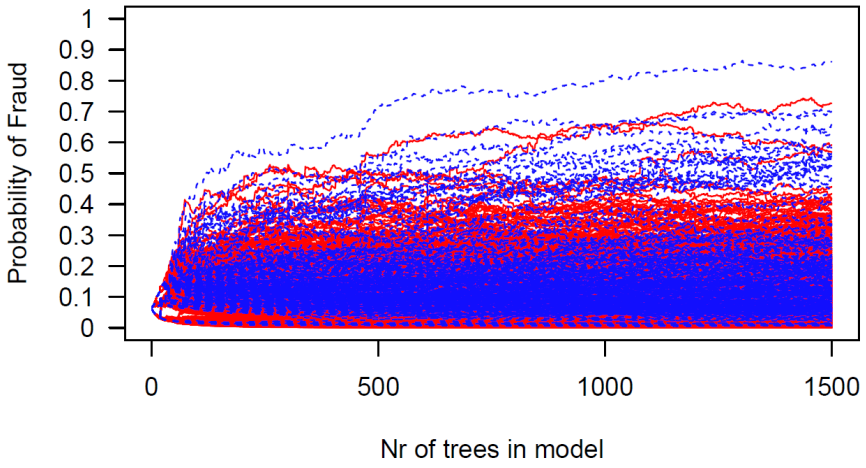

Figure 22: For model CS V2, probability of classifying a claim as fraudulent. Blue lines are actual fraudulent claims, red are legitimate. Each line represents an observation.

References

[1] Jerome Friedman: Greedy Function Approximation: A Gradient Boosting Machine. 1999. Stanford Press.
https://statweb.stanford.edu/~jhf/ftp/trebst.pdf

[2] Leo Breiman, Jerome Friedman, Charles J. Stone, R.A. Olshen: Classification and Regression Trees. 1984. Taylor & Francis.

[3] Jerome Friedman: Stochastic Gradient Boosting. 1999. Stanford Press.
https://statweb.stanford.edu/~jhf/ftp/stobst.pdf

[4] Trevor Hastie, Robert Tibshirani, Jerome Friedman: The Elements of Statistical Learning (Second Edition, corrected 12th printing). 2017. Springer.

[5] Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani: An Introduction to Statistical Learning (7th printing). 2013. Springer.

[6] Greg Ridgeway: Generalized Boosted Models: A guide to the gbm package. 2007. R Vignette.
https://cran.r-project.org/web/packages/gbm/vignettes/gbm.pdf

[7] Tin Kam Ho: Random Decision Forests. 1995. Proceedings of the Third International Conference on Document Analysis and Recognition - Volume 1. IEEE Computer Society
https://web.archive.org/web/20160417030218/http://ect.bell-labs.com/who/tkh/publications/papers/odt.pdf

[8] J.R. Quinlan: Induction of Decision Trees. 1986. Springer.
https://link.springer.com/content/pdf/10.1007/BF00116251.pdf

[9] Svensk Försäkring: Försäkringsbedrägerier i Sverige 2017. 2017.
https://www.svenskforsakring.se/globalassets/rapporter/forsakringsbedragerier/forsakringsbedragerier-i-sverige-2017.pdf

[10] Oracle: Data on Auto insurance fraud for data-mining training. 2010.
https://blogs.oracle.com/datamining/fraud-and-anomaly-detection-made-simple