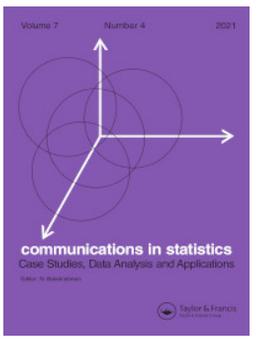

Taylor & Francis
Taylor & Francis Group

Communications in Statistics: Case Studies, Data Analysis and Applications

ISSN: (Print) (Online) Journal homepage: https://www.tandfonline.com/loi/ucas20

Automobile insurance fraud detection

Mark Anthony Caruana & Liam Grech

To cite this article: Mark Anthony Caruana & Liam Grech (2021) Automobile insurance fraud detection, Communications in Statistics: Case Studies, Data Analysis and Applications, 7:4, 520-535, DOI: 10.1080/23737484.2021.1986169

To link to this article: https://doi.org/10.1080/23737484.2021.1986169

COMMUNICATIONS IN STATISTICS: CASE STUDIES, DATA ANALYSIS AND APPLICATIONS
2021, VOL. 7, NO. 4, 520–535
https://doi.org/10.1080/23737484.2021.1986169

Taylor & Francis
Taylor & Francis Group

Automobile insurance fraud detection

Mark Anthony Caruana[a] and Liam Grech[b]

[a]Department of Statistics and Operations Research, Faculty of Science, University of Malta, Msida, Malta; [b]Phoenix GSB, Malta

ABSTRACT

The risk of incurring financial losses from fraudulent claims is an issue concerning all insurance companies. The detection of such claims is not an easy task. Moreover, a number of old-school methods have proven to be inefficient. Statistical techniques for predictive modelling have been applied to detect fraudulent claims. In this article, we compare two techniques: Artificial neural networks and the Naïve Bayes classifier. The theory underpinning both techniques is discussed and an application of these techniques to a dataset of labelled automobile insurance claims is then presented. Fraudulent claims only constitute a small percentage of the total number of claims. As a result, datasets tend to be unbalanced. This in turn causes a number of problems. To overcome such issues, techniques which deal with unbalanced datasets are also discussed. The suitability of Neural Networks and the Naïve Bayes classifier to the dataset is discussed and the results are compared and contrasted by using a number of performance measures including ROC curves, Accuracy, AUC, Precision, and Sensitivity. Both classification techniques gave comparable results with the Neural network giving slightly better results than the Naïve Bayes classifier on the training dataset. However, when applied to the test data, the Naïve Bayes classifier slightly outperformed the artificial neural network.

KEYWORDS

Fraud detection; Insurance claims; Artificial neural networks; Naïve Bayes classifier; Unbalanced datasets

1. Introduction

Insurance fraud refers to a claim made for a loss, where the policyholder (insured) has reported an accident that has not occurred or has exaggerated the extent of the circumstances. Moreover, it ranges from professional fraud, where accidents are planned by organized crime rings, to opportunistic fraud. In the US, the property and causality insurance industry estimates that, each year, 5–10% of claim costs are due to fraud, with almost a third of companies believing that this figure is as high as 20%. In Europe, it is estimated that 10% of all claim expenditure is due to fraudulent claims and particularly, in the UK roughly €2.2 bn is lost each year due to undetected fraud. Insurance fraud is also present in the Maltese Islands. In 2016, an organized automobile insurance fraud ring

CONTACT Mark Anthony Caruana mark.caruana@um.edu.mt Department of Statistics and Operations Research, Faculty of Science, University of Malta, Msida, Malta

was uncovered by Maltese police, where 22 people faced charges of defrauding 8 insurance companies of hundreds of thousands of euros by planning traffic accidents and repeating claims under different names between the years 2009 and 2013.

There are a number of different models and algorithms that have been studied for various types of fraud detection. Bolton, Richard, and Hand (2002) gave a review of the techniques for statistical fraud detection, while emphasizing the need for companies to explore the vast amounts of data that is available to them. Clifton et al. (2010) commented that there is too much emphasis on complex techniques such as Artificial neural networks (ANNs), and suggested that faster algorithms such as Naïve Bayes (NB) should be explored for fraud detection. Ngai et al. (2011) provided an exhaustive overview of the literature for fraud detection, where classification was found to be the most popular method for tackling the fraud detection problem, while Logistic models, ANNs, Bayesian Belief Networks and Decision Trees were observed to be the preferred techniques. For automobile insurance fraud, it was noted that Logistic models were the most popular, followed by ANNs, Naïve Bayes, Bayesian belief networks, and Probit models, respectively. The aim of this article is to compare the performance of ANNs and the Naïve Bayes classifier on automobile insurance fraud detection.

The rest of the article is structured as follows: in Sec. 2, we introduce some basic notation, in Secs. 3 and 4 the theory behind Neural Networks and Naïve Bayes is discussed, in Sec. 5, we apply the two techniques to a data set and compare the results, finally, Sec. 6 contains some concluding remarks.

2. Context

Let $\mathbf{X} = (X_1, \ldots, X_p)$ be a random vector where each random variable X_i represents an attribute of a claim (e.g., age of policy holder, past number of claims, the presence of a police report, etc.) and can be continuous or discrete. Moreover, let $\mathbf{x}$ denote an observed value of the random vector $\mathbf{X}$, which can also be referred to as the input of a given model. Let t denote what is known as the target variable. An observation can belong to one of K classes, represented by the class labels $\mathcal{C}_0, \ldots, \mathcal{C}_{K-1}$. C shall denote the random variable representing a general class label $\mathcal{C}_k$. The goal is that of formulating a classifier that can assign observations to the correct class by applying a classification rule. Note that $P(C = \mathcal{C}_k \mid \mathbf{X} = \mathbf{x})$ denotes the posterior probability of the class $\mathcal{C}_k$. Throughout this article, this may be replaced by $P(\mathcal{C}_k \mid \mathbf{X} = \mathbf{x})$ for ease of notation. For fraud detection, the aim is to classify a set of claims correctly. When $K = 2$, the scenario is reduced to a *two-class* problem which is appropriate for fraud detection. Therefore, the target variable t corresponds to the "fraudulent" class $\mathcal{C}_1$ when $t = 1$ and the "legitimate" class $\mathcal{C}_0$, when $t = 0$.

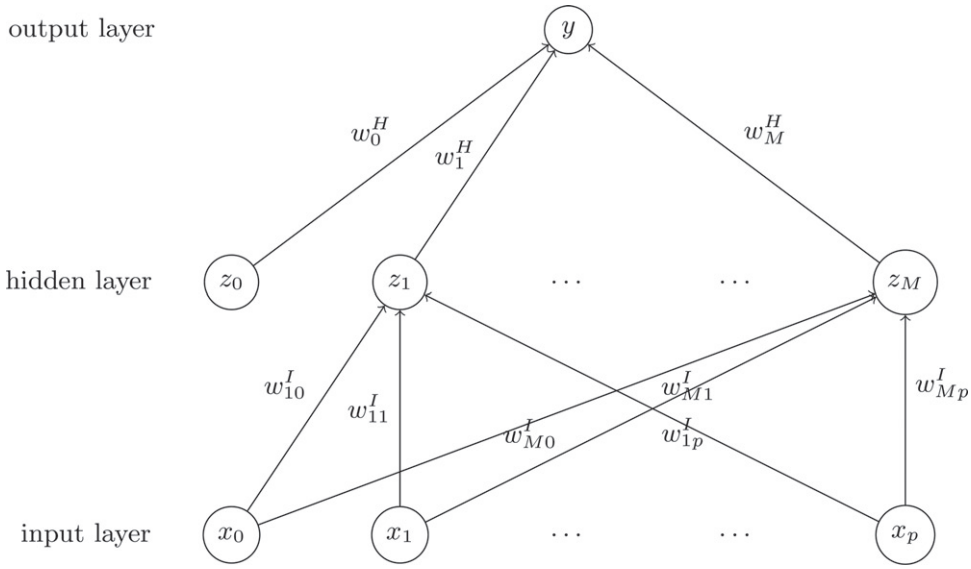

Figure 1. NN with one hidden layer and one output unit.

We will assume that we are in possession of the data set which contains n observations. A section of the this dataset, which contains N observations will be called the *training set* and will be used to fit a model to this data set. The quality of the model is then assessed using the other section of the data set consisting of $n - N$ observations. This is called the testing set.

3. Artificial neural networks

Neural networks (NN) consist of one input layer and one output layer, with an arbitrary number of hidden layers. Within each hidden layer we have an arbitrary number of hidden units and a number of continuous activation functions to approximate a continuous mapping between the input and output space. In this article, we will primarily consider a NN with a single hidden layer. Figure 1 illustrates a NN with a single hidden layer and an output layer.

In our context, the output y is a function of the input variables $x_1, \ldots, x_p$ represented by the vector $\mathbf{x}$ and all the weights (parameters) of the network are represented by the vector $\mathbf{w}$. The nodes in the input layer represent the p variables of some observation from the training set. The first step is to form M weighted sums in terms of the input variables , where the jth denoted by a_j is expressed as follows:

$$a_j = w_{j0}^I + \sum_{i=1}^{p} w_{ji}^I x_i = \sum_{i=0}^{p} w_{ji}^I x_i \tag{1}$$

for $j = 1, \ldots, M$ and the weights w_{ji}^I are associated with the links between the input layer and the first hidden layer. The bias term w_{j0}^I is absorbed into the summation and is given a fixed dummy input value of 1. The subscript *ji* implies that the weight belongs to the link that starts from the i^{th} node in the input layer and ends at the *j*th node in the hidden layer. The hidden units (activations) of the hidden layer are denoted by z_j and are calculated by applying an activation function h^H for the hidden layer to (1) as follows

$$\text{Hidden Layer} : z_j = h^H \left(\sum_{i=o}^{P} W_{ji}^I x_i \right) \tag{2}$$

Should one require, it is possible to insert more hidden layers. However, in this paper we will primarily concentrate on NN with only one hidden layer.

For the case of one output node, the output y can be expressed as follows:

$$\text{Output layer (1 unit)} \ : y = h^O \left(\sum_{s=0}^{S} w_s^H z_s \right) \tag{3}$$

where one weighted sum has been constructed in this case and w_s^H denotes the weights between the hidden layer and the single output unit. In general, the activation functions for hidden layers and for the output layer need not be the same and the choice of function depends on a number of factors, such as the type of data or the problem to be solved by the network.

In literature one finds a number of activation functions. However, the most frequently used are the Logisitc sigmoid functions and the hyperbolic tangent. The former is usually denoted by σ and is defined as follows:

$$\sigma(a) = \frac{1}{1 + \exp(-a)}, \text{ for } a \in \mathbb{R} \tag{4}$$

while the latter is defined as follows:

$$\tanh(a) = \frac{\exp(a) - \exp(-a)}{\exp(a) + \exp(-a)}, \text{for } a \in \mathbb{R} \tag{5}$$

These activation functions are used often because they are both continuous and differentiable functions. Hence they fully satisfy the assumptions of the theorem below which is central in the theory of NNs.

Theorem 1 (Universal Approximation theorem). Let h be a continuous sigmoidal function. Then finite sums of the form

$$G(\mathbf{x}) = \sum_{j=1}^{M} \alpha_j h(\mathbf{w}_j^{\mathrm{T}} \mathbf{x} + w_{j0})$$

where $\mathbf{x}, \mathbf{w}_j \in \mathbb{R}^p$ and $\alpha_j, w_{j0} \in \mathbb{R}$ are fixed, are dense in $C[I^p]$. That is, given any $f \in C[I^p]$ and $\varepsilon > 0$, $G(\mathbf{x})$ exists such that

$$|G(\mathbf{x}) - f(\mathbf{x})| < \varepsilon \; \forall \, \mathbf{x} \in I^p$$

The proof of this theorem can be found in Cybenko (1989).
It is important to note that the implication of this theorem is that a NN with a single hidden layer and continuous sigmoidal activation functions in the hidden layer, can be shown to be mathematically sufficient to give an approximation of the required nonlinear mapping; however, an exact representation is not attainable. As discussed in Hornik (1991), the interpretation of this is that a model with a single hidden layer will only be unsuccessful if not enough hidden neurons are used or if the target variable does not depend on the given data in the first place. Also the output for the theorem is taken as a single linear output; however, this can be generalized to sigmoidal output activation functions as well as multiple output nodes as discussed in Ripley (1994), and Hassoun (1995).

Now that we have defined with a certain degree of clarity a NN and its components, we will next discuss how the weights can be estimated by using the data in the training set. The algorithm used is commonly known as *error back-propagation*. This supervised learning technique makes use of the available training set $\mathcal{D} = \{(\mathbf{x}^{(n)}, t^{(n)})\}_{n=1}^{N}$ in order to obtain an error value or loss for each observation with respect to its target value, which shall be used to construct an error function with respect to the weights and biases of the model. This method sheds light on why so much emphasis was made on the differentiability of the activation functions in each layer, since differentiation plays an important role in the algorithm. In this algorithm, the following terms will be used:

- *Input Signals*: The attributes $(x_1^{(n)}, \ldots, x_p^{(n)})$ of the n^{th} observation $\mathbf{x}^{(n)}$ are called input signals.
- *Output Signals*: The output for a network with one output unit gives an output signal $y^{(n)}$.
- *Forward Pass*: The input signals are propagated forward through the network, from the input layer, to the hidden layers and finally reaching the output layer.
- *Loss function*: The output signals obtained from the forward pass and the corresponding target variable for that observation are used to obtain the loss function which shall be denoted by L and is discussed below.
- *Error Signal & Backward Pass*: The loss function L obtained after the forward pass is then differentiated with respect to the weights and biases between the final hidden layer and the output layer and by means of the error back-propagation algorithm discussed below, the gradients (error signals) are said to be propagated backwards through the network.
- *Error Function*: This shall be denoted by E and will be discussed below.

As a consequence of theorem 1, a NN with one hidden layer and a finite number of hidden units is sufficient to solve a number of problems. This can be extended by adding more hidden layers, each consisting of an arbitrary number of hidden units and continuous non-linear activation functions in order to approximate a continuous mapping between the input space and output space. Therefore, if in general an error function E is expressed in terms of the output signals of the network, which in turn can be expressed in terms of the input signals and parameters of the network, then E can be expressed simply as a function of the parameters of the model. Recall the target variable for binary classification $t \in \{0, 1\}$ for one output node. Generally for the output of a NN, the error function can be expressed as follows:

$$E(\mathbf{w}) = E(y(\mathbf{x}, \mathbf{w}), t) \tag{6}$$

where $\mathbf{w} = (w_1, \ldots, w_W)$ is a vector of all the parameters of a network consisting of W weights. This representation of the error function suggests that the output signals and the target variable shall be used to calculate a form of error of the model, while keeping in mind that the aim shall be to minimize this error with respect to the parameters of the network.

Let $\nabla E(\mathbf{w})$ denote the gradient of the error function such that

$$\nabla E(\mathbf{w}) = \begin{bmatrix} \frac{\partial E(\mathbf{w})}{\partial w_1} \\ \frac{\partial E(\mathbf{w})}{\partial w_2} \\ \vdots \\ \frac{\partial E(\mathbf{w})}{\partial w_W} \end{bmatrix} \tag{7}$$

The objective function may have local minima as well as a global minimum at which $\nabla E(\mathbf{w}) = 0$ is satisfied since it is a non-convex function. Therefore, there is not a unique value for $\mathbf{w}^*$ that satisfies $\nabla E(\mathbf{w}) = 0$, which corresponds to a global minimum of the objective function. This is an important property to take into consideration when choosing a suitable algorithm for adjusting the parameters of the network.

The method of *error back-propagation* provides an efficient algorithm for calculating the derivatives of the error function with respect to the weights and depends heavily on the properties of the NN. Bishop (1995) considered the output of the model as an overall network function and by implementing differentiable activation functions at each layer of the network, it is possible to differentiate this output function with respect to the weights and biases. If the error function is expressed in terms of the output function, then it is also differentiable with respect to the weights and biases, which allows us to use the gradient descent method. Consider a NN with an arbitrary number of hidden layers and a single output node for the two-class problem. Consider a general differentiable error function E in terms of the output function y and the target variable t. The derivation shall focus on the loss function L. For the n^{th}

observation, this can be defined as follows:

$$L = -\left(t^{(n)} \ln y^{(n)} + \left(1 - t^{(n)}\right) \ln\left(1 - y^{(n)}\right)\right) \tag{8}$$

Since it can be shown that E can be expressed as the sum of L over all observations of the training set through the equation:

$$E = \frac{1}{N} \sum_{n=1}^{N} L(y^{(n)}, t^{(n)}) \tag{9}$$

and each observation completes a forward pass and a backward pass through the network. It can be shown that the derivatives for the overall error function E with respect to the weights of the output layer for a network with a single output unit can be given as follows:

$$\frac{\partial E}{\partial w_s} = \sum_{n=1}^{N} \frac{\partial L(y^{(n)}(\mathbf{x}^{(n)}, \mathbf{w}), t^{(n)})}{\partial w_s} \tag{10}$$

while for the hidden layer this can be expressed as follows:

$$\frac{\partial E}{\partial w_{sc}} = \sum_{n=1}^{N} \frac{\partial L(y^{(n)}(\mathbf{x}^{(n)}, \mathbf{w}), t^{(n)})}{\partial w_{sc}} \tag{11}$$

Finally, to obtain $\mathbf{w}^*$ one may use techniques such as the stochastic gradient descent method.

4. Naïve Bayes classifier for categorical attributes

In this section, we assume that the attributes $X_1, \ldots, X_p$ are conditionally independent of each other. Under this assumption, Bayes' theorem can be applied to obtain

$$P(\mathcal{C}_k \mid \mathbf{X} = \mathbf{x}) = \frac{q(\mathcal{C}_k) \prod_{i=1}^{p} P(x_i \mid \mathcal{C}_k)}{P(\mathbf{x})} \tag{12}$$

or alternatively, since $P(\mathbf{x})$ is the same for all classes, it can be considered as a constant of proportionality such that (12) becomes

$$P(\mathcal{C}_k \mid \mathbf{X} = \mathbf{x}) \propto q(\mathcal{C}_k) \prod_{i=1}^{p} P(x_i \mid \mathcal{C}_k) \tag{13}$$

This assumption is considered to be "Naïve" since in practice this is not always the case. More often than not, there may in fact be correlations between the attributes such that this conditional independence does not hold. Nonetheless, it has been shown by Domingos and Pazzani (1997) and Hand and Yu (2001) that despite the presence of dependencies between the attributes, the NB classifier can perform as well as other more complex classifiers in terms of classification accuracy.

The training phase for the NB classifier corresponds to maximum likelihood estimation of the required parameters. The required parameters in this case are the prior probabilities for each class and the univariate marginal class-conditional probabilities for each observation and each attribute of the training set $\mathcal{D}$. The form of the class-conditional probabilities depends on the type of variables making up the training set. Since the available data consists only of discrete-valued variables, it is enough to consider the cases when the attributes are categorical, i.e., where an attribute $X_i \in \{1, \ldots, M\}$. The NB model can be extended for continuous attributes by considering a Normal distribution for the class-conditional probabilities as disucssed in Heckerman and Geiger (1995) or by discretizing the values as discussed in Dougherty, Kohavi, and Sahami (1995); however, this is not required for the available data since the attributes for the claims are all discrete-valued.

Consider the case when an attribute X_i is a categorical variable such that the attribute can take $M > 2$ possible values, i.e. $X_i \in \{1, \ldots, M\}$. Many algorithms (such as NN) handle categorical attributes by using a one-of-M coding scheme such that the single variable is transformed into M binary attributes, one for each possible value of the categorical attribute. For example, if a categorical variable can take values $\{1, 2, 3\}$ and for a specific observation the observed value is 2, then this is coded as $(0, 1, 0)$, i.e., one unity and the rest zero. However, as stated in Barber (2010), the issue with this coding scheme is that it violates the independence assumption of the NB model, since the three derived attributes are clearly dependent.Therefore, for the NB classifier, multi-level factors shall not be coded using a one-of-M coding scheme, but by the usual coding scheme. Since a categorical variable X_i can take M possible values, the likelihood that, conditioned on the class, the observed attribute x_i takes a value m from the M possible values is denoted as

$$P(X_i = m \mid t) = \pi_{it}^{(m)} \tag{14}$$

such that $\sum_{m=1}^{M} P(X_i = m \mid t) = 1$.

Consider that there are $p_2 \leq p$ categorical attributes in the given training set $\mathcal{D} = \{(\mathbf{x}^{(n)}, t^{(n)})\}_{n=1}^{N}$, where p is again the total number of attributes in the training set. Conditioning on the target variable t, each observation is iid such that the likelihood is given as follows:

$$\mathcal{L} = \prod_{n=1}^{N} P(\mathbf{x}^{(n)} \mid t^{(n)}) = \prod_{n=1}^{N} \prod_{i=1}^{p_2} \prod_{m=1}^{M} \prod_{t=0}^{1} (\pi_{it}^{(m)})^{\mathbb{I}\{x_i^{(n)}=m\}\mathbb{I}\{t^{(n)}=t\}} \tag{15}$$

and the class-conditional log-likelihood is given as follows:

$$\ell = \sum_{n=1}^{N} \sum_{i=1}^{p_2} \sum_{m=1}^{M} \sum_{t=0}^{1} \mathbb{I}\{x_i^{(n)} = m\}\mathbb{I}\{t^{(n)} = t\} \ln \pi_{it}^{(m)} \tag{16}$$

Now to obtain the required estimates for $\pi_{it}^{(m)}$, the Lagrange multiplier method may be used as carried out by Barber (2010) to ensure that probabilities

sum to one such that

$$\ell = \sum_{n=1}^{N}\sum_{i=1}^{p_2}\sum_{m=1}^{M}\sum_{t=0}^{1}\mathbb{I}\{x_i^{(n)}=m\}\mathbb{I}\{t^{(n)}=t\}\ln\pi_{it}^{(m)} + \sum_{t=0}^{1}\sum_{i=1}^{p_2}\lambda_{it}\left(1-\sum_{m=1}^{M}\pi_{it}^{(m)}\right)$$

and differentiate with respect to $\pi_{it}^{(m)}$ and equate to 0 to obtain

$$\lambda_{it} = \sum_{n=1}^{N}\frac{\mathbb{I}\{x_i^{(n)}=m\}\mathbb{I}\{t^{(n)}=t\}}{\pi_{it}^{(m)}} \tag{17}$$

Then the maximum likelihood estimators for $\pi_{it}^{(m)}$ are given as follows:

$$\hat{\pi}_{it}^{(m)} = \frac{\sum_n \mathbb{I}\{x_i^{(n)}=m\}\mathbb{I}\{t^{(n)}=t\}}{\sum_{m',n'}\mathbb{I}\{x_i^{(n')}=m'\}\mathbb{I}\{t^{(n')}=t\}} \tag{18}$$

which corresponds to the number of times a categorical variable X_i takes the m^{th} value out of the M possible values for a certain class.

Similarly, it can be shown that for binary attributes, such that $x_i^{(n)} \in (0,1)$ we have that the estimator for π_{it} is given as follows:

$$\hat{\pi}_{it} = \hat{P}(X_i = 1 \mid t) = \frac{\sum_n \mathbb{I}\{x_i^{(n)}=1, t^{(n)}=t\}}{\sum_n\{\mathbb{I}\{x_i^{(n)}=0, t^{(n)}=t\} + \mathbb{I}\{x_i^{(n)}=1, t^{(n)}=t\}\}} \tag{19}$$

which corresponds to the number of times $X_i = 1$ for a specific value of t (i.e. for a specific class $\mathcal{C}_k$, with $k \in \{0,1\}$) divided by the total number of observations in class $\mathcal{C}_k$. The estimator for the prior probabilities is given as

$$\hat{q}_k = \frac{N_k}{N} \tag{20}$$

which corresponds to the ratio of the number of observations in class $\mathcal{C}_k$, denoted by N_k, and the total number of observations in the training set N. In practice, this is calculated by counting the number cases from each class by observing the target value $t^{(n)}$ for each observation, hence the target variable is used in the derivation.

Consider the case when X_i is a binary attribute. As an example, suppose that for the class $\mathcal{C}_0$, the observations in the training set al. have a value of 0 for the specific attribute. This implies that the class-conditional probability $P(X_i = 1 \mid \mathcal{C}_0) = 0$. When plugging this probability into the NB model defined in (13), this would imply that, whenever a new observation $\mathbf{x}$ has a value of 1 for that attribute, then the probability of that observation belonging to class $\mathcal{C}_0$ is 0, while the probability of that observation belonging to $\mathcal{C}_1$ is 1. This prediction is unlikely to be very accurate. The same can be said for the case when the attribute X_i can take more than 2 possible values. The more possible values M that the

attribute can take, the more likely that this may become an issue, since it is more likely that certain possibilities never occur in the training set.

The most popular way of overcoming this issue is known as the Laplacian correction or Laplace estimator. It is assumed that for the training set $\mathcal{D}$, the number of observations N is large enough such that adding a small number of cases for each level of a factor for both classes would be negligible. Let $b \in \mathbb{N}$ denote the number of cases to be added for an attribute-class pair. It is important to note that b cases shall be added for each possible value of the attribute, so for binary attributes, $2b$ must be added to the denominator such that from (19) the Laplace estimator is

$$\hat{\pi}_{it}^{L} = \frac{\sum_n \mathbb{I}\{x_i^{(n)} = 1, t^{(n)} = t\} + b}{\sum_n \{\mathbb{I}\{x_i^{(n)} = 0, t^{(n)} = t\} + \mathbb{I}\{x_i^{(n)} = 1, t^{(n)} = t\}\} + 2b} \tag{21}$$

For categorical attributes, recall that there are $M > 2$ possible values such that Mb shall be added to the denominator. Therefore, from (18) the Laplace estimator is

$$\hat{\pi}_{it}^{(m)L} = \frac{\sum_n \mathbb{I}\{x_i^{(n)} = m\}\mathbb{I}\{t^{(n)} = t\} + b}{\sum_{m',n'} \mathbb{I}\{x_i^{(n')} = m'\}\mathbb{I}\{t^{(n')} = t\} + Mb} \tag{22}$$

Usually, b is taken to be 1 such that there is at least one count for each attribute-class pair in the training set. This prevents the issue of obtaining values of 0 for the marginal probabilities. Since under the NB assumption the marginal probabilities are multiplied, this would also annihilate the other marginal probabilities so it is an issue that needs to be dealt with, especially when the distribution of an attribute is skewed.

5. Data analysis

The data set was provided by Datawatch Angoss and consists of 15,419 claims over the span of two years in the U.S. and contains 25 categorial variables. There are a total of 923 fraudulent claims and 14,496 legitimate claims. This class imbalance poses a number of problems when fitting any classification technique. Before splitting the data into the test set and train set the dimension of the data was reduced by applying a number of chi-squared tests between the classification variable and each of the other 24 categorical variables. The only variables which were kept are Accident area, Sex, Fault, Vehicle category, Vehicle Price, Deductible, Past Number of Claims, Vehicle age, Age of Policy Holder, Police Report Filed, Agent type, Address Change to Claim, Base Policy, and Number of Supplements.

The data set was next randomly divided between the test set and the training set following a partitioning method which is present in Python. This method ensures that both sets have roughly the same class distribution as the original data set. The training set consists of 13,877 observations, 826 of which are

fraudulent, while the test set consists of 1,542 observations with 97 fraudulent observations. The 826 fraudulent observations in the training set are all kept, while random under-sampling is carried out on the majority class in order to reduce the training set to a size of 2,753 claims with a 70:30 ligitimate-to-fraud ratio. Phua, Alahakoon, and Lee (2004) suggested that 60:40 and 50:50 are also acceptable ratios. However, a 70:30 ratio was selected in order to obtain the largest possible sample for the training set. Under-sampling techniques have been discussed extensively in literature and have been shown to give good classifier performance. A number of researchers such as Japkowicz (2000), and Sundarkumar and Ravi (2015), had discussed this technique and its advantages in detail. Diamantini and Potena (2009) stated that the study of class imbalance problems has been a hot topic in machine learning in the recent years. This is because classification techniques tend to give poor results on such datasets.

5.1. NN for fraud detection

One hidden layer with a sigmoidal activation function turned out to be sufficient to provide a good classification for the data set. More hidden layers where attempted however, no improvement in the model accuracy was detected. Thus, a NN with one hidden layer using the hyperbolic tangent activation function and a single output note with a logistic sigmoid activation function was used on the training set. A grid search was carried out on three important hyper-parameters: the learning rate η, the number M of hidden layers, and the number of epochs. It was noticed that for $\eta = 0.01$, a network with 35 hidden neurones for 500 epochs achieved the lowest generalization error and the highest mean AUC.

Figure 2 shows the training error superimposed on the validation error against the number of epochs for a network with 35 hidden neurons. Each of the five plots corresponds to a randomly selected subset of the data set held out

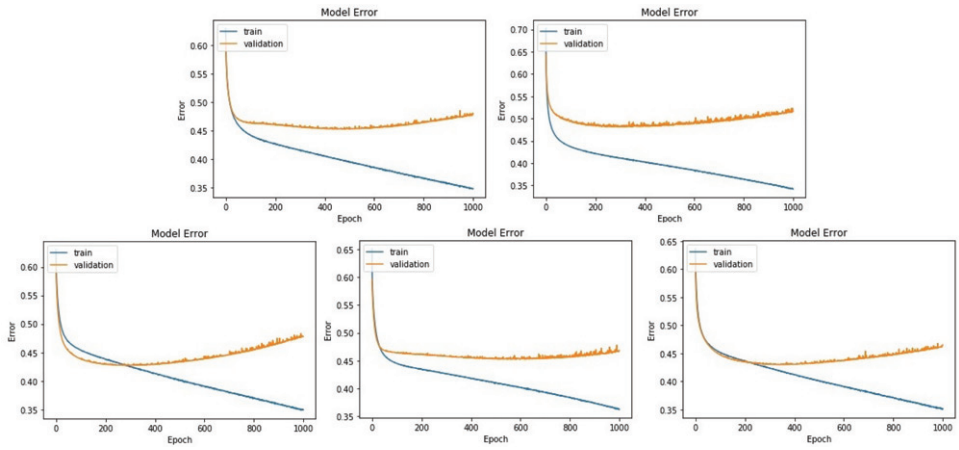

Figure 2. Training error and Validation errors vs Epochs during cross-validation.

Table 1. 2×2 contingency table for the NN on the training set.

		Predicted		
		Fraudulent	Legitimate	Total
True	Fraudulent	$TP = 516$	$FN = 310$	826
	Legitimate	$FP = 246$	$TN = 1681$	1927
	Total	762	1991	2753

Table 2. Performance measures for the NN on the training data.

Accuracy	AUC	Precision	Sensitivity	F1 Score
0.798	0.866	0.677	0.625	0.650

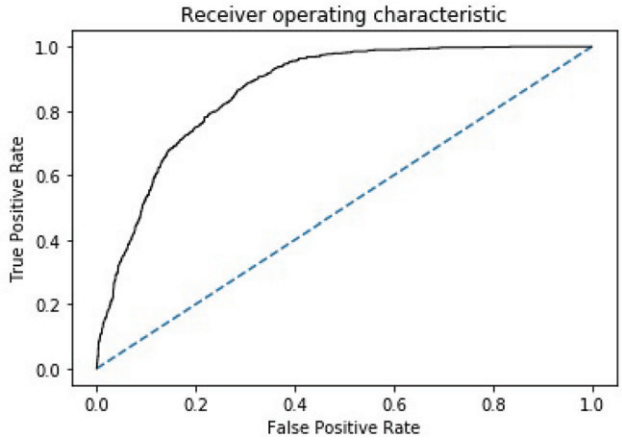

Figure 3. ROC curve for the NN on the training data.

as the validation set during stratified 5-fold cross-validation. These plots show that further training beyond 500 epochs is unjustified as it would lead to the model over-fitting the training data. This is shown by the increase in validation error, corresponding with a decrease in training error.

Next, the NN was applied to obtain predictions for the training data. Table 1 gives the confusion matrix for predictions of the chosen model on the training data. The NN classifies 87.2% of legitimate claims correctly, while classifying 62.5% of fraudulent claims correctly for the training data.

This confusion matrix can be used to calculate the following performance measures in Table 2. The NN performs well on the training data, with an accuracy of 79.8% and an AUC of 0.866. The similar scores for precision, sensitivity and F1 score suggest that the classifier has been trained to find a balance between minimizing the number of false negatives and the number of false positives.

The ROC curve for the NN on the training data can give a visual representation of the performance of the classifier, for different cut-off values. Figure 3 reflects the value for the AUC on the training data since the curve is closer to the

Table 3. 2 × 2 contingency table for NB on the training set.

		Predicted		
		Fraudulent	Legitimate	Total
True	Fraudulent	$TP = 502$	$FN = 324$	826
	Legitimate	$FP = 413$	$TN = 1514$	1927
	Total	915	1838	2753

Table 4. Performance measures for NB on the training data.

Accuracy	AUC	Precision	Sensitivity	F1 Score
0.732	0.801	0.549	0.608	0.577

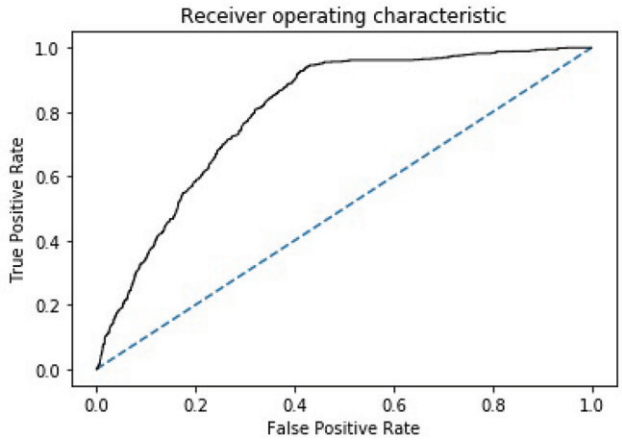

Figure 4. ROC curve for NB on the training set.

left top corner of the ROC space than to the diagonal. The Area under the Curve (AUC) of the ROC curve in Figure 3 is displayed in Table 2 and is equivalent to 0.866.

5.2. Naïve Bayes for fraud detection

The Naïve Bayes model was fit on the training set in order to obtain the posterior distribution of the two classes. The predictions for the training set can be analyzed by means of a confusion matrix. Table 3 shows that 78.6% of legitimate claims have been classified correctly, while 60.8% of fraudulent claims were classified correctly.

From the confusion matrix, the performance measures shown in Table 4 can be derived. The classifier performs reasonably well, giving an AUC of 0.801.

The ROC curve in Figure 4 has an AUC equal to 0.801. This is shown in Table 4. This clearly proves that the ROC curve is well above the diagonal line.

Table 5. 2 × 2 contingency table for NN on the test set.

		Predicted		
		Fraudulent	Legitimate	Total
True	Fraudulent	$TP = 53$	$FN = 44$	97
	Legitimate	$FP = 262$	$TN = 1183$	1445
	Total	315	1227	1542

Table 6. 2 × 2 contingency table for NB on the test set.

		Predicted		
		Fraudulent	Legitimate	Total
True	Fraudulent	$TP = 57$	$FN = 40$	97
	Legitimate	$FP = 331$	$TN = 1114$	1445
	Total	388	1154	1542

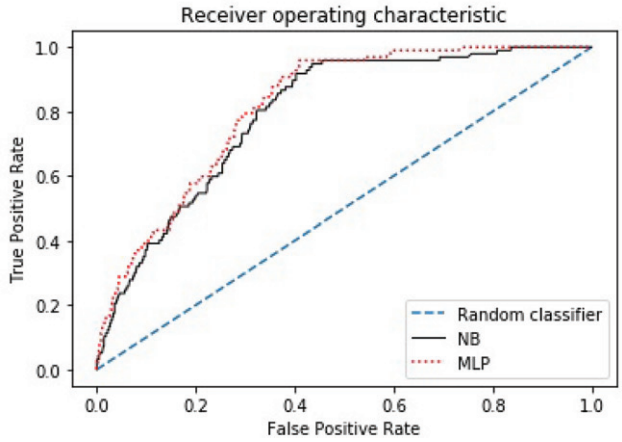

Figure 5. ROC curves for NN and NB on test data.

5.3. NN vs. Naïve Bayes

In this section, we will compare the results obtained from NN and the Naïve Bayes when applied to the test sets. First, the confusion matrix for the NN is obtained from the predictions on the test data. Table 5 shows that 82.9% of legitimate claims were correctly classified, while 54.6% of fraudulent claims were classified correctly for the test data.

Second, the NB model is used to obtain predictions for the test data which can be summarized by the confusion matrix in Table 6. 77.1% of legitimate claims were correctly classified, while 58.8% of fraudulent claims were correctly classified. The NB classifier achieved 4 more true positives than the NN; however, also predicted 69 more false positives.

The ROC curves for the NN and NB models for the test data are superimposed to obtain a visual comparison of the two classifiers' performance on new data points. Figure 5 shows that the performance of the two classifiers is comparable, with the ROC curve for NN slightly dominating the ROC curve

Table 7. Performance of NN & NB on the test data with different training ratios.

Model (training set ratio)	Accuracy	AUC	Precision	Sensitivity	F1 Score
NN (70:30)	0.802	0.817	0.168	0.546	0.257
NB (70:30)	0.759	0.795	0.147	0.588	0.235
NN (94:6)	0.937	0.832	0.500	0.010	0.020
NB (94:6)	0.923	0.503	0.038	0.010	0.016

for the NB classifier. The NB classifier slightly outperformed the NN in terms of correctly classifying fraudulent claims (accuracy). However, this comes at a cost of the precision of the classifier due to the larger number of false positives. The value of the AUC under the ROC curves in Figure 5 for the NN and NB are displayed in Table 7. These are 0.817 and 0.795, respectively.

Furthermore, Table 7 gives a comparison of the performance of the two classifiers on the test set. The NN model and NB model were also fitted on the training data with the original 94:6 legitimate-to-fraudulent distribution, before the under-sampling of the majority class was carried out, in order to highlight the need to cater for this class imbalance in the data. Note that for the models trained on the under-sampled data (70:30 legitimate:fraudulent ratio), the NN model slightly outperformed the NB model with respect to all performance measures except for sensitivity. The higher sensitivity for the NB model came at a cost of predicting more false positives as mentioned.

Upon examining the performance measures for the models fitted on data with the original class imbalance (94:6 legitimate:fraud ratio), the accuracy scores for both the NN and NB model are very high; however, as expected both models failed to detect the majority of fraudulent claims. This confirms that the under-sampling technique used was justified and effective.

6. Conclusion

In this article, the performance of NN and Naïve Bayes classifiers was compared when applied to an Insurance fraud detection dataset. Before implementing these classifiers, the data were preprocessed by selecting significant variables for the study and under-sampling the majority class to overcome the class imbalance issue. The two classifiers were then compared by considering a number of performance measures as well as the ROC curve. Model selection by cross-validation was carried out for the NN. Model selection and training time for the more complex NN proved to be a lengthy process, in contrast to the simpler NB classifier. The NN performed better on the training data than the NB model; however, for new instances (i.e., the test data), the performance of the two classifiers was comparable. The NB model slightly outperformed the NN in terms of classifying new fraudulent instances correctly; however, this was at the cost of predicting many more false positives than the NN, which is undesirable due to the possible cost of investigating claims. The under-sampling technique

proved to be effective for this application as shown by comparing the results from training the models on different class distributions.

ORCID

Mark Anthony Caruana https://orcid.org/0000-0002-9033-1481
Liam Grech http://orcid.org/0000-0001-7842-9972

References

Barber, D. 2010. *Bayesian Reasoning and Machine Learning*. Cambridge: Cambridge University Press.

Bishop, C. M. 1995. *Neural Networks for Pattern Recognition*. New York: Oxford University Press.

Bolton, R., J. Richard, and D. J. Hand. 2002. "Statistical Fraud Detection: A Review." *Statistical Science*, 235-249.

Clifton, P., L. Vincent, K.Smith, and G. Ross. 2010. "A Comprehensive Survey of Data Mining-Based Fraud Detection Research." *arXiv preprint*. arXiv:1009.6119.

Cybenko, G. 1989. "Approximation by Superpositions of a Sigmoidal Function." *Mathematics of Control, Signals and Systems*, 24, 303–314.

Diamantini, C., and D. Potena. 2009. "Bayes Vector Quantizer for Class-Imbalance Problem." *IEEE Transactions on Knowledge and Data Engineering*, 21(5):638–651.

Domingos, P., and M. Pazzani. 1997. "On the Optimality of the Simple Bayesian Classifier Under Zero-One Loss." *Machine learning*, 29(2/3):103.

Dougherty, J., R. Kohavi, and M. Sahami. 1995. "Supervised and Unsupervised Discretization of Continuous Features." *Machine Learning Proceedings*, 194–202.

Hand, D. J., and K. Yu. 2001. "Idiot's Bayes—Not So Stupid After All?" *International Statistical Review*, 69(3):385–389.

Hassoun, M. H. 1995. *Fundamentals of Artificial Neural Networks*. MIT Press.

Heckerman, D., and D. Geiger. 1995. "Learning Bayesian Networks: A Unification for Discrete and Gaussian Domains." *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, 274–284.

Hornik, K. 1991. "Approximation Capabilities of Multilayer Feedforward Networks." *Neural Networks*, 4(2):251–257.

Japkowicz, N. 2000. "The Class Imbalance Problem: Significance and Strategies," International Conference on Artificial Intelligence.

Ngai, E. W. T., Y. Hu, and Y. H. Wong, Y. Chen, X. Sun. 2011. "The Application of Data Mining Techniques in Financial Fraud Detection: A Classification Framework and an Academic Review of Literature." *Decision Support System*, 50(3):559–569.

Phua, C., D. Alahakoon, and V. Lee. 2004. "Minority Report in Fraud Detection: Classification of Skewed Data." *ACM SIGKDD Exploations Newsletter*, 6(1):50–59.

Ripley, B. D. 1994. "Neural Networks and Related Methods for Classification." *Journal of the Royal Statistical Society, Series B* (*Methodological*), 409–456.

Sundarkumar, G. G., and V. Ravi, 2015. "A Nove Hpybrid Undersampling Method for Mining Unbalanced Datasets in Banking and Insurance." *Engineering Applications of Artificial Intelligence*, 37, 368–377.