Accuracy vs. False Negative Rate Trade-off in Mushroom Classification

Guillaume Joyet[1]

[1]University of Basel, Switzerland

Abstract

In this paper, we investigate the consequences of reducing false negatives on overall accuracy in mushroom classification. A logistic regression model and a neural network are trained on descriptive data of about 8000 mushrooms in two classes, edible and poisonous. To decrease false negative rates, we proceed by either varying the threshold for classification or training the model on a weighted loss – both variants are applied on both models. Results show a clear dominance of the neural network model in all examined metrics. An issue encountered in the analysis of the trade-off between false negative rates and accuracy was the size of the data set, which constrains the precision at which to investigate the metrics of interest.

Introduction

There were 133 700 reported cases of poisoning by ingestion of wild mushrooms in the U.S. alone between 2000 and 2018, averaging over 7400 cases per year. While the majority were not serious, there are still dozens of cases resulting in major harm or even death every year [1]. On a worldwide scale, there are an estimated 100 mushroom deaths per year, which however is likely to be an underestimation [2]. Especially in periods when wild mushroom foraging rises in popularity, lacking experience in new foragers tends to lead to an increase of poisoning instances. AI solutions assisting in determining wild mushroom edibility could therefore help limit the number of cases.

In this paper, we examine two mushroom classification methods based on a visual description of a given specimen, represented as a number of categorical variables. The first is simple logistic regression, where we tested two variants of it. The second method we consider are deep neural networks. When classifying into either edible and poisonous, the risk associated with a false negative is considerably higher than for a false positive (the "positive" in this case representing toxicity). This is the case for obvious reasons: while a false positive may be slightly problematic due to it potentially leading to food waste with its associated ethical and economical implications, a false negative on the other hand may lead to death. Therefore, we want our models to reflect the heuristics used by experienced foragers [3]: "the law of the excluded middle". As much as we want to allow as many correct identifications of edible mushrooms as possible, the model should not differentiate between "potentially toxic" and "deadly". Uncertainty should inevitably lead to a classification of the inspected mushroom as toxic, where the challenge lies in defining where the threshold for certainty.

Therefore, the focus of this paper lies on both methods' false negative rates and the impact that reducing this rate has on the overall accuracy of the model. This is where we extended upon recent research [4, 5, 6, 7]: while they analyse specificity and sensitivity of different models (the latter being the complement of the false negative rate), they do not address the asymptotic behaviour of the overall accuracy as specificity goes towards 0. We also compare our results to image-based classification, which has also been investigated by several groups [8, 9, 10] in recent years.

Methods

The goal is to investigate the cost of reducing false negative frequency on the overall accuracy of a model trying to classify mushrooms into toxic and non-toxic categories based on a number of descriptive variables that can be used as predictors.

Data

The data we use is a free data set donated to the UCI Machine Learning repository in 1987 and can be downloaded from their website (). It contains 8124 hypothetical samples of mushrooms from 23 species, each sample labelled as either safe or unsafe to eat and including 22 descriptive variables. These include the appearance of the cap, the gills, and the stalk, odour, habitat, etc. This data set is the reference for research in this area, and other groups examining mushroom classification using descriptors worked with the same data [4, 5, 6, 7].

We split this data randomly into training and test set (4:1). Since all the variables are categorical, we transform them to integer variables. Other than that, the data is tidy and does not contain observations with missing values, thus requires no further transformation.

Algorithms

We examine both classical logistic regression and classification using a deep neural network. For both methods, we compare the results of two different approaches. In the first version, we train the model without any bias and vary the threshold for classifying an observation as being either edible or poisonous. The second approach consists in weighting the classes during training, meaning that the misclassification of one class as the other impacts the loss more than the other way around. Both techniques serve to reduce the false negative rate.

Logistic Regression Logistic regression aims to maximise the log likelihood l of the data with respect to the weight vector $\mathbf{w}$. This is equivalent to minimising the logistic loss, which is the negative log likelihood:

$$l(\mathbf{w}) = -\sum_{i=1}^{n} a(c_i)[c_i \log \sigma(\mathbf{w}^t\mathbf{x}_i) + (1-c_i)\log(1-\sigma(\mathbf{w}^t\mathbf{x}_i))] \tag{1}$$

with c_i the class label of sample i, $\mathbf{x}_i$ the vector containing all predictors of that sample and $\sigma(\mathbf{w}^t\mathbf{x}_i)$

$$\sigma(\mathbf{w}^t\mathbf{x}_i) = \frac{1}{1+e^{-\mathbf{w}^t\mathbf{x}_i}}. \tag{2}$$

the sigmoid function. In eq. 1, a is the weight function, $a(c_i)$ mapping the true class label of sample i to its associated weight (these weights are not to be confused with the weight vector $\mathbf{w}$, which defines the decision hyperplane). In the unweighted scenario, $a(c_i) = 1 \;\; \forall\, c_i$. If we assign different weights to each of the two classes, the misclassification of an observation i with higher $a(c_i)$ will increase the loss more. In our case, having a higher relative weight for the poisonous class will lead to the model learning to avoid false negatives more than false positives.

In this framework, $\sigma(\mathbf{w}^t\mathbf{x}_i)$ is used to represent $P(c_i = 1 \mid \mathbf{x}_i)$. The $\mathbf{w}$ that minimises the loss defines a hyperplane in the feature space, separating the data linearly into two classes, with our certainty about a class attribution being represented by its distance to the hyperplane. We can then classify the data by choosing a threshold t such that sample i is classified as poisonous (i.e. class 1) if

$$\sigma(\mathbf{w}^t\mathbf{x}_i) \geq t \tag{3}$$

where lower t decrease the probability of a false negative. We therefore have two mechanisms to try and minimise false negatives: using an unweighted loss and decreasing t in eq. 3, or keeping t fixed (the typical value is 0.5) while training the model on a loss with increasing a(poisonous).

Neural Network We choose a fully connected neural network with five layers, the output of which is one single value passed through a sigmoid function (eq. 2) and again models the probability $P(c_i = 1 \mid \mathbf{x}_i)$. The exact architecture is found in figure 1. We use ReLU activation and an Adam optimiser with a learning rate of 0.001. The criterion used is a binary cross-entropy loss, which is similar to the logistic loss (eq. 1), just replacing $\sigma(\mathbf{w}^t\mathbf{x}_i)$ by $\mathbf{w}^t\mathbf{x}_\mathbf{i}$ only. We train the model in batches of size 64. The average training loss and test loss

```
class NeuralNetwork(nn.Module):
  def __init__(self):
    super(NeuralNetwork, self).__init__()
    # layers
    self.layer1 = nn.Linear(22, 64)
    self.layer2 = nn.Linear(64, 128)
    self.layer3 = nn.Linear(128, 64)
    self.layer4 = nn.Linear(64, 32)
    self.layer5 = nn.Linear(32, 1)

  def forward(self, x):
    # forward pass
    x = torch.relu(self.layer1(x))
    x = torch.relu(self.layer2(x))
    x = torch.relu(self.layer3(x))
    x = torch.relu(self.layer4(x))
    x = torch.sigmoid(self.layer5(x))
    return x
```

Figure 1: Neural Network Architecture

is calculated at every epoch. The accuracy of the model is calculated only after completion of the training. We use the same division of the data into training and test set as for logistic regression to improve comparability.

We again compare the results obtained by training an unbiased model and subsequently varying the classification threshold to the ones produced by a model trained with a weighted loss.

Metrics

Our two main metrics of interest will be the false negative rate (FNR) and accuracy.

FNR:

$$\begin{aligned} \text{FNR} &= \frac{\text{false negatives}}{\text{false negatives} + \text{true positives}} \\ &= P(\hat{c} = 0 \mid c = 1) \end{aligned} \tag{4}$$

c being the actual class label, $\hat{c}$ the predicted one.

Accuracy:

$$\begin{aligned} \text{Accuracy} &= \frac{\text{true positives} + \text{true negatives}}{N} \\ &= P(\hat{c} = c) \end{aligned} \tag{5}$$

with N the total number of samples.

Software

We work with Python throughout, using both Jupyter notebooks and PyCharm. We use following libraries:

- **scikit-learn** for logistic regression
- **PyTorch** for neural networks
- **NumPy, Pandas** for data operations
- **Matplotlib, seaborn** for plotting

Hardware

This analysis was run on an Apple MacBook Pro (2021), Apple M1 Pro Chip, 16 GB of RAM. However, the algorithms run for this experiment are not computationally expensive, meaning hardware should not affect the results in any way.

	False Negative Rate		**False Positive Rate**		**Accuracy**	
Threshold	LR	NN	LR	NN	LR	NN
0.500	0.0684	0.0000	0.0503	0.0000	0.9409	1.0000
0.250	0.0342	0.0000	0.0635	0.0000	0.9507	1.0000
0.100	0.0051	0.0000	0.1725	0.0000	0.9089	1.0000
0.050	0.0013	0.0000	0.3605	0.0024	0.8140	0.9988
0.010	0.0000	0.0000	0.7006	0.0120	0.6398	0.9938
0.001	0.0000	0.0000	0.8491	0.0467	0.5634	0.9760

Table 1: Results from Varying Thresholds. An observation x_i is assigned to the poisonous class if $\sigma(\mathbf{w}^t\mathbf{x}_i) \geq t$.

	False Negative Rate		**False Positive Rate**		**Accuracy**	
Weight	LR	NN	LR	NN	LR	NN
1	0.0684	0.0025	0.0503	0.0000	0.9409	0.9988
2	0.0482	0.0000	0.0503	0.0000	0.9507	1.0000
5	0.0089	0.0000	0.0599	0.0000	0.9649	1.0000
10	0.0013	0.0000	0.0910	0.0000	0.9526	1.0000
25	0.0000	0.0000	0.1210	0.0000	0.9378	1.0000
50	0.0000	0.0000	0.1401	0.0000	0.9280	1.0000

Table 2: Results from training with a weighted loss function. The weight in the table indicates the relative weight of the poisonous class. Looking at eq. 1, an entry of k in the table implies $a(\text{edible}) = 1$ and $a(\text{poisonous}) = k$.

Results

In the following section, we present the resulting metrics calculated from the test set. Table 1 contains the results – false negative rate (FNR), false positive rate (FPR) and overall accuracy – for both logistic regression (LR) and the neural network (NN) in the first version, i.e. unbiased training with subsequent varying of thresholds for classification. Table 2 contains analogous result for the second version of the models where training is performed with a weighted loss function. We display numbers up to a precision of $0.5 * 10^6$. Confusion matrices for each of the configurations are found in the appendix of this paper. For ease of comparison, we choose the same parameter for both models. We examine thresholds between 0.001 and 0.5, a lower number implying a stronger bias towards classifying an observation as toxic. In the weighted version, relative weights for the poisonous class range from 1 (i.e. as if unweighted) to 50. Higher weights lead to higher cost of misclassification for false negatives, and hence bias the model towards the poisonous class. The threshold for the second variant is always 0.5.

The results show the expected tendencies. When lowering the threshold value or increasing the weight, false negative rates go down while false positive rates go up. The accuracy may go up at first, but then decreases again for more extreme thresholds and weights. The results for the two separate methods are discussed further in the sections below.

Logistic Regression With thresholding, the first tested value to reach a FNR of zero considering our precision is a threshold of 0.01. However, the corresponding FPR and accuracy are around 70% and 64%, respectively. The threshold value with highest accuracy (95%) is 0.025, where the FNR however is about 3.4%. The weighted version reaches a FNR of zero for a relative weight of 25, with FPR of 12% and accuracy of 94%. This variant reaches its highest accuracy (out of the weights we consider) of 95% at a relative weight of 2, with FNR of 4.8% and FPR of 5.0%.

Neural Network These results look quite different. The thresholding variant produces FNR of zero (at the considered precision) for all threshold values. FPRs are also mostly zero and go slightly up (at a maximum of 1.9%) for very low thresholds. Consequently, the overall accuracy is one for most threshold, and goes descreases slightly (to 99%) the lowest thresholds. The weighted scenario produces similar results. FNRs and FPRs are zero for almost all weightings of the poisonous class and accuracy therefore mostly one. The only exception is a FNR of 0.2% with a weight of 1. It is important to note that the model of the first variant with threshold 0.5 and the model of the second variant with weight 1, although having the same hyperparameters, were trained separately. The non-deterministic nature of stochastic gradient descent therefore allows for this small discrepancy in results.

Discussion

Our results display a clear superiority of neural networks over classical logistic regression. For logistic regression, out of the configurations we investigated, the weighted variant yielded better results. The relative weight of choice is 25 with zero false negatives and an accuracy of 94%. It is interesting to see that with the second variant, the FNR can be reduced to zero while loosing only 1.2% accuracy compared to the highest accuracy achieved. This looks different with the first variant, where the cost of minimising the FNR leads to a 31% decrease in accuracy. Deciding upon the optimal configuration for the neural network is less straightforward, as most trainings produced an accuracy of one. A sound choice could be the most conservative configuration (i.e. lowest threshold / highest weight) that still has perfect accuracy on the test data, which would mean either a threshold of 0.05 or a relative weight of 50. This indisputable dominance of neural networks points back to their much greater expressive power due to the non-linearities introduced by the ReLU activation functions, which is not the case in logistic regression. Since the data lives in a 22-dimensional space where each dimension only allows a few separate values, the data is not expected to be linearly separable. Even if it is, the distance between the optimal hyperplane and each data point is too short to allow perfect accuracy on the test set, which is never reached by logistic regression. The neural network we used on the other hand, though technically overparameterised, generalises well nonetheless.

However, there is an obvious caveat with every experiment run on this data set, which as mentioned before is the standard data used by research in this area [4, 5, 6, 7]. No matter the exact ratio of training to test data chosen, a data set size of 8124 means the test data itself will be in the thousands. Consequently, false negative and false positive rates will have a precision of the order of 10^{-4} at most (which is why we chose this precision in our tables). Considering the potential gravity of a misclassification, though, would we be ready to accept a one in a thousand chance that the model might be wrong? This seems highly doubtful. There does not seem to be an obvious remedy to that problem. Data augmentation techniques are not assured to lead to a meaningful increase in predictive power with categorical data of this kind, and the presence of this data set for the past 30 years indicated that there is not interest in producing more data of this type. Consideration should also be made to the applicability of these techniques. While they may be interesting to investigate for their simplicity and higher interpretability which could then be used to inform more complex, image-based neural network models, it is improbable that techniques based on descriptors only will find applications outside of research. Apart from yielding better results, the convenience coming from a smartphone app classifying mushrooms based on a picture taken with said smartphone will never be matched by a program requiring 22 measurements and descriptions.

Conclusion

In this paper, we analysed classification of mushrooms into the categories edible or poisonous based on a series of categorical descriptors. The focus of the paper was the cost on accuracy ensued when false negative rates are minimised. We compared logistic regression with neural networks, using two separated variants of each in order to minimise false negatives: unbiased training with subsequent tuning of classification threshold and training on a weighted loss with increasing the relative weight of the poisonous class. Neural networks showed superior in every aspect of this experiment, with much higher accuracy and a false negative rate of zero for almost all configurations. Logistic regression in the variant with a weighted loss managed to get the false negative rate down to zero while keeping an acceptable accuracy (94%), whereas the decrease of accuracy in order to reduce false negatives in the thresholded version was considerable. Two issues became apparent which pervade all research on this topic using this kind of data. First of all, the size of the data does not allow for an analysis of false negative rates down to the precision that the underlying problem ethically requires – which as its extremes is a matter of life and death. Secondly, classification of mushrooms based on descriptive variables probably has limited applications, if any, as image-based models have the upper hand both in accuracy and convenience. Therefore, further research should focus on the latter, not least because image data sets can much more easily be augmented in meaningful ways (i.e. ways that increases predictive power of models trained on that data) and hence also warrant analysis of metrics at higher precision.

References

[1] William E. Brandenburg and Karlee J. Ward. "Mushroom poisoning epidemiology in the United States". eng. In: *Mycologia* 110.4 (2018), pp. 637–641. ISSN: 1557-2536. DOI: 10.1080/00275514.2018.1479561.

[2] Haijiao Li et al. "Mushroom Poisoning Outbreaks — China, 2019". en. In: *China CDC Weekly* 2.2 (Jan. 2020). Publisher: China CDC Weekly, pp. 19–24. ISSN: 2096-7071. DOI: 10.46234/ccdcw2020.005. URL: https://weekly.chinacdc.cn/en/article/doi/10.46234/ccdcw2020.005 (visited on 12/04/2023).

[3] Roope O. Kaaronen. "Mycological rationality: Heuristics, perception and decision-making in mushroom foraging". en. In: *Judgment and Decision Making* 15.5 (Sept. 2020). Publisher: Cambridge University Press, pp. 630–647. ISSN: 1930-2975. DOI: 10.1017/S1930297500007841. URL: https://www.cambridge.org/core/journals/judgment-and-decision-making/article/mycological-rationality-heuristics-perception-and-decisionmaking-in-mushroom-foraging/E3B5C14DE7FADACB68AFD4C8029A233C (visited on 12/13/2023).

[4] Kemal Tutuncu et al. "Edible and Poisonous Mushrooms Classification by Machine Learning Algorithms". In: *2022 11th Mediterranean Conference on Embedded Computing (MECO)*. ISSN: 2637-9511. June 2022, pp. 1–4. DOI: 10.1109/MECO55406.2022.9797212. URL: https://ieeexplore.ieee.org/abstract/document/9797212?casa_token=8B9XdP-A1dEAAAAA:rfTf5hdr6ZvvaN4lyXlErNjCy-fDGwMpfFB3OOnASC3o_1F8Ixj9RhfIoi0H1JgpZh2PwCYtbPmc (visited on 12/13/2023).

[5] Agung Wibowo et al. "Classification algorithm for edible mushroom identification". In: *2018 International Conference on Information and Communications Technology (ICOIACT)*. Mar. 2018, pp. 250–253. DOI: 10.1109/ICOIACT.2018.8350746. URL: https://ieeexplore.ieee.org/abstract/document/8350746?casa_token=kdfEkr4HQ_kAAAAA:0TRcVnJv4Ztz3ONgY_skrsCT29AmCalgLuA7fkHHwuisRpONLmqMdbWZzijZ4dzePx8AN1zsicsx (visited on 12/14/2023).

[6] Narumol Chumuang et al. "Mushroom Classification by Physical Characteristics by Technique of k-Nearest Neighbor". In: *2020 15th International Joint Symposium on Artificial Intelligence and Natural Language Processing (iSAI-NLP)*. Nov. 2020, pp. 1–6. DOI: 10.1109/iSAI-NLP51646.2020.9376820. URL: https://ieeexplore.ieee.org/abstract/document/9376820 (visited on 12/14/2023).

[7] K. Kousalya et al. "Edible Mushroom Identification Using Machine Learning". In: *2022 International Conference on Computer Communication and Informatics (ICCCI)*. ISSN: 2329-7190. Jan. 2022, pp. 1–7. DOI: 10.1109/ICCCI54379.2022.9741040. URL: https://ieeexplore.ieee.org/abstract/document/9741040?casa_token=ga9pQzLINu4AAAAA:LiJwYdyXtkoS2HaT3Dc9hdDfMVDu6Z61ULAP5YBmv1QdUf73khWAUU4LdmsbVmCKQ1j1glm4Ih5L (visited on 12/19/2023).

[8] Hua Yin, Wenlong Yi, and Dianming Hu. "Computer vision and machine learning applied in the mushroom industry: A critical review". In: *Computers and Electronics in Agriculture* 198 (July 2022), p. 107015. ISSN: 0168-1699. DOI: 10.1016/j.compag.2022.107015. URL: https://www.sciencedirect.com/science/article/pii/S0168169922003325 (visited on 12/13/2023).

[9] Norbert Kiss and László Czùni. "Mushroom Image Classification with CNNs: A Case-Study of Different Learning Strategies". In: *2021 12th International Symposium on Image and Signal Processing and Analysis (ISPA)*. ISSN: 1849-2266. Sept. 2021, pp. 165–170. DOI: 10.1109/ISPA52656.2021.9552053. URL: https://ieeexplore.ieee.org/abstract/document/9552053?casa_token=B0AqHwqt9z4AAAAA:TAX0VwMjbzj4MPm2CsRLmK_7TPYLInY_UaJMlpiW65ZDdhy1hSxPmMyF2OqD82PogxHG1lXOfZLV (visited on 12/14/2023).

[10] Nusrat Zahan et al. "A Deep Learning-Based Approach for Edible, Inedible and Poisonous Mushroom Classification". In: *2021 International Conference on Information and Communication Technology for Sustainable Development (ICICT4SD)*. Feb. 2021, pp. 440–444. DOI: 10.1109/ICICT4SD50815.2021.9396845. URL: https://ieeexplore.ieee.org/abstract/document/9396845?casa_token=LoQ4Yqa4-Y8AAAAA:B-WZzvaLE3-GTVimyYyyhIzvKlqbzBtEzipUEzUGTTuZukUhpBSa8sjY7xtDTrIxR9z-sD4rccsh (visited on 12/14/2023).

Appendix

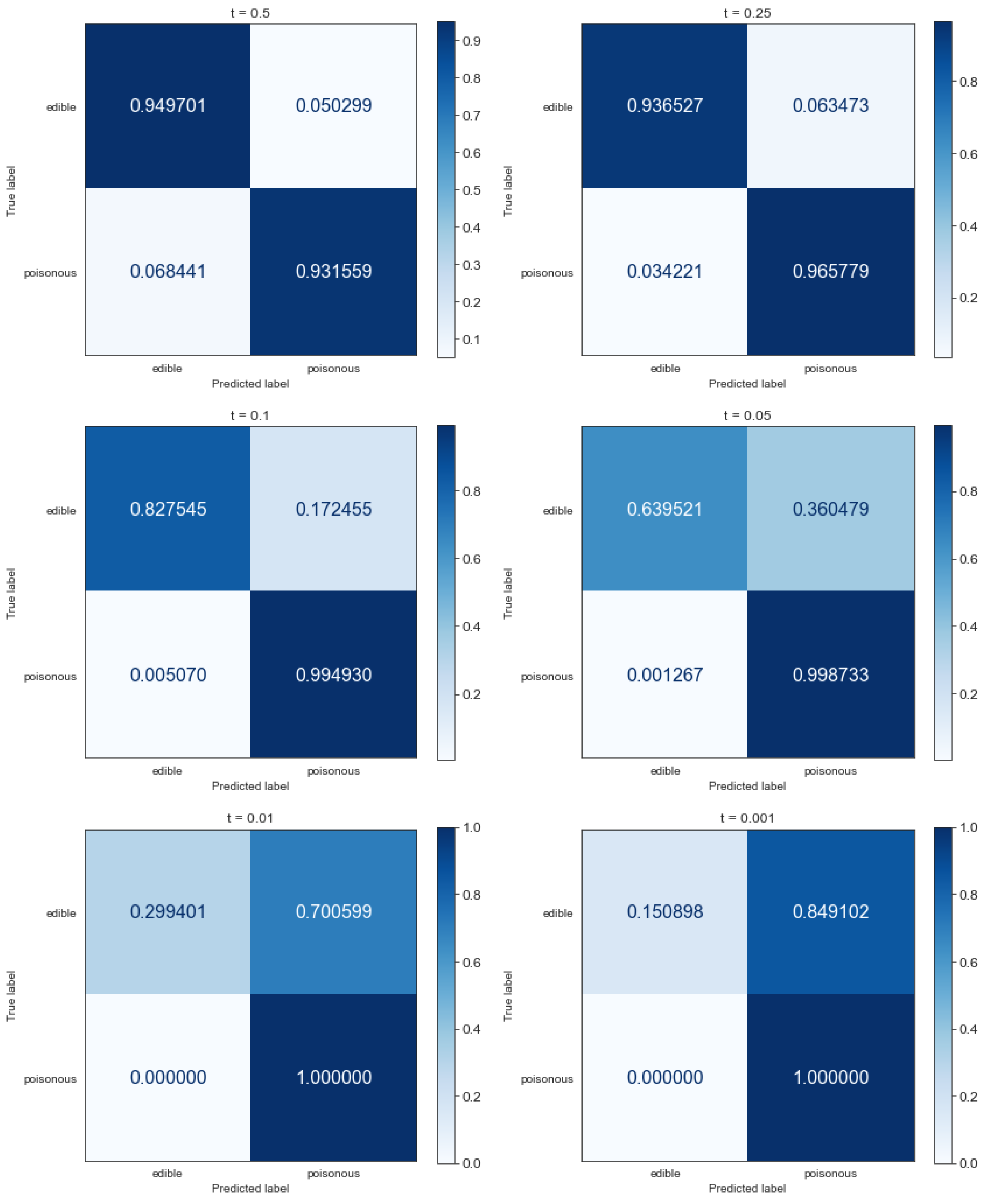

Figure 2: Confusion matrix for different thresholds t: Logistic Regression

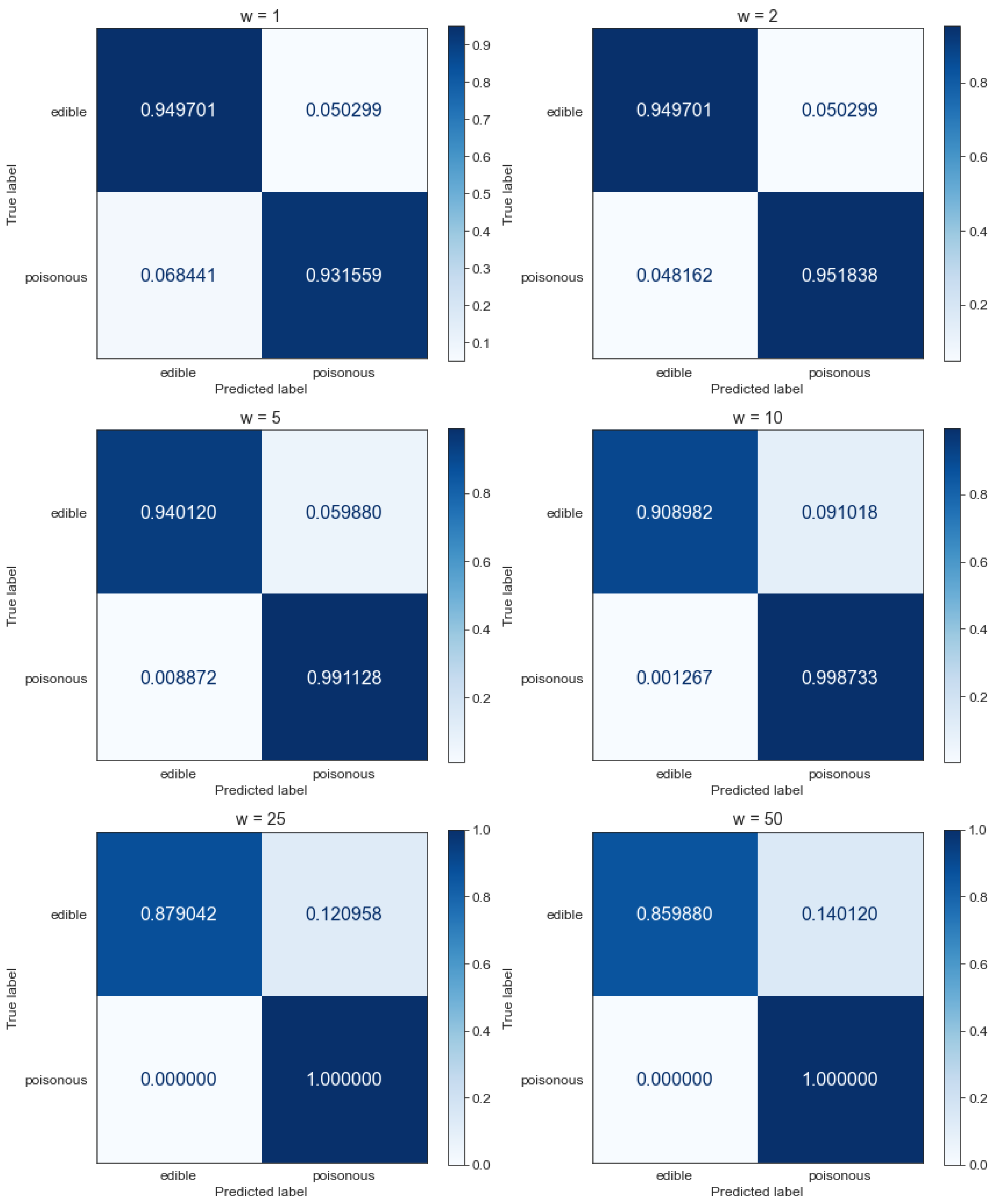

Figure 3: Confusion matrix for different weights w: Logistic Regression

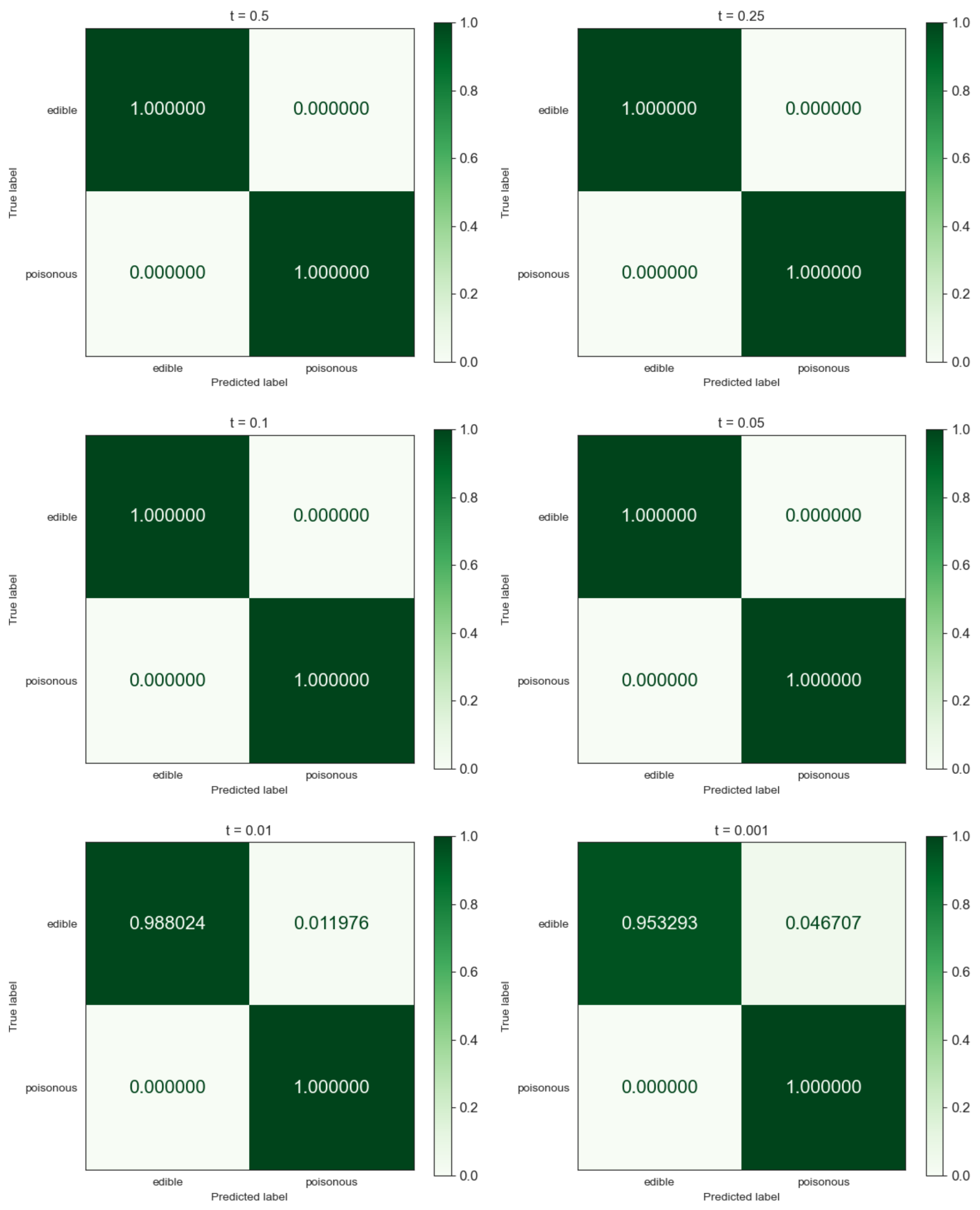

Figure 4: Confusion matrix for different thresholds t: Neural Network

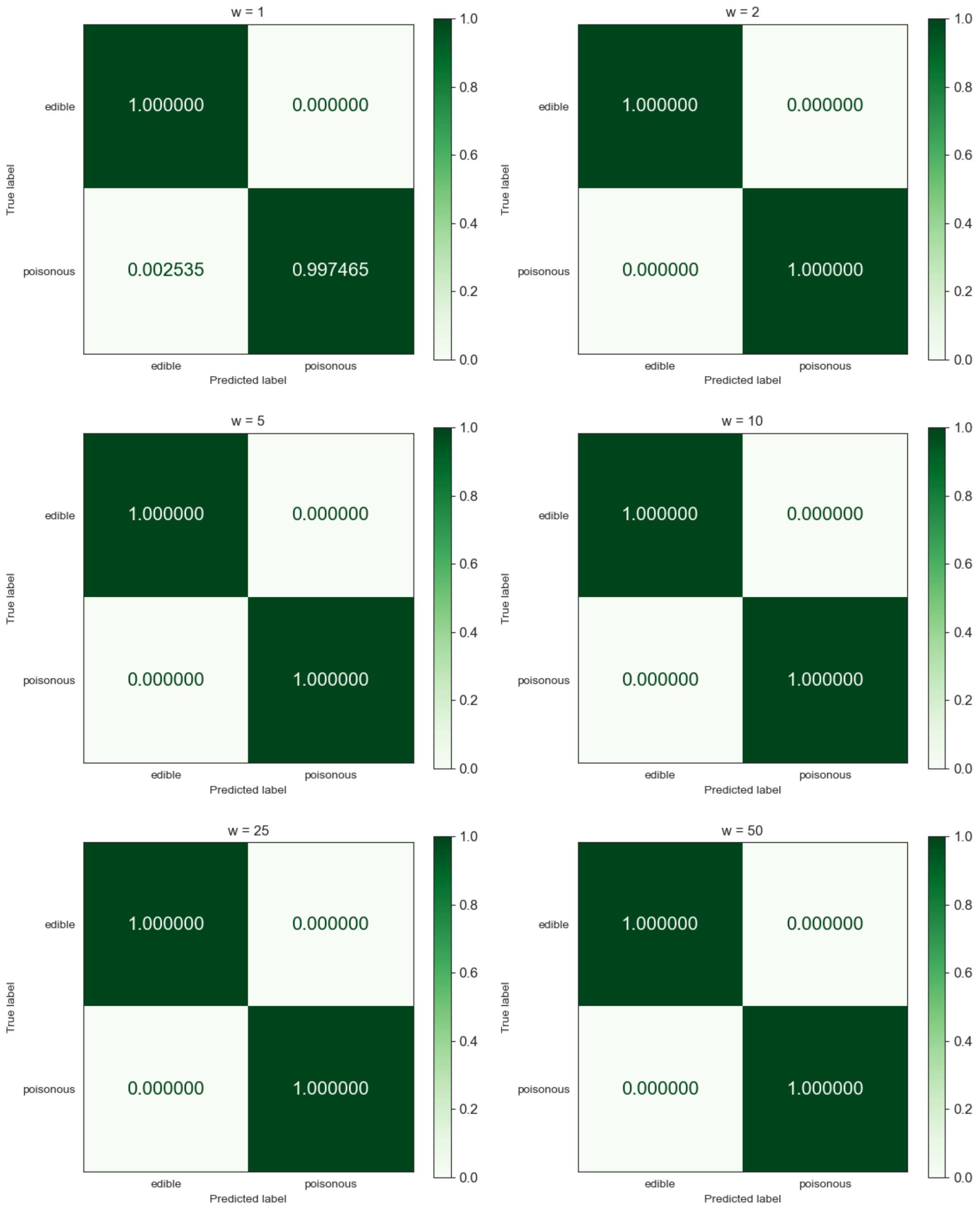

Figure 5: Confusion matrix for different weights w: Neural Network