Comparing Support Vector Machines & Multilayer Perceptrons When Classifying Diseased Pine & Oak Trees

Taranpreet Singh Rai & Gaylord Swaby

Abstract— The performance of SVM and MLP were compared, when correctly classifying the number of diseased instances in the Wilt dataset. Early data analysis showed that there was high correlation among some of the features and so a feature seletion process was initiated where Mean_Green was dropped. Initially, the dataset was highly imbalanced and so SMOTE was applied to alleviate the imbalance of the classes. A grid search was performed on both SVM and MLP individually to find the best set of parameters, where they were then tested on to the unseen test data. SVM performed better than MLP, where MLP had a tendency to select the non-diseased instances more often. Nevertheless, SVM was more computionally expensive with almost double the training time in comparison to that of MLP.

Index Terms—Multilayer perceptron, MLP, support vector machines, SVM, imbalanced dataset, SMOTE, wilt dataset,

1 INTRODUCTION

1.1 Motivation

There are insects that have the potential to cause extensive damage to forested areas in Japan. This is largely owed to the Japanese pine sawyer beetle that is responsible for the Japanese Pine Wilt and the oak platypodid beetle, which consequently causes the Japanese Oak Wilt (Johnson et al., 2013). Both diseases are rapidly spreading, causing devastation in their paths. In the summer months, both beetle types move onto attacking new trees, thus, continuing the cycle (Ohta et al. 2012; Uto et al. 2011). It is essential that the disease is detected fast so that diseased trees are either removed or treated accordingly.

The area under study is located close to Yonezawa City in Yamagata Prefecture, Japan. It is in this location that high-resolution QuickBird images have been generated in order to detect the diseased trees. Consequently, this paper will investigate and compare two classification techniques to detect diseased trees from the provided dataset.

1.2 Dataset

The dataset is the "Wilt" dataset, provided by the UCI repository (Johnson et al., 2013). In August 2012, there were QuickBird images taken from 165 different areas. The dataset is, essentially, built from image segments that are created after segmenting the pan-sharpened image. These segments contain vital information, such as spectral information from the QuickBird multispectral image bands and texture information from the panchromatic image band (Unlersen & Sabanci, 2016). As a result, 6 attributes are created, including one discrete binary class label that states whether or not the instance is diseased. Furthermore, the features are continuous. These features are the GLCM mean texture of pan, the mean of the NIR (Near Infrared) value, the mean of the green value, the mean of the blue value and, lastly, the standard deviation of the pan band. There are 4339 instances, of which only 74 are associated with the "diseased" class, in the training dataset. This is, in fact, problematic due to the high imbalance of the classes. More on this is discussed in Section 2.

Table 1 summarizes each feature of the training dataset. It can be seen that there are some differences between "Diseased" instances and "Normal" instances. This is most notable when analysing the mean values of the Mean_NIR feature, the SD_Pan and Mean_Green. It is clear that diseased instances are different to normal instances, however, there still is some data pre-processing required before moving onto the classification techniques.

		GLCM_PAN	MEAN_GREEN	MEAN_RED	MEAN_NIR	SD_PAN
Normal	COUNT	4265	4265	4265	4265	4265
	MEAN	126.799	234.446	117.278	536.125	25.004
	STD	13.816	61.103	61.183	154.415	11.06193
	MIN	0.000	164.683	59.143	86.500	0.000
	25%	118.525	206.328	91.909	425.000	18.028
	50%	127.438	221.912	101.438	531.240	23.775
	75%	135.066	242.091	116.299	644.889	30.016
	MAX	183.281	955.714	746.333	1005.516	156.508
Diseased	COUNT	74	74	74	74	74
	MEAN	128.688	202.841	118.142	417.688	20.333
	STD	7.685	16.990	19.353	108.040	5.654
	MIN	106.242	164.625	68.000	208.083	6.753
	25%	122.918	191.064	110.240	333.850	16.763
	50%	128.556	202.194	119.072	420.257	20.494
	75%	134.449	213.651	129.875	490.822	23.720
	MAX	147.771	236.487	156.243	608.900	36.442

Table 1: Summary of the Wilt Dataset

1.3 Feature Selection

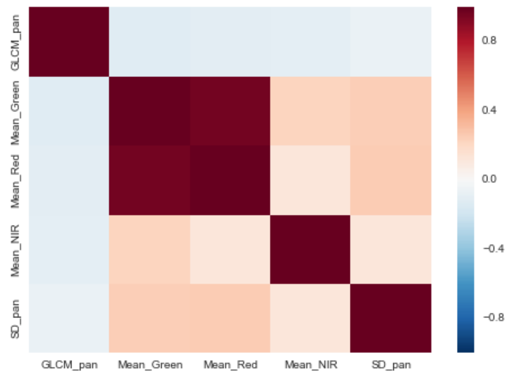

Figure 1: Correlation matrix; every feature & all instances

Figure 1 depicts the correlation amongst the features of the dataset minus the class label. Immediately, it can be seen that there is some minor negative correlation between the GLCM_pan and every other feature. There exists some positive correlation amongst the rest of the features, however, this is most notable between Mean_Green and Mean_Red.

According to Hall (1999), a good subset of features contains features that are highly correlated with the class label, but not correlated with each other. Collinearity is an undesired trait amongst classification tasks as multiple correlated features do not aid in the classification process. A highly correlated feature may not benefit the classifier and may actually hinder performance.

2 Synthetic Minority Oversampling Technique (SMOTE) & Under-sampling

As discussed in Section 1.2, the dataset is highly imbalanced. If this is not addressed, then the classifiers will favor the majority class as it is difficult for the classifier to take into account the subtle differences that may make up a certain class during training. One way of dealing with such data is through applying the Synthetic Minority Oversampling Technique (Chawla et al., 2002). This is where synthetic samples are created to closely resemble the samples that exist for the minority class.

The minority class is the diseased class which accounts for approximately 1.8% of the training set. This is extremely imbalanced and so we need to increase this to a suitable number. The SMOTE algorithm is implemented in Matlab, in which two parameters require changing. Chawla et al. use 5 nearest-neighbors to create their samples initially. We need to choose a suitable k value (for kNN), however, we do not want to use a large k values, as it smooths out important information and may create samples that do not resemble the minority class closely enough. At the same time, we do not want to use extremely small k values, as the new instances will create almost exact replicas of the training instances and, thus, being prone to over-fit. We decided to use 10-NN, as this number is not too large or too small for the given training set. However, given the scope of the paper, we will not experiment with other values.

Additionally, we under-sample the majority class in the training set, following similar methods mentioned in the paper by Chawla et al, 2002. The paper managed to increase accuracy on a number of datasets whilst combining under-sampling techniques with SMOTE. As a result, the under-represented class is raised from 1.8% to approximately 21%. Although still slightly imbalanced, the dataset is nowhere nearly as imbalanced as before and so has a much better chance at being correctly classified.

3 Background on the Two Classification Models

3.1 Support Vector Machines (SVM)

A Support Vector Machine (SVM) classifier is a popular supervised classification technique used in a wide range of domains. In order to classify, SVM tries to find an optimal hyperplane by separating the data points of two classes as much as possible. In our case, it will classify whether an instance belongs to the diseased or normal class. This can be best visualized, in Figure 2, in a two-dimensional space. We want to find the maximum-margin hyperplane, m, which maximizes the distance between the two classes, whilst at the same time correctly classifying all the given data points. If we are given training examples (x^i, y^i), we can define the functional margin of (w, b) with respect to the training example as:

$$\hat{\gamma}^{(i)} = y^{(i)}(w^T x + b)$$

From Figure 2, we want to minimize ||w|| (orthogonal to the hyperplane) whilst maximizing the margin m. However, this can be problematic as we are initially trying to solve a constrained optimization problem. Lagrange duality is used to help this problem and, thus, converting it into a convex optimization problem, where we are able to find the global minimum rather than a local minimum.

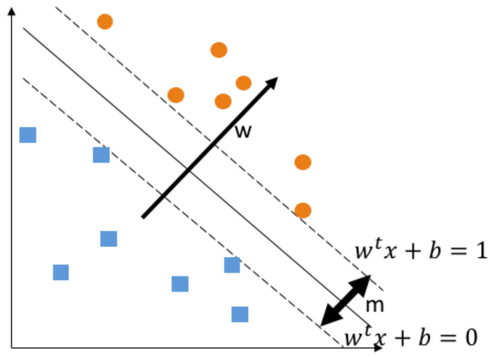

Figure 2: example of SVM in a two-dimensional space, vector "w" is orthogonal to the hyperplane, this also portrays the class separations of 1 & 0 on either side of the hyperplane

At the moment, we have assumed that the data is separable via a straight line, however, this is rarely the case with most real-world datasets. Errors are common where instances can be incorrectly labelled by falling into the wrong side of the separating hyperplane. SVM must be able to handle such cases, and so a *soft margin* is invoked. However, this is a parameter that requires adjustment as we do not want too many misclassifications. By setting such a parameter, we allow a tradeoff between the size of the margin and hyperplane violations (Noble, 2006).

The kernel function creates a new dimension to the dataset. In high dimensions, we cannot draw instances, however, we can project the SVM hyperplane in a lower dimension, thus potentially creating curved like lines when projected.

In order to transform the data points into higher dimensions, a transformation function, usually denoted as ϕ, is utilized so that the data points can be linearly divided in the new projected space. By using ϕ, a Kernel can be defined between two data points x & z:

$$K(x,z) = (\phi(x)^\wedge t)*(\phi(z))$$

In order for ϕ to exist, the Kernel must satisfy the Mercer condition. This leads onto the most useful aspect of SVM; the Kernel trick. The transformation function can be computationally expensive for high dimensional vectors, however, the use of a kernel can improve computational efficiency by replacing all inner products (x,z) from an SVM computation by the kernel K(x,z).

Nevertheless, the kernel function should be appropriately chosen as projecting the data into a very high dimensional space can cause overfitting. This happens because the number of possible solutions increases at higher dimensions, making it difficult to select a solution. The boundaries of the SVM can become very particular to that of the training examples (Noble, 2006).

3.1.1 Advantages of SVM

- There is a high generalization performance without the input of previous domain knowledge. This is because the kernel implicitly contains a non-linear transformation and thus no assumptions about the functional form of the transformation is required (Auria & Moro, 2008).
- Guaranteed optimality as the nature of the optimization problem is convex, the global minimum rather than the local minimum is found when solved.

3.1.2 Disadvantages of SVM

- Computationally expensive in terms of size and speed for both training and testing (Burges, 1998)

3.2 Multilayer Perceptron (MLPs)

The multilayer perceptron is a feed-forward artificial neural network that uses a supervised learning technique called back-propagation to train its network. Each layer in an MLP is built up of one or more artificial neurons in parallel. In fact, the MLP is based on the human brain, where a perceptron tries to mimic neuron activity. There are 3 layers; the input layer where each input is passed through a linear activation function, the hidden layer (which contributes towards making the classifier linearly separable) or layers and the output layer. In its simplest form, the MLP is used to make a binary classification by developing a separating hyperplane (the separating hyperplane is a similar concept to the separating hyperplane in SVM).

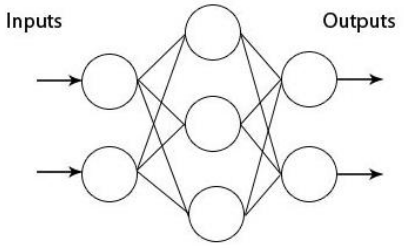

Figure 2: An example of an MLP, the middle layer being hidden

The weight of each connection is, arguably, the most important feature of an MLP as it acts resembling a synapse in a neuron. Therefore, it is reasonable to attempt finding the best weights adapted for each interneuron connection. Back-propagation, applied in training, can be described as the following eight steps in attempting to set the optimal weights (Fahmi et al., 2012).

I. Initiate a pattern to the network.
II. Compare the target output with the network output.
III. Calculate the output error in relation to each neuron within the network.
IV. Compute the correct output value that is supposed to be attained for each neuron.
V. Set the difference between the correct output and network output value.
VI. For each connection, adjust the weight of the lowest local error.
VII. To all the previous neurons, allocate an error signal
VIII. Repeat steps (IV) to (VIII) by applying them to the previous neurons and by using the error signal as the error.

3.2.1 Advantages of MLP

- MLP is an adaptive learner and so the ability to learn by examples makes the neural nets very powerful and flexible, as a result there is no need to create a specific algorithm in order to perform a task.
- "Fault Tolerance"- a neural network continues working even when some of its neurons and interconnections fail, due to its distributed nature (Sethi & Jain, 2014). Additionally, relearning after the damage can be relatively quick.

3.2.2 Disadvantages of MLP

- Unlike SVM, the optimization problem is not convex and thus has the potential to become "stuck" in some local minima.
- The hidden layers of a neural network can be considered a black box, as it can be difficult to interpret the inner workings of the algorithm.

4 TRAINING & TESTING THE MODELS

In this section, we focus on the training of the models and, then, test the trained models on the unseen test set.

4.1 Hypothesis Statement

Although MLPs offer great flexibility, SVM has outperformed MLP in papers by Frias-Martinez et al. (2006) and Fahmi et al. (2012). As a result, we hypothesize that SVM will outperform the Multilayer Perceptron in terms of a lower classification error. However, a disadvantage of SVM is that it has the potential to be computationally expensive. Although MLP can be just as expensive, for this particular domain, we hypothesize that SVM will be more computationally expensive than MLP.

4.2 Choice of Training and Evaluation Methodology

Choosing the right parameters is as important as choosing the classifiers as many tasks and domains are affected differently. As a result, a grid search is performed to find optimal parameters. We want to find optimal parameters in order to avoid over-fitting and to create a better generalization of the models.

4.2.1 Training & Evaluation Methodology for SVM

The dataset is trained using 3-fold cross-validation on numerous parameters via the grid search on SVM. The predictive error is taken as a measurement for each parameter change, where the set of parameters with the lowest predictive error is subsequently chosen as the optimal set of parameters. The optimal parameters are, then, trained once more onto the dataset before being tested onto the unseen test set.

4.2.2 Training & Evaluation Methodology for MLP

Similar to SVM, cross-validation approaches will be performed on MLP. As a measure of performance, the Mean Square Error (MSE) is taken into account for each parameter change, where the parameters with the best performance are, then, subsequently trained onto the training set once more. These are then tested onto the unseen test set.

4.2.3 Direct comparisons between SVM & MLP

To directly compare performance, we will analyze the confusion matrices of both classifiers. Here we can delve in deeper so that we can compare precision and recall measures for both classes.

Lastly, as a measure of computational expense, we will measure the amount of time it takes for each classifier to train on the grid search. The classifier with the lowest overall training time will be considered the least computationally expensive.

4.3 Grid Search: Choice of Parameters

4.3.1 Grid Search parameter changes for SVM

In section 3, we discussed the theory behind SVM. However, the success of SVM is, ultimately, down to optimizing the parameters. As mentioned earlier, most real world datasets are not linearly separable. As SVM creates a convex optimization problem, there is one unique global minima to be found. The kernel trick is employed via SVM, thus we use the Radial Basis Function kernel to help separate the classes. There are two key parameters to change:

Kernel scale, ϕ: We opt for a grid search on several sigma values:

[0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45, 0.5, 0.55, 0.6, 0.65, 0.7, 0.75, 0.8, 0.85, 0.9, 0.95]

Box Constraint, C: The box constraint parameter corresponds to the soft margin that allows some misclassifications to happen. The Lagrange multipliers (formulated in the dual optimization problem) are bounded to the range of [0, C]. Therefore, C positions itself as the box constraint on these Lagrange Multipliers. The chosen values for C are:

[0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45, 0.5, 0.55, 0.6, 0.65, 0.7, 0.75, 0.8, 0.85, 0.9, 0.95]

4.3.2 Grid Search results for SVM

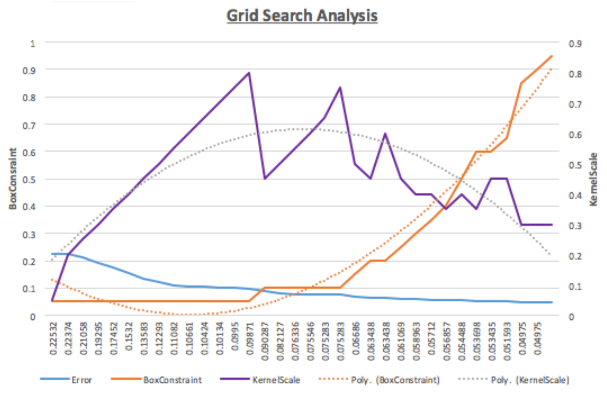

Figure 4: SVM grid search results

From Figure 4, we can see that the grid search displays the relationship between the box constraint and kernel scale. It appears that the box constraint parameter performs better at a higher value, displaying an upward trend. This is not the same for the kernel scale as it displays much more volatile behavior, gradually displaying its optimal value of 0.3. Consequently, we have determined the best values to perform on the unseen test set. The best parameter values on the training set with the error was output to:

***Box Constraint:** 0.95, **Kernel Scale:** 0.3, **CVSV Error:** 0.046328.*

4.3.3 Grid Search parameter changes for MLP

We opt for training the network through gradient descent with momentum and an adaptive learning rate for back-propagation. As a result, there are a number of parameters we can experiment through the grid search, however, we focus on four.

Epochs: The maximum number of epochs to train. One epoch equates to one forward pass and one backward pass of all the training examples. We trial the following values of epochs:

[50, 100, 150, 200, 250, 300]

Number of Hidden Layers: The hidden layers' act as feature extractors, where as the learning progresses, they discover salient features of the problem space (Simon, 2009). We opt for experimenting with the following values for hidden layers:

[5, 10, 15, 20, 25 30]

Learning Rate: This parameter is vital for MLP as it controls the size of bias and the weight changes in the learning of the training algorithm. The chosen values are:

[0.1, 0.3, 0.5, 0.7, 0.9]

Momentum: As suggested by the disadvantages, unlike SVM, MLP can get stuck in a local minimum. The momentum parameter can help avoid such situations by

[0.1, 0.3, 0.5, 0.7, 0.9]

4.3.4 Grid Search parameter results for MLP

At the end of the grid search, we are presented with the following parameter values and results to test onto the unseen test set:

Hidden Layer: *20,* ***Epochs****: 300,* ***Learning Rate****: 0.1,* ***Momentum****: 0.5,* ***Performance:*** *0.14004*

We can see from Figure 5 that, during training, MLP allows an overall accuracy of 77.4% on its best parameters. A low learning rate suggests that there are only a few local minima. This seems promising, however. A further look into the confusion matrices, shows us that MLP has a bias towards class 0. As this is the best set of parameters for the current MLP model, we shall continue to use these parameter values on the unseen test set. We further amend our hypothesis to suggest that a similar occurrence will transpire to the unseen test data.

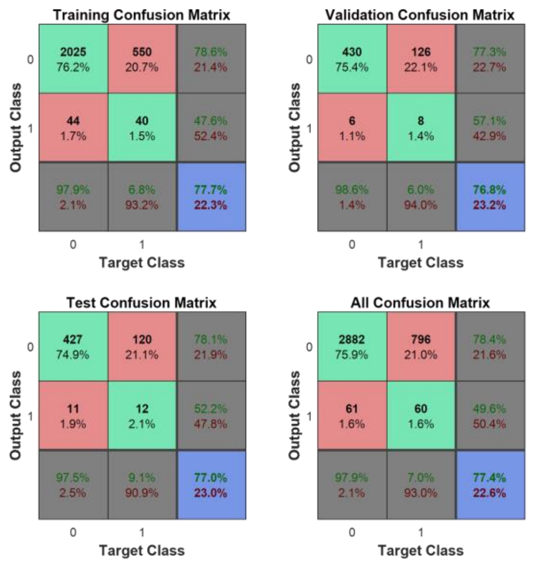

Figure 5: MLP Confusion matrices from training set data

5 Analysis and Critical Evaluation of Results

5.1 Test Set Results

SVM predictive error: 0.063175. The error displayed is marginally higher than that of the training error (0.046328).

MLP performance: 0.23752. This is a much higher error in comparison to the training performance (0.14004), suggesting that the model did not generalize well overall.

5.2 Confusion matrices: SVM versus MLP

For SVM, class 0 was output correctly 58% of the time, whereas, class 1 was output correctly 19% of the time, equating to an overall correct classification performance of 77%. Additionally, for MLP, class 0 was correctly output 59% of the time and class 1 was output correctly 1.6% of the time, equating to an overall correct classification performance of 60.6% of the time.

We can state that SVM has outperformed MLP as hypothesized earlier. A closer inspection of the results also suggest that SVM advocates less bias in these results. A simple look into the MLP confusion matrix supports this claim, where we can clearly see class 0 (non-diseased instances) being selected considerably more than class 1. We were also right to assume that MLP would select this class more often as shown in the training (Figure 5). However, for the test set, this phenomenon occurred at a grander level where class 0 is chosen incorrectly 35.8% of the time. This also suggests that MLP does not generalize well with the current model.

From Table 2, we can see the highest precision (the amount of times a class was correctly predicted) was for class 0 for MLP. Moreover, the lowest was also for MLP for class 1. Precision for SVM was reasonably accurate with both classes displaying high values, suggesting a better performance from SVM. When analyzing the recall measures, it appears that SVM, class 0 (non diseased) had the highest value whereas class 1 (diseased) for MLP had the lowest value. It is worth noting that the class with the worst performance was indeed class 1 (diseased). The slight imbalance of the classes may contribute towards this performance.

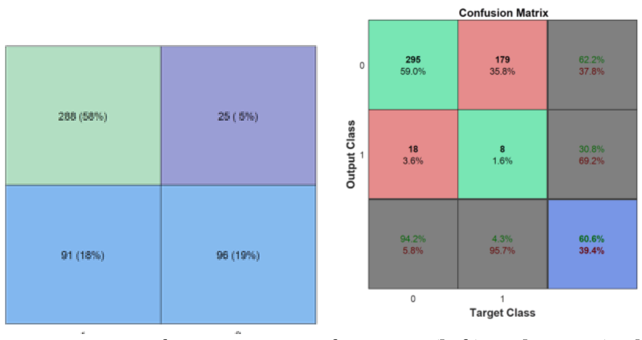

Figure 6: Confusion matrices for SVM (left) and MLP (right)

CLASSIFIER:	SVM		MLP	
	0	1	0	1
Precision	0.75989	0.79339	0.94249	0.04278
Recall	0.92013	0.51337	0.62236	0.30769

Table 2: precision and recall measures for both classes

Upon observation it would seem that MLPs results do not resonate with other papers who applied the network to the Wilt datasets. For example, Unlersen & Sabanci (2016) found much better success with an overall classification success rate of 86.6%. The number of neurons in the hidden layer for the optimal results were 10 (in comparison to ours of 20) and the MSE (square of RMSE) was lower at 0.10240 (in comparison to ours of 0.23752). Unfortunately, the paper does not portray any confusion matrices for investigation. Despite the lower MSE values, we suspect that the results are actually far worse than those being demonstrated as there is no mention of dealing with the high imbalance that is apparent in the dataset. Not dealing with this imbalance can mask, in actual fact, poor results.

We initially believed that MLPs poor generalization is due to the synthetically produced data from SMOTE. However, SVM performs significantly better and, so, the poor performance of MLP may be due to the choices of parameters.

5.3 Computational expense of both models

CLASSIFIER:	SVM	MLP
Grid Search Training time	1233.6875	641.4063
Optimal Parameters Training time	0.46875	0.71875

Table 3: Time of training on both models

Overall, SVM was the most computationally expensive in terms of running time, with almost double the time performing the grid search than MLP. However, it appears that during the training of the optimal parameters, it was slightly quicker than MLP (although a negligible difference). Overall, including the time that it takes for the grid search and training with optimal parameters, we were correct to hypothesize that SVM would be more computationally expensive than MLP.

6 CONCLUSION

6.1 Lessons Learnt & Future Work

SVM outperforms MLP with back-propagation, as initially predicted. Furthermore, SVM is much more computationally expensive in training; this is more visible when analyzing the time that it takes to complete the grid search.

The optimization of such models is considerably important. However, the quality of training data is just as important. We initially applied SMOTE to the training dataset to overcome a highly imbalanced class issue. Just as we have done with a grid search on parameter values for both models, it would be wise to overcome the imbalance issue with a similar approach. One could test several under-sampling ratios, as well as test several number of neighbors within the SMOTE algorithm. Additionally, ADYSYN could be approached as an alternative when dealing with class imbalance (He et al., 2006). The importance of creating synthetic samples that would resemble real world values is highly important and crucial for the success of these models. Furthermore, dropping "Mean_Green" may have had an adverse effect on the classifier. It may be worth investigating the effect of highly correlated features on a neural network and understand whether an increase in dimensionality will aid in better performance for classification. Conversely, we could apply dimension reductions techniques such as PCA and reduce computational expense by using the first two principal components as feature inputs.

SVM had two key parameters that could be tuned which allowed for its better success during this specific task. However, with the current method employed, MLP had several parameters that were overlooked. For example, when measuring the performance of the network, we opted to use MSE, where there are further parameters that could have been applied to avoid over-fitting. Take "regularization" as a parameter. This relates to the proportion of the performance attributed to the weight/bias values. The default value in our case was 0, however, if we applied a larger value, this would apply a penalty on the network for large weights, thus, making it more likely that the network function will avoid over-fitting.

6.2 Implications for the domain

It appears that SVM is the better model to perform on this particular domain. However, the accuracy is still not as high as needed for the given problem. The utmost accuracy is required as ideally only diseased trees should be targeted for treatment or removal. The ethics of removing trees by accident based on our current estimates would be too great a problem to deal with. In fact, high accuracy that equates to 99% or more would make SVM a reasonable model to apply. Nevertheless, MLP still has potential with further investigation (several parameter changes) due to its flexibility in adaptive learning.

REFERENCES

[1] Auria, L. and Moro, R.A., 2008. Support vector machines (SVM) as a technique for solvency analysis.

[2] Burges, C.J., 1998. A tutorial on support vector machines for pattern recognition. *Data mining and knowledge discovery*, *2*(2), pp.121-167.

[3] Chawla, N.V., Bowyer, K.W., Hall, L.O. and Kegelmeyer, W.P., 2002. SMOTE: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, *16*, pp.321-357.

[4] Fahmi, B.R., Kaouther, N. and Trabelsi, A., 2012. Support vector machines versus multi-layer perceptrons for reducing false alarms in intensive care units. *International Journal of Computer Applications*, *49*(11).

[5] Frias-Martinez, E., Sanchez, A. and Velez, J., 2006. Support vector machines versus multi-layer perceptrons for efficient off-line signature recognition. *Engineering Applications of Artificial Intelligence*, *19*(6), pp.693-704.

[6] He, H., Bai, Y., Garcia, E.A. and Li, S., 2008, June. ADASYN: Adaptive synthetic sampling approach for imbalanced learning. In *Neural Networks, 2008. IJCNN 2008.(IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on* (pp. 1322-1328). IEEE.

[7] Johnson, B.A., Tateishi, R. and Hoan, N.T., 2013. A hybrid pansharpening approach and multiscale object-based image analysis for mapping diseased pine and oak trees. *International journal of remote sensing*, *34*(20), pp.6969-6982.

[8] Ohta, K., K. Hoshizaki, K. Nakamura, A. Nagaki, Y. Ozawa, A. Nikkeshi, A. Makita, K. Kobayashi, and O. Nakakita. 2012. "Seasonal Variations in the Incidence of Pine Wilt and Infestation by Its Vector, Monochamus Alternatus, Near the Northern Limit of the Disease in Japan." *Journal of Forest Research* 17: 360–368.

[9] Noble, W.S., 2006. What is a support vector machine?. *Nature biotechnology*, *24*(12), pp.1565-1567.

[10] Sethi, I.K. and Jain, A.K. eds., 2014. *Artificial neural networks and statistical pattern recognition: old and new connections* (Vol. 11). Elsevier.

[11] Simon, H., 2009. Neural networks and learning machines. *Upper Saddle River: Pearson Education*, *3*.

[12] Uto, K., T. Massaki, Y. Kosugi, G. Saito, and T. Ogata. 2011. "Band Selection for Japanese Oak Wilt Extraction in Autumnal Tints of Forest Based on NWI." 3rd Workshop on Hyperspectral Image and Signal Processing: Evoloution in Remote Sensing (WHISPERS), 1–4, Lisbon, June 6–9.

[13] Unlersen, M.F. and Sabanci, K., 2016. The Classification of Diseased Trees by Using kNN and MLP Classification Models According to the Satellite Imagery. *International Journal of Intelligent Systems and Applications in Engineering*, *4*(2), pp.25-28.