

Towards an Automatic Diagnosis of Peripheral and Central Palsy Using Machine Learning on Facial Features

C.V. Vletter* H.L. Burger* H. Alers† N. Sourlos‡ Z. Al-Ars§

Abstract

Central palsy is a form of facial paralysis that requires urgent medical attention and has to be differentiated from other, similar conditions such as peripheral palsy. To aid in fast and accurate diagnosis of this condition, we propose a machine learning approach to automatically classify peripheral and central facial palsy. The Palda dataset is used, which contains 103 peripheral palsy images, 40 central palsy, and 60 healthy people. Experiments are run on five machine learning algorithms. The best performing algorithms were found to be the SVM (total accuracy of 85.1%) and the Gaussian naive Bayes (80.7%). The lowest false negative rate on central palsy was achieved by the naive Bayes approach (80% compared to 70%). This condition could prove to be the most severe, and thus its sensitivity is another good way to compare algorithms. By extrapolation, a dataset size of 334 total pictures is estimated to achieve a central palsy sensitivity of 95%. All code used for these machine learning experiments is freely available online at <https://github.com/cvvletter/palsy>.

1 Introduction

Peripheral palsy and central palsy are two forms of facial paralysis. Both conditions have similar characteristics, but they can have very different acuteness. Bell's Palsy is one form of peripheral nerve disease, from which patients often heal and which is not likely to be fatal [1]. A central facial palsy, however, can be indicative of a stroke. In this case, it is vital to survival that the diagnosis is made early and that prompt treatment is provided to those patients [2].

In earlier research [3], a system for an automatic palsy classification based on manually defined distances between points of the face was introduced. This system works by executing the following steps in order:

1. A picture of an individual is used as input to

the system, the location of the face is then found by a facial detection algorithm.

2. A facial feature extractor finds 68 predetermined landmarks (shown in Figure 1) on the face and records the location (x- and y-coordinates) of each one of these.
3. A deterministic algorithm makes the diagnosis ('central palsy', 'peripheral palsy', or 'healthy'), based on predetermined distances between the landmarks and predetermined thresholds for these distances.

The dataset used in this work, contains a total of 203 pictures found online, which were classified by an experienced neurologist and a neurologist in-training as having peripheral or central palsy, or being healthy. Due to privacy reasons, no new pictures taken in a clinical setting could be added to the dataset. Out of the 203 images, 103 pictures are of patients with peripheral palsy, 40 of patients with central palsy, and 60 of healthy people. For each picture, the coordinates of the 68 landmarks on the face, the output of step 2 of the system, are available. For patients, these are manually annotated, while for healthy people, these were placed automatically. The coordinates of the top-left and bottom-right point of a bounding box around the face region ("face box"), the output of step 1 of the system, are also available. The dataset as described above is called the "palsy dataset". [3]

The algorithm used in step 3 of the described system currently works by defining 49 quantitative measurements on the extracted landmarks, and choosing threshold values to distinguish between healthy people and patients, and then between a central or a peripheral facial palsy. Since all images were taken into account to define these thresholds, it was not assessed whether the algorithm works on an alternative image dataset. Therefore, it would be better to replace the current classification algorithm with a more robust one that is not based on manually defined thresholds.

The goal of this algorithm should be to classify any individual correctly. Given the data (the positions of all landmarks on a person's face), a diagnosis must be made (healthy, central, or peripheral palsy). For these kinds of problems, a machine learning algorithm (MLA) can be considered [4]. Since the

*{c.v.vletter, h.l.burger}@student.hhs.nl

† h.alers@hhs.nl

‡ n.sourlos@umcg.nl

§ z.al-ars@tudelft.nl



Figure 1: The location on a face of the 68 numbered facial landmarks [9]

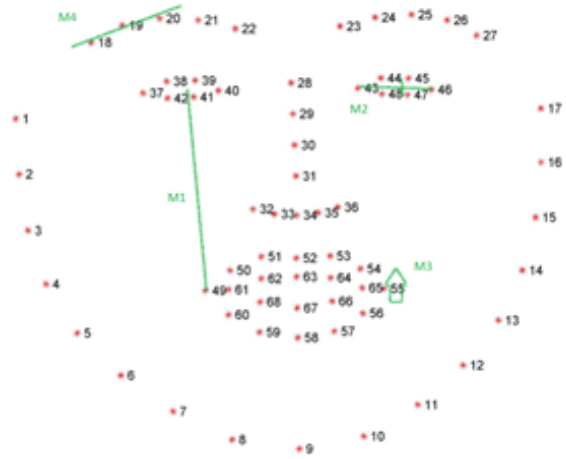


Figure 2: Example of four extracted metrics, adapted from [9]

diagnoses for all individuals in the palsy dataset are known, a supervised learning approach is preferred [5]. This means the training algorithm will use both the features (data based on the landmarks) and the corresponding label (diagnosis) of any data point (image), to train the MLA.

In a similar study, researchers used pictures taken with a smartphone to detect facial palsy. For classification, the researchers used Linear Discriminant Analysis and a Support Vector Machine [6]. This binary classification system, however, does not distinguish between the two possible types of palsy a patient has.

SVMs are a type of MLA which find clusters in high-dimensional data. The data in our datasets consists of up to 136 features per data point. A total of 203 data points are available. This ratio of high complexity to small size of the palsy dataset could mean that SVMs are a good candidate for this problem. [7]

More basic algorithm types of algorithms are also good to consider for this classification problem. An example could be a trained decision tree [7]. Using an algorithm like ID3 to train this tree, in which metrics are used step-by-step to get closer to the final answer, could lead to an algorithm resembling the one proposed in [3].

An expansion to the decision tree is the random forest. This approach uses many trees in parallel to increase higher classification certainty. [8]

2 Materials & methods

This article proposes an algorithm to replace the third step of the system [3] described in Section 1. To achieve this, different types of MLAs will be tested on features extracted from the palsy dataset. The accuracy of each will be recorded and later used

to choose the algorithm that's best suited for this problem.

It should be noted that the algorithm that was previously used in the system used hand-tweaked thresholds to achieve the highest possible accuracy on the known dataset [3]. In this article, we propose an algorithm which is tested with data not used for training. This allows for better assessment of the generalisation performance of the algorithm. Moreover, as more data become available, the algorithm can also very easily be retrained to further increase its performance.

Since the dataset is small, Leave-One-Out Cross-Validation (LOOCV) will be used to run the final algorithm accuracy tests. For a dataset of size n , this means that $n - 1$ of the data points are used for training, and the remaining 1 data point is used for testing. This process is repeated n times, with each data point being used for testing once. The total number of correct classifications is denoted c . To compute the final classification accuracy a , the ratio of correct predictions to total predictions is used:

$$a = \frac{c}{n} \cdot 100\% . \quad (1)$$

LOOCV maximises the ratio of training data to testing data, while preserving an accurate accuracy metric [6]. Even though using LOOCV, it is computationally expensive to train a model, the inference time is low which makes it a great candidate for medical applications.

Another metric of interest is the classification accuracy for patients with a central palsy, since this is the most sever condition. This is called the sensitivity for this condition. For a certain condition, the sensitivity s is calculated by using

$$s = \frac{c_1}{n_1} \cdot 100\% . \quad (2)$$

Here, c_1 is the number of correct predictions for all n_1 number of people with a certain condition.

2.1 Description of the datasets

In this study, three different adaptations of the palsy dataset are used in experiments. These are described below.

2.1.1 Landmarks dataset

For supervised machine learning algorithms, it is important to have a properly annotated and labelled dataset and to perform the right pre-processing before feeding the data to those algorithms. These are the pre-processing steps performed on the palsy dataset:

1. The faces are cropped from the image using the face box data, resulting from step 1 of the system described in Section 1.
2. The cropped images are resized to 900x900 pixels and the landmarks are resized accordingly.
3. The resized images are rotated so that the mean of all points from both eyes are in the same horizontal line. The landmarks are rotated accordingly using the resulting translation matrix.
4. Landmarks outside the face box region are shifted to the nearest point inside the face box.
5. The resulting landmark data is then normalised, meaning all landmarks are resized so that the coordinates end up between 0 and 1.

For all patient pictures, the manually annotated landmarks available in the palsy dataset are used. One data point was left out because many landmarks laid outside the face box, resulting in too many incorrect landmarks after step four of pre-processing. The dataset that results when applying these five steps to the palsy dataset is called the “landmarks dataset”. It contains 202 data points, and has 136 features: two for each landmark.

2.1.2 No-chin dataset

A second dataset that is experimented with, is created from the landmarks dataset described in Section 2.1.1. The chin points are the landmarks most often shifted by step 4 of the pre-processing steps described in Section 2.1.1, and they are unlikely to differ significantly between healthy people and patients. Leaving out the chin points will also decrease the processing time. Therefore, they were left out for this second dataset. This new dataset will be called the “no-chin dataset”. It still contains

202 data points, but the complexity is reduced to a feature size of 102: the x- and y-coordinates of the 17 chin landmarks are discarded.

2.1.3 Metrics dataset

In previous research, metrics were defined based on extracted facial features which were used to give each patient a palsy score [10]. Two of these scores are the House-Brackmann and Sunnybrook scale. What these scales generally have in common is that they look at different parts of the face, compare them to the other side and score a patient based on the difference between the left and right size of one’s face. To make use of this, 52 metrics are defined for each of the data points, based on the landmarks dataset. The metrics are based on the House-Brackmann scale, Sunnybrook scale, and previous work [3].

Some examples of facial metrics are listed here. Ratio of distances from the centre of each eye to the edge of the corresponding side of the mouth (M1). Ratio of the width of each eye divided by its eyelid opening (M2). Ratio of the difference of the highest left point of the mouth and the highest right point of the mouth (M3). Slope of the best line fitted on the 3-left most points of the left brow (M4). These four examples are illustrated in Figure 2, with the labels as specified.

When the 52 metrics are calculated from the landmarks dataset, a new dataset is formed. This set will be called the “metrics dataset”. It still contains 202 data points, but each one has 52 features.

The metrics dataset thus uses data transformation when compared to the landmarks dataset, possibly revealing distinctions between classes otherwise not easily found by a MLA.

2.2 Description of the algorithms

In this work, five algorithms are tested for finding a diagnosis algorithm. These are presented below.

2.2.1 Gaussian naive Bayes

The Gaussian naive Bayes algorithm is based on Bayes’ theorem. Naive Bayes assumes independent, equally important features and calculates a probability that a feature belongs to each class (healthy, central palsy, or peripheral palsy). It then uses the probabilities for all features to predict an outcome class. In real world situations, features often are not actually independent, yet this algorithm seems to work in practice [11].

2.2.2 Simple decision tree

A simple decision tree is another supervised MLA. The decision tree decides based on multiple different

features, each layer deep selecting the best feature to split the test data. Then walking the test data down a path through multiple nodes. Each of these nodes has two different edges, leading to more nodes and ending with a leaf: the outcome classification of the decision tree.

This algorithm follows a flow chart type structure that resembles human-like decision making. Because of its ability to handle both numerical and categorical features and to find all the possible outcomes [12], the decision tree is a widely used MLA.

2.2.3 K-nearest neighbour (KNN)

The premises behind the KNN algorithm is rather simple. The training data points are placed in an f -dimensional space, where f is the number of features per data point. After this, the data point to predict is placed upon the plane [13]. Once the test point is placed, an f -dimensional sphere is drawn so that k number of other data points are within the circle. Only these k values will then be taken into consideration, and the test point will be assigned a label based on the majority vote amongst the k points, weighed by distance.

2.2.4 Random forest

A random forest is an extension of a simple decision tree that uses multiple uncorrelated trees, called an ensemble. The combination of these will outperform any of the individual models [14]. The idea behind the random forest is that it takes the most predicted answer and thus guarantees a higher predictability chance than a single decision tree. This is because of the effect the individual trees have on each other. While in a single decision tree a wrong answer is fatal, a random forest combats this by taking the majority vote amongst all decision tree predictions.

2.2.5 Support vector machine

The idea of a support vector machine is that a hyperplane which separates the data into classes is created [15]. The training algorithm looks for the margin between both classes and places a hyperplane so that the margin is maximised

When a perfectly separating hyperplane cannot be created, the data can be transformed into a higher dimension using a kernel function. Possibly, a hyperplane can now be fitted into the plot. There are several different kernel functions.

3 Experimental results

To test which algorithm performs best in the task of diagnosing any person as being healthy, as having a peripheral or as having a central palsy, experiments

are run with the five algorithms discussed in Section 2.2. From the Scikit-learn machine learning library [16], the algorithms SVM, random forest, Gaussian naive Bayes, KNN, and decision tree are used. The dataset, the Python files to create the three and Jupyter notebooks used to run all experiments are available on Github¹. The experiments are run on the three datasets: the landmarks dataset, the no-chin dataset, and the metrics dataset (explained in Section 2.1).

For each algorithm, excluding the neural network, the LOOCV accuracies a from (1) are given for all three datasets. Some of the experiment runs are accompanied by a confusion matrix. These matrices (which all look like Table 1) show the type of error the algorithm makes most. For each actual diagnosis, the percentages of predicted diagnoses are stated. In these tables, abbreviations are used for the three classes. P means peripheral palsy, C means central palsy, and H means healthy. If an algorithm performs very good, the top-left to bottom-right diagonal will contain only values close to or at 100%, and all other cells will contain values close to or at 0%.

3.1 Gaussian naive Bayes performance

For the Gaussian naive Bayes algorithm, the results are as follows: When using the landmarks dataset in training the model, an accuracy of $a = 63.9\%$ is achieved. The accuracy improves to $a = 64.4\%$ when the no-chin dataset is used. For the no-chin dataset, a confusion matrix is shown in Table 1.

The Gaussian naive Bayes ML algorithm was then fitted on the metrics dataset. A testing accuracy of $a = 80.7\%$ is now reached. The resulting confusion matrix is given in Table 2. The Gaussian naive Bayes model has no parameters that can be tuned.

3.2 Decision tree performance

A decision tree model is also built. Using a max depth of 10 for the tree, accuracies of $a = 67.8\%$ and $a = 69.8\%$ are achieved, if using the landmarks and no-chin datasets, respectively. For the no-chin

¹<https://github.com/cvvletter/palsy>

Table 1: Confusion matrix for the Gaussian naive Bayes algorithm on the no-chin dataset

Diagnosis		Actual		
		P	C	H
Predicted	P	69.6 %	40.0 %	18.3 %
	C	28.4 %	40.0 %	10.0 %
	H	2.0 %	20.0 %	71.7 %

dataset, the confusion matrix given in Table 3 results.

The simple decision tree reached an accuracy of $a = 73.3\%$ on the metrics dataset when a maximum tree depth of 20 is used. The resulting confusion matrix is given in Table 4. Besides the tree-depth, no further parameter tuning was performed.

3.3 K-nearest neighbour performance

For the K-nearest neighbour approach, the model was specified to use the nearest 5 neighbours to the data point to predict the output class. These neighbours were weighed by distance, meaning closer neighbours count more in the final prediction. When using this approach, the model achieves an accuracy of $a = 69.8\%$ on the landmark’s dataset. When the no-chin dataset is used, the accuracy is improved to $a = 75.2\%$. The confusion matrix for the no-chin dataset is shown in Table 5.

When the nearest 7 neighbours are considered and again weighed by distance, a classification accuracy of $a = 59.4\%$ is achieved on the metrics dataset. The confusion matrix that results is shown in Table 6.

3.4 Random forest performance

The random forest model has a parameter that defines the number of parallel trees. This parameter was tuned before testing the accuracy of the model.

3.4.1 Random forest tuning

A random forest algorithm was also trained. The number of estimators was initially set to 136 random estimators, and used compare the landmarks and no-chin datasets. Using the no-chin dataset led to the same accuracy as when the landmarks set was used.

An experiment is then performed to find the optimal number of random estimators, using the landmarks dataset. A graph showing the LOOCV accuracy a and number of random estimators is given in Figure 3. From around 100 random estimators, there seems to be no more improvement in accuracy. The ‘noise’ – due to the randomness in the algorithm – seems to stabilise from around 150 random

Table 2: Confusion matrix for the Gaussian naive Bayes algorithm on the metrics dataset

Diagnosis		Actual		
		P	C	H
Predicted	P	73.5 %	20.0 %	1.7 %
	C	26.5 %	80.0 %	5.0 %
	H	0 %	0 %	93.3 %

Table 3: Confusion matrix for the decision tree algorithm on the no-chin dataset

Diagnosis		Actual		
		P	C	H
Predicted	P	81.4 %	35.0 %	13.3 %
	C	11.8 %	47.5 %	21.7 %
	H	6.7 %	17.5 %	65.0 %

Table 4: Confusion matrix for the decision tree algorithm on the metrics dataset

Diagnosis		Actual		
		P	C	H
Predicted	P	75.5 %	45.0 %	5.0 %
	C	19.6 %	42.5 %	5.0 %
	H	6.7 %	12.5 %	90.0 %

Table 5: Confusion matrix for the KNN algorithm on the no-chin dataset

Diagnosis		Actual		
		P	C	H
Predicted	P	83.3 %	42.5 %	3.3 %
	C	8.8 %	27.5 %	3.3 %
	H	7.9 %	30.0 %	93.4 %

Table 6: Confusion matrix for the KNN algorithm on the metrics dataset

Diagnosis		Actual		
		P	C	H
Predicted	P	68.6 %	57.5 %	18.3 %
	C	13.7 %	20.0 %	11.7 %
	H	17.6 %	22.5 %	70.0 %

estimators. The training time also increases rapidly when introducing more random estimators.

On the metrics dataset, the random forest algorithm was also tested with 1 until 200 random estimators. As seen in Figure 4, no meaningful improvement in accuracy is achieved when more than around 50 random estimators are used to classify the input.

3.4.2 Random forest testing

A final accuracy of $a = 77.2\%$ is obtained on the landmarks dataset when 200 random estimators are used. Table 7 shows the resulting confusion matrix for this experiment.

At last, an experiment was performed on the metrics dataset using 100 random estimators. This resulted in an accuracy of $a = 85.1\%$, with the confusion matrix given in Table 8.

3.5 Support vector machine performance

The support vector machine was also first tuned, before testing the final accuracy.

3.5.1 Support vector machine tuning

For training the SVM, the input classes are balanced, meaning the three classes are given a weight inversely proportional to the number of entries in that class to combat underrepresentation. The standard available kernels are used in the experiments. Both the radial basis function kernel and the linear kernel result in lower accuracy for the model than when a polynomial kernel of degree 5 is used. The polynomial kernel function is thus further examined.

To test the optimal degree of kernel polynomial function, values of 1 until 40 are tested using both the landmarks (Figure 5) and the no-chin (Figure 6) datasets.

3.5.2 Support vector machine testing

Figures 5 and 6 show that the highest accuracy is achieved using the no-chin dataset and a polynomial kernel function of degree 4. The final accuracy for this SVM is $a = 85.1\%$. The confusion matrix for this dataset is given in Table 9.

Table 7: Confusion matrix for the random forest algorithm on the landmarks dataset

Diagnosis		Actual		
		P	C	H
Predicted	P	91.2 %	42.5 %	10.0 %
	C	5.9 %	22.5 %	0 %
	H	2.9 %	7.5 %	90.0 %

Table 8: Confusion matrix for the random forest algorithm on the metrics dataset

Diagnosis		Actual		
		P	C	H
Predicted	P	95.1 %	55.0 %	0 %
	C	4.9 %	37.5 %	0 %
	H	0 %	7.5 %	100 %

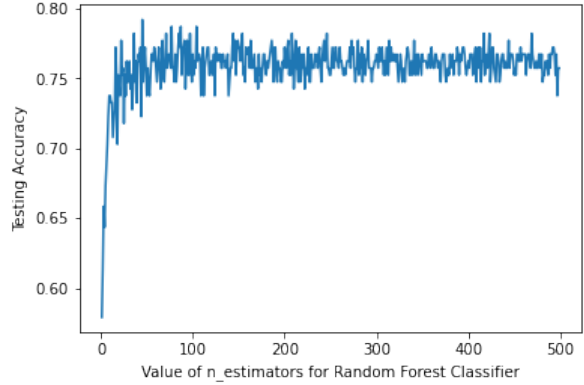


Figure 3: Random forest accuracy for different number of random estimators using the landmarks dataset

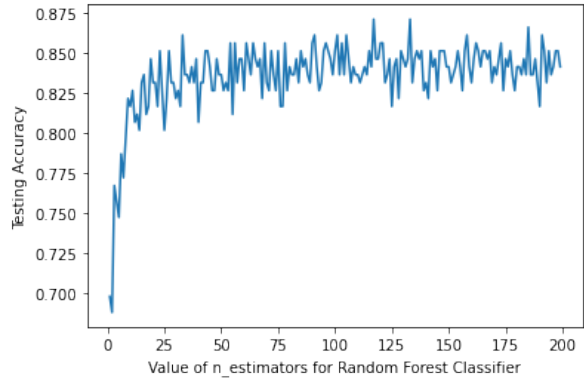


Figure 4: Random forest accuracy for different number of random estimators using the metrics dataset

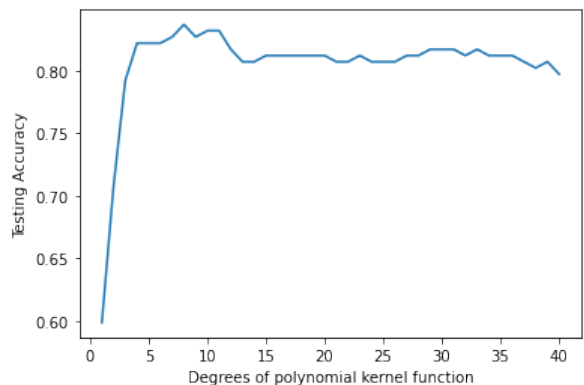


Figure 5: SVM accuracy for different degrees of polynomial kernel function using the landmarks dataset

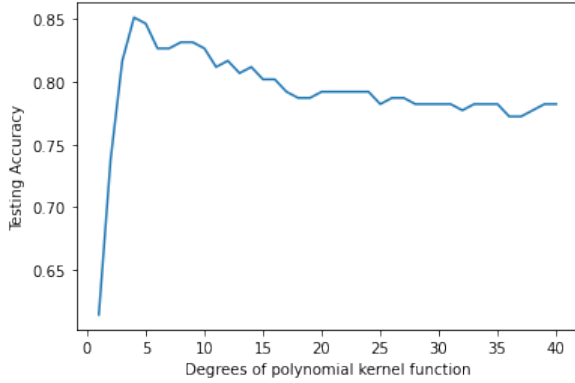


Figure 6: SVM accuracy for different degrees of polynomial kernel function using the no-chin dataset

Finally, the support vector machine was tested on the metrics dataset. The polynomial kernel function and balanced class weights still performed best, but a final accuracy of only $a = 61.9\%$ was achieved, using a polynomial kernel function of degree 15. The resulting confusion matrix is given in Table 10.

3.6 Varying the dataset size

Finally, the accuracy a and central palsy sensitivity s are investigated when the dataset varies in size. Data points are removed one at a time from the dataset. For every 10 data points removed, 5 are peripheral palsy patients, 3 are central palsy patients, and 2 are healthy people. This roughly keeps the same division between data points in the dataset. The accuracy and sensitivity are then calculated until the dataset has only 40 data points left $\sim 20\%$ of the original dataset. The resulting plots for the SVM on the landmarks dataset and the Gaussian naive Bayes on the metrics dataset are shown in Figures 7 and 8, respectively.

The behaviour at larger datasets is then estimated by fitting the following curve to all lines:

$$y = f(x) = 1 - a \cdot e^{b \cdot (x-c)}.$$

Here, a , b , and c are unknown coefficients, y and x the dependent and independent variable, and $f(x)$ is a function that approaches 1 at $x \rightarrow \infty$. This is reasonable, since we would expect a machine learning algorithm to perform perfectly at infinite training dataset size.

Table 9: Confusion matrix for the SVM algorithm on the no-chin dataset

Diagnosis		Actual		
		P	C	H
Predicted	P	82.4 %	30.0 %	0 %
	C	17.7 %	70.0 %	0 %
	H	0 %	0 %	100 %

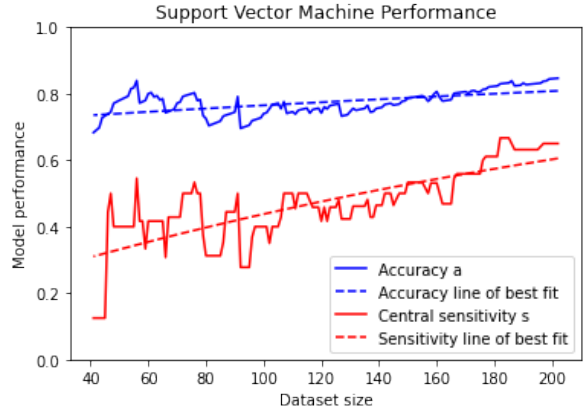


Figure 7: SVM accuracy a and sensitivity s for central palsy for different dataset sizes on the landmarks dataset

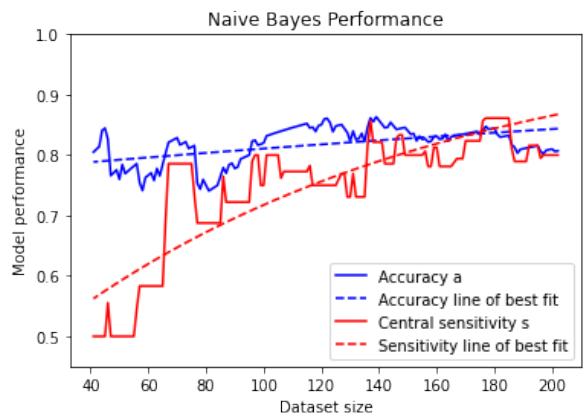


Figure 8: Gaussian naive Bayes accuracy a and sensitivity s for central palsy for different dataset sizes on the landmarks dataset

The resulting curve fit lines are also shown in Figures 7 and 8. We highlight the line fitted to the naive Bayes sensitivity, which is as follows:

$$y = 1 - 2.64 \cdot e^{-0.00741 \cdot (x+201)}.$$

This line would intersect a performance of 95 % at a dataset size of 334 pictures.

4 Discussion

The four best performing algorithms were the Gaussian naive Bayes on the metrics dataset (80.7 %), the

Table 10: Confusion matrix for the SVM algorithm on the metrics dataset

Diagnosis		Actual		
		P	C	H
Predicted	P	49.0 %	15.0 %	0 %
	C	37.3 %	57.5 %	13.3 %
	H	13.7 %	27.5 %	86.7 %

random forest on the metrics dataset (85.1%), and the support vector machine on the no-chin dataset (85.1%). The confusion matrices of these algorithms are given in Tables 2, 8, and 9, respectively.

Comparing these confusion matrices, we find that while the random forest achieved the highest accuracy, it correctly classifies individuals with a central palsy only in 37.5% of cases, which is not a good performance. It does perform well on healthy individuals (100%) and people with a peripheral facial palsy (95.1%). Since a central palsy could be a stroke, the poor performance on this condition is not ideal. This result could be caused by the unbalance of data points between the three classes.

The SVM has the highest performance on healthy people, since it correctly classifies 100% of them as healthy, and also classifies 0% of patients as being healthy. The highest central palsy accuracy is reached by the naive Bayes algorithm. This algorithm classified 80% of central palsy patients correctly, and thus ‘misses’ the least cases of this disease. It could thus be argued that this algorithm performs the best, even though the random forest and SVM both reached higher overall accuracies a .

Another noteworthy result is that five out of six algorithms –only the random forest did not– performed better on the no-chin dataset than on the landmarks dataset, even though the no-chin dataset is just the landmark dataset with the features corresponding to 17 landmarks left out. We assume this to be caused by the fact that the chin position differs very little between healthy people and patients. This could result in a distorted view of the differences between the three classes, which leads to poorer clustering abilities of the algorithms, and thus a lower testing accuracy.

Lastly, Figure 7 and 8 show steady increase in both the general accuracy and sensitivity of central palsy. The Gaussian naive Bayes however, shows a steeper slope on sensitivity. Adding more images to the dataset could give a superior central prediction earlier on compared to the SVM approach. By curve fitting, an expected dataset size of 334 images is needed to achieve a central palsy sensitivity of 95% with the Gaussian naive Bayes approach.

5 Conclusions

The goal of this study was to implement an algorithm to classify any individual as healthy or as having a peripheral or central palsy, by using the relative location of 68 facial features (landmarks) on their face. In machine learning, this is described as a three-way classification problem. The dataset used in our study contains 202 individuals.

Experiments were then run with five different algorithm types, explained in Section 2.2, on three different representations of the palsy dataset, as

described in Section 2.1. Some algorithms were first optimised by varying some of their model parameters and running accuracy tests for each.

The highest accuracies that were reached are 85.1% using a SVM with a polynomial kernel function of degree 4 on the landmarks dataset and 85.1% using a random forest with 100 random estimators on the metrics dataset. Both made no errors when an individual is healthy. The random forest on the metrics data seems to be overfitted to peripheral patients (see Table 8), while the SVM on the landmarks dataset performs similarly for both palsies (see Table 9). The highest correct classification rate for patients with a central palsy –the most severe condition– was reached by the Gaussian naive Bayes on the metrics dataset (80%), while achieving a lower accuracy than the SVM and random forest of 80.7%.

By curve fitting, it was shown that for both the SVM and naive Bayes algorithms, accuracy a and central palsy sensitivity s tend to improve when the dataset grows in size. It was estimated that by tripling the dataset in size, central palsy sensitivity would achieve 95% with the naive Bayes approach.

6 Recommendations

Adding more images to the dataset would be a first step to improving the current results. This could allow the training of a deep neural network (DNN), which could achieve much better results.

Further tuning of the different MLAs to improve the accuracy is also advised. Some model parameters of MLAs presented in this study have not yet been tuned. It is expected that tweaking some of these could still lead to better results.

Another point that needs more research, is the weight of the two palsy conditions. Since central palsy is a much more severe disease than peripheral palsy, it could be preferred that an automatic algorithm favours central palsy diagnosis. For central palsy, a false positive is less dangerous than a false negative, since the first would just alert a doctor, while the second could falsely ease one. In short, sensitivity and specificity of central palsy diagnosis could be another metric – besides accuracy – to compare different MLAs.

The significance of each of the 68 landmarks could also use more research. Since leaving out the chin landmarks resulted in higher accuracy for five out of six landmarks, there is reason to believe that some other combination of landmarks could also have this effect.

Furthermore, the line fitting on the performance curves (Figures 7 and 8) could be further investigated. The negative exponent curve we fitted to all lines might not be perfect. It could for example be

investigated if a different base number than Euler's number achieves better results.

Lastly, a platform should be made to easily implement the diagnosis system in a hospital setting, after it has surpassed the preferred accuracy. Fitting a linear, polynomial, or a curve fitted regression model on the dataset-size accuracy discussed in Section 3.6 should result in a prediction model to predict the preferred accuracy. For this, steps 1 and 2 as described in Section 1 should be combined with a diagnosis algorithm proposed in this article to form a complete automatic palsy diagnosis system.

References

- [1] E. Peitersen, "The natural history of bell's palsy," *The American Journal of Otolaryngology*, vol. 4, no. 2, pp. 107–111, 1982.
- [2] C. Gordon, R. L. Hewer, and D. T. Wade, "Dysphagia in acute stroke," *Br Med J (Clin Res Ed)*, vol. 295, no. 6595, pp. 411–414, 1987.
- [3] N. Sourlos, W. E. Amerika, Z. Jaber, B. C. Jacobs, S. F. T. M. de Bruijn, and Z. Al-Ars, "Palda: Public dataset and algorithms for facial imaging and diagnosis of neurological disorders," Note: not yet published, 2022.
- [4] P. Sajda, "Machine learning for detection and diagnosis of disease," *Annu. Rev. Biomed. Eng.*, vol. 8, pp. 537–565, 2006.
- [5] *Supervised and unsupervised learning*, Oct. 2021. [Online]. Available: <https://www.geeksforgeeks.org/supervised-unsupervised-learning/>.
- [6] H. S. Kim, S. Y. Kim, Y. H. Kim, and K. S. Park, "A smartphone-based automatic diagnosis system for facial nerve palsy," *Sensors*, vol. 15, no. 10, pp. 26 756–26 768, 2015.
- [7] M. Somvanshi, P. Chavan, S. Tambade, and S. Shinde, "A review of machine learning techniques using decision tree and support vector machine," in *2016 international conference on computing communication control and automation (ICCCUBEA)*, IEEE, 2016, pp. 1–7.
- [8] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.
- [9] C. Saganos, *The 68-points mark-up used for annotation*, 2021. [Online]. Available: <https://ibug.doc.ic.ac.uk/resources/facial-point-annotations/>.
- [10] A. Y. Fattah, A. D. Gurusinge, J. Gavilan, T. A. Hadlock, J. R. Marcus, H. Marres, C. C. Nduka, W. H. Slattery, A. K. Snyder-Warwick, *et al.*, "Facial nerve grading instruments: Systematic review of the literature and suggestion for uniformity," *Plastic and reconstructive surgery*, vol. 135, no. 2, pp. 569–579, 2015.
- [11] H. S. Chauhan, *Naive bayes algorithm: Everything you need to know*, 2020. [Online]. Available: <https://www.kdnuggets.com/2020/06/naive-bayes-algorithm-everything.html>.
- [12] Sharma, *Decision tree classification | guide to decision tree classification analytics vidhya*, 2021. [Online]. Available: <https://www.analyticsvidhya.com/blog/2021/04/beginners-guide-to-decision-tree-classification-using-python/>.
- [13] T. Srivastava, *K nearest neighbor | knn algorithm | knn in python & r. analytics vidhya*. 2018. [Online]. Available: <https://www.analyticsvidhya.com/blog/2018/03/introduction-k-neighbours-algorithm-clustering/>.
- [14] T. Yui, *Understanding random forest - towards data science*, 2021. [Online]. Available: <https://towardsdatascience.com/understanding-random-forest-58381e0602d2>.
- [15] R. Pupale, *Support vector machines (svm) - an overview - towards data science*, 2019. [Online]. Available: <https://towardsdatascience.com/https-medium-com-pupalerushikesh-svm-f4b42>.
- [16] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, *et al.*, "Scikit-learn: Machine learning in python," *the Journal of machine Learning research*, vol. 12, pp. 2825–2830, 2011.