# Image Analysis

Optimization and Data Analytics mini-project

Morten Lyng Rosenquist
*Faculty of Technical Sciences*
*Aarhus University*
Aarhus, Denmark
201706031

***Abstract*—This paper analyze the Image Classification problem for two different data sets. The classifiers used are Nearest Class Centroid, Nearest Sub-Class Centroid, Nearest Neighbor, Perceptron using Backpropagation and Perceptron using MSE. The classification is performed on the sets with full features and the sets having their features reduced to two utilizing PCA. It is seen that the methods vary in performance with respect to training time and testing accuracy.**

***Index Terms*—classification, orl, mnist, pca, data analytics**

## I. Introduction

This thesis will analyze the Image Classification problem. Utilizing only supervised learning algorithms This revolves around training a classification model with train images and thereafter being able predict correct classes for test images. The computer doesn't see the physical images as humans will, instead it sees a vector or matrix of pixels. The used classification models is described in the following sections.

### *A. Nearest Centroid*

Nearest Centroid, also called Nearest Prototype, calculates the centroids of each class and then predicts an observation to a class whose mean is the closest. Given a set of N samples, where each sample is represented by a vector $\mathbf{x}_i \in \mathbb{R}^D$ and the corresponding labels are denoted as $l_i$. The centroids are denoted as $\mu_k, k = 1, ..., K$ where $K$ is the amount of classes. Then during training the centroids are defined[1]:

$$\mu_k = \frac{1}{N_k} \sum_{i, l_i = k} \mathbf{x}_i, k = 1, ..., K, \tag{1}$$

Having calculated the centroids, a new observation $\mathbf{x}_*$ can be calculated based on the minimal distance from $\mu_k$:

$$d(\mathbf{x}_*, \mu_k) = ||\mathbf{x}_* - \mu_k||_2^2 \tag{2}$$

### *B. Nearest sub-class Centroid*

Nearest sub-class centroid is similar to the Nearest Centroid. It introduces a new sub-class element, that divides each class in a set amount of subclasses. Then during training the centroids of the subclasses are calculated. During predictions an observation is classified to the class which subclass it had the minimal distance. Given a set of N samples, where each sample is represented by a vector $\mathbf{x}_i \in \mathbb{R}^D$ and the corresponding labels are denoted as $l_i$. Having a sub classes $m$ of class $c_k$ the centroids of each class can then be calculated[1]:

$$\mu_{km} = \frac{1}{N_{km}} \sum_{i, l_i = k, q_i = m} \mathbf{x}_i, k = 1, ..., K, \tag{3}$$

The subclass label of vector $\mathbf{x}_i$ is $q_i$ and $N_{km}$ is the amount of samples of class $c_k$ composing subclass m. Classification on observations can then be done based on distance similar to the Nearest Centroid:

$$d(\mathbf{x}_*, \mu_{km}) = ||\mathbf{x}_* - \mu_{km}||_2^2 \tag{4}$$

### *C. Nearest Neighbor*

The Nearest Neighbor schema stores the training data and classifies observations based on which training data samples they are closest to. The amount of neighbors to specify is highly dependent on the data set. A high amount of neighbors reduces the effect of bad samples and noise[2]. Meanwhile a low amount of neighbors leads to spurious classifications of possible outliers.

### *D. Perceptron using Backpropagation or MSE*

Perceptron is a (binary) linear classifier that decides if an observation is in a class or not. As an example looking at only two classes $c_1$ and $c_2$. Given a weight vector $\mathbf{w}$ and a input vector $\mathbf{x} \in \mathbb{R}^D$ then a linear discrimination function g is:

$$g(\mathbf{x}) = \mathbf{x}^T x + w_0 = \sum_{d=1}^{D} w_d x_d + w_0 \tag{5}$$

$x_0$ is called the bias. The class decision of $\mathbf{x}$ is then dependent on the result of $g(\mathbf{x})$:

$$l_i = \begin{cases} c_1 & \text{if } g(x) > 0 \\ c_2 & \text{if } g(x) < 0 \\ c_1 \lor c_2 & \text{if } g(x) = 0 \end{cases} \tag{6}$$

The weights are updated after each sample:

$$\mathbf{w}(t+1) = \mathbf{w}(t) - \eta(t)\Delta J_p = \mathbf{x}(t) + \eta(t) \sum_{\mathbf{x}_i \in X} l_i \mathbf{x}_i \tag{7}$$

Where $\eta(t)$ is the learning rate that is either constant during all samples or are being according to a certain algorithm. If the learning rate is to high the classifier will converge to fast and the decision boundaries will be incorrect. Otherwise if it's to low the classifier will be slow. $\Delta J_p$ is the gradient of our optimization problem. As for utilizing this classifier for a problem with more than two classes we need to combine multiple binary classifiers. One option is the one-versus-all. If there is K classes then we need K binary classifiers. First one that determines if one class is different from the others, and a second one that determines if the second class is different from the others, and so on. This utilizes backpropagation by updating $\mathbf{w}$ such that it will be more fit to handle a similar input the next time[1].

Instead of utilizing backpropagation the Minimum Squared Error can be used. The goal is still to update $\mathbf{w}$ as optimal as possible. This is done by minimizing an error vector $\mathbf{e}$:

$$\mathbf{e} = \mathbf{X}^T\mathbf{w} - \mathbf{b} \tag{8}$$

Where X is a matrix with columns containing the training vectors and $\mathbf{b}$ is vector with the target values. Solving the optimization problem leads to[1]:

$$\mathbf{w} = (\mathbf{X}\mathbf{X})^{-1}\mathbf{X}\mathbf{b} = \mathbf{X}^{\dagger}\mathbf{b} \tag{9}$$

The backpropagation solution fits well when clear decision boundaries can be drawn between the classes. If that is not the case utilizing Minimum Square Error can be more beneficial.

### E. Principal component analysis

There is advantages and disadvantages in reducing the dimensions of the data. It makes it easier to visualize the data. Having the feature reduction as a preprocessing step the time of training and testing classifiers will reduce. As all data can not be covered in the reduced dimensions it will also result in reduced accuracy when performing classification. The goal in reducing features is keeping it uncorrelated, meaning it should be possible to still differ which class each sample belongs to. PCA works by the first dimension having the largest possible variance. Then remembering the first dimension each succeeding dimension is also to have the largest possible variance[1].

## II. Databases

The data sets that are analyzed in this paper are MNIST and ORL.

### A. ORL

The ORL data set contains 400 vectorized images. The images are of size 40x30 and depict a persons face in an upright position in a frontal view[3]. There is 10 images of 40 different persons. Examples of the images can be seen on Figure 1.

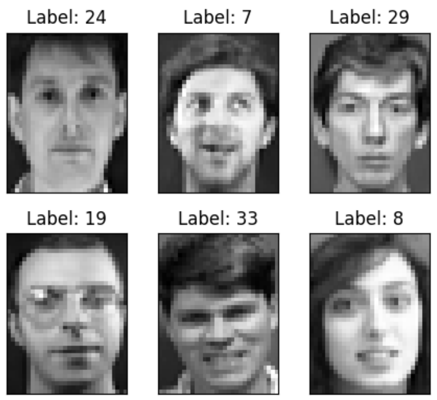


Fig. 1: ORL example images

### B. MNIST

The MNIST dataset is larger with 70,000 vectorized images. The images depict handwritten digits and are of size 28x28. Differing from the ORL data set, the set is already split in training and test data. Leading to 60,000 images for training and 10,000 for testing. Examples from the MNIST data set can be seen on Figure 2.

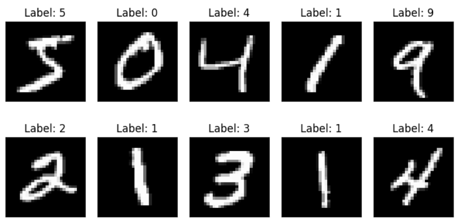


Fig. 2: MNIST example images

## III. Methods

The classification and visualization of the data is implemented in python and the source code can be found on github and in the attached zip folder. The primary packages used are sklearn for data analysis and classification and matplotlib for data visualization. The steps for each classification are:

1) Loading and reshaping data.
2) (ORL only) split train and test data
3) Tuning relevant hyperparameters using GridSearchCV
4) Fit model with train data
5) Predict test labels with test data
6) Output performance in logs and perform data visualization

## IV. Results

The performance of each classifier for both datasets will in this section be covered.

### A. PCA impact on data

Classification is performed both on the original data and on the data reduced to two dimensions using PCA. Illustrations of the data lost by reducing the dimensions can be seen on Figure 3 for ORL and Figure 4 for MNIST. These figures are the same images illustrated on Figure 1 and Figure 2, but where PCA has been applied and thereafter retransformed to their original dimensionality.

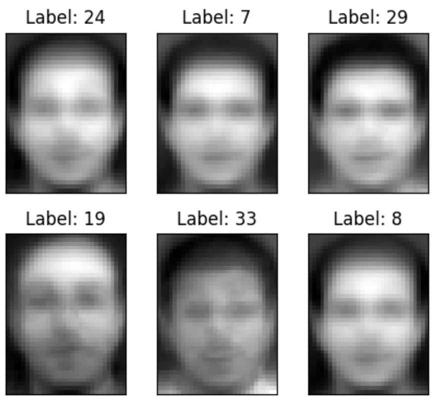


Fig. 3: ORL images reconstructed after PCA

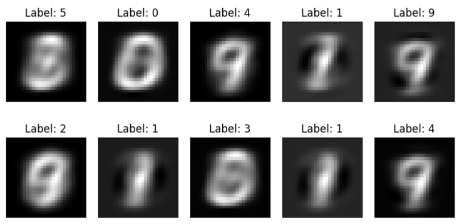


Fig. 4: MNIST images reconstructed after PCA

Having the dimensions reduced to two, makes it easy to visualize the data. On Figure 5 can a scatter plot for the MNIST test data be seen.

As for the ORL data set there is 40 classes, which is hard to illustrate in a single scatter plot. Therefore Figure 6 contains the first 20 classes while Figure 7 contains the other 20 classes.

### B. Classification

Each classifier, for both the original data and the PCA version of the data, have their hyperparameters tuned if relevant. Afterwards the classification is performed. The accuracy of class predictions of the test data can be seen on Table I. Measurements of the time spent training can aswell be seen.

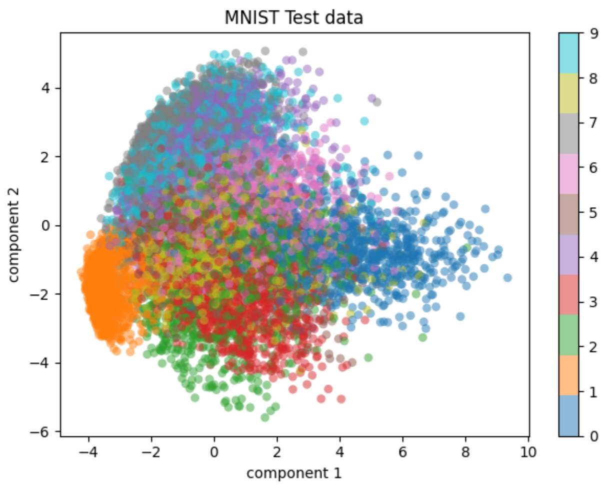


Fig. 5: MNIST test PCA images scatter plot

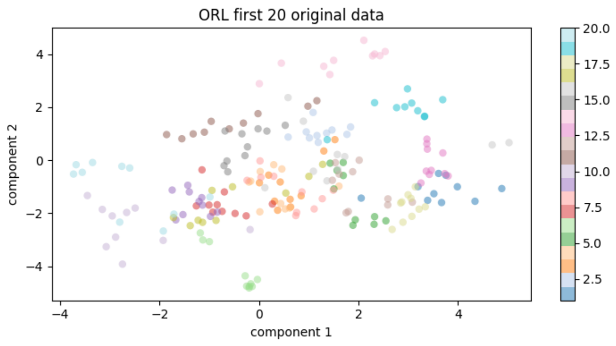


Fig. 6: ORL original PCA images first 20 classes

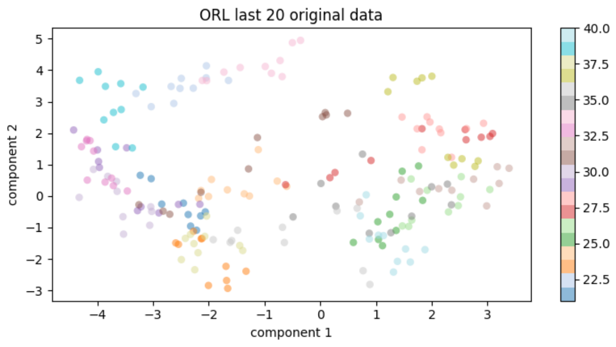
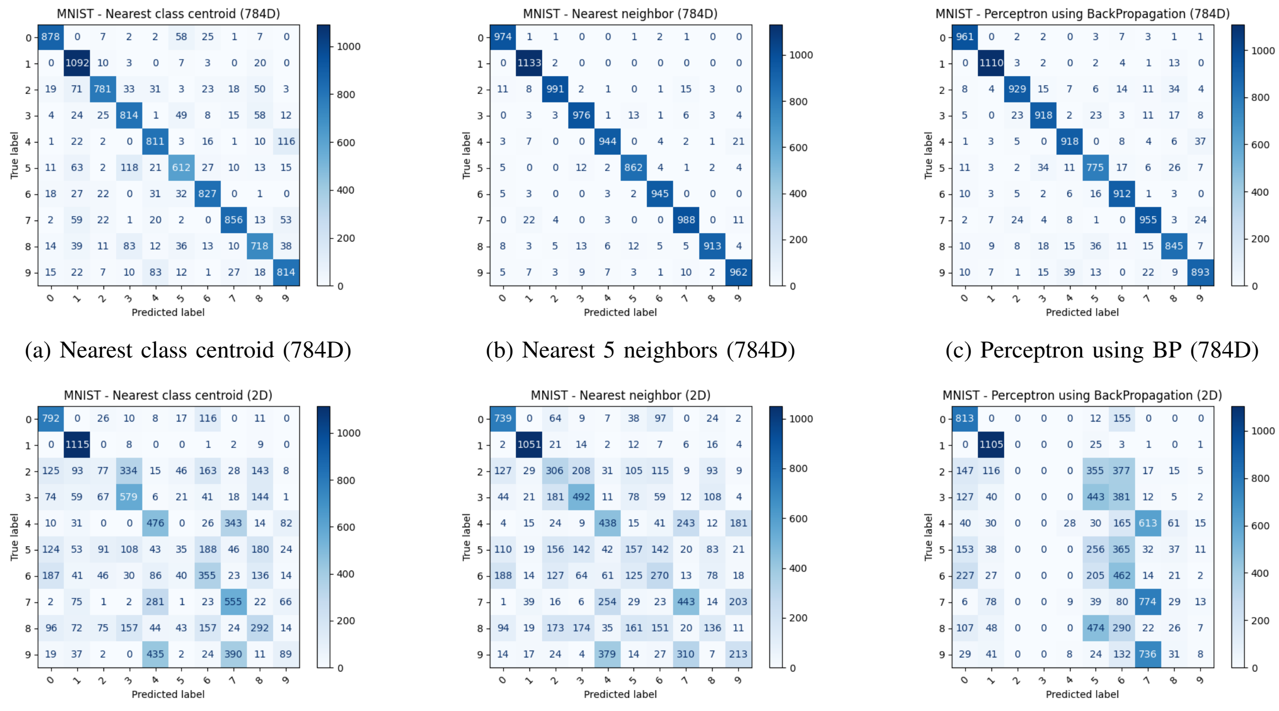


Fig. 7: ORL original PCA images last 20 classes

| Dataset | Classifier | Accuracy (Raw) | Accuracy (2d) | Time (Raw) | Time (2d) |
|---|---|---|---|---|---|
| MNIST | | | | | |
| | Nearest Class Centroid | 82.03% | 43.65% | 0.510 s | 0.012 s |
| | Nearest 2 Sub-Class Centroid | 86.07% | 43.07% | 14.306 s | 9.687s |
| | Nearest 3 Sub-Class Centroid | 88.24% | 42.39% | 16.759 s | 9.785 s |
| | Nearest 5 Sub-Class Centroid | 90.27% | 41.81% | 25.618 s | 12.077 s |
| | Nearest Neighbor | 96.88% | 42.43% | 0.023 s | 0.031s |
| | Perceptron with Backpropagation | 92.16% | 34.72% | 5.724 s | 0.093 s |
| | Perceptron with MSE | 86.08% | 32.79% | 6.121 s | 0.094 s |
| ORL | | | | | |
| | Nearest Class Centroid | 87.50% | 33.33% | 0.005 s | 0.001 s |
| | Nearest 2 Sub-Class Centroid | 94.17% | 40.00% | 4.279 s | 0.179s |
| | Nearest 3 Sub-Class Centroid | 96.67% | 35.83% | 4.853 s | 0.264 s |
| | Nearest 5 Sub-Class Centroid | 96.67% | 34.17% | 4.846 s | 0.289 s |
| | Nearest Neighbor | 95.83% | 30.83% | 0.0004 s | 0.0003 s |
| | Perceptron with Backpropagation | 95.83% | 19.17% | 0.138 s | 0.014 s |
| | Perceptron with MSE | 95.83% | 13.33% | 0.248 s | 0.013 s |

TABLE I: Performance of each classifier for both datasets



(a) Nearest class centroid (784D)
(b) Nearest 5 neighbors (784D)
(c) Perceptron using BP (784D)

(d) Nearest class centroid (2D)
(e) Nearest 5 neighbors (2D)
(f) Perceptron using BP (2D)

Fig. 8: Confusion Matrices - MNIST

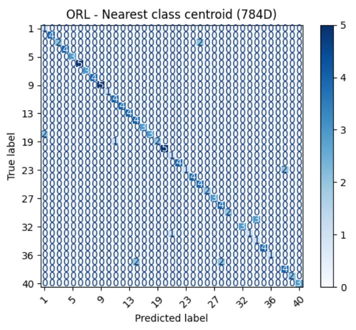
## C. *Visualization*

Confusion matrices showing how certain classifiers performed when predicting labels for the test data can be seen on Figure 8 for the MNIST data set and on Figure 9 for the ORL data set.

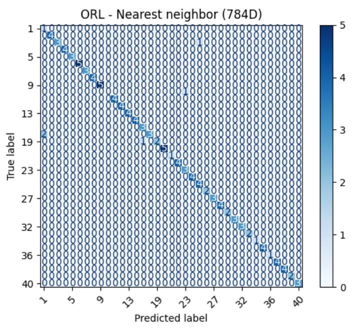
Scatter plot figures including boundary lines can be seen for certain classifiers on Figure 10-12. The figures illustrates the class's boundaries for the trained models.

## V. Discussion

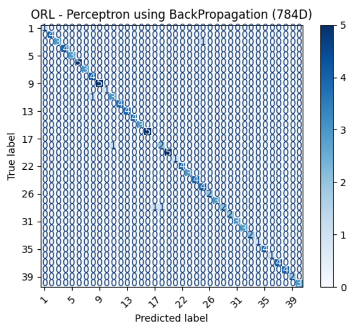
Comparing the performance of classification of the full features and to two features, it is clear that they distinguish. The accuracy is much lower using two features. As an example for Nearest Neighbor on the MNIST data set the accuracy for full features is 96.88% while being 42.43% on two features. Furthermore the training time is in general way faster on the lower features. The time can't directly be compared between classifiers, since some utilize parallelism. Looking at perceptron with BP for MNIST the full features took 6s while on two features only 0.1s. It's seen on Figure 12 that there is only eight classes, meanwhile there is supposed to be ten. As shown in the confusion matrix on Figure 8f this clearly leads to test data being misclassified, where some classes are not represented at all. As PCA only looks into the variance of variables and not including response variables, it might have been better for the accuracy to utilize linear



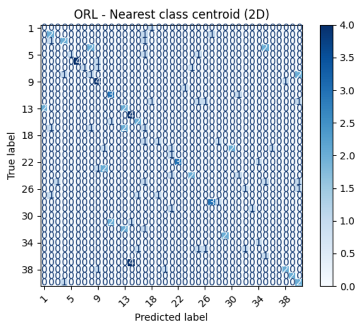
(a) Nearest class centroid (784D)



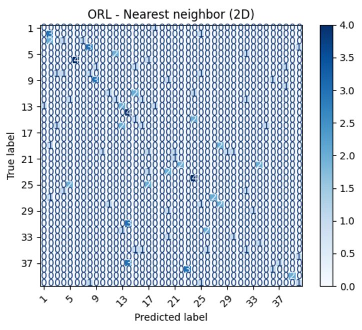
(b) Nearest 5 neighbors (784D)



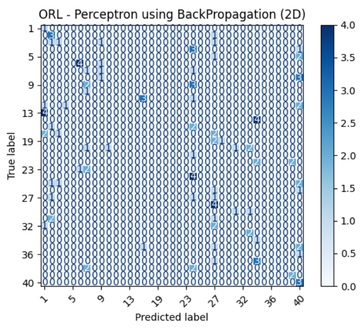
(c) Perceptron using BP (784D)



(d) Nearest class centroid (2D)



(e) Nearest 5 neighbors (2D)



(f) Perceptron using BP (2D)

Fig. 9: Confusion Matrices - ORL

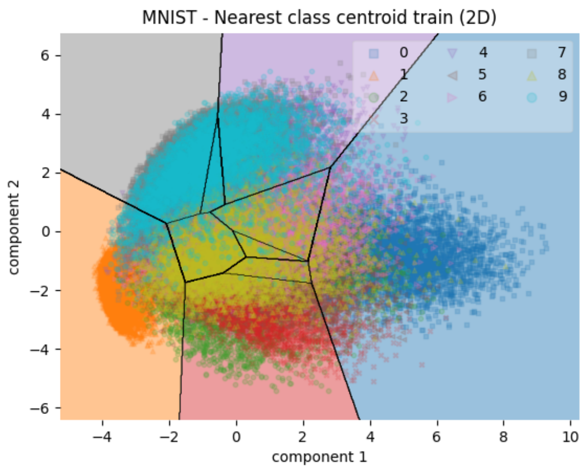


Fig. 10: Boundary Lines - Nearest Centroid train data

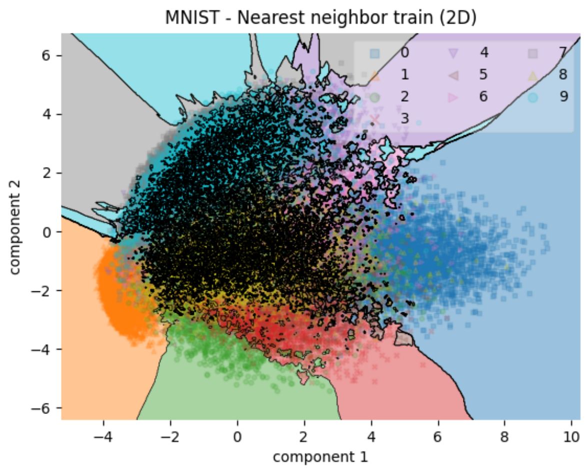


Fig. 11: Boundary Lines - Nearest Neighbor train data

discriminant analysis instead. It's seen that the classifiers make different decision boundaries and vary in performance. Regarding perceptron on both data sets it is seen on lower features, that they have the lowest accuracy for both data sets. This is possibly caused by them making linear boundary lines. Compared to nearest neighbour that makes many islands related to the nearest observations. Furthermore it's seen that classifiers that perform well on the MNIST data set, does not necessarily perform well on the ORL data set.

## VI. Conclusion

This thesis showed how various supervised learning algorithms can handle the Image Classification problem. Resulting in varying results for two different datasets. Reducing the amount of features led to lower accuracies but reduced the training time drastically. Certain algorithms performed better on one data set while others performed better on the different data set. The reduced features allowed data visualization and it was shown how certain algorithms make their decision

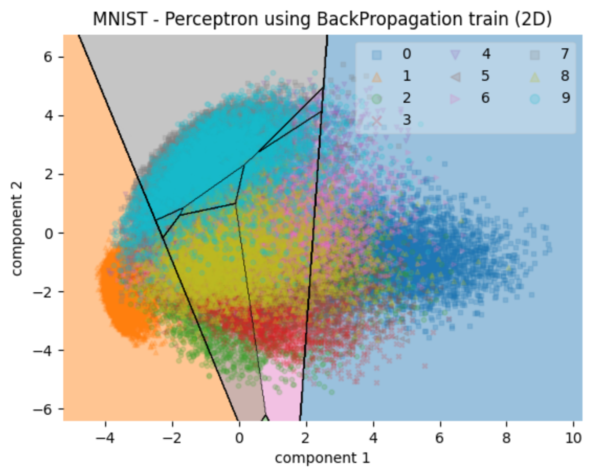


Fig. 12: Boundary Lines - Perceptron using BP train data

boundaries. It would be interesting in future work to study how the classifiers would perform on higher than 2 features.

## References

[1] Alexandros Iosifidis. *Introduction to Machine Learning*. 2018.

[2] i-king-of-ml. *KNN(K-Nearest Neighbour)*. URL: https://medium.com/@rdhawan201455/knn-k-nearest-neighbour-algorithm-maths-behind-it-and-how-to-find-the-best-value-for-k-6ff5b0955e3d. (accessed: 15.11.2021).

[3] Nimfa. *ORL Images*. URL: https://nimfa.biolab.si/nimfa.examples.orl_images.html. (accessed: 15.11.2021).