Linear versus Non-linear Dimensionality reduction

Dimensionality reduction as preproccesing of image data for use in image classification

Daniel Runge Petersen, Gustav Svante Graversen, Lars Emanuel Hansen, Raymond Kacso, Sebastian Aaholm

Computer Science, cs-22-dat-5-05, 2022-12

Semester Project

Composed and typeset by the authors using the LaTeX Document Preparation System, based on the AAU report template by Daniel Runge Petersen [1].

Electronics and IT
Aalborg University
http://www.aau.dk

AALBORG UNIVERSITY
STUDENT REPORT

Title:
Linear versus Non-linear Dimensionality Reduction

Abstract:
This is the abstract...

Theme:
Theoretical data analysis and modeling

Project Period:
Fall Semester 2022

Project Group:
cs-22-dat-5-05

Participant(s):
Daniel Runge Petersen
Gustav Svante Graversen
Lars Emanuel Hansen
Raymond Kacso
Sebastian Aaholm

Supervisor(s):
Alexander Leguizamon Robayo

Copies: 1

Page Numbers: 47

Date of Completion:
December 7, 2022

Contents

Preface

This is the preface. You should put your signatures at the end of the preface.

Aalborg University, December 7, 2022

Author 1
<username@student.aau.dk>

Author 2
<username@student.aau.dk>

Author 3
<usernamestudent.aau.dk>

Author 4
<username@student.aau.dk>

Author 5
<username@student.aau.dk>

Author 6
<username@student.aau.dk>

Chapter 1

Introduction

Chapter should probably contain: The initial problem (If we have one), motivation, and the scope or background of the project or theme. Report outline at the end.

In this project we will study some common methods of dimensionality reduction using the MNIST dataset for digit recognition.

Keywords: dimensionality reduction, linear methods, nonlinear methods, MNIST, Computer Vision (CV), Machine Learning, machine intelligence (artificial intelligence).

Use sources [2] and [3] for superficial overview and explanations of umbrella terms.

Notes to self: Humans vs computers in Neural Network (NN). Why are humans good with little training, and computers only accceptable with much more training? Consider perhaps domains (recongnizing epsilon vs. recognizing a 3)

On theory driven projects

> The overall purpose of the project module is for the student to acquire the ability to analyze and evaluate the application of methods and techniques within database systems and / or machine intelligence to solve a specific problem. **This includes analyzes of the formal properties of the techniques and an assessment of these properties in relation to any requirements for the solution to the specific problem**. [...]
>
> In this project module, the project work is primarily driven by theoretical and analytical considerations about the methods and techniques used. For a specific problem area, a project could, for example, be based on specific performance requirements for the developed software solution, and the project work can thus be guided by the solution's algorithmic time / space complexity as well as formal analyzes and considerations of its theoretical properties and performance guarantees. [4]

1.1 Motivation

High-dimensional data (i.e. data that requires more than three dimensions to be represented) can often be difficult to work with. Not only is it difficult to interpret and visualize but also can require a high use of computational resources. For these (and many more) reasons, it is important to study dimensionality reduction methods. These methods are usually used in exploratory data analysis and for visualization purposes.

The most usual methods of dimensionality reduction are **linear methods**. These methods might assume that the features in the original data are independent and they can produce reduced data by a linear combination of the original data. These assumptions might not apply to all datasets. In fact, there are cases in which linear methods do not capture important features of a dataset. For these cases one can use **nonlinear methods**/ These methods can be used for more general cases while preserving important information from data.

For now this is taken directly from the project proposal. We may rewrite this partially at a later time.

1.2 Report outline

The report is structured as follows:

- **Introduction** - This chapter
- **Problem Analysis** - Chapter 2
- **Theory** - Chapter 3
- **Methodology** - Chapter 4
- **Results** - Chapter 5
- **Discussion** - Chapter 6
- **Conclusion** - Chapter 7

The introduction describes the initial problem and the motivation for the project.

The Problem Analysis chapter dives into the initial problem and leads to a final problem statement.

The Methodology chapter describes the methods and theory used to explore the problem statement. It also describes the data used and how it was collected/created.

The Results chapter is an evaluation of the results of the project.

The Discussion chapter is a discussion of the results and the project as a whole.

The Conclusion chapter provides a summary of the project and the results. It also provides perspective and reflection on the project and the process.

Chapter 2

Problem analysis

This chapter describes the motivation for the project, culminating in a problem statement.

Machine Learning (ML) is a field of study within Artificial Intelligence (AI) concerned with learning from data, where learning means that data is analyzed and something is gathered from it. ML could be methods to solve a puzzle or patterns to recognize a number from an image. ML is a complex and growing field used in many areas. One of the reasons for the increasing interest in ML is the increasing amount of available data. The data collected worldwide is increasing at an incredible rate, which seems to continue in the future [5]. Because ML models train on data, the more data available, the better the models can be [6].

ML can be described as the discipline of teaching computers how to complete tasks where no perfect algorithm is possible. This also covers when there are many possible good ways to achieve something; for this, all the acceptable methods are labeled as acceptable ways to succeed. These answers help to "train" the computer to improve on some basic algorithm [7].

Inside a field as complex as ML, many challenges exist; one such challenge is working with high-dimensional data. This is due to the inherent properties of many dimensions, making it challenging to interpret and computationally expensive. Higher dimensions are also associated with the so-called "curse of dimensionality". Richard E. Bellman coined this term in a paper about dynamic programming [8]. According to John A. Lee:

> the curse of dimensionality also refers to the fact that in the absence of simplifying assumptions, the number of data samples required to estimate a function of several variables to a given accuracy ... on a given domain grows exponentially with the number of dimensions [9].

These difficulties associated with data in many dimensions make the study of dimensionality reduction highly relevant. The goal of dimensionality reduction is to reduce the number of dimensions in a dataset while retaining as much information as possible. Dimensionality reduction can improve interpretability on data and, if used in a ML pipeline, make the pipeline faster. Dimensionality reduction is

made by extracting features from the data, which trains a model. The extracted features help to predict new results on new data. This section is a general overview of the dimensionality reduction process and will be discussed in more detail in the next chapter. It will be explained what it is and some of its uses in the following paragraphs.

2.1 Machine learning

A pipeline is good for comparing and contrasting linear and nonlinear methods through the lens of a model efficiently. In this section, there will be a brief Introduction to pipelines, their use, and how to introduce dimensionality reduction to them.

Figure 2.1 illustrates a general ML pipeline. A ML pipeline consists of four main steps: data, Feature Engineering (FE), ML model training, and model evaluation. The first step is the data step, which is the data collection. The FE step transforms the data into a more suitable form for the ML model. The ML model training step trains the ML model on the data. The model evaluation step is used to evaluate the performance of the ML model. The ML model can then be used to make predictions on new data [10].

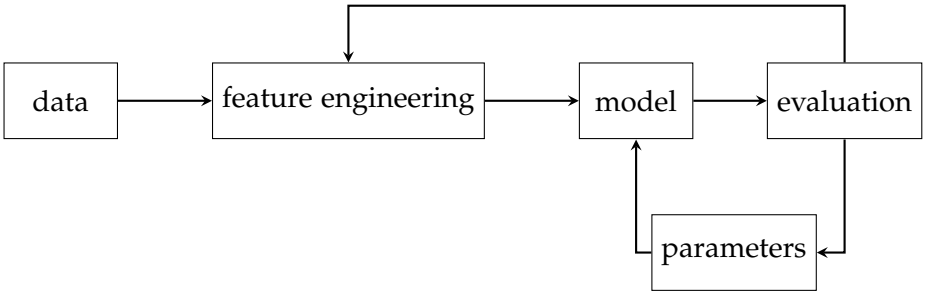

Figure 2.1: Simplified machine learning pipeline

The FE step is where dimensionality reduction is relevant - this step reduces the dimensions to increase the model's performance.

2.2 Dimensionality reduction

Dimensionality reduction is reducing the number of features in the data. This is done by removing features that are irrelevant to the task at hand or by combining features. The goal of dimensionality reduction is to reduce the number of features in the data while still retaining the essential information [11].

The advantages of dimensionality reduction are that it can reduce the amount of data that needs to be processed, making the model faster or require fewer resources. It can also make the model more interpretable because there are fewer features to look at. Dimensionality reduction can also improve the model's performance because it can remove noise from the data and make the model more robust [11].

Time is a significant factor when discussing the ML model, as the time complexity of the model is a significant factor in the overall time complexity of the system.

> Dimensionality reduction techniques can tremendously reduce the time complexity of training phase of ML algorithms hence reducing the burden of the machine learning algorithms. [12].

Therefore, when reducing features in a dataset, it is crucial to find the features that are more relevant to the output of the model [13]. There are many ways to reduce the dimensionality of a dataset; the different types of dimensionality reduction will be further discussed in Chapter 3.

Interpretability is another significant factor when discussing the ML model; interpretability is the ability to understand the model's decisions. The interpretability of the model is a significant factor in the overall understanding of the system. This project will not focus on interpretability, but it is still a significant factor when discussing the ML model.

2.2.1 Applications of dimensionality reduction methods

Dimensionality reduction has many applications, and different methods can prove to be more suitable for different applications. An example of the applications of dimensionality reduction can be seen in [14], which discusses the use of dimensionality reduction in recommender systems. Using dimensionality reduction methods helps the system's scalability by reducing the amount of data run on the system and improving the quality of the recommendations. There are many dimensionality reduction methods, and more are still being researched today [11]. Moreover, there are many different uses for these methods. An example of this could be to improve heuristic models for explaining the data from surveys better, through better visualizations as it helps to improve understanding [11].

Additionally, dimensionality reduction can be used in applications such as image compression, visualization of high-dimensional data, and feature extraction. It shows that it has many different applications and is a handy tool.

Many more examples exist, but the main takeaway is that dimensionality reduction is a powerful tool in many different fields and a vital part of the ML model. We will categorize dimensionality reduction into two general categories in the following section.

2.3 Linear versus nonlinear methods

This section will explore dimensionality reduction further. Specifically, this project will distinguish between linear and nonlinear methods. According to John A. Lee [9], several distinctions can be made for dimensionality reduction methods. This project will only focus on one distinction from [9], linear and nonlinear because it is a very straightforward way of classifying dimensionality reduction.

It is essential to distinguish between methods to classify them and to understand their differences. This will help to understand the advantages and disadvantages of each method. This will also help to understand which method is the most suitable for a specific problem [9]. The following section will discuss the differences between linear and nonlinear methods.

2.3.1 Results and differences of linear and nonlinear methods

As outlined earlier, dimensionality reduction methods can be used to remove redundancy from data, which can improve the performance of a ML model. However, the methods can also be used for other purposes, such as visualization and feature engineering [9].

A linear method assumes linear independence of the features. Linear independence means that the features are independent of each other; this is a strong assumption, which is only sometimes true [15]. A nonlinear method does not assume linear independence of the features, which means that the features are not independent; this is a weaker assumption, which is often true [16]. That means that a nonlinear method is often more robust than a linear method. However, a nonlinear method requires more parameters, which can require more data in a model [9].

Examples of how linear and nonlinear dimensionality reduction methods can be used can be seen in [17, 18], where the methods have been tested on artificial and real-world datasets. As an example, it has shown that artificial datasets, such as the swiss-roll, show that linear methods have a more challenging time finding the intrinsic dimensionality of the data than nonlinear methods [18].

The research paper [17] compares the performance of linear and nonlinear dimensionality reduction methods with some ML models.

According to Laurens [17], there is a tendency for real-world data to be nonlinear. The linear methods should have a disadvantage because they cannot capture the intrinsic dimensionality of the nonlinear data and nonlinear methods. However, the research paper states that nonlinear methods can only sometimes outperform linear methods [17].

In this project, the focus will be on whether linear or nonlinear methods positively impact the performance of the ML model implemented in this project. The differences between the methods used in this project will be presented in section 3.5.

2.4 Problem statement

Rewrite sectiop, potentially include stuff from: The following is an example of how the problem statement should look like This problem aims to answer the question: Are nonlinear reduction techniques better than linear ones when doing machine learning? As a part of this question we need to answer the following smaller questions * where in the pipeline are we gona use the methods? * what data are we gonna use? is it only synthetic or real world or are you gonna use both? * how are we gonna evaluate performance? after classification? as a way to explain errors ? And so on

This project aims to explore the impact of dimensionality reduction, comparing linear and non-linear dimensionality reduction techniques. This will be done in regards to

the performance of a machine learning model, for the computer vision problem of image classification and recognition, of the mnist dataset.

Chapter 3

Theory

This chapter will discuss the theory behind ML and FE. We will briefly introduce a sketch of the steps in how the theory will be implemented, then describe the theory behind ML and FE, and then go into more detail about the different methods used in this project on the thoughts behind why choosing them.

3.1 Images

As mentioned before, the project will use the Modified National Institute of Standards and Technology (MNIST) dataset; the dataset comprises images; therefore, a digital image consists of pixels. In a grayscale image, each pixel has a value between 0 and 255, where 0 is white, and 255 is black. In a color image, each pixel has three values - one for each color: red, green, and blue; the amount of pixels in a color picture is the same as in a grayscale picture. A 28x28x1 grayscale image has 784 pixels, while a 28x28x3 color image also has 784 pixels [19].

In this project, dimensionality reduction is shown by each pixel representing the image's feature (dimension). As the size of images increases, this quickly becomes a large number of features.

3.1.1 Data

In this project, it was chosen to work with MNIST. MNIST was chosen because it has real-world uses inside image recognition, particularly recognizing handwritten numbers. The real-world use comes from detecting patterns in how numbers are written, which can be attributed to the writer's style, artificial errors, or both.

Another reason is that many projects revolve around recognizing handwritten digits using machine learning, therefore is well documented, which makes it easy to find information about it. That means this project can compare the results to similar projects for better discussion of results. The dataset is also tiny, which means it is easy to work with and does not require a lot of computing power [20].

The project could also have gone another route, such as collecting and cleaning data. However, choosing the MNIST dataset has simplified the overall work and

allowed more focus on the project's machine learning/dimensionality reduction part.

The group has also considered the Iris dataset and the CIFAR-10 dataset. However, the Iris dataset might have too few data samples [21], and the CIFAR-10 dataset would be too complex for this project [22]. Therefore the group has chosen to work with MNIST. As outlined before, there has been some preprocessing regarding the MNIST, which further solidified the choice of MNIST.

3.2 Machine Learning

Understanding the basics of machine learning is important to determine the appropriate dimensionality reduction methods to compare for the project. This section will describe the basics of machine learning by describing a simplified model of a machine learning model pipeline.

3.2.1 Machine learning pipeline

Figure 2.1 shows the simplified and generalized steps in the pipeline of a machine-learning model. The arrows represent how machine learning models continuously learn. The model is trained on the training set and then evaluated on the validation set. The model then updates with the new information, and the process repeats. This loop is called the training loop. The training loop repeats until the model converges or until the model is no longer improving. The model gets evaluated on the test set, and the evaluation gets used to determine the model's performance. The reason for having a pipeline for a machine learning model states as follows:

> A machine learning pipeline (or system) is a technical infrastructure used to manage and automate ML processes in the organization. The pipeline logic and the number of tools it consists of vary depending on the ML needs. But, in any case, the pipeline would provide data engineers with means of managing data for training, orchestrating models, and managing them on production. [10]

For this simplified example, the machine learning pipeline shown in this report has four main steps: data collection, feature engineering, model training, and model Evaluation. This model is not entirely identical to the model shown in [10], as this is a simplified model with slight modifications to fit this project's scope. The four steps will get described in the following subsections.

Data collection

The `data` box in Figure 2.1 represents the collection of data to be used in training the machine learning model [10]. There are many different types of data, but some of the most commonly used, as stated by [23], are `Numerical Data`, `Categorical Data`, `Time Series Data`, and `Text Data`. Often the data is in some type or format

which is not directly usable by the machine learning model. For this, the `feature engineering` step is used, which will get described in the following subsection.

Fearture engineering

`Feature engineering` represents the step where the data gets transformed through dimensionality reduction. In [10], `Data preparation` and `feature engineering` are both in a step called `prepare data`, for this model, `feature engineering` covers those steps. `feature engineering` is also the step where data is preprocessed [10]. Preprocessing is done to ensure quality data [24], and feature engineering is done, among other reasons, to improve the quality of the results, and to improve the performance of the machine learning model, by reducing the burden on the machine learning algorithms [25].

Model

`model` is where the process of training the model with the data takes place. Model training splits the data into a training set and a validation set. An example of model training gets seen in further detail in the model shown in [10], at the `Split data` step. As stated in 3.8, it is important to split the data in this manner, as it helps to reduce the risk of overfitting.

Evaluation

`evaluation` represents the step where the model gets evaluated. The evaluation compares the model's predictions with the actual values. The model's predictions get evaluated on the validation set, and this step can repeat multiple times [10]. Using accuracy, precision, recall, and F1 score metrics helps provide information regarding the performance of the model [26].

Parameters

`Parameters` represents the step where hyperparameters of the machine learning model get set and tuned. The algorithm sets some parameters through training, then there are hyperparameters, which are parameters given to the algorithms to train the model [27]. This step is purely for hyperparameters. As mentioned in the previous section 3.7, there are multiple ways to tune hyperparameters. The main reason this tuning is important is that it helps improve the model's performance and reduce the risk of overfitting [28].

3.3 Classification

When making a model, it is only sometimes possible to predict the value of a variable. For example, if one wants to predict a house's price, one can not predict the exact price, but one can predict if the price is high or low. This is called classification.

In ML, classification is when a quantitative answer is not required; a qualitative answer is satisfying. The goal is to classify the input into one of a set of categories set in the dataset. As long as the dataset supports supervised learning, classification can be applied to it [29]. This is done by training a model on a labeled dataset and then using the model to predict the category of new, unlabeled data [29].

The training data consists of input and class label pairs. The input can be a single value, such as numbers or strings, or vectors of values, such as a list of numbers or strings. The input can also be a matrix of values, such as a grayscale or color image. The class label is a discrete value, such as a number or a string. In this project, a class label is a number representing the digit in the image [29].

The model is trained to minimize the error between the predicted and actual class labels. The error is calculated by comparing the predicted class label with the actual class label [29].

The model is trained by finding the best parameters for the model, such as weights in a neural network or parameters in models, such as Support Vector Machine (SVM) [29]. These parameters help the model predict the category of new data. Hyperparameter optimization is finding the best parameters; this will be discussed further in Section 3.7.

One must consider the data and the problem when deciding which model to use. The model must be able to handle the data and the problem. In this project, the ML model is used for image classification, as the MNIST dataset is used. Several ML models have been used for image classification with MNIST in general; the ones the group has looked into is [3, 20, 30, 31]. These models can be used to classify images into one of ten categories, representing the digits 0-9, as this is the project's focus.

Support Vector Machine

SVM is a supervised ML model that can be used for classification or regression problems. It is a linear model for classification and regression problems. It is a binary classifier, meaning it can only classify two classes. It is a linear classifier, meaning it makes a decision based on a linear function of the input features [29].

SVM is a supervised learning model that classifies data by finding a mapping (hyperplane) that separates the classes in data [32]. SVM is also known as a large-margin classifier, which means that it relies on finding "a maximum-margin hyperplane to separate classes" [32]. As the name implies, SVM uses support vectors, which are the 'vectors' closest to the function defining the mapping, and alterations on those data points will influence the hyperplane. An exciting property of SVM is that it can have, among other parameters, a kernel function, which can be tuned whether the data is linearly separable or not [32]. The kernel function that SVM resembles the way Kernel Principal Component Analysis (kPCA) works, as it also uses a kernel function.

For the project's purposes, SVM is a promising model since "SVMs have been shown to perform well in a variety of settings and are often considered one of the best "out of the box" classifiers." [29]. This is because SVM is a linear model, which

means it is fast to train and predict. This is important for the project, as the model needs to classify the images quickly. The model also has a high accuracy, which is vital for the project, as the model needs to classify the images correctly. Though it is not as accurate as other models, such as NN, it is still a good model for the project. On the downside of SVM, [29] states "Though it is elegant and simple, we will see that this classifier, unfortunately, cannot be applied to most data sets, since it requires that the classes be separable by a linear boundary" [29]. It means there can be some errors in the ML model, as it is only sometimes possible to separate the classes in MNIST by a linear boundary; this is something the group has considered choosing the model.

This project is not mainly concerned with the choice of ML model but rather with the choice of dimensionality reduction methods. Therefore, SVM is chosen as the ML model as it has already been used with MNIST without dimensionality reduction [20].

3.3.1 Multi-class classification

In classification, there are two types of classification, binary classification, and multi-class classification. Binary classification is the classification of two classes, while multi-class classification is the classification of more than two classes. The MNIST dataset used in this project presents a multi-class classification problem, as the images can represent any of the ten digits. The SVM model, however, is a binary classification model and thus has to be adapted to the multi-class classification problem. there will explained two approaches to this problem: One-versus-One (OvO) and One-versus-All (OvA) [29].

One-vs-One

OvO is a method where the model trains on all possible combinations of two classes. For example, if there are five classes, the model is trained on ten different models, one for each combination of two classes; this makes it computationally expensive as it has to go through every combination. Then the model is evaluated against all other models, and the class with the highest score is picked as the predicted class [29].

One-vs-All

However, OvA is faster than OvO, as it only uses one class to distinguish if the data is similar. For example, if there are five classes, the model is trained on five variations of the model, one for each class. This makes OvA good to distinguish between the current class that is being modeled from the other classes. However, in OVA, it is harder to distinguish between the other classes that are not being trained. Then the model is then evaluated on all models, and the class with the highest score is chosen as the predicted class [29].

One-vs-One vs One-vs-All

OvO is more computationally expensive than OvA, but is more accurate. The choice of OvO or OvA is, therefore, a trade-off between accuracy and computational cost [29]. The SVM model is chosen because it is a relatively simple model, and thus OvO is chosen as the method for multi-class classification.

We use OvO because it is faster than OvA, make that clear.

3.4 Preprocessing

In this section, the theory of preprocessing and what effects it has on the ML model is discussed.

3.4.1 Definition of preprocessing

The proper definition of preprocessing or data preparation varies depending on the source. The explanation that will be used in this report is given as follows:

"Data preparation comprises those techniques concerned with analyzing raw data so as to yield quality data, mainly including data collecting, data integration, data transformation, data cleaning, data reduction, and data discretization." [24]

From this, it can be gathered that preprocessing is a broad term that can be divided into several subcategories. Not all subcategories will be relevant to this project; therefore, the following sections will only discuss appropriate subcategories of preprocessing.

3.4.2 Reasons for preprocessing

As stated in the definition, preprocessing is used, among other reasons, to yield quality data. The importance of this stems from the fact that real-world data is only sometimes clean or complete. Meaning that can be a lot of noise, which is data containing errors or outliers. This noise can be removed by preprocessing, which creates a more accurate, higher quality, and smaller dataset to gather information. This results in a reduced amount of data that the model is trained on, but the data it is trained on should be more accurate, which should then train the model more accurately and efficiently [24].

Data collected may be in a form or shape that is not compatible with the process that is needed to work with it. Therefore, to use the data, it may need to be transformed into a form that fits with the process. Transformation of data can be many things, as [33] mentions normalization as a part of the transformation, to scale the data so that it fits into a new range, Subsection 3.4.3 will give a more detailed explanation of normalization.

3.4.3 Steps of preprocessing

The amount of preprocessing depends on what is needed and what is available. In the report [33], these steps are shown.

They start by cleaning data, transforming it, reducing it, and balancing it. This section will explain the theory of the steps used in this report.

Normalization

Normalization, also called scaling, is the process of scaling the data. This is done by changing the range of the data; for example, if the data is in the field of 0-100, it can be scaled to be in the range of 0-1. This is done to make the data more suitable for the model and to make it easier to compare the data. Scaling is also a standard practice in most ML problems. There are many ways to scale the data; for example, there is a min-max scaler [13].

Min-max scaler The min-max scaler is a method that transforms the data to be in the range between 0-1 by subtracting the minimum value and dividing by the scope of the data. The formula for this is:

$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{3.1}$$

Where x is the original value, x_{scaled} is the scaled value, x_{min} is the minimum value in the data, in the case of this project, this will be 0, and x_{max} is the maximum value in the data, in this project's case, this would be 255.

The min-max scaler is a simple way to scale the data, but it is not robust to outliers. If there are outliers in the data, the min-max scaler will scale the data to be in the range of 0-1, but the outliers will be scaled very close to 0 or 1. While other data will be clustered together, this can be a problem if the outliers are essential to the model. The min-max scaler is also sensitive to the presence of zeros in the data. If there are zeros in the data, the zeros will remain zero, and this can be a problem if the zeros are essential to the model, however for this project, this will not be a concern, as this does not affect the model [13].

Variance scaling Variance scaling is a more robust way to scale the data, but it is more complex. The variance scaling is a method that transforms the data to have a mean of 0 and a variance of 1, by subtracting the mean and dividing by the standard deviation. The formula for the variance scaling is:

$$x_{scaled} = \frac{x - mean(x)}{\sqrt{var(x)}} \tag{3.2}$$

Where x is the original value, x_{scaled} is the scaled value, $mean(x)$ is the mean of the data, and $var(x)$ is the variance of the data [13].

Data augmentation

This section will describe data augmentation, its relevance, and which specific augmentation will be used.

"Data augmentation in data analysis are techniques used to increase the amount of data by adding slightly modified copies of already existing data or newly created synthetic from existing data. It acts as a regularizer and helps reduce overfitting when training a machine learning model." [34]

Data augmentation has many reasons, typically to make a model more general and less overfitted. Overfitting happens because there is only a limited amount of data, often from multiple sources in the real world. So augmentation is used to make a model less reliant on these less-than-ideal data sources [35]. In the case of this project, augmentation is chosen, so it is possible to show how the FE and the model handle different data shapes while still having a baseline. Augmentation helps make more general inferences and intuition about FE and ML. It will, in particular, show which FE methods best handle different data shapes. It will also help show how general augmentation can affect performance ML models and FE. There was chosen augmentations from the following list: Rotation, removing pixels, scaling. These were selected because they are all well-suited for images since they create variations of images [36]. There exist others, but they were not considered for this project.

All the chosen preprocesses for the project are now described. The following section concerns the next point in FE dimensionality reduction.

3.4.4 Choice of Preprocessing

Following the theory of preprocessing from 3.4, this section will cover the decisions made for preprocessing the data and how there were taken.

As stated in section 3.4.3, the amount of preprocessing needed varies depending on what is needed for the model. For this project, it was deemed sufficient to reshape the data, normalize it, and augment it. A detailed description of each decision can be found in the following sections.

Reshape

This subsection describes the decisions made regarding reshaping the data loaded from the MNIST dataset. This subsection only describes the decisions for the training data as an example, but similar decisions were also made for the test data.

The data, when loaded in at first, is shaped as a long array of 47.040.000 integers, ranging from 0-255, representing the values of the pixels in all 60.000 images, and then another array of 60.000 integers representing what number a given image is, between 0-9. The images are in a 28x28 matrix, meaning that the first 784 integers represent the first image, and the successive 784 integers represent the second image. These numbers proved challenging, as it was unclear what image was what.

Additionally, the function used from scikit-learn (sklearn) did not accept the data in this format because the function `fit` expects: "X: array-like of shape (n_samples, n_features)" [37], which means that the function expects the data to be in array of samples(images), and an array of features(pixels). Because of the input `fit` expects, it was necessary to reshape the data into a 60.000x784 matrix. From this, there was a clear distinction between each image, and the data could easily be used in the functions given by sklearn.

Normalization

The following section describes the decisions of normalizing the data loaded from the MNIST dataset.

In section 3.4.3, two methods of normalizing data are described: min-max scaler and variance scaler. From these two methods, variance scaler was chosen. As variance scaler, also called StandardScaler, is a commonly used method of normalizing data, it was deemed a good choice. Variance scaler also ensures that it is comparable to prevent bias in the model [38]. Both methods would have been good choices for normalization, variance scaler was chosen, but either would be acceptable for this project.

Data augmentation

The following section describes the decisions made regarding augmenting the data loaded from the MNIST dataset.

As mentioned in section 3.4.3, data augmentation increases the amount of data and makes the model more general. Augmentation creates new data from the existing data by applying different augmentations to the original data and using the new data to train the model. For this project, rotation and pixel removal augmentations were chosen. They were also deemed to represent real-world errors in data. A number could become slightly rotated due to an angle of writing, or a faulty printer could cause missing text parts. These are some of the cases thought of when deciding on the augmentations.

3.5 Dimensionality reduction

In general, dimensionality reduction transforms data with many features (dimensions) into a new representation of the data with fewer features while preserving as much relevant information as possible. Dimensionality reduction is made by finding a transformation of the data that maps the data to some lower dimensional space while keeping the mean distances between the data points [17].

Dimensionality reduction can be divided into two categories: linear and nonlinear methods, described in Section 2.3. This section presents some of the standard methods from both categories.

3.5.1 Linear methods

The linear methods covered are: Principal Component Analysis (PCA), Factor Analysis (FA), Non-negative Matrix Factorization (NMF), and Linear Discriminant Analysis (LDA).

Principal Components Analysis

PCA is a linear method used to reduce a dataset's dimensionality by projecting it onto a lower dimensional subspace, retaining as much of the variance as possible. The best projection is found through the directions of maximum variance in the data based on the covariance matrix and then projecting the data onto those directions. The directions of maximum variance are principal components, and the projection results from the PCA [17].

Factor Analysis

FA is very similar to PCA, and PCA may be called a type of FA. However, FA is based on the assumption that the data is generated by a linear combination of a few latent variables called factors and that the observed variables are a linear combination of the latent variables plus some noise.

The goal of FA is to find the factors that explain the most covariance in the data. Intuitively, related items have stronger mathematical correlations and can thus be grouped with less loss of information. Once the correlations are determined, a rotation is performed to make the factors easier to interpret FA [39].

Non-negative matrix factorization

NMF constructs approximate factorizations of a non-negative data matrix V into two non-negative matrices W and H such that $V \approx WH$. The non-negativity constraints permit the intuition that data may represent parts forming a whole, which can be thought of as a sum of parts [40]. For example, a text document may represent a combination of topics, and each may represent a combination of words.

Linear Discriminant Analysis

LDA is quite similar to PCA since they both try to project data into a hyperplane. Still, instead, LDA tries to maximize class separability, making it easier to contrast classes. LDA finds the hyperplane with the most separation between the classes and keeps the data points with the same class as close together as possible. This is a three-step process. The first step is to determine the separation between different classes (between-class variance) as the distance between the means of each class. Secondly, the distance between the mean and samples of each class (within-class variance) is calculated. The third step is creating a lower dimensional representation that maximizes the between-class variance and minimizes the within-class variance [41].

3.5.2 Non-linear methods

Sometimes high-dimensional data may contain non-linear relationships, which linear methods cannot capture. It has been shown that PCA fails to find significant components in the swiss-roll dataset [18]. In such cases, non-linear methods are used. The non-linear methods covered are kPCA, Multi Dimensional Scaling (MDS), and Isometric Feature Mapping (ISOMAP).

Kernel Principal Component Analysis

kPCA is an extension of PCA, and it projects the data onto a higher dimensional plane, where PCA is performed. The first step is to construct the kernel matrix from the data. The kernel matrix is a matrix of the dot products of the data points, the kernel matrix is used like the covariance matrix in PCA [42]. In the kernel matrix the data is in a higher dimensional space, where the data is linear separable. By using the kernel function, kPCA exposes the kernel trick, which is instead of calculating the dat into a higher dimensional space, it calculates the inner product between the datapoints, which is computationally cheap to calculate instead of calculating every datapoint into a higher dimensional space. The kernel matrix is then centered by subtracting the mean of each row and column. This matrix is also called the Gram matrix. The Gram matrix is then used to find eigenvectors and eigenvalues. By using the kernel trick, kPCA can compute eigenvalue decomposition, as in PCA.The eigenvectors are then used to project the data onto a higher dimensional space, where PCA is performed. The eigenvectors with the highest eigenvalues are the principal components, and the projection is the data projected onto the principal components [42].

Isometric Feature Mapping

ISOMAP is another non-linear method that is a special case of Classical Multi Dimensional Scaling (cMDS). It is assumed that ISOMAP is suited to discover manifolds of arbitrary dimensionality and that it guarantees an approximation of the true structure of the manifold [18].

The first step in ISOMAP is to map the data points in a graph. The graph is constructed by connecting each point to its nearest neighbors. The user sets the number of nearest neighbors. The nearest neighbors are found by using the Euclidean distance. The Euclidean distance is the linear distance between two points in a Euclidean space [43].

In the second step, ISOMAP finds the Geodesic distance between each point [43]. The Geodesic distance is the shortest distance between two points on the surface of a sphere. The Geodesic distance is calculated by finding the shortest path between two points on a graph. The graph is constructed by connecting each point to its nearest neighbors. The shortest path can be found by using the Dijkstra algorithm [44].

The final step in ISOMAP is finding the data's MDS projection. The MDS projection is found by minimizing the stress function. The stress function is the

sum of the squared differences between the distances in the original data and the distances in the projected data [44], so the stress function finds the slightest change in the data. The minimum of this stress function will be the best reproduction of the data in lower dimensional space according to the ISOMAP algorithm.

3.5.3 Choice of dimensionality reduction

Some dimensionality reduction methods were presented in the Section 3.5. Due to the project's limited scope, not all of the dimensionality reduction methods will be chosen in the implementation. The implementation of the methods will be based on scikit-learn's implementation of the methods. The hyperparameters of the respective methods will also be presented.

Linear methods

The reason for choosing PCA is, among others, because it is a popular dimensionality reduction method [17] and because it tries to find a linear embedding that retains as much information as possible from the original data, which is done by choosing a k components. A difference between PCA and FA is that FA, as described in 3.5.1, further assumes that the linear combinations have some noise. The hyperparameters for the PCA method are the number of components and whether the components will be whitened or not. According scikit-learn, whitening "can sometime improve the predictive accuracy of the downstream estimators" [45].

LDA was also chosen because of its ability to project the data by maximizing the separation between classes. Furthermore, the number of dimensions that would be reduced on MNIST would correspond to the number of $dimensions - 1$čitescikit-learn, sklearn-api. Such a property proves helpful because it simplifies the classification task and may also improve the performance of the machine learning model. The fact that the number of dimensions reduced for MNIST will be maximally about the same as the number of classes might provide a good starting point for PCA and LDA to be compared. The comparison can be based on how much information they can retain by reducing the data to nine dimensions. The number of components is the hyperparameters of the LDA method.

Nonlinear methods

kPCA was chosen because of its ability to project the nonlinear data onto a hyperplane where PCA can be used. That fact leads to the possibility of using kPCA with the hyperparameters of PCA, and kernels. Additionally, some kernels will have a kernel coefficient, which can be thought of sensitivity of the kernel. Another reason for choosing kPCA is that it uses PCA, so the differences between linear PCA and nonlinear PCA could be compared.

ISOMAP is another nonlinear dimensionality reduction method that takes another approach to reduce the dimensions of nonlinear data. Instead of using a kernel, ISOMAP constructs a graph of the data and then applies cMDS to the data.

ISOMAP also tries to preserve the distances between the data points as much as possible [17]. Therefore, the hyperparameters used for ISOMAP are the number of components and the number of neighbors used to construct the graph. The group has also worked with other nonlinear methods, such as T-SNE. However, it was ultimately not chosen because it is a method that is mostly used for data visualization [46].

This section has presented the dimensionality reduction methods that will be used in the implementation. The methods are the following: PCA, LDA, ISOMAP, kPCA, where the number of components is the hyperparameters for the methods. Additionally, PCA has the hyperparameter of whether the components will be whitened or not, ISOMAP has the number of neighbors, and kPCA has the kernel and the kernel coefficient.

3.6 Examples of methods

This section will present how the linear and nonlinear methods behave on linear and nonlinear data. It should be noted again that the methods have different purposes. As a quick reminder, PCA tries to maximize the variance in the embedded data (and so does kPCA). In contrast, LDA tries to maximize the separability of classes in the data, and ISOMAP tries to preserve the distances from the high-dimensional data when projecting it onto a lower space. Therefore their results will not be a one-to-one comparison.

3.6.1 Linear data example

As linear data, we will use the Iris dataset, which contains three different species of Iris flowers, with 50 examples each. Each class has four dimensions: the length and width of the sepals and petals [47]. The dataset is a simple and well-known dataset, which has been used in many machine learning papers [47] and should give intuition as to how the methods behave on the dataset.

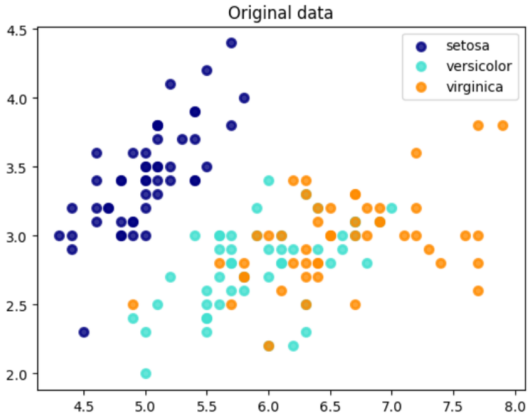

Figure 3.1: Linear data as Iris dataset

Figure 3.1 represents the iris flowers on a 2D plot. According to [47], only one class is separable, which can further be solidified by the fact that the species Setosa is the only visually linearly separable class.

Linear methods

In figure 3.2, one can see that PCA has successfully separated the Setosa species from Versicolor and Virginica in the dataset. The other species are not separated on the coordinates [1.20, -0.20]. Having only Setosa separated from Versicolor and Virginica in the dataset is not concerning, as the test on the Iris dataset is to separate Setosa from the other classes by linear separability.

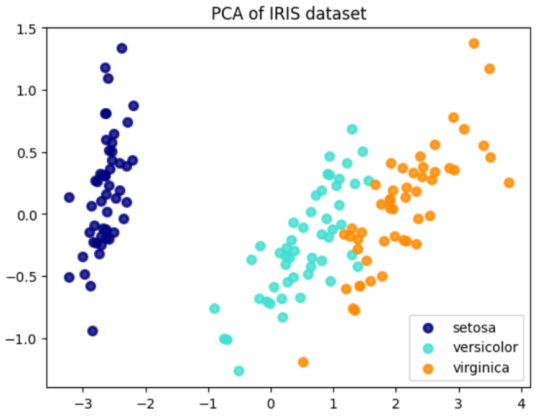

Figure 3.2: PCA on iris

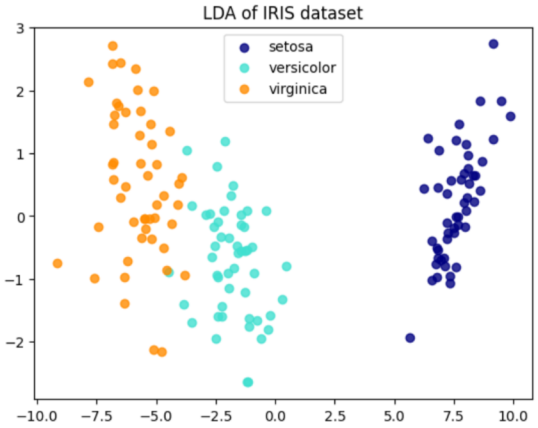

Figure 3.3: LDA on iris

In Figure 3.3, LDA has also successfully separated the Setosa species from the other classes in the dataset. In contrast with PCA, LDA has managed to separate somewhat better as the distinction between those classes can be made more accessible. An advantage that LDA poses is that LDA is a supervised method, which means

that LDA knows the labels of the data and may be better suited for classifying the data.

Nonlinear methods

In Figure 3.4, ISOMAP has tried to separate the Setosa and map the data on the first two dimensions. ISOMAP, as opposed to the aforementioned linear methods, has found little diversity on the second projection, as the values range from approximative 0.0 to -0.25. Such a range is short in comparison with the linear methods. ISOMAP clusters the Setosa species, but that may not be helpful if the data needs to be projected on two dimensions. If, for example, a point is in the upper left corner, it is impossible to tell if it is a Setosa or a Versicolor. ISOMAP is nevertheless a nonlinear method, and it is expected that it does not reduce the data as efficiently as the linear methods.

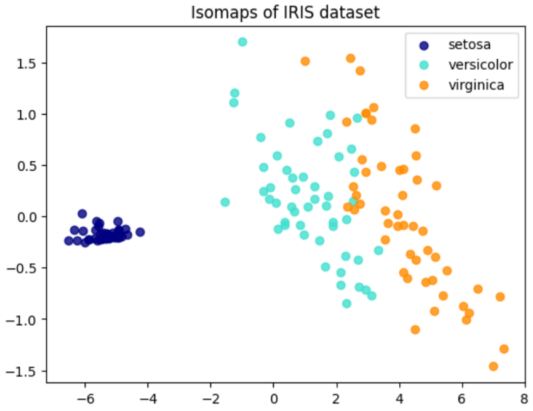

Figure 3.4: Isomap on iris

In figure 3.5, kPCA has tried to separate the Setosa. In the figure, it can be seen that kPCA has clustered the other two species on the first projection, but there cannot be any clear separation between them, especially on the second projection. kPCA has also mapped the data points for Setosa with a high degree of variance, but it did not separate the Setosa from the other two species. Likewise, ISOMAP and kPCA were not supposed to separate the data and the linear methods. However, it is interesting that kPCA has yet to separate the Setosa species from the rest of the data.

3.6.2 Nonlinear data example

As nonlinear data, we will construct two classes of circles, an inner- and an outer circle, which will look like in Figure 3.6.

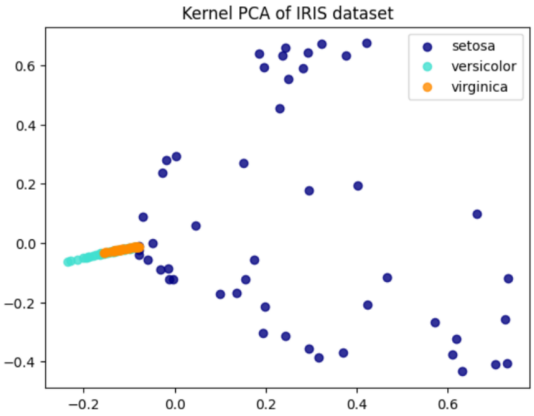

Figure 3.5: KernelPCA on iris

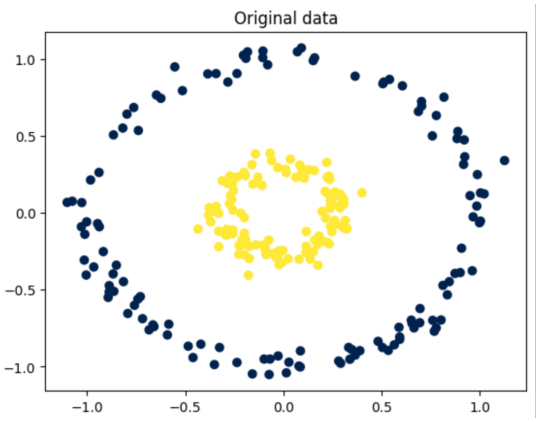

Figure 3.6: Nonlinear data as two circles

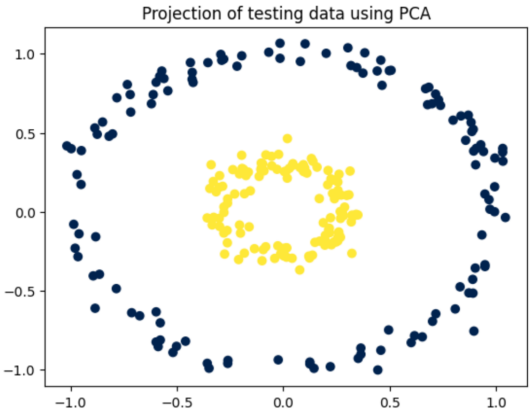

Figure 3.7: PCA on circles

Linear methods

In Figure 3.7, it can be seen that PCA's transformation could have managed to map the nonlinear data. In Figure 3.8, LDA has reduced the data's dimensions to one

dimension, but the two classes are clustered, which means that LDA could not separate the two classes.

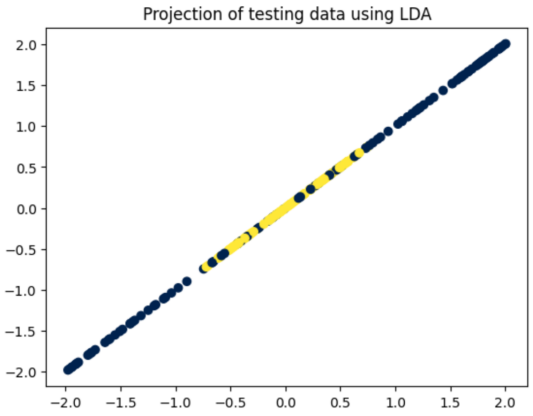

Figure 3.8: LDA on circles

Nonlinear methods

In Figure 3.9, ISOMAP has separated the data points from the circles. Based on the first projection ISOMAP is capable of separating the data. On the second projection, the method can capture more information about the outer circle, thus approximating the original data. The outer circle has a higher variance than the inner circle. The second projection, ISOMAP, revealed little information about the inner circle.

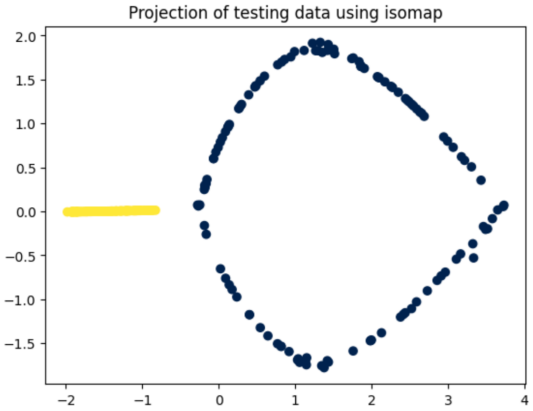

Figure 3.9: Isomap on circles

In Figure 3.10, kPCA has also separated the data points from the circles. kPCA does an excellent job of maximizing the variance for the inner circle and separating the data points from each other. As can be seen, kPCA needs at least two dimensions to better differentiate between the two classes, unlike ISOMAP, which only needs one dimension.

Until now, we have presented some simple examples of the methods. More often than not, the number of reduced dimensions will not be two, as there will be much

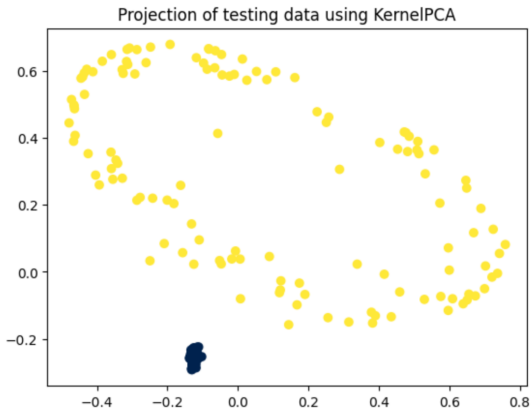

Figure 3.10: KernelPCA on circles

more relevant information in the other dimensions.

3.7 Hyperparameter optimization

This section introduces the concept of hyperparameter optimization and the importance of this process in machine learning.

3.7.1 Hyperparameters

Hyperparameters are the parameters set, for algorithms, before training the model, which does not get learned from the data. The values of hyperparameters can have a significant impact on the performance of the model. How hyperparameters differ from model parameters is that those model parameters get learned from the data during model training [48]. Hyperparameters exist for both FE and for ML models, examples of these could be SVM and PCA. Here PCA should at least have the hyperparameter which corresponds to the amount of dimensions it should reduce down to. SVM would have hyperparameters like which kind of kernel it should use.

3.7.2 Methods of optimization for hyperparameters

Usually, when selecting hyperparameters for a given algorithm, users can resort to default values based on the algorithm's documentation, read literature for recommendations, or try different values and see which one works best. However, this approach could be more efficient, as it is time-consuming and requires a lot of manual work [48].

Instead, the process of finding optimal hyperparameters can be given to the computer [49], given a set of configurations. Hyperparameter optimization is complex because it is unknown which hyperparameters will significantly affect the model, which hyperparameters will interact with each other, and how their interactions

will change the model's performance. According to Marc Claesen [50], the number of hyperparameters that have a significant impact may be small, but that does not mean that the number of meaningful combinations may be small too.

Many different techniques can get used to optimize the hyperparameters automatically [49]. An example of such a technique is grid search. This technique allows the user to define "a set of finite values for each hyperparameter, and grid search evaluates the Cartesian product of these sets" [49]. Another example of a method is random search, a technique similar to grid search. However, instead of evaluating all the combinations of hyperparameters, it randomly evaluates hyperparameters, given a limited amount of time. Both methods have limitations; grid search is inefficient when the number of hyperparameters is significant due to the curse of dimensionality, which means that for algorithms that require large amounts of hyperparameters, it is not feasible to use this technique [51]. Random search can prove helpful, but there is no guarantee that it will find the optimal hyperparameters, given its limited time to evaluate them. Knowing the optimal time limit for random search is tricky, as it depends on the number of hyperparameters, the number of possible values for each hyperparameter, and the number of times the hyperparameters are evaluated [51].

From this, it can be concluded that no single method is optimal for all cases. The optimal method depends on many variables, such as the number of hyperparameters and possible values for each hyperparameter. Therefore, it is essential to evaluate the different methods and find the one that works best for the given problem. Of course, many other techniques could be discussed. Nevertheless, these two methods should sufficiently demonstrate the strengths and weaknesses of different techniques.

3.7.3 Choice of hyper parameters

This section will describe why hyperparameter tuning is part of the pipeline.

Choosing parameters is necessary when working with models and with FE. In this pipeline, in particular, there are many parameters to tune because it has both model and FE. Tuning the parameters would take very long due to not only having to pick different model parameters and try each of these configurations with different FE, which increases the time exponentially. The pipeline implements hyperparameter tuning, specifically using grid search To solve this. This hyperparameter tuning using grid search makes the tuning process more effective and avoids missing exemplary configurations due to human error.

3.8 Cross-validation

This section will describe cross-validation, why it is used in general and why it is used in this project.

A common problem in machine learning is that the test set is supposed to be used only in the final evaluation. This problem makes it hard to, for example, hyperparameter tune or chooses the correct model for the data. On the other hand,

if the test data is used to tune or choose a model, this often leads to overfitting [52]. Overfitting can be described as when the model is very good at predicting the test data but will not do well with new data, not from the test set. This section will introduce one technique that will reduce the chances of overfitting the model: cross-validation. Cross-validation is a technique that splits the data into three sets: training data, validation data(comprising of training data), and test data. By partitioning the training data into two sets, the model can learn from the training data and evaluate on the validation data. The model can be tuned with the validation data and evaluated on the test data after the model is optimized. Such an approach is practical because the model gets evaluated based on data it has never seen, which shows how good the model is at predicting data it has not yet seen [53]. Cross-validation is particularly useful with hyperparameter tuning because it allows picking the correct parameters without using the test set.

Cross-validation, more practically can be described as "The basic form of cross-validation is the basic form of k-fold cross-validation. (. . .) In k-fold cross-validation, the data is first partitioned into k equally (or nearly equally) sized segments or folds. Subsequently, k iterations of training and validation are performed such that within each iteration a different fold of the data is held-out for validation while the remaining $k - 1$ folds are used for learning." [54]

There are also exists other forms of cross-validation, which are special cases of k-fold. As described before, shuffling the data might be necessary, and there is a version of k-fold called Stratified k-fold, where the data gets reshuffled for each round. Alternatively, instead of shuffling the data, one can use the leave-one-out cross-validation, where for each fold, all of the data except one sample is used for training [54]. Stratifying or shuffling the data may be necessary so that each fold has a balanced distribution of classes. There is a risk that the folds contain imbalanced samples for each class, which may affect the overall model performance because the model is biased towards the majority class [54].

In order to avoid overfitting, one can use k-fold cross-validation or derivatives of it, but k-fold implies that a specific number must be chosen. The choice of k could be made through trial and error. However, the train and data samples need to be large enough to be statistically representative of the data set [54].

3.8.1 Choice of cross-validation

This section describes why cross-validation was chosen in this project. Cross-validation is a part of the pipeline and the project in general because it now only allows hyperparameter tuning without the test set but also because it allows for exploration of data while keeping the test set clean [52].

3.9 Evaluation and metrics

A pipeline is generally evaluated based on quantitative metrics in machine learning and data science. These metrics are in classification general based on the confusion

matrix. An example of a confusion matrix can be seen in Figure 3.11.

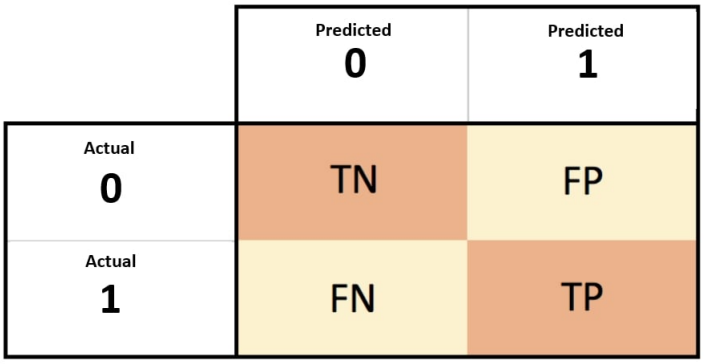

Figure 3.11: Confusion matrix [55]

Figure 3.11 has the values: `TN`, `FP`, `FN`, and `TP`, which stand for True Negative, False Positive, False Negative, and True Positive, respectively. A confusion matrix represents all an ML model's guesses, with the number of guesses placed at the corresponding label where the correctly predicted elements are on the diagonal, in this case `TN` and `TP` [52].

3.9.1 Metrics

There are many different metrics, but there are generally four basic ones. These are accuracy, precision, recall, and F1. Outside these, there are more esoteric metrics such as the Mattheus correlation coefficient, Cohen's kappa, and more [56]. The following section will describe each of the four basic metrics.

Accuracy The most straightforward and simple of the metrics is accuracy, as it simply describes the percentage of correct guesses out of all guesses. Accuracy works well in cases where the sizes of the different classes are similar but struggles in cases where one class is much larger than another. In a case where one class is much larger than others, it is possible to achieve very high accuracy by only guessing the larger class, as the larger classes will have a higher weight when compared to the smaller classes [56]. The formula for accuracy can be seen in 3.9.1.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.3}$$

Precision In the case where one class is smaller when compared to the other classes, the precision metric is better suited than accuracy. Precision is an excellent metric to use when the cost of false positives is high, as it measures how many of the positive guesses were positive. Precision can also measure how well the model

can be trusted when predicting a class[56]. The formula for precision can be seen in 3.9.1.

$$Precision = \frac{TP}{TP + FP} \tag{3.4}$$

Recall The third metric is recall, which measures the model's ability to find all the positive samples. In the case of this project, positive samples refer to correctly guessing numbers from the MNIST dataset. [56]. The formula for recall can be seen in 3.9.1.

$$Recall = \frac{TP}{TP + FN} \tag{3.5}$$

F1 The last basic metric is F1. F1 is commonly used when both recall and precision are essential; F1 can be considered a weighted average. The formula for F1, is shown in [56]. The formula for f1 can be seen in 3.9.1.

$$F1 - Score = \frac{2}{precision^{-1} + recall^{-1}} = 2 \cdot (\frac{precision \cdot recall}{precision + recall}) \tag{3.6}$$

These metrics generally get used in a pipeline for hyperparameter tuning and general evaluation of the model's performance. For this project, hyperparameter tuning will generally only use a single metric to focus on to maximize it. In evaluation, the metrics generally explain what the model does well and does not do well [52].

3.9.2 Choices of evaluation metrics

This section covers the decisions made regarding the evaluation metrics of the results.

As mentioned in subsection 3.9.1, many different metrics can be used to evaluate a model. The basic metrics used in this project were accuracy, precision, recall, and F1. These metrics were used as they each have their strengths and weaknesses. Some of these strengths and weaknesses of the metrics were described in subsection 3.9.1. Another metric used to measure the performance of the models was the time it took to train them. Depending on the circumstances, the time it takes to train a model could be more important than the accuracy of the model, depending on the loss of accuracy. Such as, if a model takes 10 minutes to train but only loses 1% accuracy, it might be worth using that model compared to another model that takes a few hours.

Based on the reasons stated, the metrics used in this project were chosen: accuracy, precision, recall, F1, and time. These metrics were used in the pipeline to evaluate the results and performance of the models.

Chapter 4

Methodology

Based on the problem statement in chapter 2, the following methodology is used to solve the problem.

4.1 Data collection

The data used is the MNIST database [20], which is a collection of handwritten digits. The data is split into a training set of 60,000 images and a test set of 10,000 images. The images are 28x28 pixels grayscale, and each pixel is the intensity of the pixel represented by a value between 0 and 255, where 0 is black and 255 is white. The images are labeled with the digit they represent creating 10 classes, and the labels are integers between 0 and 9.

4.1.1 Data pre-preprocessing

The data is normalized by dividing each pixel value by 255, so that the values are between 0 and 1. The data is then augmented by rotating the images by 90, 180 and 270 degrees, and flipping the images horizontally and vertically. This results in 10 times as much data as the original MNIST dataset.

Include all the augmentations we actually use.

4.1.2 Data preprocessing

The data is then preprocessed by applying linear and non-linear dimensionality reduction techniques. The linear dimensionality reduction techniques are PCA and LDA. The non-linear dimensionality reduction techniques are kPCA and ISOMAP. The data is reduced to 2 dimensions, so that it can be visualized.

4.2 Model training

The SVM and Convolutional Neural Network (CNN) models are then trained on both the original and the preprocessed data. The SVM model is trained using the sklearn library, and the CNN model is trained using the Keras library.

4.2.1 Evaluation

The models are evaluated using k-fold cross validation, and the results are averaged. The evaluation metrics are accuracy, precision, recall, f1-score, speed/run time, and memory usage. The confusion matrix covers the first of these. The speed/run time is measured by timing the training of the models, and the memory usage is measured by using the psutil library.

Lastly the models are evaluated on the test set and the results are compared to the results from the cross validation.

Do we want to compare our results to the results of other papers?

4.3 Pipeline

An automation pipeline is created based on the model in Figure 4.1.

dataset mnist

pre-preprocessing normalization, data augmentation

preprocessing linear and non linear dimensionality reduction (PCA, LDA, Kernel PCA, and ISOMAP)

models SVM and CNN

evaluation accuracy, precision, recall, f1-score, speed/run time, and memory usage

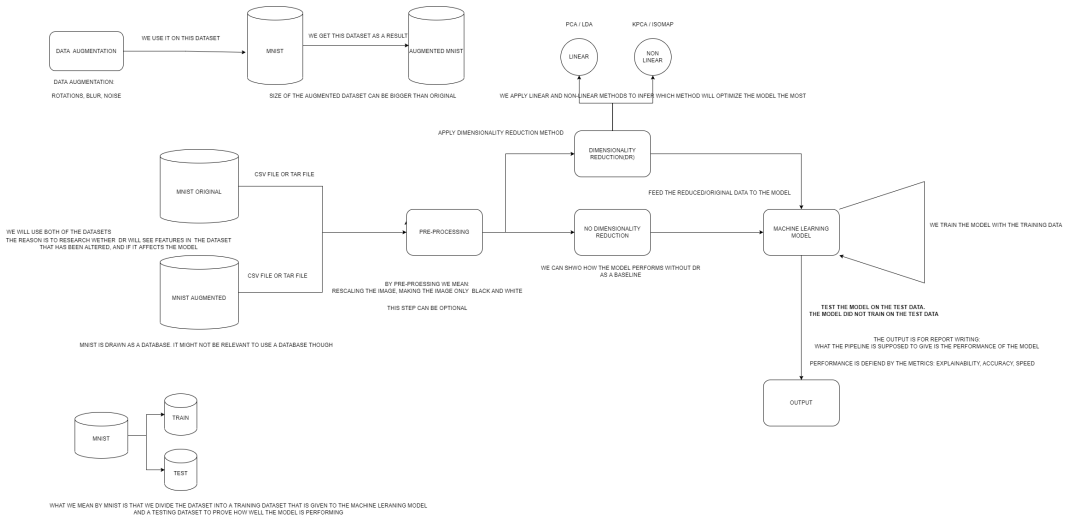

Figure 4.1: Python project pipeline model.

write about the pipeline, and update the pipeline to include the cross validation.

4.4 Metrics

This section will describe our choices in metric what they are and why they were chosen. In this project the differens dimmensionality reduction methods are eval-

uated based on how much it improves the metrics of our models. The following paragraphs will go through these chosen metrics. After that the reason for the choices will be explained.

Model metrics

To be able to to make decisions and show understanding of the different kinds of feature engineering we must have some sort of system to evaluate effectiveness of the feature engineering. In our project we have chosen some metrics to evaluate the future engineering through our model. There exists some very common metrics for general model evaluation we will in this project also look at some of these common metrics. These common metrics are for example accuracy precision recall F1 score. We also looked at some less common once like Mattheus correlation coefficient and Cohen's kappa. All of these metrics are essentially builds upon the confusion matrix. The confusion matrix is a 4 by 4 matrix which describes the possibilities of a binary classifier. Hearing each part of the confusion matrix exists true positives false positives true negatives false negatives. Hear things like accuracy are quite simple operations made on this computer matrix where for example accuracy is the amount of true positives devidedthe the amount of guesses. Where are things like Cohen's kappa are more advanced operations on this confusion matrix [56].

4.4.1 The chosen metric

There are 4 metrics which will be used to evaluate the FE through the model. These metrics are acurracy, precision, recall, F1-score. And there is 2 metrics used to evaluate FE directly. Here the speed/computations is used to evaluate the effeciency. The other metric used is how well the dimmensionality reduction clusters the data, this will be calculated for each class as the average distance between class members divided by the average distance between all data points.

4.4.2 Reason for choice of metrics

The four model metrics were chosen because they the metric which generally describes how good the model does in infering the data. These are all essentially just variations on how many does the model get correct devided by diffrent parts of the confusion matrix. The last 2 metrics was chosen to show the tradeoff most models will have in the more time it takes the better results. This makes it possible to show which metric will actually be most useful in most cases since accurracy is often important only to a certain point. The second FE metric also takes into acount how well the data is clustered after aplying the FE which gives insight into how well the FE works in a vacuum and how much performance a actual model whill achiewe from data which has more defined clusters.

Chapter 5

Results

Describe the results of the project.

Chapter 6

Discussion

Discuss the results from chapter 5 and compare them to the problem statement in section 2.4. Also, discuss the methodology and the theoretical background in chapter 4. Finally, discuss the project as a whole and the process of the project.

What went well? What could have been done better? What would we do differently next time? Perhaps include thoughts on UN sustainability goals.

Chapter 7

Conclusion

Based on the discussion in chapter 6, the results from chapter 5 and the problem statement in section 2.4, the following conclusions can be drawn:

This chapter contains the concluding remarks of the project. It is based on the discussion in chapter 6, the results from chapter 5 and the problem statement in section 2.4. The chapter concludes with a reflection and perspectives for future work.

Acronyms

AI Artificial Intelligence. 3

cMDS Classical Multi Dimensional Scaling. 19, 20

CNN Convolutional Neural Network. 31

FA Factor Analysis. 18, 20

FE Feature Engineering. 4, 9, 16, 26, 27

ISOMAP Isometric Feature Mapping. 19–21, 23, 25, 31

kPCA Kernel Principal Component Analysis. 12, 19–21, 23, 25, 31

LDA Linear Discriminant Analysis. 18, 20–25, 31

MDS Multi Dimensional Scaling. 19

ML Machine Learning. 3–6, 9, 12–16, 26, 29

MNIST Modified National Institute of Standards and Technology. 9, 10, 12, 13, 16, 17, 20, 30, 31

NMF Non-negative Matrix Factorization. 18

NN Neural Network. 1, 13

OvA One-versus-All. 13, 14

OvO One-versus-One. 13, 14

PCA Principal Component Analysis. 18–22, 24, 26, 31

sklearn scikit-learn. 17, 31

SVM Support Vector Machine. 12–14, 26, 31

Bibliography

[1] Daniel Runge Petersen. *AAU-Dat templates*. URL: https://github.com/AAU-Dat/templates (visited on 08/17/2022).

[2] *Machine Learning*. IBM. URL: https://www.ibm.com/cloud/learn/machine-learning (visited on 09/27/2022).

[3] *What is computer vision?* IBM. URL: https://www.ibm.com/topics/computer-vision (visited on 09/27/2022).

[4] *Theory-driven Data Analysis and Modeling*. Aalborg University. URL: https://moduler.aau.dk/course/2022-2023/DSNDATB521 (visited on 09/27/2022).

[5] *Domo Releases 10th Annual "Data Never Sleeps" Infographic*. Sept. 2022. URL: https://www.proquest.com/wire-feeds/domo-releases-10th-annual-data-never-sleeps/docview/2716042192/se-2%7D.

[6] Alon Halevy, Peter Norvig, and Fernando Pereira. "The Unreasonable Effectiveness of Data". In: (2009). DOI: 10.1109/MIS.2009.36.

[7] Etham Alpaydin. *Introduction to Machine Learning. Fourth*. 2020.

[8] Richard Ernest Bellman. "Rand Corporation (1957)". In: *Dynamic programming* ().

[9] John A. Lee and Michel Verleysen. *Characteristics of an Analysis Method*. Ed. by John A. Lee and Michel Verleysen. New York, NY: Springer New York, 2007, pp. 17–45. ISBN: 978-0-387-39351-3. DOI: 10.1007/978-0-387-39351-3_2. URL: https://doi.org/10.1007/978-0-387-39351-3_2.

[10] Altexsoft. *Machine Learning Pipeline: Architecture of ML Platform in Production*. URL: https://www.altexsoft.com/blog/machine-learning-pipeline/ (visited on 11/23/2022).

[11] Zhun Cheng and Zhixiong Lu. "A Novel Efficient Feature Dimensionality Reduction Method and Its Application in Engineering". In: *Complexity* (2018). DOI: 10.1155/2018/2879640.

[12] G. Thippa Reddy, M. Praveen Kumar Reddy, Kuruva Lakshmanna, Rajesh Kaluri, Dharmendra Singh Rajput, Gautam Srivastava, and Thar Baker. "Analysis of Dimensionality Reduction Techniques on Big Data". In: *IEEE Access* 8 (2020), pp. 54776–54788. DOI: 10.1109/ACCESS.2020.2980942.

[13] Alice Zheng and Amanda Casari. *Feature Engineering for Machine Learning: Principles and Techniques for Data Scientists*. 2018. ISBN: 1491953241.

[14] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. *Application of dimensionality reduction in recommender system-a case study*. Tech. rep. Minnesota Univ Minneapolis Dept of Computer Science, 2000.

[15] Dan Margalit, Joseph Rabinoff, and L Rolen. "Interactive linear algebra". In: (2017).

[16] Mordecai Avriel. *Nonlinear programming: analysis and methods*. 2003.

[17] Laurens van der Maaten, Eric Postma, and H. Herik. "Dimensionality Reduction: A Comparative Review". In: *Journal of Machine Learning Research - JMLR* 10 (Jan. 2007).

[18] John C. Langford Joshua B. Tennenbaum Vin de Silva. *A Global Geometric Framework for Nonlinear Dimensionality Reduction*. URL: https://wearables.cc.gatech.edu/paper_of_week/isomap.pdf (visited on 10/11/2022).

[19] *digital imagery questions*. 2018. URL: https://northcoastphoto.com/digital-explained/.

[20] *MNIST database*. Yann LeCun. URL: http://yann.lecun.com/exdb/mnist/ (visited on 10/04/2022).

[21] Matteo Kimura. *Introductory Datasets*. URL: https://lamfo-unb.github.io/2019/05/17/Introductory-Datasets/ (visited on 11/15/2022).

[22] Uniqtech. *Famous Machine Learning Datasets You Need to Know*. 2019. URL: https://medium.com/data-science-bootcamp/famous-machine-learning-datasets-you-need-to-know-dd031bf74dd.

[23] DataRobot. *The importance of machine learning data*. URL: https://www.datarobot.com/blog/the-importance-of-machine-learning-data/ (visited on 11/23/2022).

[24] Zhang Shichao, Zhang Chengqi, and Yang Qiang. "Data preparation for data mining". In: (2003). DOI: 10.1080/713827180. URL: https://doi.org/10.1080/713827180.

[25] G. Thippa Reddy, M. Praveen Kumar Reddy, Kuruva Lakshmanna, Rajesh Kaluri, Dharmendra Singh Rajput, Gautam Srivastava, and Thar Baker. "Analysis of Dimensionality Reduction Techniques on Big Data". In: *IEEE Access* (2020). DOI: 10.1109/ACCESS.2020.2980942.

[26] Bichitrananda Behera, G. Kumaravelan, and Prem Kumar B. "Performance Evaluation of Deep Learning Algorithms in Biomedical Document Classification". In: *2019 11th International Conference on Advanced Computing (ICoAC)*. 2019. DOI: 10.1109/ICoAC48765.2019.246843.

[27] Anyscale. *What is hyperparameter tuning?* URL: https://www.anyscale.com/blog/what-is-hyperparameter-tuning (visited on 02/08/2022).

[28] Anyscale. *What is hyperparameter tuning?* URL: https://www.anyscale.com/blog/what-is-hyperparameter-tuning (visited on 02/08/2022).

[29] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An Introduction to Statistical Learning: with Applications in R*. 2013. URL: https://hastie.su.domains/ISLR2/ISLRv2_website.pdf.

[30] Sanghyeon An, Minjun Lee, Sanglee Park, Heerin Yang, and Jungmin So. *An Ensemble of Simple Convolutional Neural Network Models for MNIST Digit Recognition*. 2020. DOI: 10.48550/ARXIV.2008.10400. URL: https://arxiv.org/abs/2008.10400.

[31] Dan Ciregan, Ueli Meier, and Jürgen Schmidhuber. "Multi-column deep neural networks for image classification". In: 2012. DOI: 10.1109/CVPR.2012.6248110.

[32] Sebastian Schlag, Matthias Schmitt, and Christian Schulz. *Faster Support Vector Machines*. 2018. DOI: 10.48550/ARXIV.1808.06394. URL: https://arxiv.org/abs/1808.06394.

[33] Leonardo Moreira, Christofer Dantas, Leonardo Oliveira, Jorge Soares, and Eduardo Ogasawara. "On Evaluating Data Preprocessing Methods for Machine Learning Models for Flight Delays". In: 2018. DOI: 10.1109/IJCNN.2018.8489294.

[34] Connor Shorten and Taghi M. Khoshgoftaar. "A survey on Image Data Augmentation for Deep Learning". In: (2019).

[35] Kiran Maharana, Surajit Mondal, and Bhushankumar Nemade. "A review: Data pre-processing and data augmentation techniques". In: *Global Transitions Proceedings* (2022). DOI: https://doi.org/10.1016/j.gltp.2022.04.020. URL: https://www.sciencedirect.com/science/article/pii/S2666285X22000565.

[36] Rafael C. Gonzalez and Richard E. Woods. *Digital image processing*. 2008. ISBN: 9780131687288 013168728X 9780135052679 013505267X.

[37] *sklearn.decomposition.PCA*. Nov. 2022. URL: https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html#sklearn.decomposition.PCA.fit.

[38] TowardsAi. *How, When, and Why Should You Normalize / Standardize / Rescale Your Data?* URL: https://towardsai.net/p/data-science/how-when-and-why-should-you-normalize-standardize-rescale-your-data-3f083def38ff (visited on 11/30/2022).

[39] Jamie DeCoster. *Overview of factor analysis*. 1998.

[40] Daniel D Lee and H Sebastian Seung. "Learning the parts of objects by non-negative matrix factorization". In: *Nature* 401.6755 (1999), pp. 788–791.

[41] Alaa Tharwat, Tarek Gaber, Abdelhameed Ibrahim, and Aboul Ella Hassanien. "Linear discriminant analysis: A detailed tutorial". In: *AI communications* 30.2 (2017), pp. 169–190.

[42] Quan Wang. *Kernel Principal Component Analysis and its Applications in Face Recognition and Active Shape Models*. 2012. DOI: 10.48550/ARXIV.1207.3538. URL: https://arxiv.org/abs/1207.3538.

[43] Benyamin Ghojogh, Ali Ghodsi, Fakhri Karray, and Mark Crowley. *Multidimensional Scaling, Sammon Mapping, and Isomap: Tutorial and Survey*. 2020. URL: https://arxiv.org/abs/2009.08136.

[44] Jan de Leeuw. "Multidimensional Scaling". In: (2000).

[45] Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake VanderPlas, Arnaud Joly, Brian Holt, and Gaël Varoquaux. "API design for machine learning software: experiences from the scikit-learn project". In: (2013), pp. 108–122.

[46] Laurens Van der Maaten and Geoffrey Hinton. "Visualizing data using t-SNE." In: *Journal of machine learning research* 9.11 (2008).

[47] Dheeru Dua and Casey Graff. *UCI Machine Learning Repository*. 2017. URL: http://archive.ics.uci.edu/ml.

[48] Philipp Probst, Anne-Laure Boulesteix, and Bernd Bischl. "Tunability: Importance of hyperparameters of machine learning algorithms". In: *The Journal of Machine Learning Research* (2019).

[49] Matthias Feurer and Frank Hutter. *Automated Machine Learning: Methods, Systems, Challenges*. 2019. ISBN: 978-3-030-05318-5. URL: https://doi.org/10.1007/978-3-030-05318-5_1.

[50] Marc Claesen and Bart De Moor. "Hyperparameter Search in Machine Learning". In: (2015). URL: http://arxiv.org/abs/1502.02127.

[51] Li Yang and Abdallah Shami. "On hyperparameter optimization of machine learning algorithms: Theory and practice". In: *Neurocomputing* (2020).

[52] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An Introduction to Statistical Learning: with Applications in R*. Springer, 2013. URL: https://faculty.marshall.usc.edu/gareth-james/ISL/.

[53] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[54] Payam Refaeilzadeh, Lei Tang, and Huan Liu. *Cross-Validation*. Ed. by LING LIU and M. TAMER ÖZSU. Boston, MA: Springer US, 2009, pp. 532–538. URL: https://doi.org/10.1007/978-0-387-39940-9_565.

[55] *Confusion Matrix*. 2022. URL: https://subscription.packtpub.com/book/data/9781838555078/6/ch06lvl1sec34/confusion-matrix.

[56] Margherita Grandini, Enrico Bagli, and Giorgio Visani. *Metrics for Multi-Class Classification: an Overview*. 2020. DOI: 10.48550/ARXIV.2008.05756. URL: https://arxiv.org/abs/2008.05756.