UNIVERSITY OF WEST ATTICA

&

UNIVERSITY OF LIMOGES

Master's Thesis

Forgery detection using machine learning and image processing

Student: Alexandros Karystinos

(aivc21007)

Supervisor: Anastasios L. Kesidis, Professor

Athens, 2024

Forgery detection using machine learning and image procesing

Μέλη εξεταστικής επιτροπής συμπεριλαμβανομένου και του Εισηγητή

Η διπλωματική εργασία εξετάστηκε επιτυχώς από την κάτωθι Εξεταστική Επιτροπή

Α/α	ΟΝΟΜΑ ΕΠΩΝΥΜΟ	ΒΑΘΜΙΔΑ/ΙΔΙΟΤΗΤΑ	ΨΗΦΙΑΚΗ ΥΠΟΓΡΑΦΗ
1	Αναστάσιος Κεσίδης	Καθηγητής	ANASTASIOS KESIDIS Digitally signed by ANASTASIOS KESIDIS Date: 2024.07.17 13:20:13 +03'00'
2	Πάρις Μαστοροκώστας	Καθηγητής	Paris Mastorokostas Digitally signed by Paris Mastorokostas Date: 2024.07.18 18:08:48 +03'00'
3	Παναγιώτα Τσελέντη	ΕΔΙΠ	Panagiota Tselenti Digitally signed by Panagiota Tselenti Date: 2024.07.17 13:53:24 +03'00'

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΜΕΤΑΠΤΥΧΙΑΚΗΣ ΕΡΓΑΣΙΑΣ

Ο κάτωθι υπογεγραμμένος Καρυστινός Αλέξανδρος του Νικολάου με αριθμό μητρώου aivc21007 φοιτητής του Προγράμματος Μεταπτυχιακών Σπουδών Τεχνητή Νοημοσύνη και Οπτική Υπολογιστική του Τμήματος Μηχανικών Πληροφορικής της Σχολής Μηχανικών του Πανεπιστημίου Δυτικής Αττικής, δηλώνω ότι:

«Είμαι συγγραφέας αυτής της μεταπτυχιακής εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της, είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, οι όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε ακριβώς είτε παραφρασμένες, αναφέρονται στο σύνολό τους, με πλήρη αναφορά στους συγγραφείς, τον εκδοτικό οίκο ή το περιοδικό, συμπεριλαμβανομένων και των πηγών που ενδεχομένως χρησιμοποιήθηκαν από το διαδίκτυο. Επίσης, βεβαιώνω ότι αυτή η εργασία έχει συγγραφεί από μένα αποκλειστικά και αποτελεί προϊόν πνευματικής ιδιοκτησίας τόσο δικής μου, όσο και του Ιδρύματος.

Παράβαση της ανωτέρω ακαδημαϊκής μου ευθύνης αποτελεί ουσιώδη λόγο για την ανάκληση του πτυχίου μου».

Ο Δηλών

Contents

ABSTRACT

In an era where digital transactions and document authentication play a central role, the need for strong and secure methods to verify handwritten signatures is paramount. This thesis explores the intersection of traditional signature analysis and modern machine learning techniques to develop a reliable signature verification system. The survey begins with a comprehensive review of existing signature verification methods, including both conventional image processing techniques and advanced machine learning approaches. Challenges posed by the inherent variability of human signatures are identified, including differences in writing style and the presence of noise. The proposed solution leverages advanced machine learning algorithms, specifically adapted for signature recognition such as SVM, Decision Trees, kNN, and others. As in all cases of machine learning, in this one too, we use datasets. The datasets used for training and evaluation include a diverse collection of handwritten signatures, capturing variations in styles, angles, and levels of complexity. In addition, a key feature of this work is that feature extraction techniques are used to transform the raw signature data into meaningful representations for the machine learning models. For this purpose, four different feature sets of increasing complexity are proposed to improve the overall system performance.

ACKNOWLEDGMENTS

I would like to express my deepest gratitude to professor Anastasios Kesidis for his invaluable guidance, unwavering support, and continuous encouragement throughout the entirety of this thesis. His expertise, patience, and mentorship have been instrumental in shaping my research journey and refining my academic skills. I am also grateful to him for his insightful feedback and constructive criticism, which have significantly enriched the quality of this work. Additionally, I extend my heartfelt appreciation to my family and friends for their unwavering support, understanding, and encouragement during this challenging yet rewarding endeavor. Finally, I would like to acknowledge the University of West Attica for providing me with the resources and opportunities necessary for the completion of this thesis.

1 INTRODUCTION

The most widely used and recognized method of authenticating a person is through a signature, which makes it a target for sophisticated attacks. Forged signature detection and biometric applications rely heavily on signature verification. To identify or verify a person's identity, biometrics measure their distinctive physical or behavioral characteristics. Iris, hand shape, face, and fingerprints are examples of physical features that make up a biometric property. Iris and fingerprints are two of those that don't change over time and have very little variation. However, they require specialized equipment that is very expensive to capture the biometric image. Behavioral features that make up a biometric feature include voice, typing patterns, gait, and signature [1]. Speech and signature technologies are among the most advanced.

One well-known biometric is the handwritten signature. Handwritten signatures have a significant advantage over other authentication technologies because they can only be used when a person is conscious and willing to write, unlike fingerprints which can be captured even when a person is unconscious. Handwritten Signature Verification (HSV) systems are divided into online and offline categories. Offline signature verification requires scanning a document with an existing signature to create a digitized image. Electronic signature verification, also known as online signature verification, uses specialized hardware such as a pressure-sensitive pen or digital tablet to capture both the form and dynamics of the writing [2,3]. We will discuss these types in detail in the next chapter.

To extract dynamic features of a signature in addition to its structure, devices such as the previously mentioned pressure-sensitive tablets are used in electronic signature verification. Dynamic features, that increase the individuality of the signature and the difficulty of forgery, include the number and sequence of strokes, the general rhythm of the signature, the pressure of the pen at each point, and others [4].

Users first register in an electronic signature verification system as shown in Figure 1 by providing sample signatures (reference signatures). This test signature is compared to the reference signatures for that person when a user submits a signature (test signature) claiming to be that person. The user will be banned if the difference exceeds a predetermined level. During verification the test signature is compared to each signature in the reference set, resulting in a series of distance values [5]. To determine how much the test signature differs from the reference

set, a method must be chosen to combine these distance values into a single number and compare to a threshold. The minimum, maximum, or average of all distance values can be used to obtain the single dissimilarity value. A signature verification system often chooses one of these and ignores the others. The variables False Rejection Rate (FRR) of genuine signatures and False Acceptance Rate (FAR) of forged signatures are critical in determining how well a signature verification system works. Because of the inverse relationship between these two errors, Equal Error Rate (EER) where FAR=FRR is often quoted [6].

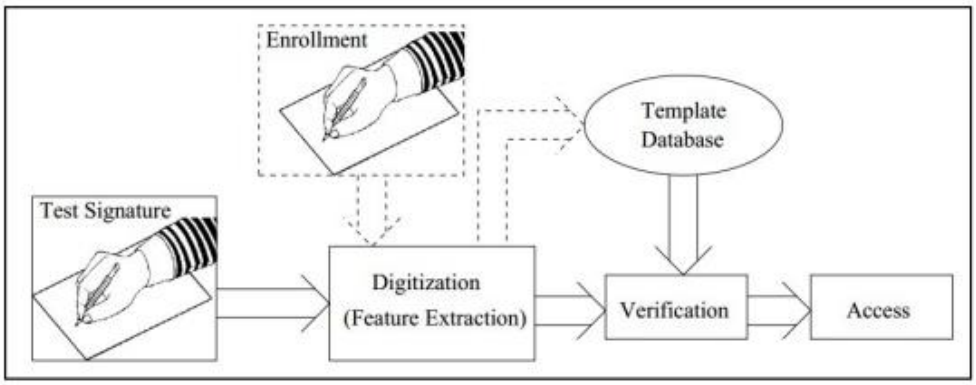


Figure 1: Handwritten signature verification system [48]

In general, the steps for verifying a handwritten signature, such as feature extraction and selection of appropriate features, contribute to the final result, but not by themselves. At this point, machine learning plays an important role. Specifically, the machine learning models that a user trains. These, combined with the classic handwritten signature verification steps, are the ones that help to better understand the problem and its solution. The reason these models are useful is that, through machine learning, allows us to draw conclusions from some metrics such as accuracy and from graphs that are even more familiar to the user. Machine learning is useful in these types of systems in general, and not just for signature verification which is our case. Finally, with the results that machine learning models provide, we can also understand where such a system has "weaknesses" in order to correct them.

During the thesis we used both feature extraction and machine learning for forgery detection. However, apart from these two techniques, we also used some morphological operations. More specifically, regarding the feature extraction, we extracted some specific features that essentially

represent each of our images and help us study them better. Regarding the morphological operations, we used them for our own convenience of the whole process. Regarding the machine learning algorithms we used the following ones:

- Decision Tree
- kNN
- Naïve Bayes
- Support Vector machine (SVM)
- Neural Network
- Ensemble Algorithms
- Naïve Bayes
- Logistic Regression

About the structure of this thesis is as follows: Chapter 1, provides a general introduction to handwritten signatures, focusing on the problem of forgery and its solution. In Chapter 2, we will discuss basic features such as feature extraction, feature selection, and required data preprocessing. In Chapter 3, we will discuss machine learning models, including decision trees and support vector machines, among others and we will mention their advantages and disadvantages. Chapter 4 provides a detailed description of the experimental part, including figures and tables of the results. We will also refer to the three specific datasets used in our experiments. Finally, in Chapter 5, we will present the conclusions drawn from our experiments and discuss the potential applications of this system in the future.

2 HANDWRITTEN SIGNATURE VERIFICATION TECHNIQUES

In this chapter we will discuss about the differences between offline handwritten signatures and online handwritten signatures in more detail, by going into some details about handwritten signature feature extraction, data preprocessing, feature selection, and other characteristics.

2.1 Handwritten signature data preprocessing

Preprocessing is an important step in signature verification where the the signature image is prepared for further analysis and feature extraction. Some common preprocessing steps for signature verification include; [7]:

- Image capture: The signature image is captured using a scanner or camera. The quality of the image significantly affects the accuracy of the verification.
- Image enhancement: The captured signature image may contain noise or artifacts that may affect the resolution of the signature. Image enhancement techniques can be used to improve image quality and reduce the effects of these artifacts. Common techniques include image smoothing, edge detection, and contrast enhancement.
- Image normalization: Signature size and orientation can vary from case to case, making it difficult to compare signatures. Image normalization techniques can be used to scale and rotate the signature image to a standard size and orientation, which can simplify the comparison process.
- Data augmentation: To improve the robustness of the signature verification system, it is often useful to augment the training data by generating additional signature variations. This can include adding noise, rotating the image or resizing the signature.

Apart from these steps, there are several others that are important and they are as follows:

- Noise Removal - Noise is often present in scanned signature images. Applying a denoising filter, such as a median filter [8] to the image is a typical solution to this problem. Morphological techniques are often used to close small gaps and eliminate small patches of related components [8, 9].

- Signature Representation - Other representations have been considered besides simply using the grayscale image as input to the feature extractors. For example signature skeleton, outline, ink distribution, high-pressure-areas and directional boundaries are used [8].
- Signature Alignment - Alignment is a standard approach for online signature verification, while not often used in offline scenarios. Yilmaz [9] proposes applying rotation, scaling and translation to align signatures for training. In their spin normalization approach, Kalera et al. [10] propose to exploit the first and second order moments of the signature image.

In general, the preprocessing steps in signature verification aim to improve the quality and consistency of the signature image, making it easier to extract relevant features that can be used for comparison and classification.

2.2 Feature Extraction

As mentioned earlier, handwritten signature recognition is the process of identifying and verifying a person's signature from an image or set of images. One of the critical steps in signature recognition is feature extraction, which involves identifying relevant features or patterns that can be used to distinguish one signature from another.

Here are some common feature extraction techniques for handwritten signature recognition [11], [12]:

- Grid-based features
- Structural features
- Texture features
- Statistical characteristics
- Fourier transform based features
- Global features
- Local features
- Geometric Features

Grid-based features include dividing the signature image into smaller cells or blocks (which we will also do in our experimental part) and computing statistical metrics such as the mean, standard deviation, and entropy of the pixel values in each cell. These features can capture the spatial distribution of the signature and its overall shape. The size and shape of the grid can vary

depending on the size and complexity of the signature. Grid-based features can be further enhanced by using techniques such as Gray Level Co-occurrence Matrix (GLCM) to capture the spatial relationship between pixels in the signature image.

Structural features capture the shape and structure of the signature by analyzing the outline and shape of the signature. Examples of structural features include the number of vertices, the curvature of the signature and the angle between different parts of the signature. These features can be extracted using techniques such as chain code or Freeman code, which represent the contour of the signature as a sequence of directions.

Texture features capture the fine details and patterns in the signature, such as the texture of pen strokes or the presence of small dots or lines. Texture features can be extracted using techniques such as Gabor filters or local binary patterns (LBP). Gabor filters are spatial frequency filters that can capture signature texture at different scales and orientations. LBP is a binary pattern that captures the local texture of the signature by comparing the intensity values of a pixel with its neighbors.

Statistical features capture the statistical distribution of pixel values in the signature image, such as mean, variance, skewness, and kurtosis. These features can be useful for detecting forged signatures or signature variations caused by different writing styles. Statistical features can be enhanced using techniques such as the wavelet transform, which can capture the statistical distribution of the signature at different scales and resolutions.

Fourier transform-based features analyze the frequency content of the signature by computing the Fourier transform of the signature image. These features can capture the overall shape and frequency components of the signature. Fourier-based features can be further enhanced using techniques such as the Discrete Cosine Transform (DCT), which can capture the frequency content of the signature in a compact form.

The signature is described or represented by global functions. These features, such as length and width, are typically extracted from all pixels in the region surrounding the signature image. However, global features are easy to extract and are less susceptible to noise than individual parts of the signature with small distortions. The global feature vector is not significantly affected by them. However, because they depend on the overall orientation of the page, they are susceptible to distortion and stylistic changes.

Critical intersections and gradients are examples of local features that represent a segment or small region of a signature image. These features are typically derived from the pixel distribution of a signature, such as its local density. Local features are less affected by other regions of the signature and more sensitive to noise in the region of interest. They are much more accurate than generic features, but require more computation.

Geometric features are the features that describe the characteristic geometry of a signature, preserving both its global and local feature properties. They have the ability to tolerate deformation, style variations, rotation variations, and some degree of translation. In conclusion, the choice of feature extraction technique depends on the specific requirements of each application and the characteristics of the signature dataset. A combination of different feature extraction techniques can be used to capture different aspects of signature features and improve recognition performance.

2.3 Feature Selection

Feature selection is the process of selecting a subset of relevant features that are most informative for the classification task, while discarding irrelevant or redundant features that may introduce noise or degrade performance. Some common approaches to feature selection in handwritten signature verification systems include [13];

- Shape-based features: These features describe the shape of the signature, such as the number of strokes, curvature, and aspect ratio. Shape-based features can be extracted using techniques such as Fourier descriptors or chain codes.
- Texture-based features: These features describe the texture or pattern of the signature, such as the distribution of ink or the presence of certain strokes.
- Structural features: These features describe the structural properties of the signature, such as the location and orientation of key points or landmarks. They can also be extracted using techniques such as Hough transforms or Harris angles.
- Hybrid approaches: These approaches combine different types of features to capture different aspects of the signature. For example, a combination of shape-based and texture-based features can provide better performance than either approach alone.

The choice of feature selection method depends on the specific requirements of the signature verification system, as well as the available data and computing resources. It is important to evaluate system performance using different feature selection methods and compare their effectiveness to improve accuracy and reduce complexity.

The goal of feature selection is to minimize the loss of information caused by reducing the feature set, so in theory the selection process should not have a negative impact on classification performance. A search strategy, a subset evaluation, and a stopping criterion are the three basic steps that make up the feature selection process. According to a predefined subset evaluation criterion, a typical search technique uses a search strategy to identify the best solution. The search process is repeated until a stopping condition is met [13].

The feature selection problem requires selecting the subset of features with the highest discriminative power from the entire collection of available features. For several reasons, the selection of a robust subset of features is essential in any classification process. No matter how good the learning algorithm is, the performance achieved may not be sufficient if the considered feature set does not contain all the data needed to classify samples from different classes. On the other hand, the extent of the search space to be explored during the learning phase depends on the size of the feature set used to characterize the samples. Finally, the number of features used to define the patterns affects the computational cost of the classification [13].

2.4 Online handwritten signatures

Electronic handwritten signatures (or online handwritten signatures) are captured using electronic devices such as tablets, touchscreens, or pen-based devices. The signing process involves the signer physically writing their signature on the device using a pen or stylus, which is recorded as a sequence of digital data points. The sequence of data points captures the dynamic properties of the signature, such as stroke order, pen pressure, and pen speed. These dynamic properties make electronic signatures more difficult to forge than offline signatures [14].

The recognition process for electronic signatures involves analyzing the dynamic properties of the signature, which are unique to each signer. The authentication algorithm can compare the signer's signature with a stored template or reference signature to verify their identity. Online signature recognition systems fall into into two categories: static and dynamic. Static systems analyze the shape of the signature and rely on features such as line thickness, curvature, and

texture. Dynamic systems, on the other hand, analyze the temporal properties of the signature, such as velocity, acceleration, and pressure. They tend to be more accurate than static systems because they capture more information about the signature.

Electronic signature recognition systems can be used in a variety of applications, including banking, e-commerce and document management. In banking, electronic signature recognition can be used to verify the identity of customers when they sign checks or other financial documents. In e-commerce, electronic signature recognition can be leveraged to authenticate users during online transactions. In document management, e-signature recognition can be used to verify the authenticity of digital documents. These systems face many challenges, such as variability in writing styles, differences in hardware, and noise in the data. To overcome these challenges, techniques such as feature extraction (discussed above), machine learning algorithms (discussed in the introduction), and finally normalization are used. These three methods combined help to improve the recognition accuracy. A online handwritten signature is shown below:

Figure 2: Online Handwritten Signature [49]

2.5 Offline handwritten signature

Offline handwritten signatures are captured on a piece of paper or other surface using a pen or pencil. The signing process involves the signer physically writing his signature on the surface,

which creates an image of it. The signature image captures its static properties, such as its shape, size, and style. The static properties are less secure than the dynamic properties of electronic signatures, because they are more easily reproduced. The recognition process for offline signatures involves the analysis of these properties, which are less unique to each signer than the dynamic properties of electronic signatures. The authentication algorithm can compare the signer's signature with a stored template or a reference signature to verify its identity [15].

Offline signature recognition systems can be divided into two categories: holistic and partial. Holistic systems analyze the entire signature as a single unit, while partial systems divide the signature into smaller units such as strokes or characters. They can also be used in a variety of applications, including document authentication and forensics. In document authentication, offline signature recognition can be used to verify the authenticity of handwritten signatures on physical documents. In the case of forensics, offline signature recognition can be leveraged to analyze signatures on legal documents or other evidence.

These systems also face many challenges, such as variations in writing styles, differences in paper quality and lighting, and image noise. To overcome these challenges, signature recognition systems use the same techniques as online signature recognition systems. Below is an example of a handwritten offline signature:

Figure 3: Handwritten offline signature [50]

2.6 Differences between online and offline handwritten signature

Here are some more details about the differences between online and offline handwritten signatures [16]:

- Data format: Offline handwritten signatures are usually captured as a 2D image, where the signature is represented as a set of black and white pixels. The signature image can be captured using a scanner or a camera, and the recognition process involves analysing the shape, texture, and statistical properties of the signature image. Electronic handwritten signatures, on the other hand, are captured as a sequence of digital data points, where each point represents the position and pressure of the pen or stylus at a particular time. The sequence of data points can be captured using a tablet, touch screen, or stylus-based device. The recognition process involves analysing the dynamic properties of the signature, such as stroke order, pen pressure, and pen speed.
- Dynamic properties: Offline signatures are static and do not record the dynamic properties of the signature, unlike electronic signatures which do.
- Forgery: Offline signatures are more vulnerable to forgery compared to online signatures. For example, an attacker can easily forge an offline signature by copying or scanning the image. This type of forgery is difficult to detect, as the forged signature looks like the genuine. Electronic signatures are more difficult to forge, as they capture the dynamic properties of the signature, which as mentioned earlier, are unique to each signer.
- Recognition accuracy: Online signatures are generally more accurate than offline signatures in terms of identification performance. They record more information about the signature, including dynamic properties. This additional information helps to improve the accuracy of signature recognition systems. Offline signatures, on the other hand, are more susceptible to noise and distortion, which can reduce recognition accuracy.

2.7 Forgeries

In the context of handwritten signatures there are several types of forgery such as the following [20]:

- Random forgery: It is produced by the forger without knowing the name of the author or the authenticity of the signature.
- Simple forgery: In this category, the forger has no idea what the signature they are forging looks like. This is the easiest type of forgery to detect, because it usually doesn't look anything like a genuine signature. It is also sometimes possible for an examiner to identify who did the forger from the writing style of the forged signature.
- Skilled forgery: In this type of forgery, the forger has a sample of the signature to be forged. The quality of a forgery depends on:
    a) how much the forger practices before attempting the actual forgery
    b) his skill
    c) his attention to detail when simulating the signature.
- A skilled forgery looks more like the genuine signature. The problem of signature verification becomes more difficult as we move from simple to sophisticated forgery.
- Freehand forgery: Freehand forgery involves creating a forged signature without attempting to duplicate the appearance of the original signature. Instead, the forger may create a signature in a different style or format.
- Detected forgery: This forgery involves tracing over the genuine of the signature to create a forgery. Generally this type of forgery is often easier to detect, as the signature may contain unnatural or inconsistent events.
- Cut and paste forgery: Cut and paste involves cutting out a genuine signature from a document and pasting it into a different document or location to create a forged signature.

For the most part, signature forgery is a serious crime that can have serious legal and financial consequences, which is why it's important to take the appropriate steps to both prevent it and detect it if it does occur.

2.8 Handwritten signature verification

The test signature and the reference signature are compared using the minimum, average, and maximum values of the dissimilarity values after applying the feature extraction method. In this comparison, a threshold value is used to compare each reference and test signature. If the value is nearly the same as the reference signal value, the signature is accepted as authentic, while if the difference is greater than this threshold value, the signature is ignored. This threshold may be the same for each signature or may vary from signature to signature [17] [18]. Below are presented two methods to help verify a handwritten signature:

- Common limit: As the common limit has the best limit for all authors, it is more advantageous. After calculating the differences between the data signatures, a common threshold is selected based on the minimum error requirement.
- Author dependent threshold: The author can only use one person in this type of threshold. Compared to ordinary data, the data for this limit should be larger. The writer changes the value after each entry in this limit selection format.

Besides the common threshold and the author-dependent threshold, there are two more methods we can use to verify the handwritten signature, which are as follows [19]:

- Feature-based methods: Feature-based methods analyse the signature image to extract relevant features that can be used to distinguish between genuine and forged signatures. Common features used in signature verification, include: a) stroke direction, b) stroke width, c) curvature and d) texture. These attributes are then compared with a set of reference attributes to determine whether the signature is genuine or not.
- Machine learning methods: Machine learning methods involve training models on a large dataset of signature images and their corresponding labels (genuine or forged). These models can then be used to classify new signature images as genuine or forged based on their similarity to the training data.

Some of the very common machine learning techniques used for signature verification include:

- Support Vector Machine (SVM)
- Random Forest
- Convolutional Neural Networks (CNN)

In the following chapter we will investigate machine learning techniques and their applicability to handwritten signature verification.

3 MACHINE LEARNING APPROACHES

Can a machine be considered to think and learn like a human? was the main issue that led to machine learning. This research was done mostly in response to Alan Touring's work, "Computing Machine and Intelligence", and his study of the possibility of machine thinking [21]. Perceptrons and neural networks were discovered as a result of focused study in machine learning. Machine Learning (ML) is considered as a filed of Artificial Intelligence (AI), flourished on its own in the 1990s and began to progress and develop rapidly.

"A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance on tasks in T, as measured by P, improves with experience E [22]", is Tom Mitchell's statement as a succinct explanation of machine learning. Machine learning is the science of using existing data and methods to train machines to learn on their own. The learning process is often performed using a model that is learned from existing data and used to predict and act in the future. The model is constantly updated, learning as it receives new data. The machine learning process is illustrated very clearly in Figure 4 below.

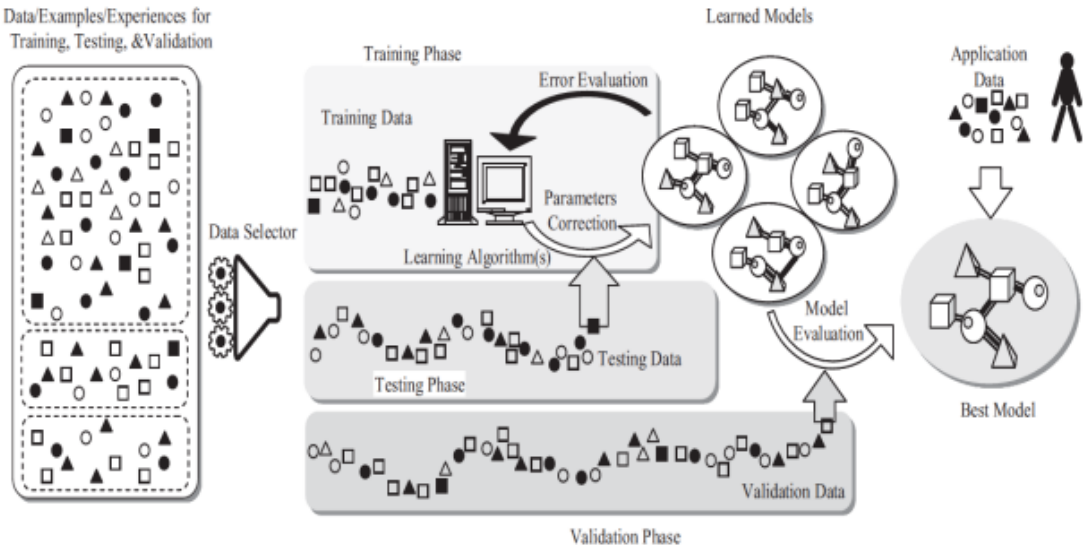


Figure 4: Machine Learning Process [51]

The first step in the process is to take the data to be used for machine learning purposes are taken and put it into the correct format in the first step of the process. This information is then divided into three categories: training, testing, and validation data. It is worth noting that the validation data is used to avoid an event called overfitting. However, the data is usually split into training

and testing halves. The training data is used in the process phase to learn the data and build models. In addition, the models are tested using independent test data to either evaluate or correct them. The best models are selected during the testing phase. Models are evaluated at the validation level if a particular technique, such as neural networks, requires an additional degree of validation. The process continues until a certain level of quality is reached or is stopped if none of the models perform at a sufficient level. Once models are selected, they are able to make predictions, learn from experience, and grow with the system, making them ready for real-world applications.

3.1 Machine Learning types

There are many different machine learning algorithms and each has a unique set of techniques or structures. Nevertheless, algorithms can be distinguished from each other at some level and categorised according to certain characteristics that they all have. As a result, algorithms are divided into four learning categories: a) supervised learning, b) unsupervised learning, c) semi-supervised learning, and d) reinforcement learning.

3.1.1 Supervised learning

In supervised learning, the input features for the data must be labelled, but more importantly, there must be a labelled feature for the desired output value [23]. Each data instance should have a single variable that, given its input values or variables, indicates the desired output value. The output value should be determined by considering the input variables, which should be limited to a manageable and efficient level. It can also be categorical (for requiring classification tasks) or continuous (for regression tasks).

The goal of supervised learning is to build models that represent the training data correctly and in a simple way. The accuracy and recall rate of the models are evaluated before they are selected from alternative models [24]. It could also be evaluated and improved after being applied to real-world problems. There are many supervised learning techniques such as Support Vector Machines (SVM), Random Forests, k-Nearest Neighbor algorithms (kNN), simple Bayes classifiers, artificial neural networks (which we will analyse in the next units). In addition to bioinformatics, database marketing, information retrieval, and pattern recognition fields,

supervised learning algorithms have other applications, such as image, voice, and speech recognition.

3.1.2 Unsupervised learning

Unlike supervised learning, in unsupervised learning the data has no prior output label. Knowing that the data is unlabelled, the primary goal of the algorithms is to allow the data to learn on its own. To arrange the data samples so that those that are related to each other are in the same group, regularities, patterns or any other commonalities between the data samples are examined [24]. Clustering, Principal Component Analysis (PCA), and Expectation-Maximization (EM) are three of the most important unsupervised learning algorithms. The k-means algorithm, hierarchical clustering, and other methodologies are some of the different approaches used by the clustering algorithm to cluster the data. However, the primary goal, is to group data instances so that they are more similar to each other within each cluster than to any other instances belonging to different clusters. In other words, there is high similarity within clusters and little similarity between clusters. In order to minimise the number of variables, or dimensions in the data and to maximise for example learning performance, Principal Component Analysis is performed.

3.1.3 Semi-supervised learning

Semi-supervised learning, as the name suggests, is a class of supervised learning algorithms and tasks that also use unsupervised learning or unlabeled data. The majority of data used in semi-supervised learning is unlabeled, with a small amount of labeled data. The combination of the two learning strategies is mainly done to improve the overall learning accuracy. Under certain conditions, it has been shown to be superior to other supervised approaches [24]. Semi-supervised has one drawback and it is this: the labeled data must be produced by highly skilled people, increasing the cost of the whole process.

Inductive learning is another name for semi-supervised learning. By combining the advantages of supervised and unsupervised learning algorithms, a semi-supervised method is created. Self-training, mixed models, co-training, multi-projection learning, graph-based techniques, and semi-supervised support vector machines are some examples of semi-supervised techniques.

3.1.4 Reinforcement Learning

Powerful machine learning systems are produced using the Reinforcement Learning (RL) method in artificial intelligence, which integrates the fields of dynamic programming and supervised learning [24]. A decision maker, hypothetically a robot, is assigned a goal and works to achieve it by learning on its own and interacting with its environment. As a result, certain requirements must be met for a fundamental RL model, and these are:

- ✓ A set of environmental situations.
- ✓ A set of actions.
- ✓ A set of rules that determine the rewards given at the end of transitions.
- ✓ A set of rules that describe what the robot observes.

3.2 Machine Learning Algorithms

3.2.1 Decision Trees

One of the most popular learning techniques is the Decision Tree algorithm, a classification method that focuses on an understandable form of representation. Decision Trees use datasets consisting of feature vectors, which in turn include a collection of classification features characterize the vector and a class feature that identifies the data as belonging to a particular class. A decision tree is constructed by iteratively dividing the data set along the feature that best categorizes the data into the various existing classes until a predefined stopping criterion is met. Since decision trees can be easily displayed in a structured tree style that is simple for humans to understand, the representation form helps users get a quick overview of the data.

The Iterative Dichotomizer 3 (ID3) algorithm and its successor, C4.5, were two of the first for decision tree training algorithms. They were created by Ross Quinlan in 1986 and 1993, respectively [25] [26]. Many subsequent advances have been built upon these algorithms. Directed trees are used as decision support tools and are called decision trees. They serve as decision guidelines and examples for later decisions.

The root node, internal nodes, and terminal nodes, often known as leaves, can be used to categorize nodes in decision trees. The root node has no incoming edges and serves as the origin of the decision support process. Internal nodes contain at least two outgoing edges and exactly

one incoming edge. They include a test based on one of the characteristics of the data set [27]. Such a test might ask, for example, "Is the customer older than 35 for the feature age?" A solution to a decision problem, which is mainly represented by a class prediction, is located at the leaf nodes. Using the category predictions yes and no as an example, a decision problem would be whether a customer will make a purchase in an online store or not. Regarding leaf nodes, they have only one incoming edge and no outgoing edge.

Given a node «n», all nodes that follow it and are separated from it by exactly one edge are referred to as its children, while «n» is referred to as the parent of all its child nodes. An example of a decision tree is shown in Figure 5. For example, a data record with the attributes cold, polarBear would be sent to the left subtree, since its temperature property is cold, and then to the "North Pole" leaf, which would carry the label with appropriate information.

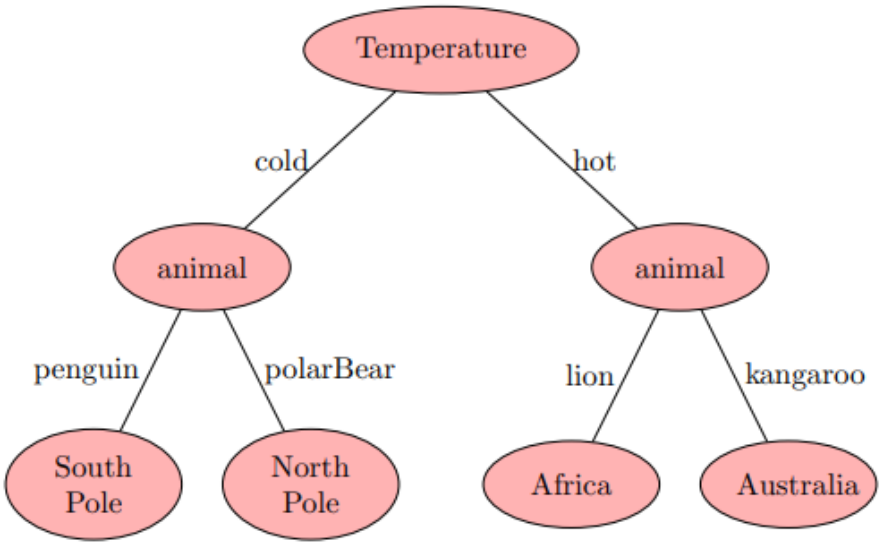


Figure 5: Decision Tree Algorithm [52]

A popular data mining technique used primarily for classification purposes, is to train a decision tree. Its goal is the prediction of the value of a target feature using a variety of input features. When training a decision tree in a supervised setting, patterns in the data are discovered using a training set and then the decision tree is constructed. The value of their target property can be predicted using a set of samples they have never seen before. The training set includes data records of the following format:

$$(\vec{x}, Y) = (x_1, x_2, x_3, \ldots, x_{n,} Y)$$

where «Y» is the desired feature value, «x» is a vector of n input values and «n» is the total number of features in the dataset [27].

A training set including a target feature, input features, a split criterion, and a stop criterion is required in order to train a decision tree and then build a classifier. The separation criterion specifies a value for each property at a particular node. By splitting the node using this attribute, the information which obtained is quantified by this value. Then, the node is split into the various results of each feature using the best value which obtained from all the features. At this stage, all generated subtrees are subjected to an iterative application of the process of determining the best separation between features until a stopping criterion is reached.

Common stopping criteria are:

- The maximum height of the tree when reached.
- The number of entries in the node should be less than the minimum allowed.
- The criterion that the best separation should not to exceed a certain threshold in terms of the information obtained.

There is no way to split records across all attribute results if the split attribute is a numeric type. This is one of the major improvements of the C4.5 decision tree over ID3. The C4.5 can also determine the optimal points for splitting numeric properties by splitting them using the greater than or equal to and less than operators.

This automated approach to decision tree training, can produce very large decision trees with relatively weak classification power. Furthermore, the trees are often (over)fitted, meaning they are over-matched during training. When these trees are applied to unknown data, the training does not perform well. Due to this phenomenon, the practice of pruning has evolved. The goal of this technique is to eliminate the least useful or inefficient elements of the decision tree, such as those that are over-fitted or based on noisy or inaccurate data. This often results in a further improvement in accuracy, while also reducing the size of the tree. Since every real-world dataset contains imprecise or noisy data, this step is very important.

Advantages of Decision Trees [28]

- Compared to other algorithms, the decision tree algorithm requires less effort to prepare data during preprocessing.

- Does not require data normalization.
- Does not require data scaling.
- Missing values in the data do not significantly affect the process of building a decision tree.

Disadvantages of Decision Trees

- A small change in the data can cause a large change in its structure.
- Sometimes the computation can be much more complicated compared to other algorithms.
- They often require more time to train the model.
- It is relatively expensive to train because of its complexity.
- It is not suitable for regression tasks and predicting continuous values.

3.2.2 kNN Algorithm

One of the simplest machine learning algorithms is the one called k-nearest neighbors (k-nearest neighbors or kNN). Generally, kNN is used for regression and classification. The «k» closest training instances in the sample space serve as input for both classification and regression, and the result depends on whether case 1 or case 2 was used respectively. Statistical estimation and pattern recognition have used both cases of kNN [29]. Regarding the different distance metrics, there are a variety available, including Euclidean, squared Euclidean, and Chebyshev. The Euclidean distance is the most preferred choice among them to determine the distance between two points [30]. Between two points x and y of dimension M, the Euclidean distance [30] (d) is given by the mathematical formula:

$$d(x, y) = \sqrt{\sum_{i=1}^{M}(x_i - y_i)^2}$$

To categorize an unknown object from known objects, the kNN classifier is used. Let's look at a simple example. Figure 6 shows a question mark with a red circle and the plus and minus symbols. By calculating the distance between each point and the question point (red circle), we can determine its class and determine whether it belongs to the plus or minus (plus and minus). The class of the object changes according to the k-value [30].

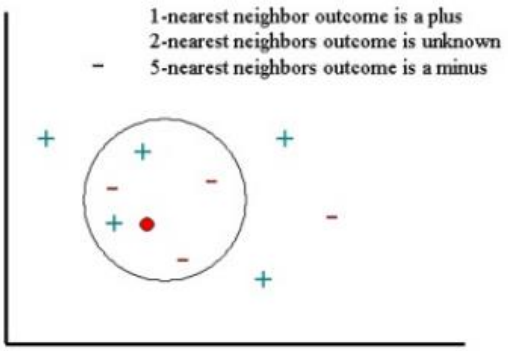


Figure 6: KNN Algorithm [53]

Advantages of KNN [31]

- No Training Period Required- kNN modeling does not require any training period, as the data itself is a model that will be the benchmark for future prediction and therefore it is very time efficient in improvising for a random modeling on the available data.
- Easy implementation – kNN is very easy to implement, as all that needs to be calculated is the distance between different points based on data of different features and this distance can be easily calculated using distance formula like Euclidean or Manhattan.
- As there is no training period, this new data can be added at any time without affecting the model.

Disadvantages of KNN [31]

- It does not work well with large amounts of data, as calculating the distances between each data instance would be very expensive.
- It does not work well with high dimensions, as this will complicates the distance calculation process to calculate the distance for each dimension.
- It is sensitive to noisy and missing data.
- Attribute scaling- Data should be scaled properly across the dimension.

3.2.3 Support Vector Machines (SVM)

The SVM algorithm uses a method known as kernel functions to transform nonlinear problems into linear ones, which essentially boils down to calculating the distance between two observations. This algorithm, also referred to as a large space classifier, determines whether to maximize the limit of the sample interval. The most important application of SVM is to simulate

the non-linear decision boundary using a nonlinear kernel function. It is considered to be one of the best classifiers. It uses a variety of kernels to handle both linear and nonlinear functions. Without overcomplicating a system, SVM can be applied to datasets with many features [32]. As shown in Figure 7, this algorithm attempts to divide the given points into two classes.

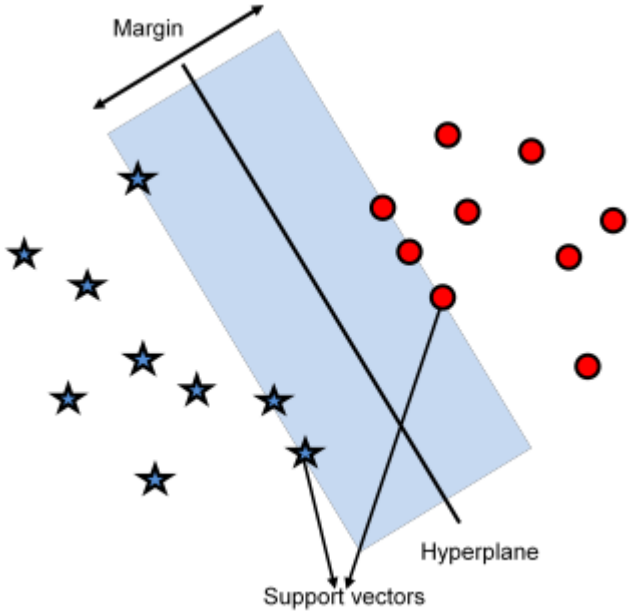


Figure 7: SVM Algorithm [54]

A data point is misclassified using hyperlayer B. According to Figure 8, hyperlayer A works well for categorizing the given data.

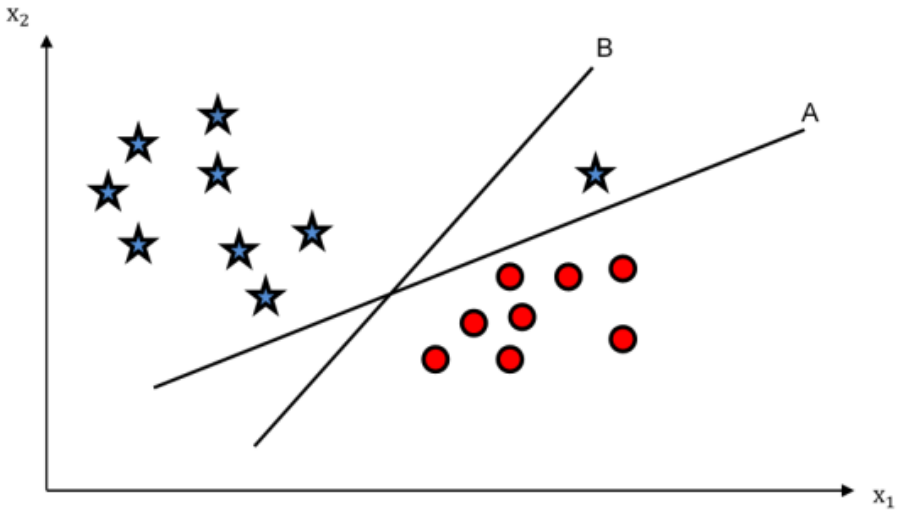


Figure 8: Categorization through two hyperlayers [54]

The logistic sigmoid function, an s-shaped curve that takes any real number and converts it to a value between 0 and 1, is used in the case of logistic regression when we analyze the output of the linear function and determine the value within the range 0 and 1. If the value exceeds a threshold value, the label 1 is often applied. If not, the label 0 is applied. The logistic regression function seeks to determine the probability that $y = 1$ given the input « x » where « x » and « y » are the input and output variables, respectively:

$$P(y = 1 \mid x)$$

To convert the expected values into probabilities, we use the logistic sigmoid function:

$$h(x) = \frac{1}{1 + e^{-x}}$$

where «e» is the base of the natural logarithm, «x» is the input to the function, and $h(x)$ is the output, expressed as a number between 0 and 1. The model aims to keep the training example as far away from the decision surface as possible. The term "decision surface" refers to a surface in a multidimensional state space that divides the space into various segments. Data points on one side of a decision surface are from a different category than data points on the other side. A margin is the distance between the nearest point and the decision surface, as illustrated in Figure 9.

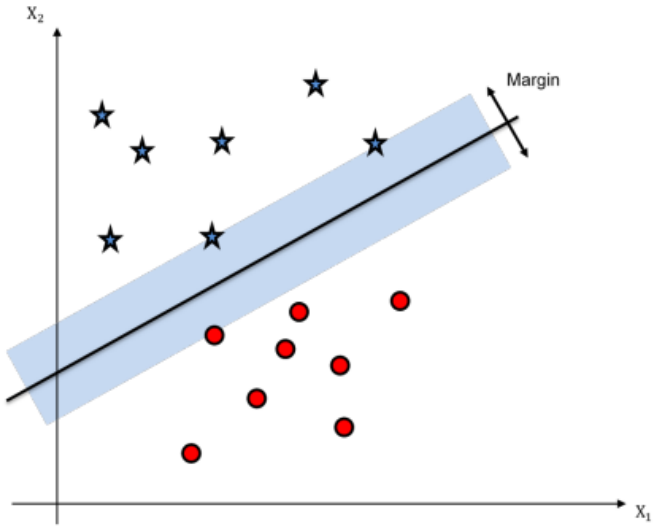


Figure 9: separation between the nearest point and the decision surface [54]

The decision surface that offers the largest margin is selected among all other decision surfaces. The support vectors are the points closest to the decision surface. There are at least two of them for a linearly separable two-dimensional feature space. The decision surface is defined by these support vectors [33].There are also many SVM kernels that can be used for classification. SVM with a linear kernel performs poorly when a data set is not linearly separable [34]. In this case, polynomial nonlinear SVM kernels are used. In order to create nonlinear boundaries between classes, nonlinear kernels expand the feature space of the input data. This is achieved by mapping the data to a higher dimensional space. This algorithm is extended to nonlinear surfaces using such a boundary produced by nonlinear kernels, which resembles a linear hyperplane.

In our case, the SVM algorithm will essentially try to separate with a straight line (as far as possible), those points that belong to class 0 (i.e. fake signature points) and those that belong to class 1 (i.e. real signature points).

Advantages of SVM [35]

- It is very efficient even with high-dimensional data.
- When the classes in the data are points are well separated, it works very well.
- It can be used for both regression and classification problems.
- It can also work well with image data ((which is perfect for our case because signatures are essentially images).

Disadvantages of SVM [35]

- If the classes in the data are points that are not well separated, meaning there are overlapping classes, it will not perform well.
- We need to choose an optimal kernel for SVM and this task is difficult.
- In large data set it requires comparatively more time for training.
- It is not a probabilistic model, so we cannot explain the classification in terms of probability.
- It is difficult to understand and interpret SVM model compared to Decision Tree as SVM is more complex (in case of application where for some reason we want to choose only these two algorithms).

3.2.4 Logistic Regression Algorithm

Logistic regression is the commonly used statistical method in empirical studies involving categorical dependent variables. As Allison (1999) demonstrates, a dichotomous dependent variable violates the assumptions of homoscedasticity and normality of the error term for the linear regression model. Consequently, the standard error estimates will not be consistent estimates of the true standard errors, and the coefficient estimates will no longer be efficient. In addition, estimating a linear probability model by the ordinary least squares technique will result in predicted values that are outside the reasonable range of the probability (0,1). For these reasons, the logistic regression model is used when the dependent variable is dichotomous. This model converts the probability to odds and then takes the logarithm of them. In this way, both the lower and upper bounds of the probability are removed. The logistic regression model has the following mathematical form [36]:

$$\log\left[\frac{p_i}{1-p_i}\right] = \text{a} + b_1 x_{i1} + b_2 x_{i2} + b_k x_{ik}$$

where «i» denotes individual, «p_i» represents the probability that the event occurs, «$1 - p_i$» represents the probability that the event does not occur, the ratio of the two represents the probabilities of the event, and the expression on the left side represents the log probability, or else the logit. On the right-hand side of the equation, " a " represents the intercept, " b " represents the regression coefficient, and " x " is the independent variable (for a more detailed background on logistic regression, see also Kleinbaum et al. 1998; McCulloch and Searle 2001; Menard 2010; Pedazur 1997; Rencher 2000; Tabachnick and Fidell 2001).

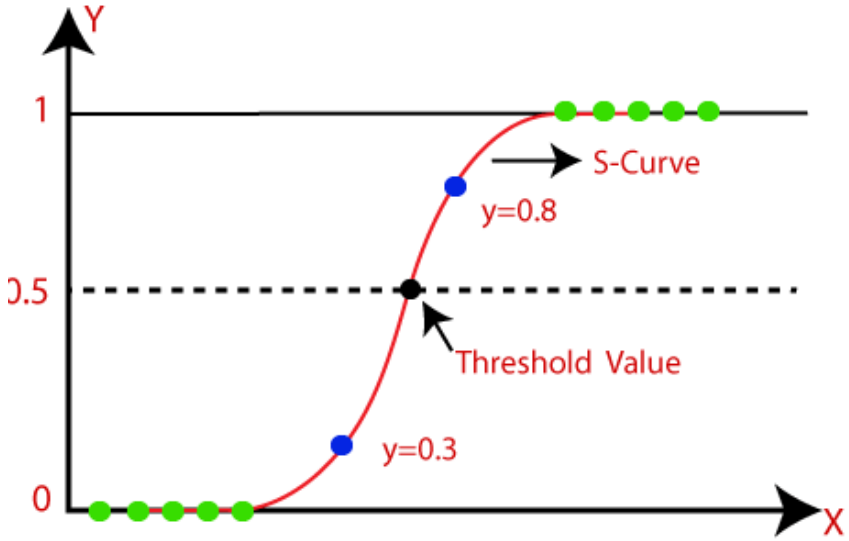


Figure 10: Logistic Regression Algorithm [55]

Advantages of Logistic Regression [37]

- Easy to apply and interpret but also effective in training
- Predicted parameters give inferences about the importance of each feature
- Performs well on low-dimensional data.
- Very effective when the data set has features that are linearly separated.
- It outputs well-calibrated probabilities along with the classification results.

Disadvantages of Logistic Regression [37]

- Exaggerations in high-dimensional data
- Nonlinear problems cannot be solved by this algorithm since it has a linear decision surface
- Assumes linearity between dependent and independent variables.
- It fails to capture complex relationships.
- Only important and relevant features should be used, otherwise the predictive value of the model will be degraded.

3.2.5 Naïve Bayes Algorithm

Naive Bayes is a machine learning algorithm that is often used for classification tasks, such as predicting whether an email is spam or not and it is probabilistic. At a high level, the algorithm works by calculating the probability that a known data point belongs to each possible class and then selecting the class with the highest probability. It is based on Bayes' theorem, which states that the probability of a hypothesis (in this case, the class) given some observed evidence (the data) is proportional to the probability of the evidence given the hypothesis multiplied by the prior probability of the hypothesis [38]. Mathematically this is expressed as follows:

$$P(A \mid B) = \frac{P(B \mid A)\, P(A)}{P(B)}$$

The "naive" part of the algorithm's name comes from the assumption that all features (i.e. the input variables) are independent of each other. This is a simplifying assumption that allows the algorithm to calculate probabilities more easily, but may not always be true in practice. There are many variants of Naive Bayes, including Gaussian Naive Bayes, Multinomial Naive Bayes, and Bernoulli Naive Bayes. The choice of which variant to use, depends on the nature of the data and the problem at hand [38].

In general, it is a relatively simple and computationally efficient algorithm, which can perform well in many classification tasks, especially when the number of features is large and the data is sparse. However, its performance may not be good if the independence assumption is violated or if the data contains too much noise or outliers. The Naive Bayes algorithm is shown schematically below:

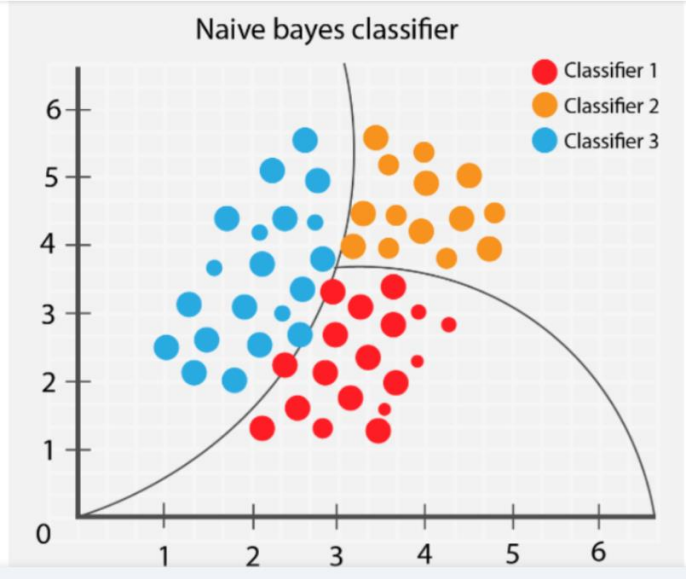


Figure 11: Naïve Bayes Algorithm [56]

In our case, the Naive Bayes algorithm can be useful by calculating the probability that some points (and more specifically values of features) belong to the fake signature and some other points belong to the genuine signature.

Advantages of Naive Bayes [39]

- It is fast and can save a lot of time.
- It is suitable for solving multi-class prediction problems.
- If its feature independence assumption holds, it can outperform other models and requires much less training data.
- It is more suitable for categorical input variables than for numerical variables.

Disadvantages of Naive Bayes [39]

- It assumes that all predictors (or traits) are independent, which is rarely the case in real life. This, limits the applicability of this algorithm to real-world use cases.
- It addresses the "zero frequency problem" where it assigns zero probability to a categorical variable whose category in the test data set was not available in the training data set. This problem is addressed by a smoothing technique.

- Its estimation can be wrong in some cases, so its output probabilities should not be taken too seriously in those cases.

3.2.6 Ensemble Algorithms

By mixing predictions from multiple learning algorithms/estimators, ensemble methods aim to increase the robustness of an estimator [40]. Ensemble methods are divided into two categories, which are as follows:

- Averaging techniques: The goal is to make predictions using various estimators and then average these results [40]. It is argued that a reduced variance makes the combined estimator using the mean approximation perform better on average than a single estimator [40].

- Boosting techniques: In contrast to average ensemble techniques, the goal of these techniques is to combine a number of weak estimators to create a strong ensemble [40]. By sequentially generating simple estimators, these techniques aim to reduce the bias of the generated population [40].

The most popular ensemble algorithms are the following:

- Random Forest
- AdaBoost
- XGBoost
- Gradient Tree Boosting
- Stacked Generalization

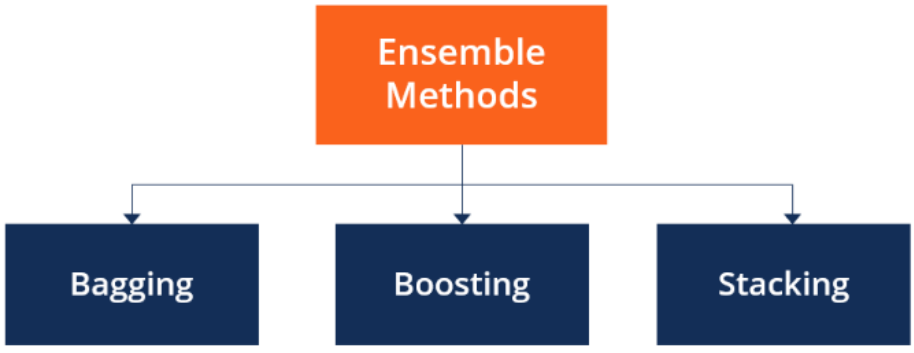


Figure 12: Ensemble Algorithms [57]

Advantages of Ensemble Algorithms [41]

- Provide accurate prediction results
- Stable and more robust models. The overall effect of many models is always less noisy than the individual models. This leads to stability and robustness of the model.
- Capture both linear and nonlinear relationships in the data. This is achieved by using two different models and forming a set from the two.

Disadvantages of Ensemble Algorithms [41]

- Decrease model interpretability. Using ensemble methods reduces the interpretability of the model due to increased complexity and makes it very difficult to extract any critical business insights at the end.
- Computational and planning time is high
- Selecting the models to create an ensemble is an art that is really difficult to dominate.

3.2.7 Neural Networks

Artificial Neural Network (ANN), commonly called Neural Network (NN), is an algorithm that was originally motivated by the goal of having machines that could imitate the human brain. A neural network consists of an interconnected group of artificial neurons. They are natural cellular systems that are able to receive and store information and use experiential knowledge, just like the human mind. ANN's knowledge comes from examples they encounter. In the human nervous system, the learning process involves modifications in the synaptic connections between neurons. Similarly, ANNs adapt their structure based on the output and input information flowing through the network during the learning phase.

The data processing process in a typical neural network consists of two main steps: the learning step and the application step. In the first step, a training database or historical price data is required to train the networks. This data set includes an input vector and a known output vector. Each of the inputs and outputs represents a node or neuron. There are also one or more hidden layers. The goal of the learning phase is to adjust the synaptic weights of the connections between different layers or nodes. After setting up the learning samples, in an iterative approach a sample is fed to the network and the resulting results are compared with the known outputs. If the result and the

unknown output are not equal, the synaptic weights of the connections are changed until the difference is minimized. After achieving the desired convergence for the networks in the learning process, we move to the application step, where the validation dataset is applied to the network for validation [42].

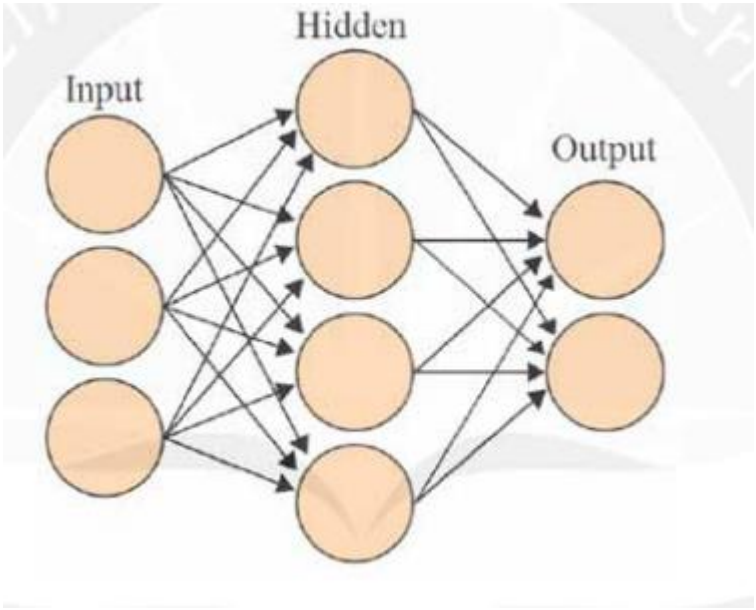


Figure 13: Neural Networks with hidden layers [58]

Advantages of Neural Networks [43]

- They can implement tasks that a linear program cannot.
- When an element is rejected for them, they can continue without problems because of its parallel characteristics.
- They do not need to be reprogrammed.
- They can be run in any application.

Disadvantages of Neural Networks [43]

- They require training to operate
- Their structure is different from that of microprocessors, so emulation is required.
- Large neural networks require a lot of processing time.

4 METHODOLOGY AND EXPERIMENTS

4.1 Motivation

The experimental part of this thesis is implemented using Matlab code. For this purpose three image datasets with signatures were tested. In practice, we created a large table, where each row corresponds to an image and the columns correspond to the attributes. In addition, besides to the columns with the attributes of each image, two additional columns were also created (the last two to be precise). The first column consists only of ones and zeros where one means a signature that is genuine and zero means a signature that is forged. The second column contains the active pixels, i.e. the sum of pixels in a given area where the signature is present. For the implementation we relied on a piece of web code that we have developed further. The main goal of the experiment was to improve the accuracy metric as much as possible. That is, the machine learning models to distinguish, as far as possible, the forged from the real handwritten signature. Two methods were used to achieve this. The first method was to further extract features related to the existing features. The second method referred to the creation of patches (parts) where each image is divided into four equal parts in order to improve the training process (in practice, each image corresponds to four smaller ones). The first patch corresponds to the upper right part of the image, the second corresponds to the upper left part of the image, the third to the lower left part of the image and finally the fourth to the lower right part of the image. A total of twelve experiments were performed. In six of them, our data were the initial features withoud additivies while in the remaining six it was the initial features were combined with the additions. In both cases, we performed the experiment in the whole image as well as with the patch approach. We performed this specific procedure for three different datasets, resulting in a total of twelve experiments. In the following sections, we present the datasets, describe the feature extraction and demonstrate the experimental results.

4.2 Datasets

In the present work, we have used three different datasets of signature images, which differ in size, i.e. the number of signatures under consideration ranges from a few to thousands of images.

1st Dataset

This dataset comes from an open project on github [44]. It is a dataset consisting of 36 signature images in total. In fact, it contains 19 real ones and 17 fake ones. The Figure 14 below shows some sample signature images from this dataset. In particular, the first six are genuine signatures, while the last three are forged.

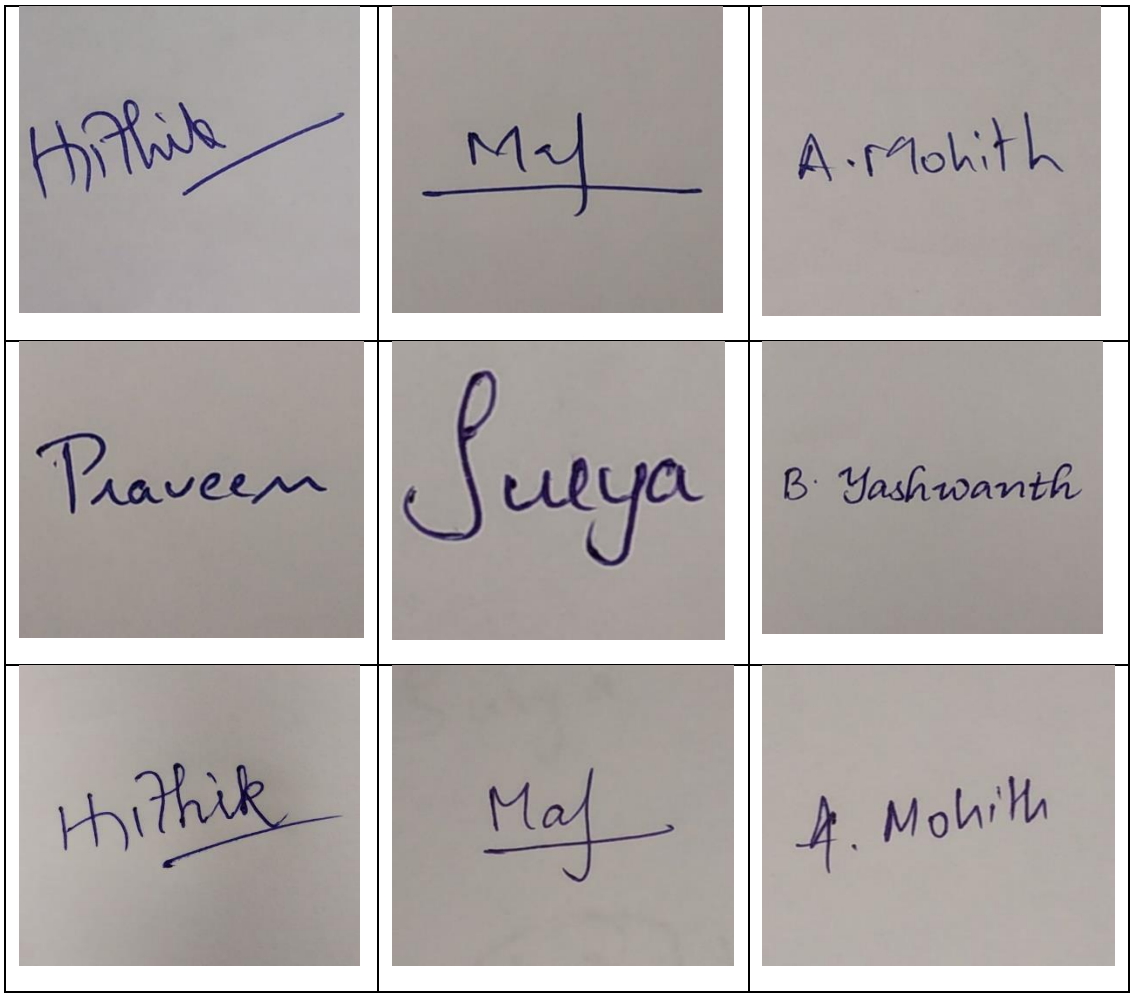


Figure 14: First Dataset

2nd Dataset

This dataset is retrieved from the kaggle site [45]. It contains 60 genuine signatures and 60 forged ones. The Figure 15 below shows examples from this dataset where, as before, the first six signatures are genuine and the rest are forged.

Figure 15: Second Dataset

3rd Dataset

The last dataset we examined is CEDAR [46] a large well-known dataset that contains 3121 forged signatures and 3121 genuine ones. The Figure 16 below shows some examples where the first six signatures are genuine and the rest are forged.

Figure 16: CRDAR Dataset

4.3 Feature Extraction

Feature extraction is an important part of the proposed methodology, since each signature image is expressed as a set of carefully selected features. First, nine features already introduced in [47] are listed and then the proposed additional features are also presented.

4.3.1 Initial features

Normalized Signature Area (NSA)

The normalized signature area is a measure of how much space the signature occupies within its bounding box. It indicates the relative size of the signature compared to the total available space.

Aspect Ratio

The aspect ratio describes the shape of the bounding box of the signature. It indicates whether the signature is more elongated (aspect ratio > 1) or more compact (aspect ratio < 1). It can provide insight into the overall shape and proportions of the signature. Mathematically it is expressed as follows:

$$AR = \frac{L}{W}$$

Maximum Horizontal Project

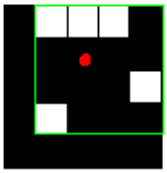
The maximum horizontal projection represents the maximum width of the signature. It refers to the total span of the signature along the horizontal axis.

Endpoints

Endpoints are points where lines or curves in the signature image end. Counting endpoints can be useful in detecting breaks or discontinuities in the signature strokes, that may indicate forgeries or irregularities. If $\mathrm{M}(x_1, y_1)$ is the midpoint and $\mathrm{A}(x, y)$ is the known endpoint, then the coordinates $\mathrm{B}(x_2, y_2)$ of the other endpoint are given by $x_2 = 2x_1 - x$ and $y_2 = 2y_1 - y$.

Centroid of vertically divided images

It is the pair of centroids (four values) of the two images formed by vertically dividing the main image vertically (so it is essentially four numbers when it is a pair).



Skew

The skew angle measures the tilt or slant of the signature image. A positive skew angle indicates a clockwise rotation, while a negative skew angle indicates a counterclockwise rotation. Skew detection is useful for aligning and normalizing signatures for further analysis.

4.3.2 Additional features

In addition to the features mentioned above, we have included additional features to strengthen the description of the signature. These features are:

Entropy

Entropy quantifies the degree of randomness or information content in the grayscale distribution of the signature. Higher entropy values indicate greater variability in the intensity levels, while lower entropy values indicate more uniform intensity distribution. Entropy is defined as:

$$H(X) = -\sum_{i=1}^{n} P(x_i) log_2 P(x_i)$$

where X is a random variable with has x_i possible outcomes and $P(x_i)$ denotes the corresponding probabilities.

Contrast

Contrast refers to a calculation of the amount of variation between a pixel and its neighbor across the entire image. The range of values for contrast is from 0 to the square of the size of the Gray Level Co-occurrence Matrix (GLCM) minus 1. When an image is constant, the contrast value is 0. In addition, the property known as contrast can also be referred to as variance and inertia. The mathematical formula is shown below:

$$Contrast = \sum_{i,j} |i-j|^2 \, p(i,j)$$

where

- i and j represent the intensity values of adjacent pixel pairs in the image,
- $p(i,j)$ is to the co-occurrence matrix, computed using the gray-matrix function in MATLAB. The co-occurrence matrix measures the frequency of intensity value pairs (i,j) occurring at a given distance and angle in the image and
- The $|i-j|^2$ term calculates the squared difference between the intensity values of the adjacent pixel pairs. Squaring the difference ensures that positive and negative differences contribute equally to the contrast measurement.

Correlation

Correlation calculates the degree of correlation between a pixel and its neighbor across the entire image. The range of correlation values is between -1 and 1. When an image is perfectly positively correlated, the correlation value is 1, while a perfectly negatively correlated image yields a correlation of -1. In the case of a constant image, the correlation value is NaN (Not a Number). The mathematical formula is shown below:

$$\sum_{i,j} \frac{(i-\mu_\iota)(j-\mu_j)p(i,j)}{\sigma_\iota \sigma_j}$$

where:

- i and j represent the intensity values of neighboring pixel pairs in the image,
- $p(i,j)$: corresponds to the co-occurrence matrix,
- μ_ι and μ_j: represent the means of the intensity values along the x-axis and y-axis, respectively. They are computed using the formulas:

$$\mu_i = \sum i \, p(i,j) \quad \text{for all } i,j$$

$$\mu_j = \sum j \, p(i,j) \quad \text{for all } i,j$$

- σ_χ and σ_y: represent the standard deviations of the intensity values along the x-axis and y-axis, respectively. They are calculated with the formulas:

$$\sigma_\chi = \sqrt{\sum (i-u_x)^2 P(i,j)} \quad \text{for all } i,j$$

$$\sigma_j = \sqrt{\sum (j - u_y)^2 P(i,j)} \quad \text{for all } i, j$$

Action

Returns the sum of the squares in the GLCM. Action also represents the activity or energy present in the signature image. It can be related to the overall stroke dynamics and speed of the signer, providing information about the fluidity or hesitation in the signature.

Homogeneity

Homogeneity calculates a value that quantifies how closely the elements in the GLCM are distributed along its diagonal. The range of homogeneity values is between 0 and 1. A homogeneity value of 1 indicates that the GLCM is perfectly diagonal, i.e. the elements are distributed exclusively along the diagonal. The mathematical formula is shown below:

$$\sum_{i,j} \frac{p(i,j)}{1 + |i - j|}$$

where

- i, j represent the intensity values of adjacent pixel pairs in the image,
- $p(i,j)$: corresponds to the co-occurrence matrix, and
- The $|i - j|$ term calculates the absolute difference between the intensity values of adjacent pixel pairs. Taking the absolute difference ensures that the homogeneity measurement is independent of the direction of the intensity difference.

Area

Area represents the total number of pixels in the signature image. It provides a measure of the overall size or extent of the signature.

Bounding box

The bounding box is the smallest rectangular region that encloses the signature image. It provides information about the position, size, and orientation of the signature within the image.

Convex Area

Convex area measures the total number of pixels within the convex hull of the signature image. The convex hull is the smallest convex polygon that contains all of the signature pixels.

Eccentricity

Eccentricity characterizes the curvature or roundness of the signature. It measures how much the shape deviates from a perfect circle. A value closer to 0 indicates a more circular shape, while a value closer to 1 indicates a more elongated shape.

Isodiameter

The isodiameter represents the maximum distance between any two pixels in the signature image. It provides an estimate of the overall size or span of the signature.

Euler number

The euler number measures the topological connectivity of the signature image. It is calculated by subtracting the number of holes from the number of objects. A higher Euler number indicates a simpler and more connected structure, while a lower Euler number indicates a more complex and fragmented structure.

Extent

Extent is the ratio of the number of pixels in the signature image to the total number of pixels in the bounding box. It provides information about the occupancy of the signature within its bounding box. A higher extent value indicates a larger portion of the bounding box is occupied by the signature.

Filled area

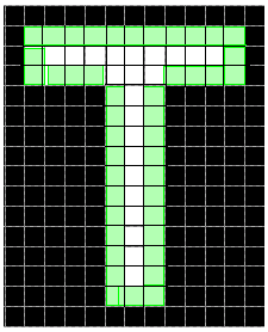
Filled area measures the number of pixels within the closed boundary of the signature image. It provides a measure of the total colored area in the signature.

Orientation

Orientation specifies the angle between the major axis of the bounding ellipse and the x-axis. It provides information about the tilt or rotation of the signature.

Perimeter

Perimeter calculates the length of the boundary or perimeter of the signature image. It provides information about the complexity and intricacy of the signature shape.



Solidity

Solidity is the ratio of the area of the signature image to the area of its convex hull. It describes the compactness or solidity of the signature shape. A higher solidity value indicates a more solid or filled shape.

Major axis length

The major axis length represents the length of the major axis of the bounding ellipse. It provides information about the overall size or extent of the signature along its longest dimension.

Minor axis length

Minor axis length represents the length of the minor axis of the bounding ellipse. It provides information about the overall size or extent of the signature along its shortest dimension.

4.3.3 Data preparation

In order to investigate the applicability of the proposed setup, a total of twelve experiments were performed. The first four experiments concern the first dataset [44], the next four experiments concern the second dataset [45], and the last four experiments concern the CEDAR dataset [46]. The three experiments follow the same methodology: in the first experiment (of the four), the classification is applied to the whole image using the basic features, in the second, the basic features are applied to the four patches that divide the image, in the third, the extended feature set is applied to the whole image, while in the fourth, the extented feature set is applied to the four patches. The following table summarizes the experiments performed.

Table 1. Experiments performed for each dataset

	Initial features Without patches	**Initial features With patches**	**Extented features Without patches**	**Extented features With patches**
1st Dataset	Experiment 1.1 Result: 36×11 values	Experiment 1.2 Result: 36×41 values	Experiment 1.3 Result: 36×33 values	Experiment 1.4 Result: 36×133 values
2nd Dataset	Experiment 2.1 Result: 120×11 values	Experiment 2.2 Result: 120×41 values	Experiment 2.3 Result: 120×33 values	Experiment 2.4 Result: 120×133 values
3rd Dataset	Experiment 3.1 Result: 2240×11 values	Experiment 3.2 Result: 2240×41 values	Experiment 3.3 Result: 2240×33 values	Experiment 3.4 Result: 2240×133 values

All experiments used the following machine learning models: (i) Support Vector Machines, (ii) Decision Trees, (iii) kNN, (iv) Neural Networks, (v) Naïve Bayes, (vi) Logistic Regression, and (vii) Ensemble Algorithms.

4.4 Experiments

For each dataset, the training and testing accuracy results are summarized to determine which one of the seven methods under consideration provides the best performance. In the following tables, case 1 refers to initial features without patches, case 2 refers to initial features with patches, Case 3 refers to extended features without patches and finally case 4 refers to extended features with patches. Also, for each dataset sample images are provided showing correctly and incorrectly classified samples, while t-SNE visualizations show the disctrubition of the data. t-SNE (t-distributed Stochastic Neighbor Embedding) is an unsupervised non-linear dimensionality reduction technique for data exploration and visualization of high-dimensional data. It provides an intuition of how data is arranged in higher dimensions. It is often used to visualize complex datasets in two (or three) dimensions, highlighting underlying patterns and relationships in the data. The provided t-SNE visulization refers to case 4 of extended features with patches. Different distances are visualized, namely, Cosine, Chebyshev and Euclidean. Specifically:

- Cosine distance: is a measure of the dissimilarity between two non-zero vectors in an inner product space. It is derived from the cosine of the angle between the two vectors.

- Chebyshev distance: measures the distance between two points as the maximum difference over any of their axis values.
- Euclidean distance: is a measure of the linear distance between two points in Euclidean space.

4.4.1 Experiments for dataset 1

The first dataset contains 36 images in total (19 images with genuine signatures and 17 images with forged signatures). We have downloaded this dataset from the following link:

https://github.com/Kevv-J/Signature-Verification-using-MATLAB

Since there are 36 images in total, it can be considered a small dataset for classification problems.

Table 2. Training accuracy in the 1st dataset for all cases

Model	Accuracy % (Training) Case 1	Accuracy % (Training) Case 2	Accuracy % (Training) Case 3	Accuracy % (Training) Case 4
Naive Bayes	60,6	81,8	72,7	60,6
KNN	72,7	78,7	78,7	72,7
Ensemble	72,7	60,6	78,7	87,8
SVM	69,6	78,7	81,8	84,8
Neural Network	69,6	78,7	75,7	81,8
Decision Tree	60,6	51,5	69,6	90,9
Logistic Regression	54,5	54,5	54,5	51,5

The experimental results show that Decision Tree provides the maximum validation accuracy in case 4 (extended features with patches) with 90.9%. Regarding the accuracy for the test set, all algorithms (although in different cases) achieved 100% accuracy. This means that forged signatures were correctly distinguished from genuine ones, as shown in the following table.

Table 3. Testing accuracy in the 1st dataset for all cases

Model	Accuracy % (Testing) Case 1	Accuracy % (Testing) Case 2	Accuracy % (Testing) Case 3	Accuracy % (Testing) Case 4
Naive Bayes	100	66,6	100	66,6
KNN	66,6	66,6	66,6	100
Ensemble	66,6	100	100	66,6
SVM	33,3	66,6	100	100
Neural Network	33,3	66,6	100	100
Decision Tree	33,3	66,6	100	66,6
Logistic Regression	33,3	33,3	33,3	100

The next figures show some examples of correct and incorrect classification examples.

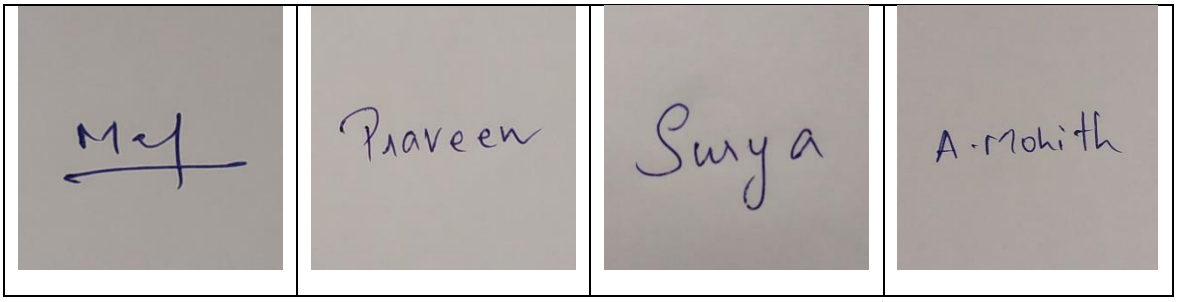


Figure 17 Correctly classified signature examples in the 1st dataset.

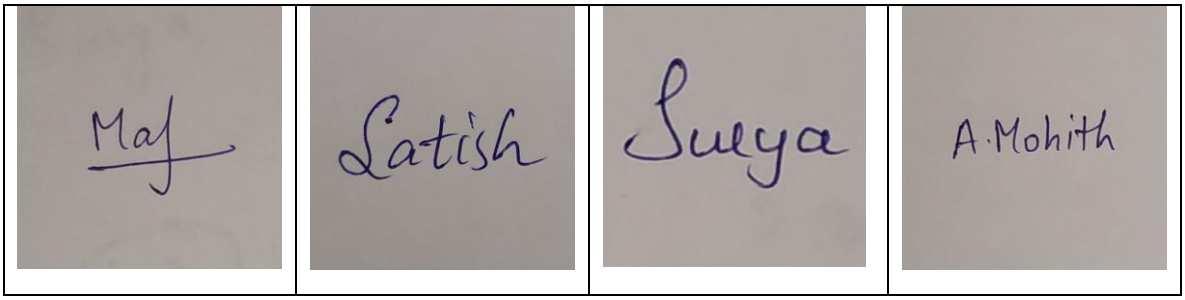


Figure 18 Misclassification examples in the 1st dataset.

The next three figures demonstrate the 2D t-SNE visualization for the three different distances.

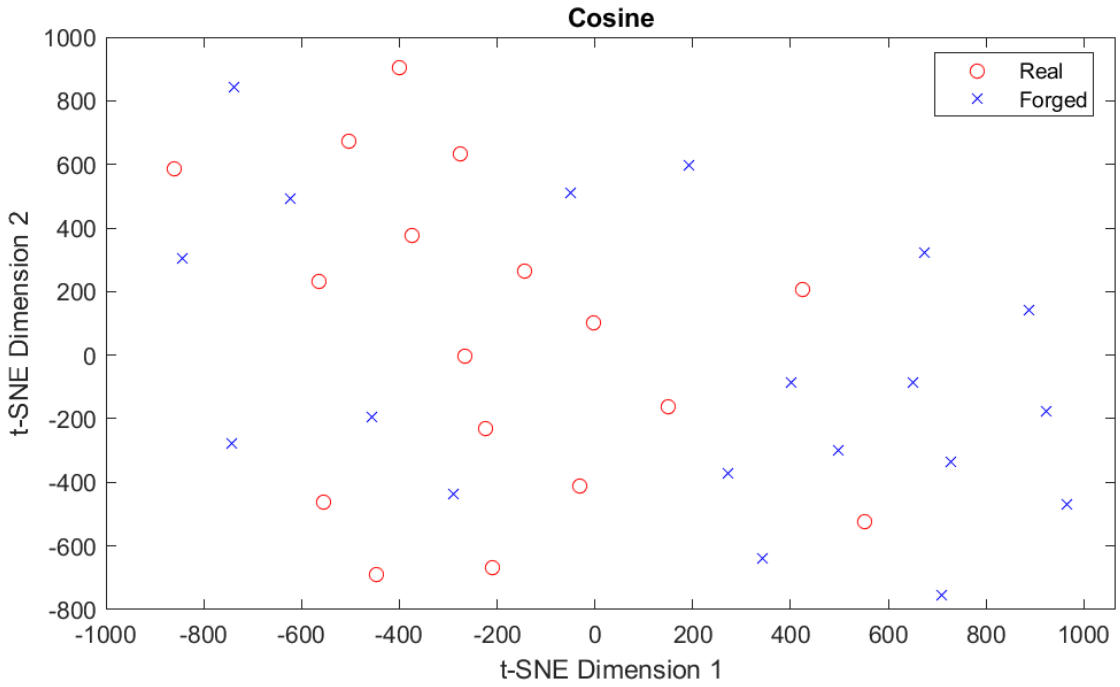


Figure 19: Cosine distance of real and forged images of dataset 1

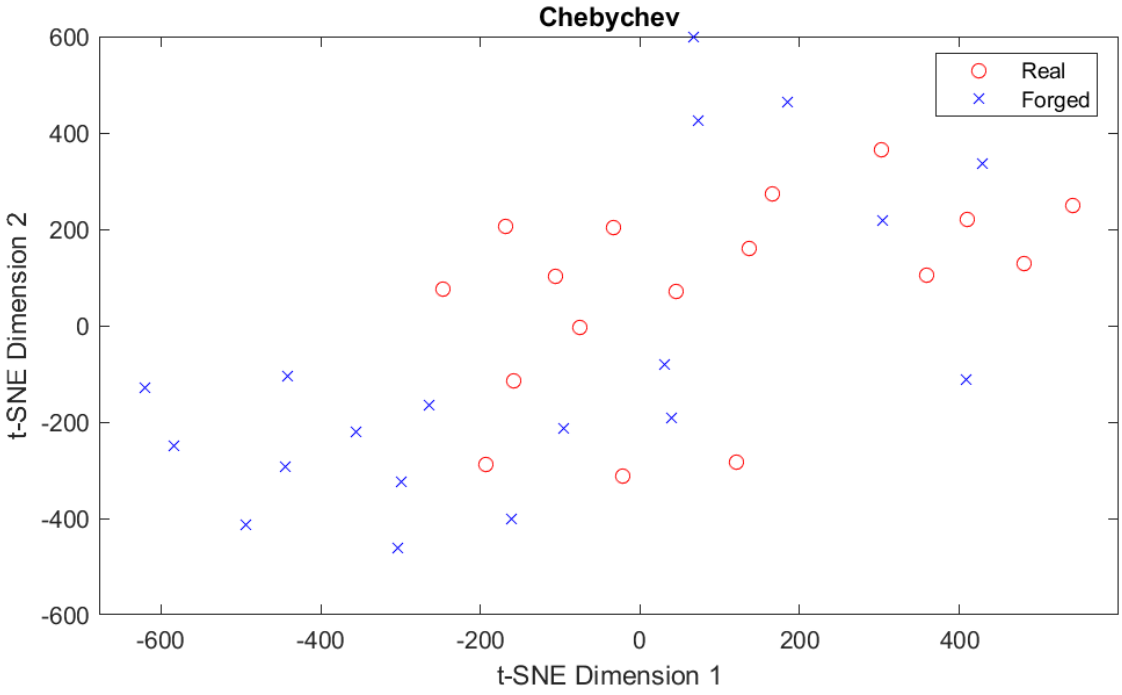


Figure 20: Chebychev distance of real and forged images of dataset 1

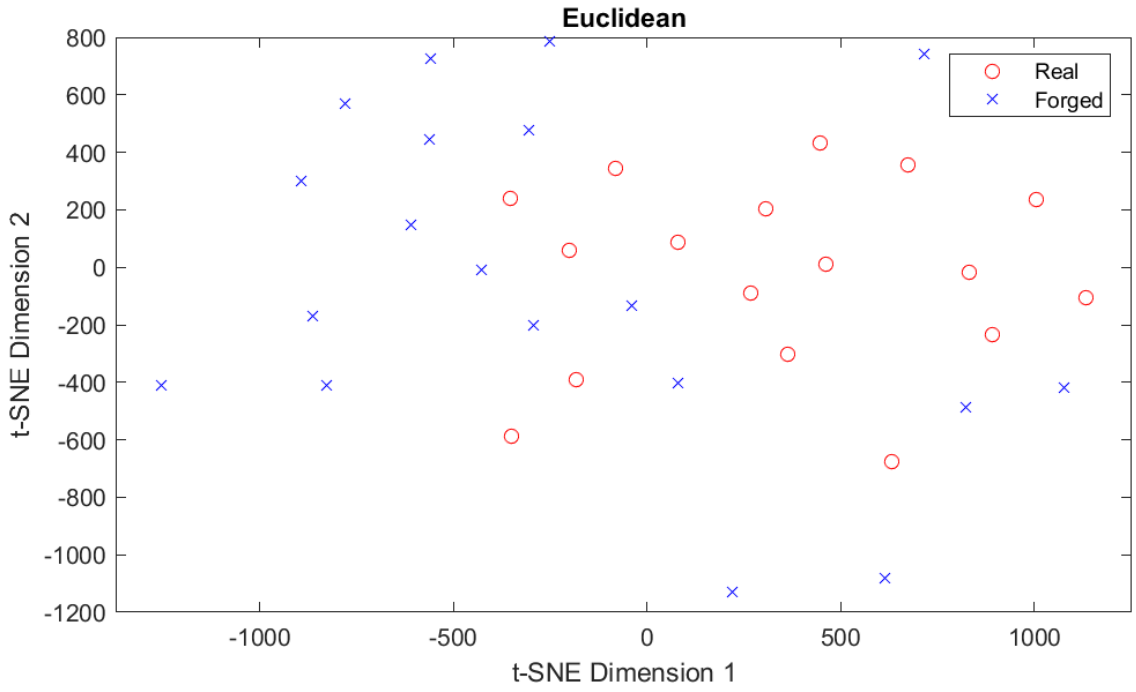


Figure 21: Euclidean distance of real and forged images of dataset 1

4.4.2 Experiments for dataset 2

The second dataset contains a total of 120 images (60 images with genuine signatures and 60 images with forged signatures). This is a medium sized dataset and we have downloaded it from the following link: https://www.kaggle.com/datasets/divyanshrai/handwritten-signatures

Regarding the training process, it can be observed that the algorithm that provides the maximum accuracy validation is Neural Network in Case 3 with 85,1%. In case of testing, we observe that the algorithms with the maximum testing accuracy are SVM and Neural Network in case 1 with 91,6% and Naïve Bayes in case 4, with the same percentage.

Table 4. Training accuracy in the 2nd dataset for all cases

Model	Accuracy % (Trining) Case 1	Accuracy % (Training) Case 2	Accuracy % (Training) Case 3	Accuracy % (Training) Case 4
SVM	74,1	69,4	78,7	63,8
Neural Network	75,9	62	85,1	64,8
KNN	75,9	75	73,1	62,9
Logistic Regression	59,2	57,4	64,8	51,8
Decision Tree	61,1	56,4	59,2	58,3
Ensemble	71,2	62,9	75,9	63,8
Naive Bayes	65,7	62	63,8	52,7

Table 5. Testing accuracy in the 2nd dataset for all cases

Model	Accuracy % (Testing) Case 1	Accuracy % (Testing) Case 2	Accuracy % (Testing) Case 3	Accuracy % (Testing) Case 4
SVM	91,6	66,6	83,3	83,3
Neural Network	91,6	75	83,3	83,3
KNN	83,3	50	83,3	83,3
Logistic Regression	58,3	66,6	66,6	50
Decision Tree	66,6	66,6	66,6	75
Ensemble	83,3	75	83,3	83,3
Naive Bayes	66,6	83,3	75	91,6

The next figure provides examples of correctly classified images. Specifically, the first two signatures are fake and were classified as fake and the other two are real and were classified as real.

Figure 22 Correctly classified signature examples in the 2nd dataset.

The next figure provides some examples of misclassification. The first two signatures are fake and were classified as real and the other two are real and were classified as fake.

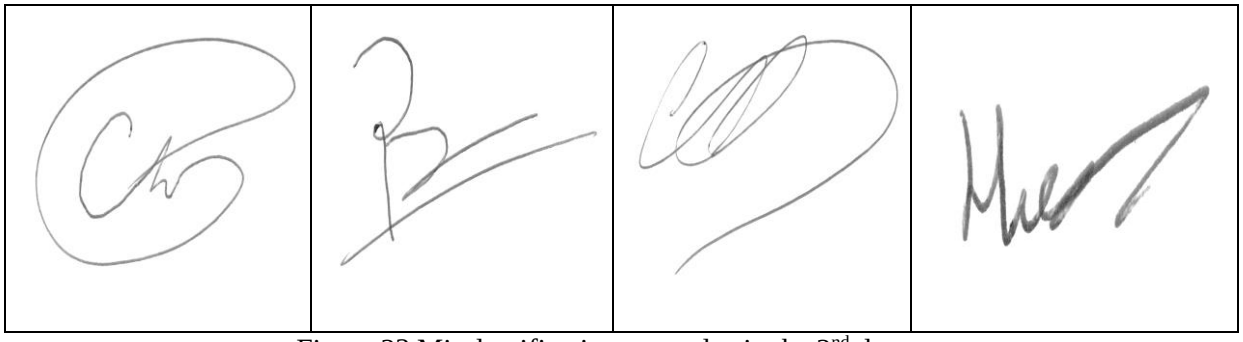


Figure 23 Misclassification examples in the 2nd dataset.

The t-SNE analysis for the 2nd dataset is shown in the following three figures. Once again, we see that there are difficulties in the separating the two classes.

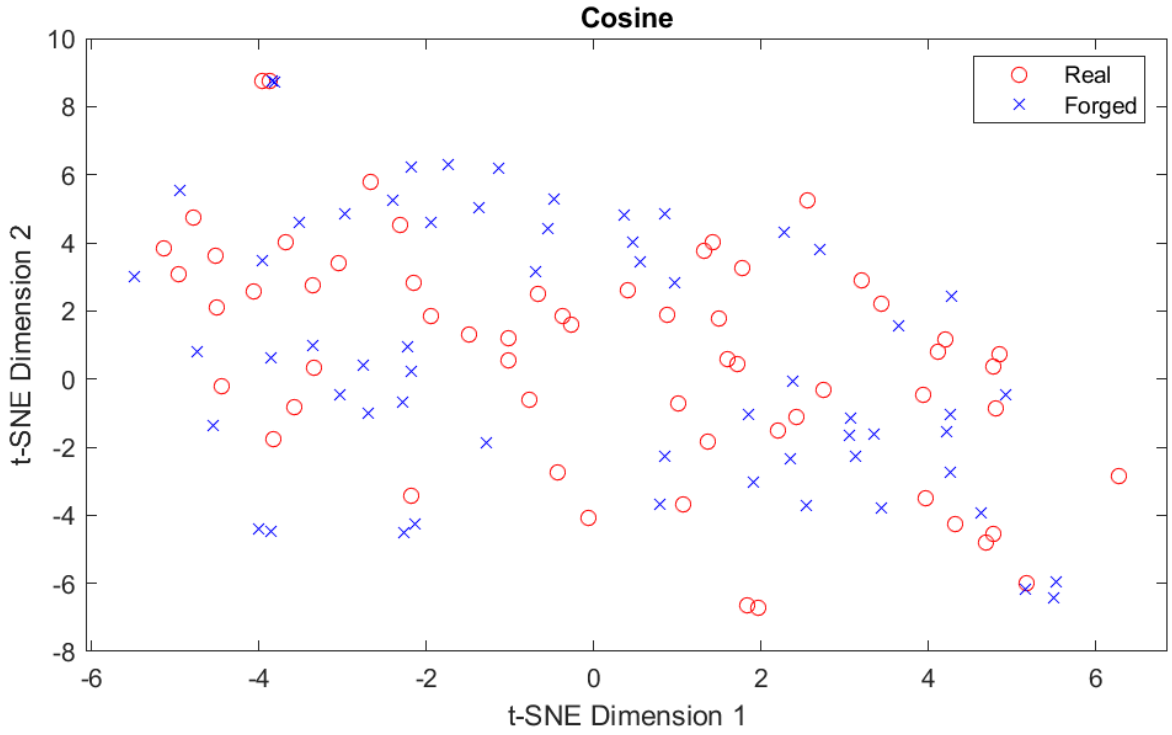


Figure 24: Cosine distance of real and forged images of dataset 2

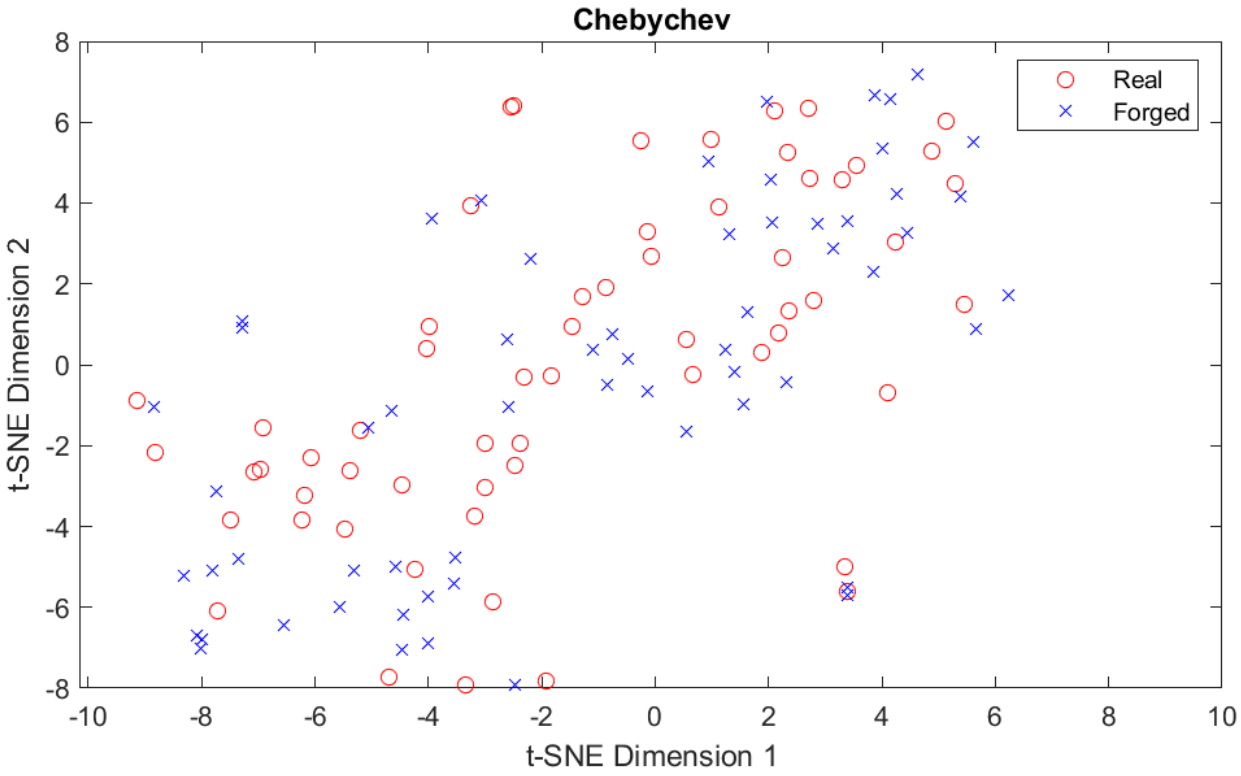


Figure 25: Chebyshev distance of real and forged images of dataset 2

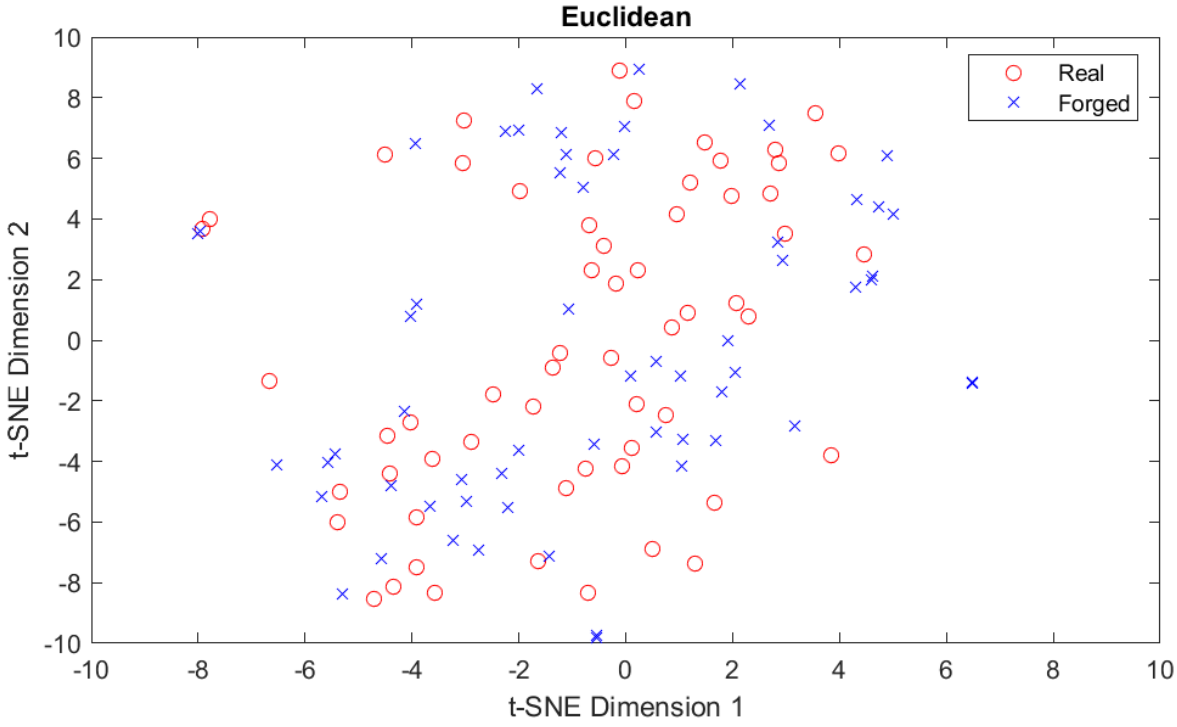


Figure 26: Euclidean distance of real and forged images of dataset 2

4.4.3 Experiments for dataset 3

The third dataset (CEDAR) contains 2640 images (1320 images with forged signature and 1320 images with genuine signature). We have downloaded it from the following link: https://paperswithcode.com/dataset/cedar-signature

and it is a much larger dataset than the other two.

The best training results are provided by the Ensemble algorithm with 83,4% for case 3. The same method provides the best testing accuracy with 87.5% for case 2.

Table 6. Training accuracy in the 3rd dataset for all cases

Model	Accuracy % (Training) Case 1	Accuracy % (Training) Case 2	Accuracy % (Training) Case 3	Accuracy % (Training) Case 4
SVM	81,2	55,9	81,1	53,7
Neural Network	77,5	55,5	78,8	49,9
KNN	49,9	49,9	49,9	49,9
Logistic Regression	68,6	53,6	71,2	49,9
Decision Tree	72,4	76,5	74,2	76,5
Ensemble	80,6	82,6	83,4	83,3
Naive Bayes	69,1	73,5	65,3	53,7

Table 7. Testing accuracy in the 3rd dataset for all cases

Model Type	Accuracy % (Testing) Case 1	Accuracy % (Testing) Case 2	Accuracy % (Testing) Case 3	Accuracy % (Testing) Case 4
SVM	56,8	80,3	50	78
Neural Network	77,6	50	82,9	50
KNN	50	50	50	50
Logistic Regression	72,3	50	58,7	50
Decision Tree	79,9	78,7	79,9	71,5
Ensemble	79,9	87,5	83,7	84,8
Naive Bayes	54,5	67	71,9	70,4

The next figure provides examples of correctly classified images where the first two signatures are correctly classified as fakes while the other two are correctly classified as real.

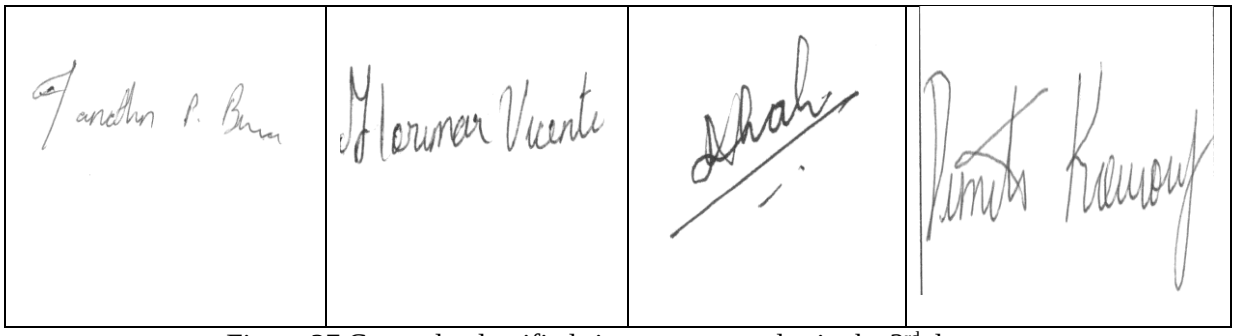


Figure 27 Correctly classified signature examples in the 3rd dataset.

The next figure shows examples of misclassification. The first two signatures are fake and were classified as real and the other two are real and were classified as fake.

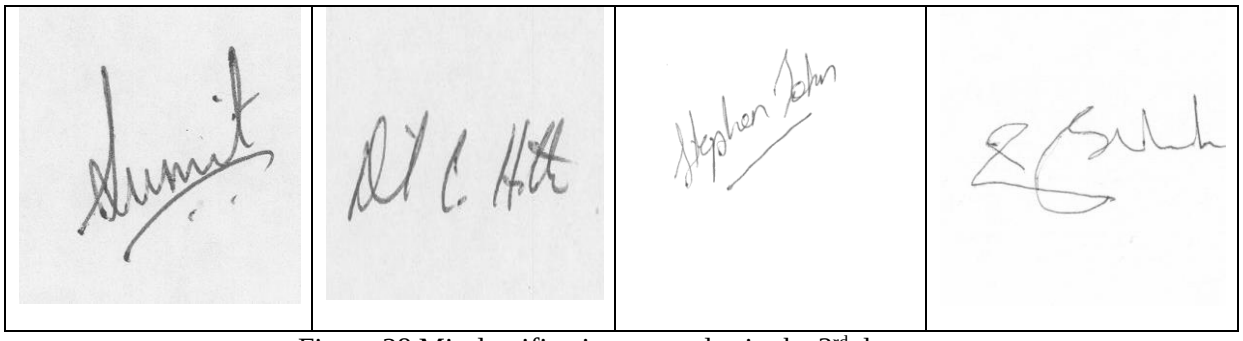


Figure 28 Misclassification examples in the 3rd dataset.

The following three figures domenstrate the results of the t-SNE visualization regarding the 3rd dataset.

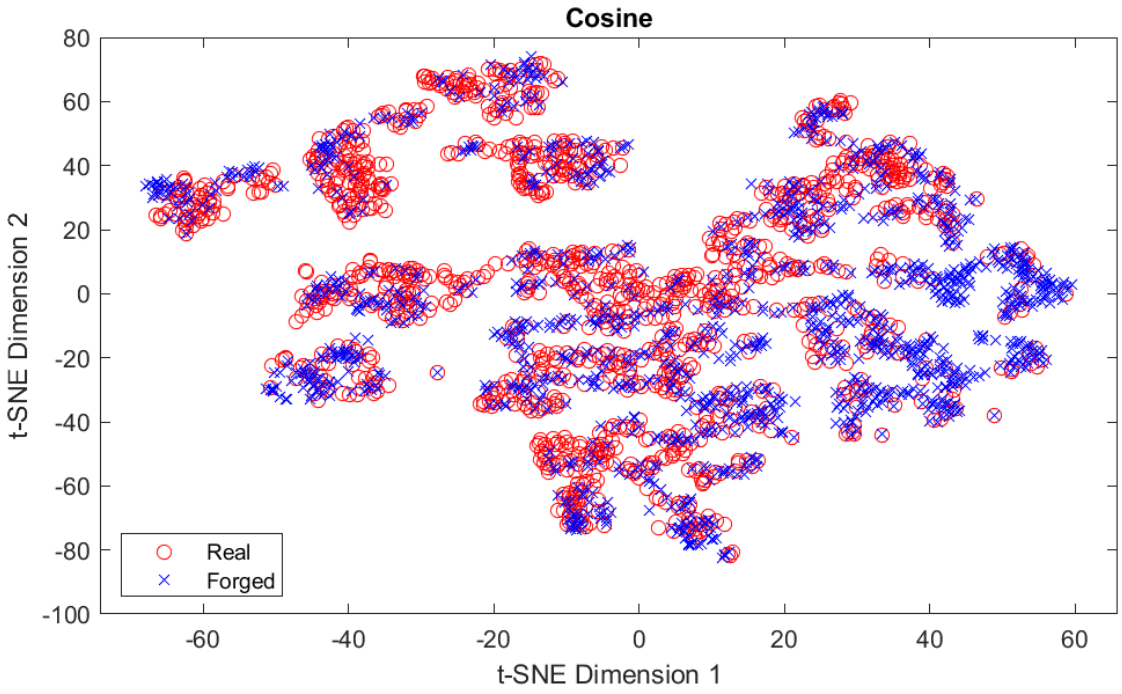


Figure 29: Cosine distance of real and forged images of dataset 3

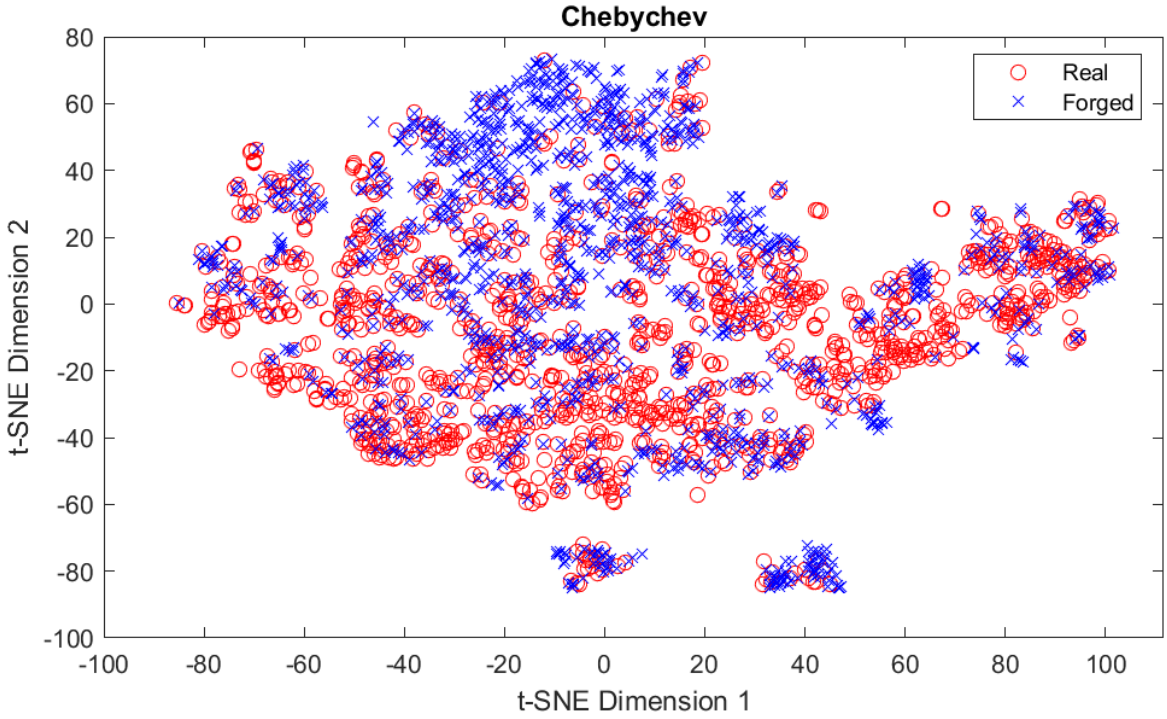


Figure 30: Chebychev distance of real and forged images of dataset 3

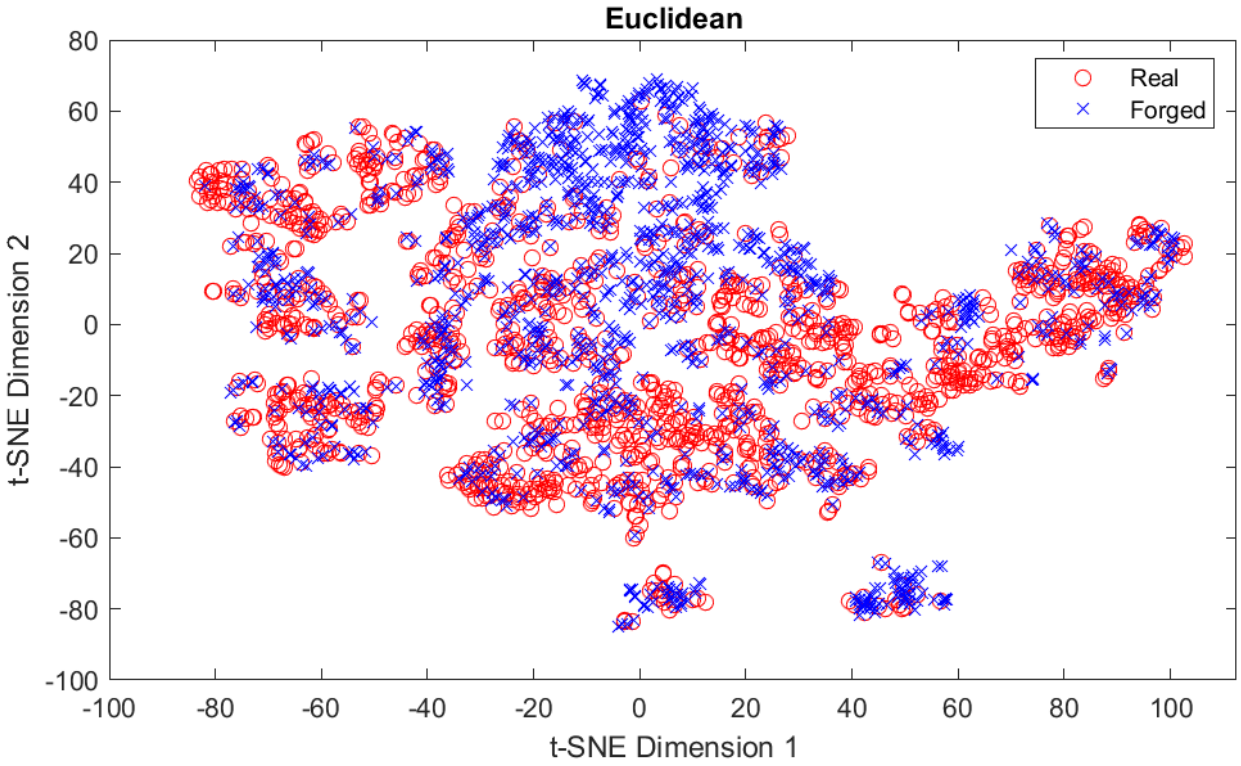


Figure 31: Euclidean distance of real and forged images of dataset 3

From the above results it can be seen that the separation in to the two classes is not an easy task even in the case of the 1st dataset which has fewer samples. Nevertheless, the experiments with different machine learning algorithms showed that some of them provide reasonable classification results.

5 CONSLUSIONS

In this thesis, we conducted a literature review on handwritten signature verification systems. First, we gave a general introduction to handwritten signature verification systems, as we mentioned that the most important steps for forgery detection are feature extraction and machine learning. We then discussed in detail the preprocessing of handwritten signature data, feature extraction, feature selection, and forgeries and their types. We then focused on the concept of machine learning, its importance in forgery verification, its types and the most common classifiers, as well as the advantages and disadvantages of each.

Regarding the experimental part of the thesis, we studied 3 datasets containing genuine and forged signatures. The datasets were of different sizes in terms of volume, so we examined the performance of the classifiers on few data (dataset 1), on a medium sized dataset (dataset 2), and on a large dataset with thousands of samples (dataset 3). In the experimental part, we ran a total of twelve experiments, i.e. four cases for each dataset. The first case refers to the original features extracted from the whole image without dividing it into patches. In the second case, the image is divided into four patches and features are extracted from each patch separately. In the third case, the proposed extended set of features is extracted from the whole image. Finally, the fourth case refers to extended features extracted from each of the four patches, separately. We presented results showing the training and testing accuracy in all cases for all three datasets. In addition, especially for the fourth case, we graphically presented the t-SNE results for each dataset when Chebyshev, cosine, and Euclidean distances are applied.

Regarding the experimental result, we observed that in the case where we have few data (specifically in all experiments of dataset 1), many classifiers achieve 100% test accuracy. This is because the less training data we have, the more likely the classifiers are to correctly predict the entire test set. Our experiments also showed that both enriching the extracted features and dividing the image into patches have a positive effect on most of the classifiers in terms of both training and testing accuracy. This is true for all three datasets. However, in certain cases, some of the classifiers achieved low training and testing accuracy, which can be attributed to overfitting due to the nature of these classifiers. The t-SNE plots have shown that there is no good separation of the data, especially in datasets 2 and 3. The results are characterized by strong class overlap, which hinders the separation efficiency of the classifiers. This conclusion does not hold (or at

least not to a large extent) for dataset 1, which contains much fewer data samples than the other two datasets. In genral, regarding the performance of the classifiers, the experimental results were satisfactory. In all datasets and in all cases, good training/testing accuracies were reported.

In a future work, the efficiency could be further increased by applying deep learning techniques. Such an approach would be especially helpful when the amount of data is large, as in the case of dataset 3. The high-dimensional enriched features presented in this thesis would potentially benefit from the deep nature of such classifiers, resulting in even better class separation between forged and genuine signatures.

6 REFERENCES

[1] D. Impedovo, G. Pirlo, and R. Plamondon, “Handwritten signature verification: New advancements and open issues,” in 2012 International Conference on Frontiers in Handwriting Recognition, Sept 2012, pp. 367–372.

[2] H. Lei and V. Govindaraju, “A comparative study on the consistency of features in on-line signature verification,” Pattern Recogn. Lett., vol. 26, no. 15, pp. 2483–2489, Nov. 2005.

[3] F. J. Zareen and S. Jabin, “A comparative study of the recent trends in biometric signature verification,” in 2013 Sixth International Conference on Contemporary Computing (IC3), Aug 2013, pp. 354–358.

[4] G. Padmajadevi and K. S. Aprameya, “A review of handwritten signature verification systems and methodologies,” in 2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT), March 2016, pp. 3896–3901.

[5] D. Morocho, J. Hernandez-Ortega, A. Morales, J. Fierrez, and J. OrtegaGarcia, “On the evaluation of human ratings for signature recognition,” in 2016 IEEE International Carnahan Conference on Security Technology (ICCST), Oct 2016, pp. 1–5.

[6] R. Plamondon and S. N. Srihari, “On-line and off-line handwriting recognition: A comprehensive survey,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 22, no. 1, pp. 63–84, Jan. 2000.

[7] Rathgeb, C., & Ratha, N. K. (2017). A survey on preprocessing techniques for handwritten signature verification. ACM Computing Surveys, 50(6), 87.

[8] Kai Huang and Hong Yan. Off-line signature verification based on geometric feature extraction and neural network classification. Pattern Recognition, 30(1):9–17, January 1997. (data processing)

[9] Mustafa Berkay Yilmaz and Berrin Yanikoglu. Score level fusion of classifiers in off-line signature verification. Information Fusion, 32, Part B:109–119, November 2016. (data processing)

[10] Meenakshi K. Kalera, Sargur Srihari, and Aihua Xu. Offline signature verification and identification using distance statistics. Int. Journal of Pattern Recogn., 18(07):1339–1360, November 2004. (data processing)

[11] Liao, S.; Hu, Y.; Zhu, X.; Li, S.Z. Person re-identification by local maximal occurrence representation and metric learning. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 2197–2206. [Google Scholar]

[12] Osama M. Elrajubi, "Off-line Signature Verification Based on Fuzzy Logic", Master's Thesis, Academy of Graduate Studies, Tripoli, Libya, 2009.

[13] C. De Stefano, F. Fontanella , C. Marrocco, A. Scotto di Freca, 2014. A GA-based feature selection approach with an application to handwritten character recognition. Pattern Recognition Letters 35 (2014) 130–141.

[14] J. Almajhdi, M. T. Hasan, and M. Bennamoun, "Online signature verification using dynamic time warping and recurrent neural networks," in Proceedings of the 2018 IEEE International Conference on Image Processing, 2018.

[15] L. G. Hafemann, R. Sabourin, and L. S. Oliveira, "Offline Handwritten Signature Verification-Literature Review," arXiv preprint arXiv:1507.07909, 2015.

[16] R. K. Sharma and A. Kumar, "Comparison of offline and online signature verification systems using feature extraction techniques," in Proceedings of the 2016 IEEE 6th International Conference on Advanced Computing, 2016.

[17] S. Z. Li, Encyclopedia of Biometrics, 1st ed. Springer Publishing Company, Incorporated, 2009.

[18] A. K. Jain, A. Ross, and S. Prabhakar, "An introduction to biometric recognition," IEEE Trans. Cir. and Sys. for Video Technol., vol. 14, no. 1, pp. 4–20, Jan. 2004.

[19] A. K. Jain, A. Ross, and S. Prabhakar, "An introduction to biometric recognition," IEEE Trans. Cir. and Sys. for Video Technol., vol. 14, no. 1, pp. 4–20, Jan. 2004.

[20] S. I. Abuhaiba, "Offline Signature Verification Using Graph Matching," Turk J Elec Engine, vol. 15, no. 1, 2007.

[21] Turing, A. M. (1950). Computing machinery and intelligence. Mind, 433–460.

[22] Mitchell, T. (1997). Decision tree learning. Machine Learning, 414.

[23] Alpaydin, E. (2014). Introduction to machine learning. MIT press.

[24] Lai, C.-C., Doong, S.-H., and Wu, C.-H. (2007). Machine Learning. In Wiley Encyclopedia of Computer Science and Engineering. John Wiley & Sons, Inc. Retrieved from http://dx.doi.org/10.1002/9780470050118.ecse228

[25] J. R. Quinlan, "Induction of decision trees," Machine Learning, vol. 1, no. 1, pp. 81– 106, 1986.

[26] S. Salzberg, "Book review: C4.5: programs for machine learning by j. ross quinlan. morgan kaufmann publishers, inc., 1993," Machine Learning, vol. 16, no. 3, pp. 235–240, 1993.

[27] L. Liu and M. T. Özsu, eds., Encyclopedia of Database Systems. Springer US, 2009.

[28]https://dhirajkumarblog.medium.com/top-5-advantages-and-disadvantages-of-decision-tree-algorithm-428ebd199d9a

[29] KNN algorithm http://www.saedsayad.com/k_nearest_neighbors.htm

[30] k-Nearest Neighbors http://www.statsoft.com/textbook/k-nearest-neighbors

[31] https://medium.com/@anuuz.soni/advantages-and-disadvantages-of-knn-ee06599b9336

[32] V. Vural and J. G. Dy, "A hierarchical method for multi-class support vector machines," in Proc. Int. Conf. Mach. Learn., ICML 2004, Banff, Alberta, Canada, July 2004, pp. 831–838.

[33] J. C. Platt, Probabilistic Outputs for Support Vector Machines and Comparison to Regularized Likelihood Methods, MIT Press, 1999, pp. 61–74.

[34] M. M. Hossain and M. S. Miah, "Evaluation of different SVM kernels for predicting customer churn," in Proc. 18th Int. Conf. Comput. and Inform. Technol., Dhaka, Bangladesh, Dec. 2015, pp. 1–4.

[35]https://botbark.com/2019/12/19/top-5-advantages-and-disadvantages-of-support-vector-machine-algorithm/

[36] Allison, P. D. 1999. Logistic Regression Using SAS: Theory and Application. Cary, NC: SAS Institute Inc.

[37] https://www.kaggle.com/discussions/general/352871

[38] Igor Kononenko. Comparison of inductive and naive bayesian learning approaches to automatic knowledge acquisition. In B. Wielinga, J. Boose, B. Gaines, G. Schreiber, and M. van Someren, editors, Current trends in knowledge acquisition. IOS Press., 1990.

[39] https://www.upgrad.com/blog/naive-bayes-explained/

[40] F. Pedregosa et al., “Scikit-learn: Machine Learning in Python,” MACHINE LEARNING IN PYTHON, p. 6.

[41] https://blog.knoldus.com/ensemble-learning-its-methods-in-machine-learning/

[42] Chollet, F. (2017). Deep Learning with Python. Manning

[43]https://www.tutorialspoint.com/what-are-the-advantages-and-disadvantages-of-artificial-neural-networks

[44]https://github.com/Kevv-J/Signature-Verification-using-MATLAB/tree/master/DSAA%20Project%20-%20Signature%20Verification/Dataset

[45] https://www.kaggle.com/datasets/divyanshrai/handwritten-signatures

[46] https://www.kaggle.com/datasets/shreelakshmigp/cedardataset

[47] Srinivasan, Harish, Sargur N. Srihari, and Matthew J. Beal. "Machine learning for signature verification." Computer Vision, Graphics and Image Processing: 5th Indian Conference, ICVGIP 2006, Madurai, India, December 13-16, 2006. Proceedings. Springer Berlin Heidelberg, 2006.

[48] https://www.diva-portal.org/smash/get/diva2:1177227/FULLTEXT01.pdf

[49] https://www.researchgate.net/figure/Example-of-o-ffl-ine-preprocessing_fig3_276108909

[50]https://www.researchgate.net/publication/276108909_Online_and_Offline_Signature_Verification_A_Combined_Approach

[51] http://dx.doi.org/10.1002/9780470050118.ecse228

[52] https://sci-hub.hkvisa.net/10.1002/9780470050118.ecse228

[53] https://www.diva-portal.org/smash/get/diva2:861804/FULLTEXT01.pdf

[54] https://www.sfu.ca/~ljilja/cnl/pdf/Thesis_prerna.pdf

[54] https://www.sfu.ca/~ljilja/cnl/pdf/Thesis_prerna.pdf

[54] https://www.sfu.ca/~ljilja/cnl/pdf/Thesis_prerna.pdf

[55] https://www.javatpoint.com/logistic-regression-in-machine-learning

[56]https://www.analyticsvidhya.com/blog/2022/03/building-naive-bayes-classifier-from-scratch-to-perform-sentiment-analysis/

[57] https://corporatefinanceinstitute.com/resources/data-science/ensemble-methods/

[58] https://www.tibco.com/reference-center/what-is-a-neural-network